Theorie des maschinellen Lernens

Hans U. Simon

23. Oktober 2018

Inhaltsverzeichnis

1 Einleitung

Der in diesem Skriptum vermittelte Stoff wird weitgehend abgedeckt durch die folgenden Literaturquellen:

- Kapitel 1–11 und 15–20 aus [3]

- Kapitel 1 und 6 aus [2]
- Kapitel 11 aus [4]
- Kapitel 7 aus [1]

Die Hauptquelle, [3], ist eines der neueren Lehrbücher in der Theorie des maschinellen Lernens. Auf diese wird im Skriptum häufiger mit der Kurzbezeichnung „Lehrbuch“ verwiesen.

1.1 Ein einleitendes Beispiel

Nehmen wir an, wir wollten anhand einer Kollektion $\mathbf{x} = (x_1, \ldots, x_k)$ von medizinischen Messwerten (zum Beispiel die Auswertungen einer Blutprobe) feststellen, ob die betreffende Person gesundheitliche Komplikationen zu befürchten hat. Jeder Messwerte-Vektor $\mathbf{x}$ soll also entweder mit dem Prädikat „normal“ oder mit dem Prädikat „auffällig“ markiert werden. Es gibt offensichtlich zwei Arten der Fehlklassifikation:

- Klassifikation „normal“, aber in der Folge treten gesundheitliche Komplikationen auf,
- Klassifikation „auffällig“, ohne dass in der Folgezeit gesundheitliche Komplikationen auftreten.

Eine erfahrene Medizinerin bzw. ein erfahrener Mediziner wird diese Aufgabe i.A. ohne allzuviele Fehlklassifikationen lösen. Im Rahmen des maschinellen Lernens stellen wir die folgende Frage: können wir ein Computerprogramm entwerfen, dass

- zum Zwecke des Trainings Zugriff auf eine korrekt markierte Datenbank mit Records der Form $(\mathbf{x}, b)$, $b \in \{\text{normal,auffällig}\}$, bekommt
- und eine Hypothese $h : \mathbb{R}^k \to \{\text{normal,auffällig}\}$ zurückliefert, die neue Datensätze $\mathbf{x}$ möglichst korrekt mit $h(\mathbf{x})$ klassifiziert?

Dabei gehen wir insbesondere folgenden Fragen nach:

- Wie werden die Trainingsdaten für einen Lernalgorithmus generiert?
- Welche Form haben die Hypothesen und nach welcher Regel wählt der Lernalgorithmus eine Hypothese aus der Menge der möglichen Hypothesen aus?
- Woran messen wir den Erfolg eines Lernverfahrens?
- Für welche Lernprobleme existieren effiziente erfolgreiche Lernverfahren und welche Lernprobleme sind inhärent schwer?
- Welche bereits entwickelten Lernverfahren und Lernmaschinen[1] stehen uns zur Verfügung?

[1] Lernmaschine = recht allgemein anwendbares Lernverfahren, das mit Hilfe von frei wählbaren Parametern an eine konkrete Aufgabe angepasst werden kann

1.2 Einsatzfelder für maschinelles Lernen

Warum kümmern wir uns um maschinelles Lernen, anstatt eine Maschine direkt so zu programmieren, dass sie macht, was sie machen soll? Wir nennen zwei Gründe.

„Unprogrammierbare Aufgaben“: Es gibt viele Aufgaben, die wir programmierbaren Maschinen nicht auf direktem Weg übertragen können, obwohl sie von Menschen ohne großes Aufhebens erledigt werden. Beispielhaft seien genannt:

- Auto fahren
- Sprechen und Sprache verstehen
- Bilderkennung

Die betreffenden Fähigkeiten haben wir weitgehend unbewusst erworben. Unser Wissen darüber ist zu diffus, um daraus ein Computerprogramm abzuleiten.
Die folgenden Aufgaben sind *theoretisch von Menschen lösbar* (wieder ohne echte Chance, das Lösungsverfahren direkt zu programmieren), *praktisch jedoch nicht*, da die auszuwertende Datenmenge riesig ist (was für Computer kein Problem darstellt):

- Wetter- oder Klimadaten auswerten
- Schlüsse aus der Kenntnis des menschlichen Genoms ziehen
- Resultate von Internetsuchmaschinen auswerten

Wunsch nach Adaptivität: Eine nicht lernfähige sondern auf direktem Weg programmierte Maschine verfährt beim Lösen einer Aufgabe stets nach dem selben Muster. Dies ist hinderlich bei Problemstellungen, die dynamische Anpassungen erfordern. Als Beispiele seien genannt:

- Erkennung von handgeschriebenen Texten (Anpassung an die individuelle Handschrift)
- Spracherkennung (Anpassung an den jeweiligen Sprecher)
- Spam-Filter (Anpassung an neuartige Formen von Spam)

1.3 Arten des Lernens

Mathematische Modelle für maschinelles Lernen können sich entlang verschiedener Dimensionen unterscheiden:

- *überwachtes* versus *unüberwachtes Lernen*
- *aktives* versus *passives Lernen*
- *Datengenerierung wohlwollend* versus *neutral* versus *übelwollend*
- *online* versus *batch-Lernen*

Wir erklären die Unterschiede zwischen den diversen Modellen anhand von Klassifikationsaufgaben. Beim *überwachten Lernen* sind die Trainingsdaten mit den korrekten Klassifikationslabels markiert. Beim *unüberwachten Lernen* liegen nur die unmarkierten Daten vor. Beim *aktiven Lernen* kann der Lernalgorithmus sich auf aktive Weise neue Information beschaffen, indem er Fragen an ein Orakel stellt. Zum Beispiel könnte er einen Datenrecord (etwa eine Kollektion medizinischer Messwerte) vorlegen und das Orakel nach dem korrekten Klassifikationslabel fragen. Beim *passiven Lernen* hat der Lernalgorithmus keinen Einfluss auf die Generierung der Trainingsdaten. Werden die Trainingsdaten *wohlwollend generiert*, so enthalten sie besonders informative Beispiele, die den Lernalgorithmus in die Richtung einer guten Hypothese lenken. Bei übelwollender Datengenerierung sind die Beispiele im Gegensatz dazu besonders uninformativ gewählt, so dass sie den Lernprozess maximal behindern. Die *neutrale Datengenerierung* liegt zwischen diesen Extremen (wie es etwa bei zufällig ausgewählten Trainingsdaten der Fall wäre).

Unüberwachte Lernmodelle sind zum Beispiel für Clustering-Verfahren angemessen. Ein aktives Lernmodell (mit Fragen an ein Orakel) wurde 1988 von Dana Angluin vorgeschlagen. Ein berümtes Verfahren in diesem Kontext ist Angluin's Algorithmus zum aktiven Lernen von endlichen Automaten. Wohlwollende Datengenerierung finden wir bei sogenannten „models for teaching“ (mit hilfreichen Trainingsbeispielen, die von einer Lehrerin bzw. einem Lehrer ausgewählt werden). Übelwollende Datengenerierung spielt eine Rolle bei der Worstcase-Analyse sogenannter Online-Algorithmen. Während es bei Batch-Algorithmen zunächst eine Trainingsphase gibt und erst anschließend einen Übergang in den Testmodus, lernen Online-Verfahren im laufenden Betrieb. In der Vorlesung konzentrieren wir uns auf das Modell des *statistischen Lernens*. Es handelt sich dabei um ein passives, überwachtes batch-Lernen mit zufälliger Datengenerierung. Auf der Skala von „wohl-“ bis „übelwollend“ ist die Datengenerierung nicht gänzlich neutral sondern zumindest etwas übelwollend: es wird zwar vorausgesetzt, dass die Daten gemäß einer zufälligen Verteilung $\mathcal{D}$ generiert werden, aber es erfolgt eine Worstcase-Analyse die Wahl von $\mathcal{D}$ betreffend.[2]

Der Startschuss für die statistische Lerntheorie erfolgte in den 1970ern in Verbindung mit grundlegenden Arbeiten von Vapnik und Chervonenkis. Die Arbeit dieser russischen Forscher (teilweise auf russisch publiziert) wurde in westlichen Ländern nicht in ihrer Gänze erfasst. In der westlichen Hemissphäre erfolgte der Startschuss etwas verspätet durch Leslie Valiant, der 1984 in der Zeitschrift „Communications of the ACM“ das sogenannte pac-Lernmodell vorschlug und einige erste Resultate präsentierte. In den 1990ern verfeinerte Vapnik die frühen Ideen aus den 1970ern und entwarf die Support-Vektor-Maschine (SVM). Diese und die sogenannte kernbasierte Methode sorgten in der Folgezeit für Furore. Es begann die Blütezeit der kernbasierten Methoden; im Internet wurde eine Vielzahl enstprechender „Tools“ zur Verfügung gestellt. In der letzen Dekade kam es zu einer Renaissance der neuronalen Netzwerke (Stichwort: deep learning), die zwischenzeitig durch die Entdeckung der SVM und der kernbasierten Methoden etwas in den Hintergrund geraten waren. Insbesondere bei Sprach- und Bildverarbeitung spielen „Deep Neural Networks“ derzeit eine führende Rolle.

[2]Im Lernmodell wird sich das darin äußern, dass ein Lernalgorithmus die zugrunde liegende Verteilung $\mathcal{D}$ nicht kennt und grundsätzlich mit jeder Wahl von $\mathcal{D}$ zurecht kommen muss.

1.4 Beziehungen zu und Abgrenzung von anderen Gebieten

Die statistische Lerntheorie benutzt Methoden und Resultate aus folgenden Bereichen der Mathematik und der theoretischen Informatik:

Wahrscheinlichkeitstheorie und Statistik: Grundlegende Kenntnisse in Wahrscheinlichkeitstheorie werden in unserer Vorlesung vorausgesetzt. Die statistische Maschinerie, die wir bemühen, weist die folgenden Unterschiede zur klassischen Maschinerie auf:

- Berücksichtigung endlicher Stichproben anstelle von reiner Asymptotik
- uniforme Konvergenzsätze von stärkerer Relevanz als nicht-uniforme
- Verteilungsfreiheit des Modells statt Annahme spezieller Verteilungen
- starke Berücksichtigung des algorithmischen und komplexitätstheoretischen Aspektes

Algorithmik, Komplexitätstheorie: Grundlegende Kenntnisse in Algorithmik und Komplexitätstheorie, wie sie zum Beispiel in den Vorlesungen über Datenstrukturen und theoretische Informatik vermittelt werden, werden in unserer Vorlesung vorausgesetzt. In der statistischen Lerntheorie wird neben der Zeitkomplexität auch die sogenannte Informations- oder Stichprobenkomplexität analysiert.

Die statistische Lerntheorie ist ein Teilgebiet der Theorie des maschinellen Lernens, diese wiederum ein Teilgebiet der künstlichen Intelligenz. Die engen Beziehungen, welche die statistische Lerntheorie zu Wahrscheinlichkeitstheorie, Statistik, Algorithmik und Komplexitätstheorie aufweist, finden sich in anderen Bereichen der künstlichen Intelligenz in dieser Ausprägung jedoch eher nicht.

Es sei abschließend darauf hingewiesen, dass wir in der statistischen Lerntheorie nicht untersuchen, wie menschliches Lernen vonstatten geht. Weiterhin unternehmen wir keinen Versuch, mit den aufgezeigten Lernverfahren, menschliche Formen des Lernens zu imitieren. Vielmehr versuchen wir die spezifischen Stärken maschineller Systeme (wie zum Beispiel die schnelle Verarbeitung riesiger Datenmengen) voll auszunutzen.

2 Ein formales Lernmodell

In Abschnitt 2.1 führen wir die Leserin und den Leser in das Grundvokabular der statistischen Lerntheorie ein und treffen nebenbei notationelle Vereinbarungen. Das Basismodell in Abschnitt 2.2 beschäftigt sich mit dem PAC-Lernen binärer Hypothesen unter der Realisierbarkeitsannahme (Realizability Assumption). Abschnitt 2.3 behandelt Verallgemeinerungen des Basismodells. Abschnitt 2.4 erweitert das PAC-Lernmodell um Effizienzkriterien. Zu diesem Zweck werden parametrisierte Lernprobleme eingeführt. Ein wichtiges Markenzeichen des im Folgenden vorgestellten Lernmodells ist seine „Verteilungsunabhängigkeit“. Darauf gehen wir in Abschnitt 2.5 kurz ein.

2.1 Das Rahmenwerk der statistischen Lerntheorie

Die Aufgabe für einen Lernalgorithmus A wird darin bestehen, einen Datenpunkt x, genannt *Instanz*, mit einer (im Idealfall) korrekten Markierung zu versehen. Die Instanzen stammen aus einem *Instanzenraum* $\mathcal{X}$ und die Markierungen aus einer *Markierungsmenge* $\mathcal{Y}$. Gilt $\mathcal{Y} = \{0, 1\}$, so sprechen wir von einem *binären Klassifikationsproblem*. Ist $\mathcal{Y}$ eine endliche Menge mit evtl. mehr als zwei Elementen, so sprechen wir von einem *Multiklassen-Klassifikationsproblem*. Ein *Regressionsproblem* liegt vor, falls $\mathcal{Y} = \mathbb{R}$. Wir setzen im Folgenden voraus, dass $\mathcal{Y}$ endlich ist und dass somit ein Klassifikationsproblem vorliegt. Regressionsprobleme diskutieren wir zu einem späteren Zeitpunkt der Vorlesung.

Damit ein Lernalgorithmus A seine Aufgabe erfüllen kann, versorgen wir ihn in einer Trainingsphase mit einer Sequenz

$$S = [(x_1, y_1), \ldots, (x_m, y_m)] \in (\mathcal{X} \times \mathcal{Y})^m$$

korrekt markierter Instanzen. Eine korrekt markierte Instanz wird auch *Beispiel* genannt. Beispiele stammen also aus der Menge $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. Die Beispielsequenz $S \in \mathcal{Z}^m$ heißt *Trainingssequenz* oder auch *Stichprobe*. Aus S soll A eine *Hypothese* $h = A(S)$ ableiten. Dies ist eine Funktion $h : \mathcal{X} \to \mathcal{Y}$, welche einer Instanz $x \in \mathcal{X}$ eine Markierung $h(x) \in \mathcal{Y}$ zuordnet. Salopp gesprochen gilt: A ist der Ansicht, dass $h(x)$ die korrekte Markierung für x ist. Im Rahmen der statistischen Lerntheorie werden folgende Grundannahmen gemacht:

- Es gibt eine (Wahrscheinlichkeits-)Verteilung $\mathcal{D}$ auf der Beispielmenge $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ und die Beispiele (x_i, y_i) aus S sind unabhängig voneinander zufällig (bezüglich $\mathcal{D}$) gewählt, d.h., $S \sim \mathcal{D}^m$.

- Eine Hypothese $h : \mathcal{X} \to \mathcal{Y}$ wird anhand ihrer *realen Fehlerrate*

$$L_{\mathcal{D}}(h) = \Pr_{(x,y)\sim\mathcal{D}}[h(x) \neq y] \tag{1}$$

  bewertet. $L_{\mathcal{D}}(h)$ ist Wahrscheinlichkeit, mit der h auf einem neuen zufälligen Testbeispiel einen Fehler produziert.

Beachte, dass (fairerweise) die reale Fehlerrate mit der selben Verteilung $\mathcal{D}$ ermittelt wird, welche auch bei der Generierung der Trainingssequenz zugrunde gelegt wurde. Beachte weiterhin, dass der Lernalgorithmus die zugrunde gelegte Verteilung $\mathcal{D}$ nicht kennt. Die Ermittlung von $h = A(S)$ beruht einzig und allein auf der Kenntnis der Trainingssequenz S.

Bei einem zufälligen Beispiel $(x, y) \sim \mathcal{D}$ ist die Markierung y i.A. keine Funktion von x: es kann Instanzen $x \in \mathcal{X}$ und verschiedene Markierungen $y_0, y_1 \in \mathcal{Y}$ mit $\mathcal{D}((x, y_0)) > 0$ und $\mathcal{D}((x, y_1)) > 0$ geben. In diesem Fall kann es keine fehlerfreie Hypothese geben. Andererseits ist auch nicht grundsätzlich ausgeschlossen, dass die Verteilung $\mathcal{D}$ eine fehlerfreie Hypothese zulässt. Die folgende Definition behandelt diesen Sonderfall.

Definition 2.1 *Wir sagen eine Verteilung $\mathcal{D}$ auf $\mathcal{X} \times \mathcal{Y}$ und eine Hypothesenklasse $\mathcal{H} \subseteq \{h : \mathcal{X} \to \mathcal{Y}\}$ erfüllen die* Realisierbarkeitsannahme*, falls ein $h \in \mathcal{H}$ mit $L_{\mathcal{D}}(h) = 0$ existiert.*

Mitunter werden wir zu gegebenem $\mathcal{H}$ eine Verteilung $\mathcal{D}$ entwerfen, welche die Realisierbarkeitsannahme erfüllt, indem wir

- eine Verteilung $\mathcal{P}$ auf $\mathcal{X}$ angeben,
- eine Hypothese $h \in \mathcal{H}$ auswählen,
- und schließlich $\mathcal{D}$ als die von $\mathcal{P}$ und h induzierte Verteilung auf $\mathcal{X} \times \mathcal{Y}$ definieren:

Die Instanz x eines bezüglich $\mathcal{D}$ zufälligen Beispieles wird gemäß $\mathcal{P}$ zufällig gewählt und dann mit $h(x)$ markiert. Beachte, dass $\mathcal{P}$ dann mit der Randverteilung $\mathcal{D}_{\mathcal{X}}$ von $\mathcal{D}$ auf $\mathcal{X}$ übereinstimmt. Wenn $S = [(x_1, y_1), \ldots, (x_m, y_m)] \sim \mathcal{D}^m$ eine zufällige Trainingssequenz ist, dann bezeichnet $S_{\mathcal{X}} = (x_1, \ldots, x_m) \sim \mathcal{D}_{\mathcal{X}}^m$ die zugehörige Sequenz ohne die Labels. Zur besseren Unterscheidbarkeit sprechen wir im Falle von $S_{\mathcal{X}}$ von einer *unmarkierten* Trainingssequenz (wohingegen der Terminus „Trainingssequenz“ in der Regel eine *markierte* Trainingssequenz S meint).

Das allgemeine Lernszenario (ohne die Realisierbarkeitsannahme) wird auch als *agnostisches Szenario* bezeichnet. Im agnostischen Szenario ist die reale Fehlerrate einer bestmöglichen Hypothese i.A. echt größer als 0. In der Regel unterstellen wir, dass wir uns im agnostischen Szenario befinden, es sei denn wir machen ausdrücklich die Realisierbarkeitsannahme.

Welche Hinweise kann uns eine Trainingssequenz $S = [(x_1, y_1), \ldots, (x_m, y_m)]$ für die Beurteilung einer Hypothese geben? Einer der relevanten Parameter ist die *empirische Fehlerrate*

$$L_S(h) = \frac{1}{m} \cdot |\{i \in \{1, \ldots, m\} |\ h(x_i) \neq y_i\}|$$

von h, also die mittlere Anzahl der von h vorgenommenen Fehlklassifikationen auf der Trainingssequenz S. Dabei ist Vorsicht geboten! Folgendes Beispiel für ein binäres Klassifikationsproblem zeigt auf, dass Hypothesen mit empirischer Fehlerrate 0 eine reale Fehlerrate von 1/2 besitzen können (womit sie nicht besser abschneiden als zufälliges Erraten der korrekten Markierung).

Beispiel 2.2 *Es sei $\mathcal{X} = [0, 1]$, $\mathcal{Y} = \{0, 1\}$, $\mathcal{P}$ die uniforme Verteilung auf $\mathcal{X}$,*

$$f(x) = \mathbb{1}_{[x \geq 1/2]} = \begin{cases} 1 & \text{if } x \geq 1/2 \\ 0 & \text{otherwise} \end{cases},$$

und $\mathcal{D}$ die von $\mathcal{P}$ und f induzierte Verteilung auf $\mathcal{X} \times \{0, 1\}$. Wir befinden uns also im nicht-agnostischen Szenario. Es sei $S = [(x_1, y_1), \ldots, (x_m, y_m)]$ eine Trainingssequenz. Betrachte die Hypothese h, welche alle Instanzen $x_1, \ldots, x_m$ korrekt markiert und allen nicht in S vorkommenden Instanzen die Markierung 0 zuordnet. Offensichtlich gilt $L_S(h) = 0$ und $L_{\mathcal{D}}(h) = 1/2$.

Es ist leicht zu sehen, woran die Hypothese h aus Beispiel 2.2 scheitert: die Markierungen für die Instanzen aus S wurden auswendig gelernt und auf allen anderen Instanzen wird die willkürliche Entscheidung für die 0-Markierung getroffen. Man spricht in diesem Kontext

von einer *Überanpassung (overfitting)* an die Trainingssequenz. Wenn dem Lernalgorithmus alle Funktionen $h : \mathcal{X} \to \mathcal{Y}$ als potenzielle Hypothesen zur Verfügung stehen, dann ist die Gefahr von Überanpassung an S sehr groß. Eine Möglichkeit, dieser Gefahr zu begegnen, besteht darin, eine *Hypothesenklasse* $\mathcal{H}$ zu wählen, welche nicht alle Funktionen von $\mathcal{X}$ nach $\mathcal{Y}$ umfasst, und dem Lernalgorithmus nur Hypothesen $h \in \mathcal{H}$ zur Verfügung zu stellen.[3] Tendenziell sollte $\mathcal{H}$

einerseits so reichhaltig wie nötig sein (damit eine Anpassung an die Daten möglich ist und eine kleine empirische Fehlerrate realisiert werden kann), aber

andererseits so spärlich wie möglich (damit Überanpassung an die Trainingssequenz vereitelt wird).

Eine wichtige erste Strategie zur Auswahl einer Hypothese $h \in \mathcal{H}$ liefert die folgende

ERM-Lernregel: Gegeben eine Trainingssequenz S wähle eine Hypothese $h \in \mathcal{H}$ mit kleinstmöglicher empirischer Fehlerrate, d.h.,

$$h = \text{argmin}_{h \in \mathcal{H}} L_S(h) \ .$$

„ERM“ steht hierbei für „Empirische Risikominimierung“.

Aus der Realisierbarkeitsannahme für $\mathcal{D}$ und $\mathcal{H}$ folgt leicht, dass eine zufällige Trainingssequenz S fast sicher mit einer geeigneten Hypothese $h \in \mathcal{H}$ fehlerfrei markiert werden kann. Dies impliziert, dass die ERM-Lernregel eine Hypothese $h \in \mathcal{H}$ mit $L_S(h) = 0$ auswählt.

Selbst unter der Realisierbarkeitsannahme liefert die Trainingssequenz S nur eine bruchstückhafte Information über die Verteilung $\mathcal{P} = \mathcal{D}_{\mathcal{X}}$ auf $\mathcal{X}$ und eine perfekte Klassifikationsregel $h \in \mathcal{H}$. Wir können daher nicht erwarten, dass A eine Hypothese mit realer Fehlerrate 0 findet. Aus ähnlichen Gründen können wir im allgemeinen Fall nicht erwarten, dass A eine Hypothese $h \in \mathcal{H}$ mit der kleinstmöglichen realen Fehlerrate $\inf_{h' \in \mathcal{H}} L_{\mathcal{D}}(h')$ findet. Es sei $0 < \varepsilon < 1$. Wir sagen eine Hypothese $h \in \mathcal{H}$ ist *ε-akkurat*, falls

$$L_{\mathcal{D}}(h) \leq \varepsilon + \inf_{h' \in \mathcal{H}} L_{\mathcal{D}}(h')$$

bzw., äquivalent dazu, falls

$$\forall h' \in \mathcal{H} : L_{\mathcal{D}}(h) \leq \varepsilon + \mathcal{D}(h') \ .$$

Unter der Realisierbarkeitsannahme gilt dann wegen $\inf_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') = 0$ sogar $_{\mathcal{D}}(h) \leq \varepsilon$. Der Parameter ε heißt *Akkuratheitsparameter*. Er ist ein Maß für die zugelassene „Ungenauigkeit“ oder „Suboptimalität“ einer Hypothese $h \in \mathcal{H}$ im Vergleich zur optimalen Hypothese aus $\mathcal{H}$.

Zu erwarten, dass $h = A(S)$ eine ε-akkurate Hypothese ist, ist immer noch überehrgeizig, da (zumindest mit einer kleinen von 0 verschiedenen Wahrscheinlichkeit) die Trainingssequenz S uninformativ sein kann:

[3] Eine weitere Möglichkeit ist die Verwendung von Kostentermen, sogenannte *Regularisierungsterme*, welche komplizierte Klassifikationsregeln mit höheren Kosten belegen als einfache solche Regeln. Die Gesamtkosten ergeben sich dann als Summe aus empirischer Fehlerrate und Regularisierungsterm.

Beispiel 2.3 *Betrachten wir eine Datenbank, in welcher eine große Zahl von Tier-Individuen mit Merkmalsvektoren repräsentiert ist. U.a. enthalte ein Merkmalsvektor eine Boolesche Komponente mit dem Wert* 1*, falls das betreffende Tier-Individuum Eier legt, und dem Wert* 0 *sonst. Mit einer winzigen, aber von* 0 *verschiedenen, Wahrscheinlichkeit enthält die Trainingssequenz* $S \sim \mathcal{D}^m$ *nur Individuen die entweder Schnabeltiere*[4] *oder Nicht-Säugetiere sind. Nehmen wir an, der Lernalgorithmus* A *soll auf Basis von* S *das binäre Klassifikationsproblem „Säugetier, ja oder nein?“ lösen. Es erscheint dann plausibel, dass die Hypothese* $h = A(S)$ *ein weibliches Individuum höchstens dann als Säugetier klassifiziert, wenn es Eier legt. Eine Hypothese dieser Art hätte zwangsläufig eine hohe reale Fehlerrate.*

Die Schlussfolgerung, die wir ziehen, lautet: bei Vorliegen einer miesen Trainingssequenz dürfen wir das Versagen der Hypothese $h = A(S)$ nicht dem Lernalgorithmus A anlasten. Wir werden daher nur verlangen, dass A mit einer Wahrscheinlichkeit von mindestens $1 - \delta$ eine ε-akkurate Hypothese vorlegen soll. Der Parameter $0 < \delta < 1$ heißt *Konfidenzparameter*. Er ist ein Maß für die „tolerierte Unzuverlässigkeit“ der zufälligen Trainingssequenz. Unter der Realisierbarkeitsannahme hat das genannte (ε, δ)-Erfolgskriterium die Form: Mit einer Wahrscheinlichkeit von mindestens $1-\delta$ soll A eine Hypothese $h = A(S)$ mit realer Fehlerrate $L_{\mathcal{D}}(h) \leq \varepsilon$ abliefern. Die Hypothese h ist in diesem Sinne „Probably Approximately Correct (PAC)“. Das resultierende Lernmodell ist daher auch unter der Bezeichnung „PAC-Lernen“ bekannt geworden.

Es folgt eine tabellarische Übersicht über die in diesem Abschnitt eingeführten Symbole und ihre Bedeutung:

Symbol	Bedeutung
$\mathcal{X}$	Instanzenraum
$\mathcal{Y}$	Markierungsmenge
$\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$	Menge der Beispiele
$\mathcal{D}$	Verteilung auf $\mathcal{Z}$
$\mathcal{P}$	Verteilung auf $\mathcal{X}$ (meist identisch zu $\mathcal{D}_{\mathcal{X}}$)
$S \sim \mathcal{D}^m$	markierte zufällige Trainingssequenz
$S_{\mathcal{X}} \sim \mathcal{P}^m$	unmarkierte zufällige Trainingssequenz
A	Lernalgorithmus
$h = A(S) : \mathcal{X} \to \mathcal{Y}$	aus S abgeleitete Hypothese von A
$L_{\mathcal{D}}(h)$	reale Fehlerrate der Hypothese h
$L_S(h)$	empirische Fehlerrate der Hypothese h
$\mathcal{H}$	Hypothesenklasse
$\varepsilon \in (0, 1)$	Akkuratheitsparameter (zugelassene Ungenauigkeit von h)
$\delta \in (0, 1)$	Konfidenzparameter (tolerierte Unzuverlässigkeit von S)

2.2 PAC-Lernen unter der Realisierbarkeitsannahme

Wir verwenden die Standardnotationen, die aus dem vorangegangenen Kapitel bekannt sind. In diesem Abschnitt beschäftigen wir uns mit binären Klassifikationsproblemen und setzen

[4] das einzige Eier legende Säugetier auf dem Planeten Erde

daher $\mathcal{Y} = \{0, 1\}$ und $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$. Folgende Definition legt ein erstes Basismodell des PAC-Lernens fest:

Definition 2.4 (PAC-Lernen unter der Realisierbarkeitsannahme) *Eine binäre Hypothesenklasse $\mathcal{H}$ heißt* PAC-lernbar unter der Realisierbarkeitsannahme*, wenn eine Funktion $m_{\mathcal{H}} : (0, 1)^2 \to \mathbb{N}$ und ein Lernalgorithmus A existieren, so dass für alle $0 < \varepsilon, \delta < 1$ sowie für jede Wahl von $\mathcal{D}$ mit der Eigenschaft*

$$\exists h' \in \mathcal{H} : L_{\mathcal{D}}(h') = 0 \quad \textit{(Realisierbarkeitsannahme)}$$

die folgende Bedingung erfüllt ist: A, gestartet auf einer bezüglich $\mathcal{D}$ zufälligen Trainingssequenz S der Größe $m \geq m(\varepsilon, \delta)$, ermittelt eine Hypothese $h : \mathcal{X} \to \{0, 1\}$ und es gilt

$$\Pr_{S \sim \mathcal{D}^m}[L_{\mathcal{D}}(h) \leq \varepsilon] \geq 1 - \delta \ . \tag{2}$$

In Worten: mit einer Wahrscheinlichkeit von mindestens $1 - \delta$ wird eine zufällige Trainingssequenz generiert, aus der A eine Hypothese mit einer realen Fehlerrate von maximal ε extrahiert.

Im Folgenden bezeichnen wir mit $m_{\mathcal{H}}$ die Funktion $m_{\mathcal{H}}$, die der Definition 2.4 genügt und unter dieser Nebenbedingung die kleinstmöglichen Werte $m_{\mathcal{H}}(\varepsilon, \delta)$ hat. Sie wird als *Stichprobenkomplexität (sample complexity)* oder auch *Informationskomplexität* von $\mathcal{H}$ bezeichnet. Wir ermittlen nun eine obere Schranke für die Informationskomplexität endlicher Hypothesenklassen.

Theorem 2.5 *Für das PAC-Lernen endlicher binärer Hypothesenklassen $\mathcal{H}$ gilt im PAC-Lernmodell unter der Realisierbakeitsannahme*

$$m_{\mathcal{H}}(\varepsilon, \delta) \leq \left\lceil \frac{\ln(|\mathcal{H}|/\delta)}{\varepsilon} \right\rceil \ .$$

Beweis Unter der Realisierbarkeitsannahme enthält $\mathcal{H}$ eine Hypothese, die auf der Trainingssequenz S fehlerfrei ist. Die ERM-Lernregel wird daher eine Hypothese $h \in \mathcal{H}$ mit $L_S(h) = 0$ wählen. Es sei $\mathcal{H}_\varepsilon$ die Menge aller Hypothesen aus $\mathcal{H}$ mit einer realen Fehlerrate von mindestens ε. Wenn alle Hypothesen aus $\mathcal{H}_\varepsilon$ auf S mindestens einen Fehler machen, dann heißt das im Umkehrschluss: die gemäß der ERM-Lernregel ermittelte Hypothese ist ε-akkurat. Es genügt daher im Folgenden, die Wahrscheinlichkeit für das Ereignis

$$\text{BAD} \Leftrightarrow (\exists h \in \mathcal{H}_\varepsilon : L_S(h) = 0)$$

mit δ nach oben zu beschränken. Betrachte das Ereignis

$$\text{BAD}(h) \Leftrightarrow L_S(h) = 0 \ .$$

Nach den Gesetzen der Wahrscheinlichkeitstheorie gilt für jede individuelle Hypothese $h \in \mathcal{H}_\varepsilon$:

$$\Pr_{S\sim\mathcal{D}^m}[\mathrm{BAD}(h)] \leq (1-\varepsilon)^m < e^{-\varepsilon m} \ .$$

Wegen $\mathrm{BAD} \Leftrightarrow (\exists h \in \mathcal{H}_\varepsilon : \mathrm{BAD}(h))$ folgt

$$\Pr_{S\sim\mathcal{D}^m}[\mathrm{BAD}] < |\mathcal{H}_\varepsilon| e^{-\varepsilon m} \leq |\mathcal{H}| e^{-\varepsilon m} \ .$$

Wir machen den Ansatz

$$|\mathcal{H}| e^{-\varepsilon m} \leq \delta \ .$$

Auflösen nach m liefert

$$m \geq \frac{\ln(|\mathcal{H}|/\delta)}{\varepsilon} \ . \tag{3}$$

Wir haben nachgerechnet, dass jede Wahl von m gemäß (3) die (ε, δ)-Bedingung des PAC-Lernens gewährleistet. Eine Trainingssequenz der Größe

$$m = \left\lceil \frac{\ln(|\mathcal{H}|/\delta)}{\varepsilon} \right\rceil$$

reicht daher aus. •

Im Beweis zu Theorem 2.5 wurde die ERM-Lernregel verwendet. Daher gilt:

Folgerung 2.6 *Jede endliche binäre Hypothesenklasse ist unter der Realisierbarkeitsannahme PAC-lernbar, und zwar mit Hilfe der ERM-Lernregel.*

2.3 Verallgemeinerungen des Basismodells

Wir betrachten zunächst wieder binäre Klassifikationsprobleme, d.h., wir setzen $\mathcal{Y} = \{0, 1\}$. Die Realisierbarkeitsannahme ist bei Anwendungen fast nie erfüllt. Realistischer ist es anzunehmen, dass markierte Beispiele (x, y) bezüglich einer (dem Lerner unbekannten) Verteilung $\mathcal{D}$ auf $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ generiert werden (agnostisches Szenario). Anstatt eine reale Fehlerrate nahe bei Null anzustreben, müssen wir uns dann mit einer realen Fehlerrate begnügen, die nah an der besten mit Hypothesen aus $\mathcal{H}$ erzielbaren realen Fehlerrate liegt. Diese Überlegungen führen zu folgender Variante des Basismodells:

Definition 2.7 (Agnostisches PAC-Lernen) *Eine binäre Hypothesenklasse $\mathcal{H}$ heißt* agnostisch PAC-lernbar*, wenn eine Funktion $m_\mathcal{H} : (0,1)^2 \to \mathbb{N}$ und ein Lernalgorithmus A existieren, so dass für alle $0 < \varepsilon, \delta < 1$ sowie für jede Verteilung $\mathcal{D}$ auf $\mathcal{Z}$ folgende Bedingung erfüllt ist: A, gestartet auf einer zufälligen Trainingssequenz $S \sim D^m$ mit $m \geq m(\varepsilon, \delta)$, ermittelt eine Hypothese $h : \mathcal{X} \to \{0, 1\}$ und es gilt*

$$\Pr_{S\sim D^m}\left[L_\mathcal{D}(h) \leq \varepsilon + \inf_{h'\in\mathcal{H}} L_\mathcal{D}(h')\right] \geq 1 - \delta \ . \tag{4}$$

In Worten: mit einer Wahrscheinlichkeit von mindestens $1-\delta$ wird eine zufällige Trainingssequenz generiert, aus der A eine Hypothese extrahiert, deren reale Fehlerrate die bezüglich $\mathcal{H}$ bestmögliche reale Fehlerrate um maximal ε überschreitet.

Unter der Realisierbarkeitsannahme ergibt sich $\inf_{h' \in \mathcal{H}} L_{\mathcal{D}}(h') = 0$ und das Erfolgskriterium (4) kollabiert zu dem aus Definition 2.4 bekannten Kriterium (2).

Die Definitionen 2.4 und 2.7 sind auf binäre Klassifikationsprobleme zugeschnitten. Implizit haben wir eine sogenannte *Null-Eins-Verlustfunktion* der Form

$$\ell_{0-1}(h, (x, y)) = \begin{cases} 0 & \text{falls } h(x) = y \\ 1 & \text{falls } h(x) \neq y \end{cases} \tag{5}$$

verwendet. Beachte, dass

$$L_{\mathcal{D}}(h) = \Pr_{(x,y)\sim\mathcal{D}}[h(x) \neq y] = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell_{0-1}(h, (x, y))] \ ,$$

d.h., die Fehlerrate von h deckt sich mit dem für h zu erwartenden Null-Eins-Verlust.
Es ist wünschenswert eine breitere Palette von Problemen zu erfassen. Zum Beispiel:

Multiklassen-Klassifikationsprobleme: Es gibt $k \geq 2$ mögliche Klassifikationslabels. Wir setzen dann $\mathcal{Y} = [k]$. Ein Beispiel $(x, i) \in \mathcal{X} \times \mathcal{Y}$ zeigt an, dass x das i-te Klassifikationslabel erhalten hat. Wir können weiterhin mit der Null-Eins-Verlustfunktion agieren.

Regressionsprobleme: Es geht um die Vorhersage reeller Werte. Wir setzen dann $\mathcal{Y} = \mathbb{R}$. Als Verlustfunktionen bieten sich an der *quadratische Fehler*

$$\ell_{sq}(h, (x, y)) = (h(x) - y)^2$$

oder auch der absolute Fehler

$$\ell_{abs}(h, (x, y)) = |h(x) - y| \ ,$$

den h auf dem Beispiel (x, y) produziert.

Unüberwachtes Lernen Während beim überwachten Lernen Beispiele die Form $z = (x, y)$ haben und aus der Menge $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ stammen, sind beim unüberwachten Lernen die Beispiele $z \in \mathcal{Z}$ unmarkiert. Was geeignete Verlustfunktionen betrifft, sei auf die einschlägige Literatur zu „Clustering“ verwiesen (zum Beispiel Kapitel 22 des Lehrbuches von Shalev-Shwartz und Ben-David).

Diese Überlegungen führen schließlich zu folgender recht allgemein gefassten Variante des PAC-Lernens:

Definition 2.8 (Agnostisches PAC-Lernen (allgemeiner Fall)) *Eine Hypothesenklasse $\mathcal{H}$ heißt* agnostisch PAC-lernbar bezüglich einer Verlustfunktion $\ell : \mathcal{H} \times \mathcal{Z} \to \mathbb{R}_+$, *wenn eine Funktion $m_{\mathcal{H}} : (0, 1)^2 \to \mathbb{N}$ und ein Lernalgorithmus A existieren, so dass für alle $0 < \varepsilon, \delta < 1$ sowie für jede Verteilung $\mathcal{D}$ auf $\mathcal{Z}$ folgende Bedingung erfüllt ist: A, gestartet auf einer zufälligen Trainingssequenz $S \sim D^m$ mit $m \geq m_{\mathcal{H}}(\varepsilon, \delta)$, ermittelt eine Hypothese $h \in \mathcal{H}$ und es gilt*

$$\Pr_{S\sim D^m}\left[L_{\mathcal{D}}(h) \leq \varepsilon + \inf_{h' \in \mathcal{H}} L_{\mathcal{D}}(h')\right] \geq 1 - \delta \ , \tag{6}$$

wobei $L_{\mathcal{D}}$ auf $\mathcal{H}$ definiert ist wie folgt:

$$L_{\mathcal{D}}(h') = \mathbb{E}_{z\sim\mathcal{D}}[\ell(h', z)] \ .$$

In Worten: mit einer Wahrscheinlichkeit von mindestens $1-\delta$ wird eine zufällige Trainingssequenz generiert, aus der A eine Hypothese aus $\mathcal{H}$ extrahiert, deren erwarteter Verlustwert den bezüglich $\mathcal{H}$ kleinstmöglichen erwarteten Verlustwert um maximal ε überschreitet.

Eine Variante des agnostischen PAC-Lernmodells ist „Darstellungs-unabhängiges agnostisches Lernen“. Hier wird dem Algorithmus erlaubt, seine Hypothesen aus einer evtl. größeren Klasse $\mathcal{H}' \supseteq \mathcal{H}$ zu wählen. Die Verlustfunktion hat dann die Form $\ell : \mathcal{H}' \times \mathcal{Z} \to \mathbb{R}_+$, d.h., sie ist auch auf Hypothesen der erweiterten Klasse $\mathcal{H}'$ definiert. Es gilt aber weiterhin das Erfolgskriterium (6). Die von A abgelieferte Hypothese (evtl. kein Mitglied von $\mathcal{H}$) wird also weiterhin mit der besten Hypothese aus $\mathcal{H}$ verglichen. Wie wir später sehen werden, ist diese Variante insbesondere unter Effizienzkriterien interessant.

Die uns aus dem Abschnitt 2.2 bekannte Definition der Informationskomplexität können wir auf die allgemeine Definition 2.8 übertragen: die Funktion $m_{\mathcal{H}}$, die der Definition 2.8 genügt und unter dieser Nebenbedingung die kleinstmöglichen Werte $m_{\mathcal{H}}(\varepsilon, \delta)$ hat, wird als *Informationskomplexität* von $\mathcal{H}$ (bezüglich der Verlustfunktion ℓ) bezeichnet.

2.4 Parametrisierte Lernprobleme und effizientes PAC-Lernen

Eine (mit einem Komplexitätsparameter n) parametrisierte Hypothesenklasse hat die Form $\mathcal{H} = (\mathcal{H}_n)_{n\geq 1}$. Entsprechend ist die Beispielmenge $\mathcal{Z} = (\mathcal{Z}_n)_{n\geq 1}$ und die Verlustfunktion $\ell = (\ell_n)_{n\geq 1}$ parametrisiert. Die Verlustfunktion ℓ_n hat die Form $\ell_n : \mathcal{H}_n \times \mathcal{Z}_n \to \mathbb{R}_+$. $\mathcal{H}$ gilt dann bezüglich ℓ als agnostisch PAC-lernbar, wenn jede Unterklasse $\mathcal{H}_n$ bezüglich ℓ_n agnostisch PAC-lernbar ist. $\mathcal{H}$ heißt bezüglich ℓ *effizient agnostisch PAC-lernbar*, falls zusätzlich gilt:

1. Es gibt einen allgemeinen Algorithmus[5] A, der für jede Wahl von $n \geq 1$ die Hypothesenklasse $\mathcal{H}_n$ agnostisch bezüglich ℓ_n lernt, sofern er auf einer (hinreichend großen) Trainingssequenz mit Beipielen aus $\mathcal{Z}_n$ gestartet wird.

2. Die für A zum agnostischen Lernen von $\mathcal{H}_n$ erforderliche Anzahl von Trainingsbeispielen ist polynomiell in $n, 1/\varepsilon, 1/\delta$ beschränkt (beherrschbare Informationskomplexität).

3. Die Laufzeit von A auf Trainingssequenzen $S \in \mathcal{Z}_n^m$ ist polynomiell in m und n beschränkt (beherrschbare Berechnungskomplexität).

In Anwendungen wie Booleschen bzw. geometrischen Klassifikationsproblemen, gilt häufig $\mathcal{Z}_n = \mathcal{X}_n \times \{0, 1\}$ mit $\mathcal{X}_n = \{0, 1\}^n$ bzw. $\mathcal{X}_n = \mathbb{R}^n$. Die Hypothesen aus $\mathcal{H}_n$ sind dann Boolesche oder geometrische Klassifikatoren $h : \mathcal{X}_n \to \{0, 1\}$. Für jede Wahl von n wird $\ell_n : \mathcal{H}_n \times \mathcal{Z}_n \to \mathbb{R}_+$ als Null-Eins-Verlustfunktion gewählt.

[5] im Unterschied zu einem separaten Algorithmus A_n für jede Wahl von n

2.5 Verteilungsunabhängigkeit des PAC-Lernmodells

Das PAC-Lernmodell legt fest, dass ein Lernalgorithmus A sein Erfolgskriterium für jede Wahl der Verteilung $\mathcal{D}$ erreichen muss. Er hat dabei keinerlei Vorwissen über $\mathcal{D}$ und kann lediglich über die zufällige Trainingssequenz S Rückschlüsse auf $\mathcal{D}$ tätigen. Zum Beispiel könnte der empirische Verlustwert (bei Klassifikationsproblemen die empirische Fehlerrate)

$$L_S(h) = \frac{1}{m} \cdot \sum_{i=1}^{m} \ell(h, z_i)$$

für eine Trainingssequenz $S = (z_1, \ldots, z_m) \sim \mathcal{D}^m$ ein guter Schätzwert für $L_D(h)$ sein.

Lernmodelle, bei denen der Lernerfolg unabhängig von der Wahl der Verteilung $\mathcal{D}$ erreicht werden muss, heißen *verteilungsunabhängige Modelle*. Im Gegensatz dazu stehen *verteilungsabhängige Modelle*, bei denen die Algorithmen auf bestimmte Verteilungen (die evtl. auch noch schöne mathematische Eigenschaften haben) spezialisiert sein dürfen. Verteilungsunabhängige Modelle stellen wesentlich höhere Anforderungen an den Lerner.

3 Lernen und uniforme Konvergenz (endlicher Fall)

In Abschnitt 3.1 wird gezeigt, dass die uniforme Konvergenz[6] von $L_S(h)$ gegen $L_{\mathcal{D}}(h)$ garantiert, dass die Hypothesenklasse $\mathcal{H}$ agnostisch PAC-lernbar ist. Zum Beispiel ist dann „Empirical Risk Minimization (ERM)“ eine geeignete Lernstrategie. In Abschnitt 3.2 weisen wir nach, dass alle endlichen Hypothesenklassen (bezüglich beschränkter Verlustfunktionen) agnostisch PAC-lernbar sind. In dem abschließenden Abschnitt 3.3 gehen wir kurz auf den sogenannten „bias-complexity tradeoff“ ein.

3.1 Uniforme Konvergenz als Garant für Lernbarkeit

„Empirical Risk Minimization (ERM)“ ist folgende Lernstrategie: gegeben eine Trainingssequenz $S \subseteq Z^m$, ermittle eine Hypothese $h_S \in \mathcal{H}$, die $L_S(h)$ minimiert, d.h., es gilt

$$L_S(h_S) = \min_{h \in \mathcal{H}} L_S(h) \ . \tag{7}$$

Erinnern wir uns daran, dass der Lerner eigentlich $L_{\mathcal{D}}(h)$ anstelle von $L_S(h)$ minimieren möchte (aber leider kennt er ja die Wahrscheinlichkeitsverteilung $\mathcal{D}$ nicht). Wenn $L_S(h)$ für alle $h \in \mathcal{H}$ eine gute Schätzung von $L_{\mathcal{D}}(h)$ ist, dann sollte ERM eine fast-optimale Hypothese finden, richtig? Die folgenden Ausführungen präzisieren diese Intuition.

Definition 3.1 *Eine Trainingssequenz $S \in Z^m$ heißt* ε-repräsentativ für eine Hypothese $h : X \to \{0, 1\}$, *falls* $|L_S(h) - L_{\mathcal{D}}(h)| \leq \varepsilon$. *Ferner heißt S* ε-repräsentativ für $\mathcal{H}$, *falls*

$$\forall h \in \mathcal{H} : |L_S(h) - L_{\mathcal{D}}(h)| \leq \varepsilon \ , \tag{8}$$

d.h. S ist ε-repräsentativ für alle Hypothesen aus $\mathcal{H}$.

[6] uniform über alle Wahlen von $h \in \mathcal{H}$

Falls $\mathcal{H}$ aus dem Kontext ersichtlich ist, werden wir einfach „ε-repräsentativ“ statt „ε-repräsentativ für $\mathcal{H}$“ sagen.

Lemma 3.2 *Wir setzen voraus, dass $\mathcal{H}$ endlich ist, so dass eine „optimale Hypothese“ $h^* \in \mathcal{H}$ mit $L_{\mathcal{D}}(h^*) = \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h)$ existiert. Weiter sei S $\varepsilon/2$-repräsentativ und h_S die von ERM (angewendet auf S) ermittelte Hypothese. Dann gilt $L_{\mathcal{D}}(h_S) \leq \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h) + \varepsilon$.*

Beweis Der Beweis nutzt (8) aus (mit $\varepsilon/2$ in der Rolle von ε) und die Tatsache, dass ERM den „empirischen Champion“ (im Sinne von (7)) ermittelt:

$$L_{\mathcal{D}}(h_S) \overset{(8)}{\leq} L_S(h_S) + \frac{\varepsilon}{2} \overset{(7)}{\leq} L_S(h^*) + \frac{\varepsilon}{2} \overset{(8)}{\leq} L_{\mathcal{D}}(h^*) + \varepsilon \ .$$

•

Definition 3.3 (uniforme Konvergenzbedingung (UKB)) *Eine Hypothesenklasse $\mathcal{H}$ erfüllt die* uniforme Konvergenzbedingung (UKB) *(bezüglich einer Verlustfunktion ℓ), falls eine Funktion $m_{\mathcal{H}}^{UC} : (0,1)^2 \to \mathbb{N}$ existiert, so dass für alle $0 < \varepsilon, \delta < 1$, alle $m \geq m_{\mathcal{H}}^{UC}(\varepsilon, \delta)$ und jede Wahl von $\mathcal{D}$ folgende Bedingung erfüllt ist:*

$$\Pr_{S\sim\mathcal{D}^m}[S \text{ ist } \varepsilon\text{-repräsentativ}] \geq 1 - \delta \ .$$

Im Folgenden bezeichne $m_{\mathcal{H}}^{UC}$ die Funktion, die der Definition 3.3 genügt und unter dieser Nebenbedingung die kleinstmöglichen Werte $m_{\mathcal{H}}^{UC}(\varepsilon, \delta)$ hat.[7] Es ergibt sich unmittelbar die

Folgerung 3.4 *Wenn $\mathcal{H}$ eine endliche Hypothesenklasse ist, welche die UKB bezüglich einer Verlustfunktion ℓ erfüllt, dann ist $\mathcal{H}$ agnostisch PAC-lernbar bezüglich ℓ (und zwar mit ERM). Darüberhinaus gilt:*

$$m_{\mathcal{H}}(\varepsilon, \delta) \leq m_{\mathcal{H}}^{UC}\left(\frac{\varepsilon}{2}, \delta\right) \ .$$

Wir merken kurz an, dass für beliebige (evtl. unendliche) Klassen $\mathcal{H}$ die Folgerung 3.4 ebenso gültig ist bis auf folgende kleine Abschwächung:

$$m_{\mathcal{H}}(\varepsilon, \delta) \leq m_{\mathcal{H}}^{UC}\left(\frac{\varepsilon}{3}, \delta\right) \ .$$

Diese Behauptung wird evtl. in den Übungen überprüft werden.

[7] UC = Uniform Convergence.

3.2 Agnostische PAC-Lernbarkeit endlicher Hypothesenklassen

Nach den Ausführungen in Abschnitt 3.1 sollte klar sein, dass wir lediglich nachweisen müssen, dass $\mathcal{H}$ die UKB erfüllt. Wir werden in diesem Abschnitt zeigen, dass dies für alle beschränkten Verlustfunktionen der Form $\ell : \mathcal{H} \times Z \to [a, b]$ der Fall ist. Durch Normierung können wir uns natürlich, wann immer uns das besser passt, auf Funktionen der Form $\ell : \mathcal{H} \times Z \to [0, 1]$ zurückziehen.
Um die UKB zu verifizieren, benötigen wir einige Hilfsmittel aus der Statistik:

„Union Bound“: Für Ereignisse $E_1, \ldots, E_s$ gilt:

$$\Pr\left[\bigcup_{i=1}^{s} E_i\right] \leq \sum_{i=1}^{s} \Pr[E_i] \ . \tag{9}$$

Im Deutschen ist diese Regel unter dem etwas sperrigen Namen „Subadditivität von Wahrscheinlichkeitsmaßen“ bekannt. Wir merken kurz an, dass sich die mengentheoretische Vereinigung, sprachlich gesehen, oft hinter einem Existenzquantor versteckt, da das Ereignis $\cup_{i=1}^{s} E_i$ eintritt genau dann, wenn ein $i \in [s]$ existiert mit: das Ereignis E_i ist eingetreten.

Hoeffding's Ungleichung: Es sei $\theta_1, \ldots, \theta_m$ eine Sequenz von unabhängigen, identisch verteilten Zufallsvariablen — im Folgenden kurz iid-Variablen genannt[8] — mit Werten in einem beschränkten Intervall $[a, b]$ $(a < b \in \mathbb{R})$ und mit Erwartungswert μ. Dann gilt für alle $\varepsilon > 0$:

$$\Pr\left[\frac{1}{m}\sum_{i=1}^{m} \theta_i > \mu + \varepsilon\right] \leq \exp\left(\frac{-2m\varepsilon^2}{(b-a)^2}\right) \ . \tag{10}$$

$$\Pr\left[\frac{1}{m}\sum_{i=1}^{m} \theta_i < \mu - \varepsilon\right] \leq \exp\left(\frac{-2m\varepsilon^2}{(b-a)^2}\right) \ . \tag{11}$$

$$\Pr\left[\left|\frac{1}{m}\sum_{i=1}^{m} \theta_i - \mu\right| > \varepsilon\right] \leq 2\exp\left(\frac{-2m\varepsilon^2}{(b-a)^2}\right) \ . \tag{12}$$

Die Hoeffding-Ungleichungen begrenzen die Wahrscheinlichkeit, dass der empirische Mittelwert der iid-Variablen weit vom Erwartungswert abweicht. Die Funktion $\exp\left(\frac{-2m\varepsilon^2}{(b-a)^2}\right)$ konvergiert mit wachsendem m in negativ exponentieller Geschwindigkeit gegen Null. Freilich folgt (12) direkt aus (10) und (11) vermöge der „Union Bound“.

Wir wählen eine Hypothese $h \in \mathcal{H}$ beliebig aber fest aus. Mit einer zufälligen Trainingssequenz $S = (z_1, \ldots, z_m) \sim \mathcal{D}^m$ assoziieren wir die iid-Variablen $\theta_i = \ell(h, z_i)$ für $i = 1, \ldots, m$ und setzen

$$\mu = \mathbb{E}[\theta_i] = \mathbb{E}_{z\sim\mathcal{D}}[\ell(h, z_i)] = L_{\mathcal{D}}(h)$$

[8] iid = independent and identically distributed

Beachte, dass dann

$$\frac{1}{m}\sum_{i=1}^{m}\theta_i = \frac{1}{m}\sum_{i=1}^{m}\ell(h,z_i) = L_S(h) \ .$$

Da wir anfangs des Abschnittes vorausgesetzt hatten, dass die Verlustfunktion ℓ Werte im Intervall $[a,b]$ annimmt, ist dies auch der Wertebereich der Variablen θ_i. Anwendung von (12) liefert:

$$\Pr_{S\sim\mathcal{D}^m}[|L_S(h) - L_{\mathcal{D}}(h)| > \varepsilon] \leq 2\exp\left(\frac{-2m\varepsilon^2}{(b-a)^2}\right) \ . \tag{13}$$

Damit S ε-repräsentativ ist, muss aber das „schlechte Ereignis“ $|L_S(h)-L_{\mathcal{D}}(h)| > \varepsilon$ nicht nur für eine fest ausgewählte Hypothese h vermieden werden, sondern wir müssen das Ereignis

$$\exists h \in \mathcal{H} : |L_S(h) - L_{\mathcal{D}}(h)| > \varepsilon$$

vermeiden. Durch Kombination von (13) mit der „Union Bound“ erhalten wir

$$\Pr_{S\sim\mathcal{D}^m}[\exists h \in \mathcal{H} : |L_S(h) - L_{\mathcal{D}}(h)| > \varepsilon] \leq 2|\mathcal{H}|\exp\left(\frac{-2m\varepsilon^2}{(b-a)^2}\right) \ .$$

Wenn wir den Ausdruck auf der rechten Seite dieser Ungleichung durch δ nach oben begrenzen können, haben wir im Umkehrschluss erreicht, dass S mit einer Wahrscheinlichkeit mindestens $1-\delta$ ε-repräsentativ ist. Wir wünschen uns daher, dass

$$2|\mathcal{H}|\exp\left(\frac{-2m\varepsilon^2}{(b-a)^2}\right) \leq \delta \ . \tag{14}$$

Wenn wir m hinreichend groß wählen, können wir uns diesen Wunsch selber erfüllen. Durch Auflösen von (14) nach m sehen wir, dass

$$m \geq \frac{(b-a)^2\ln(2|\mathcal{H}|/\delta)}{2\varepsilon^2}$$

eine hinreichend große Wahl ist. Um m nicht größer als nötig zu machen, setzen wir

$$m = \left\lceil \frac{(b-a)^2\ln(2|\mathcal{H}|/\delta)}{2\varepsilon^2} \right\rceil \ .$$

Die gesamte Diskussion lässt sich zusammenfassen wie folgt:

Theorem 3.5 *Eine endliche Hypothesenklasse $\mathcal{H}$ erfüllt bezüglich einer Verlustfunktion mit Wertebereich $[a,b]$ die UKB und ist daher bezüglich ℓ agnostisch PAC-lernbar (und zwar mit ERM). Darüberhinaus gilt*

$$m_{\mathcal{H}}^{UC}(\varepsilon,\delta) \leq \left\lceil \frac{(b-a)^2\ln(2|\mathcal{H}|/\delta)}{2\varepsilon^2} \right\rceil$$

und daher auch

$$m_{\mathcal{H}}(\varepsilon,\delta) \leq m_{\mathcal{H}}^{UC}\left(\frac{\varepsilon}{2},\delta\right) \leq \left\lceil \frac{2(b-a)^2\ln(2|\mathcal{H}|/\delta)}{\varepsilon^2} \right\rceil \ .$$

Für den Spezialfall $[a,b]=[0,1]$ ergibt sich

$$m_{\mathcal{H}}^{UC}(\varepsilon,\delta) \leq \left\lceil \frac{\ln(2|\mathcal{H}|/\delta)}{2\varepsilon^2} \right\rceil \quad \textit{und} \quad m_{\mathcal{H}}(\varepsilon,\delta) \leq \left\lceil \frac{2\ln(2|\mathcal{H}|/\delta)}{\varepsilon^2} \right\rceil \ .$$

3.3 Verlustwertzerlegung und „bias-complexity-tradeoff“

Der Verlustwert (= Fehlerrate im Falle von Klassifikationsproblemen) einer ERM-Hypothese h_S lässt sich zerlegen wie folgt:

$$L_{\mathcal{D}}(h_S) = \underbrace{(\min_{h\in\mathcal{H}} L_{\mathcal{D}}(h))}_{=:\varepsilon_{app}} + \underbrace{(L_D(h_S) - \min_{h\in\mathcal{H}} L_{\mathcal{D}}(h))}_{=:\varepsilon_{est}} \quad .$$

Der Term ε_{app} wird *Approximationsfehler* genannt. Er hängt nur von der Hypothesenklasse, nicht aber von der ERM-Hypothese oder der Größe der Trainingssequenz ab. Der Term ε_{est} heißt *Schätzfehler*. Er ist abhängig von der Größe der Trainingssequenz und wächst mit der Informationskomplexität von $\mathcal{H}$. Wenn wir viel Vorwissen in die Wahl von $\mathcal{H}$ legen (strong bias), also $\mathcal{H}$ sehr speziell wählen, dann wird i.A. der Approximationsfehler groß und der Schätzfehler klein sein (low information complexity). Die Hypothesen können sich dann nicht gut an die Daten anpassen — ein Phänomen, was im Englischen „underfitting“ genannt wird. Wählen wir $\mathcal{H}$ hingegen sehr ausdrucksmächtig (weak bias), dann verhält es sich genau umgekehrt (high information complexity). Die Hypothese h_S ist dann in der Gefahr, an die Daten übermäßig angepasst zu sein (etwa „Rauschen“ oder andere Zufälligkeiten in den Daten abzubilden) — ein Phänomen, was im Englischen „overfitting“ genannt wird. Wir können also den Approximationsfehler auf Kosten eines höheren Schätzfehlers verkleinern, und umgekehrt. Im Englischen spricht man von dem „bias-complexity tradeoff“. Die Kunst besteht natürlich darin, eine geeignete Balance zu finden, so dass möglichst weder „under-“ noch „overfitting“ auftritt (good fit with the data).

4 Lernen via uniforme Konvergenz (allgemeiner Fall)

Wir diskutieren in diesem Kapitel die PAC-Lernbarkeit beliebiger (evtl. unendlich großer) binärer Hypothesenklassen und stellen die Frage, ob sie PAC-lernbar sind. Dass unendlich große Klassen grundsätzlich PAC-lernbar sein können, wird in Abschnitt 4.1 anhand eines einfachen Beispiels illustriert. In Abschnitt 4.2 beginnen wir mit allgemeineren Überlegungen und definieren zu diesem Zweck einen Parameter, genannt die VC-Dimension von $\mathcal{H}$ und notiert als $\mathrm{VCD}(\mathcal{H})$. Wir geben zudem ein paar Beispiele zur Berechnung dieses Parameters für konkrete Hypothesenklassen an. In Abschnitt 4.3 weisen wir u.a. nach, dass $m_{\mathcal{H}}(\varepsilon, \delta)$ mindestens linear mit $\mathrm{VCD}(\mathcal{H})$ wachsen muss. Somit sind binäre Hypothesenklassen mit unendlicher VC-Dimension nicht mal unter der Realisierbarkeitsannahme PAC-lernbar. Abschnitt 4.4 ist Sauer's Lemma gewidmet. Dieses Lemma ist ein wichtiger Baustein im Beweis, dass Klassen endlicher VC-Dimension PAC-lernbar sind. In Abschnitt 4.5 behandeln wir obere Schranken von $m_{\mathcal{H}}(\varepsilon, \delta)$, die höchstens linear mit $\mathrm{VCD}(\mathcal{H})$ wachsen (und polynomiell in $1/\varepsilon$ und $\log(1/\delta)$). Somit sind binäre Hypothesenklassen mit endlicher VC-Dimension sogar agnostisch PAC-lernbar. Als Folgerung ergibt sich in Abschnitt 4.6 das sogenannte Fundamentaltheorem der PAC-Lerntheorie, welches die PAC-Lernbarkeit einer binären Hypothesenklasse auf mehrfache Weise charakterisiert. Der zugehörige Beweis erfordert Konzepte

(wie zum Beispiel die Rademacher-Komplexität), die uns noch nicht bekannt sind. Daher werden wir diesen nur sehr grob andeuten.

4.1 Beispiel einer unendlich großen PAC-lernbaren Klasse

Die Indikatorfunktion $x \mapsto \mathbb{1}_{[x \geq a]} \in \{0,1\}$ bildet eine reelle Zahl x genau dann auf 1 ab, falls $x \geq a$. Sie wird auch als *Schwellenfunktion* mit Schwellwert a bezeichnet. Wir lassen auch den Schwellwert $a = \infty$ zu und $\mathbb{1}_{[x \geq \infty]}$ bezeichne die konstante Nullfunktion.

Lemma 4.1 *Die binäre Hypothesenklasse* $\mathcal{H} = \{\mathbb{1}_{[x \geq a]} |\ a \in \mathbb{R} \cup \{\infty\}\}$ *ist unter der Realisierbakeitsannahme PAC-lernbar.*

Beweis Wegen der Realisierbarkeitsannahme dürfen wir davon ausgehen, dass ein (dem Lerner unbekanntes) $a \in \mathbb{R} \cup \{\infty\}$ existiert, so dass $L_{\mathcal{D}}(\mathbb{1}_{[x \geq a]}) = 0$. Es sei $\mathcal{P}$ die Randverteilung von $\mathcal{D}$ auf $\mathbb{R} \cup \{\infty\}$ und $S \sim \mathcal{D}^m$ (so dass $S_{\mathcal{X}} \sim \mathcal{P}^m$) sei die zufällige Trainingssequenz. Es sei A der Lernalgorithmus, der auf S fehlerfrei ist und unter dieser Randbedingung den Schwellwert $\hat{a} \in \mathbb{R} \cup \{\infty\}$ seiner Hypothese so groß wie möglich wählt. Genauer: falls S kein positives Beispiel (also kein Beispiel mit Markierung 1) enthält, dann gilt $\hat{a} = \infty$; andernfalls ist $\hat{a}$ die kleinste reelle Zahl, die mit Markierung 1 in S vorkommt. Im Falle $a = \infty$ würde S kein Beispiel mit Markierung 1 enthalten, A würde $\hat{a} = \infty$ setzen und somit eine fehlerfreie Hypothese abliefern. Wir dürfen im Folgenden also $a \in \mathbb{R}$ annehmen. Aus der oben beschriebenen Wahl von $\hat{a}$ folgt $\hat{a} \geq a$ (was den Fall $\hat{a} = \infty$ einschließt). Somit wäre $[a, \hat{a})$ die Fehlermenge der Hypothese $A(S)$ mit Schwellwert $\hat{a}$. Falls $\mathcal{P}([a, \infty)) \leq \varepsilon$, dann ist $A(S)$ ε-akkurat. Wir dürfen daher $\mathcal{P}([a, \infty)) > \varepsilon$ annehmen. Es sei

$$a' = \inf\{x \in \mathbb{R} |\ x \geq a \text{ und } \mathcal{P}([a, x]) \geq \varepsilon\} \ .$$

Es folgt

$$\mathcal{P}([a, a']) = \lim_{n \to \infty} \mathcal{P}\left(\left[a, a' + \frac{1}{n}\right]\right) \geq \varepsilon$$

und, wegen der Minimalität von a', gilt weiterhin

$$\mathcal{P}([a, a')) \leq \varepsilon \quad .$$

Wenn S ein (positives) Beispiel aus dem Intervall $[a, a']$ enthält, dann ergibt sich $\hat{a} \leq a'$. Somit ist die Fehlermenge von $A(S)$ eine Teilmenge von $[a, a')$ und hat eine Gesamtwahrscheinlichkeit von maximal ε. Zur Vervollständigung des Beweises genügt es daher zu zeigen, dass (bei geigneter Wahl von m) die Wahrscheinlichkeit für $S_{\mathcal{X}} \cap [a, a'] = \emptyset$ durch δ nach oben beschränkt werden kann. Offensichtlich gilt:

$$\Pr_{S_{\mathcal{X}} \sim \mathcal{P}^m}[S_{\mathcal{X}} \cap [a, a'] = \emptyset] \leq (1 - \varepsilon)^m < e^{-\varepsilon m} \ .$$

Wir machen den Ansatz $e^{-\varepsilon m} \leq \delta$. Auflösen nach m liefert

$$m \geq \frac{1}{\varepsilon} \ln\left(\frac{1}{\delta}\right) \ .$$

Eine Trainigssequenz der Größe

$$m = \left\lceil \frac{1}{\varepsilon} \ln\left(\frac{1}{\delta}\right) \right\rceil$$

ist also hinreichend groß, um das (ε, δ)-Kriterium des PAC-Lernens zu gewährleisten. •

4.2 Definition der VC-Dimension und Beispiele

Es bezeichne wieder $\mathcal{H}$ eine binäre Hypothesenklasse über dem Grundbereich $\mathcal{X}$ und für $M \subseteq \mathcal{X}$ bezeichne $\mathcal{H}_M$ die Menge der auf $M \subseteq \mathcal{X}$ eingeschränkten Abbildungen aus $\mathcal{H}$.

Definition 4.2 (aufspaltbare Mengen) *$M \subseteq \mathcal{X}$ heißt* aufspaltbar *durch $\mathcal{H}$, wenn $\mathcal{H}_M$ die Menge aller Abbildungen von M nach $\{0,1\}$ ist.*

Intuitiv bedeutet das, dass jedes „Bitmuster“ auf M durch eine Abbildung in $\mathcal{H}$ realisiert werden kann.

Definition 4.3 (VC-Dimension) *Die* VC-Dimension *von $\mathcal{H}$, notiert als* $\mathrm{VCD}(\mathcal{H})$*, ist definiert als die Anzahl der Elemente einer größten durch $\mathcal{H}$ aufspaltbaren Teilmenge von $\mathcal{X}$ (bzw. als ∞, falls beliebig große durch $\mathcal{H}$ aufspaltbare Teilmengen von $\mathcal{X}$ existieren). Formal:*

$$\mathrm{VCD}(\mathcal{H}) = \sup\{|M| : \ M \text{ ist durch } \mathcal{H} \text{ aufspaltbar}\} \ .$$

Der Nachweis von $\mathrm{VCD}(\mathcal{H}) = d$ kann nach folgendem Schema geschehen:

1. Wir geben eine Menge $M \subseteq \mathcal{X}$ der Größe d an, auf der sich alle 2^d Bitmuster von $\mathcal{H}$ realisieren lassen (was $\mathrm{VCD}(\mathcal{H}) \geq d$ beweist).

2. Wir weisen nach, dass für jede Menge $M \subseteq \mathcal{X}$, der Größe $d+1$ ein Bitmuster $b \in \{0,1\}^{d+1}$ existiert, das von $\mathcal{H}$ nicht realisiert werden kann (was $\mathrm{VCD}(\mathcal{H}) \leq d$ beweist).

Wir betrachten ein paar konkrete Beispielklassen. Beachte dabei, dass eine Familie von Teilmengen von $\mathcal{X}$ auch als binäre Hypothesenklasse mit Grundbereich $\mathcal{X}$ aufgefasst werden kann, indem wir $H \subseteq \mathcal{X}$ mit der Indikatorfunktion $\mathbb{1}_{[x \in H]}$ identifizieren.

Schwellenfunktionen Die Klasse der Schwellenfunktionen hat VC-Dimension 1.
Offensichtlich wird die Menge $\{0\}$ von dieser Klasse aufgespalten. Daher ist die VC-Dimension mindestens 1. Seien $x_1 < x_2$ beliebige reelle Zahlen. Dann kann das Bitmuster $(1,0)$ nicht von einer Funktion vom Typ $\mathbb{1}_{[x \geq a]}$ realisiert werden. Daher kann die VC-Dimension den Wert 1 nicht überschreiten.

Intervalle: Die Klasse der Intervalle der Form $[a,b] \subset \mathbb{R}$ hat VC-Dimension 2.
Offensichtlich wird die Menge $\{-1,1\}$ (wie jede andere 2-elementige Menge) von Intervallen aufgespalten. Daher ist die VC-Dimension mindestens 2. Seien $x_1 < x_2 < x_3$ beliebige reelle Zahlen. Dann kann das Bitmuster $(1,0,1)$ nicht von einem Intervall realisiert werden. Daher kann die VC-Dimension den Wert 2 nicht überschreiten.

Achsenparallele Rechtecke: Die Klasse der (topologisch abgeschlossenen) achsenparallelen Rechtecke in $\mathbb{R}^2$ hat VC-Dimension 4.
Offensichtlich wird die Punktmenge $\{(-1,0),(0,1),(1,0),(0,-1)\}$ durch achsenparallele Rechtecke aufgespalten. Daher ist die VC-Dimension mindestens 4. Seien $z_1,\ldots,z_5$ beliebige fünf verschiedene Punkte in $\mathbb{R}^2$. Wir wählen für jede Himmelsrichtung (Westen, Norden, Osten, Süden) jeweils einen der fünf Punkte mit einer extremalen Lage in dieser Richtung aus (zum Beispiel einen nördlichsten Punkt). Ggf. nach Umnummerierung seien z_1, z_2, z_3, z_4 die vier ausgewählten Punkte. Es sei R das kleinste achsenparallele Rechteck, das die vier ausgewählten Punkte enthält. Man überlegt sich leicht, dass z_5 in R liegen muss. Daher ist das Bitmuster $(1,1,1,1,0)$ von keinem achsenparallelen Rechteck realisierbar und die VC-Dimension kann den Wert 4 nicht überschreiten.

Konvexe Polygone: Die Klasse der konvexen Polygone mit k Ecken hat VC-Dimension $2k+1$. Der Nachweis hierfür wird evtl. in den Übungen geführt.

Singleton-Mengen: Die Klasse $\mathcal{H}$ der 1-elementigen Teilmengen einer Grundmenge $\mathcal{X}$ mit $|\mathcal{X}| \geq 2$ hat VC-Dimension 1.
Offensichtlich wird jede einelementige Teilmenge von $\mathcal{X}$ durch $\mathcal{H}$ aufgespalten. Somit gilt $\mathrm{VCD}(\mathcal{H}) \geq 1$. Auf zwei beliebigen Punkten $x_1 \neq x_2 \in \mathcal{X}$ kann das Bitmuster $(1,1)$ von $\mathcal{H}$ nicht realisiert werden. Somit gilt $\mathrm{VCD}(\mathcal{H}) \leq 1$.

Potenzmengen: Die Klasse $P(\mathcal{X})$ aller Teilmengen einer d-elementigen Grundmenge $\mathcal{X}$ hat VC-Dimension d.
Freilich kann jedes Bitmuster auf $\mathcal{X}$ von $P(\mathcal{X})$ realisiert werden. Daher ist die VC-Dimension mindestens d. Da $\mathcal{X}$ keine Teilmenge mit $d+1$ Elementen hat, kann die VC-Dimension den Wert d nicht überschreiten.

Wir machen ein paar abschließende Bemerkungen:

- Es werden mindestens 2^d verschiedene Hypothesen in $\mathcal{H}$ benötigt, um alle 2^d Bitmuster über einer d-elementigen Teilmenge von $\mathcal{X}$ zu realisieren. Mit $d = \mathrm{VCD}(\mathcal{H})$ ergibt sich hieraus $|\mathcal{H}| \geq 2^d$ oder, anders ausgedrückt, $\mathrm{VCD}(\mathcal{H}) \leq \log(|\mathcal{H}|)$. Diese obere Schranke für $\mathrm{VCD}(\mathcal{H})$ ist gelegentlich scharf (wie zum Beipiel bei der Potenzmenge). Andererseits gibt es Klassen beliebiger (evtl. sogar unendlicher) Größe, deren VC-Dimension lediglich 1 beträgt (was wir am Beispiel der Singleton-Mengen sehen). Die Lücke zwischen $\mathrm{VCD}(\mathcal{H})$ und der oberen Schranke $\log(|\mathcal{H}|)$ kann also beliebig groß werden.

- Man könnte anhand vieler konkreter Klassen $\mathcal{H}$ den Eindruck gewinnen, dass die VC-Dimension von $\mathcal{H}$ übereinstimmt mit der Anzahl der Parameter, die zum Spezifizieren der Elemente von $\mathcal{H}$ benötigt werden (ein Schwellwert zur Spezifikation einer Schwellenfunktion, zwei Randpunkte eines Intervalls, vier Ecken eines achsenparallelen Rechtecks, usw.). In den Übungen werden wir sehen, dass es aber Klassen unendlicher VC-Dimension gibt, deren Elemente mit lediglich einem reellen Parameter spezifizierbar sind.

4.3 Untere Schranken zur Informationskomplexität

In diesem Abschnitt sei $\mathcal{H}$ eine binäre Hypothesenklasse der VC-Dimension d über einem Grundbereich $\mathcal{X}$. Wir betrachten PAC-Lernen von $\mathcal{H}$ unter der Realisierbarkeitsannahme. Unser Ziel ist es, die folgende untere Schranke

$$m_{\mathcal{H}}(\varepsilon, \delta) \geq \Omega\left(\frac{1}{\varepsilon}\left(d + \ln\left(\frac{1}{\delta}\right)\right)\right) \tag{15}$$

für das asymptotische Wachstum von $m_{\mathcal{H}}(\varepsilon, \delta)$ in den Parametern d, ε, δ nachzuweisen.

Untere Schranken für $m_{\mathcal{H}}(\varepsilon, \delta)$ gelten natürlich erst recht für das PAC-Lernen, wenn sie sogar für abgeschwächte Erfolgskriterien gelten, welche das Leben für den Lerner tendenziell leichter machen. Dies ist insbesondere dann von Interesse, wenn die Analyse sich unter den veränderten Modellannahmen leichter und übersichtlicher gestalten lässt. In diesem Abschnitt machen wir von dieser Technik wie folgt Gebrauch:

1. Wir fixieren Wahrscheinlichkeitsverteilungen $\mathcal{P}$ auf $\mathcal{X}$ und $\mathcal{Q}$ auf $\mathcal{H}$.
2. Wir generieren eine zunächst unmarkierte Trainingssequenz $M = S|_{\mathcal{X}} \sim \mathcal{P}^m$.
3. Wegen der Realisierbarkeitsannahme dürfen wir uns vorstellen, dass die Labels zu M gemäß einer (dem Lerner unbekannten) Funktion $h \in \mathcal{H}$ erstellt werden. Für diesen Job benutzen wir eine zufällige Hypothese $h \sim \mathcal{Q}$. Die resultierende markierte Trainingssequenz nennen wir S. Im Folgenden bezeichne $\mathcal{D} = \mathcal{D}_{\mathcal{P},h}$ die von $\mathcal{P}$ und h induzierte Verteilung auf $\mathcal{X} \times \{0,1\}$ (so dass $\mathcal{P}$ mit der Randverteilung $\mathcal{D}_{\mathcal{X}}$ übereinstimmt und die Labels von h produziert werden).
4. Der Lerner erhält S und muss seine Hypothese h_S abliefern. Das Erfolgskriterium für h_S lautet
$$\Pr_{S_{\mathcal{X}} \sim \mathcal{P}^m, h \sim \mathcal{Q}}[L_{\mathcal{D}}(h_S) \leq \varepsilon] \geq 1 - \delta \ . \tag{16}$$

Wir demonstrieren nun, dass das Kriterium (16) leichter (oder zumindest nicht schwerer) zu erreichen ist, als das Erfolgskriterium

$$\forall h \in \mathcal{H} : \Pr_{S \sim \mathcal{D}^m}[L_{\mathcal{D}}(h_S) \leq \varepsilon] \geq 1 - \delta \tag{17}$$

im PAC-Lernmodell.

Lemma 4.4 *Ein Algorithmus A, der $\mathcal{H}$ unter der Realisierbarkeitsannahme PAC-lernt erfüllt das Erfolgskriterium (16).*

Beweis Die Implikation (17)⇒(16) ergibt sich aus folgender Rechnung:

$$\begin{aligned}
\Pr_{S|_{\mathcal{X}} \sim \mathcal{P}^m, h \sim \mathcal{Q}}[L_{\mathcal{D}}(h_S) \leq \varepsilon] &= \mathbb{E}_{S|_{\mathcal{X}} \sim \mathcal{P}^m, h \sim \mathcal{Q}}[\mathbb{1}_{[L_{\mathcal{D}}(h_S) \leq \varepsilon]}] \\
&\geq \inf_{h \in \mathcal{H}} \mathbb{E}_{S \sim \mathcal{D}^m}[\mathbb{1}_{[L_{\mathcal{D}}(h_S) \leq \varepsilon]}] \\
&= \inf_{h \in \mathcal{H}} \Pr_{S \sim \mathcal{D}^m}[L_{\mathcal{D}}(h_S) \leq \varepsilon] \ .
\end{aligned}$$

•

Lemma 4.4 liefert die Rechtfertigung dafür, zum Nachweis unterer Schranken für $m_{\mathcal{H}}(\varepsilon, \delta)$ mit der oben vorgestellten Variante des Lernmodells zu arbeiten.

Definition 4.5 *$\mathcal{H}$ heißt* nicht-trivial*, wenn $h_1, h_2 \in \mathcal{H}$ und $x_1, x_2 \in \mathcal{X}$ existieren, so dass*

$$h_1(x_1) = h_2(x_1) \quad \textit{und } h_1(x_2) \neq h_2(x_2) \ . \tag{18}$$

Wir haben schon des Öfteren die Ungleichung $1 + a \leq e^a$ (mit Gleichheit nur für $a = 0$) verwendet. Es ist nicht schwer zu zeigen, dass umgekehrt auch Folgendes richtig ist:

$$\forall 0 \leq a \leq \frac{1}{2} : 1 - a \geq e^{-2a} \ . \tag{19}$$

Das folgende Resultat liefert eine erste untere Schranke für $m_{\mathcal{H}}(\varepsilon, \delta)$, die für nicht-triviale Hypothesenklassen gültig ist:

Theorem 4.6 *Für alle nicht-trivialen binären Hypothesenklassen $\mathcal{H}$ und alle $0 < \varepsilon \leq 1/4$, $0 < \delta < 1$ gilt (sogar unter der Realisierbarkeitsannahme):*

$$m_{\mathcal{H}}(\varepsilon, \delta) > \frac{1}{4\varepsilon} \ln\left(\frac{1}{4\delta}\right)$$

Beweis Wähle $x_1, x_2 \in \mathcal{X}$ und $h_1, h_2 \in \mathcal{H}$ so aus, dass die Bedingung (18) erfüllt ist. Wähle $\mathcal{P}$ bzw. $\mathcal{Q}$ so, dass die gesamte Wahrscheinlichkeitsmasse auf x_1, x_2 bzw. h_1, h_2 konzentriert ist, wobei $\mathcal{P}(x_1) = 1 - 2\varepsilon$, $\mathcal{P}(x_2) = 2\varepsilon$ und $\mathcal{Q}(h_1) = \mathcal{Q}(h_2) = 1/2$. Die von $\mathcal{P}$ und der fehlerfreien Labelfunktion h induzierte Verteilung auf $\{x_1, x_2\} \times \{0, 1\}$ sei wieder mit $\mathcal{D}$ bezeichnet. Es sei B_1 das Ereignis, dass $S|_{\mathcal{X}} \sim \mathcal{P}^m$ aus m Duplikaten von x_1 besteht. Wegen $h_1(x_1) = h_2(x_1)$ unterscheidet die resultierende Trainingssequenz S dann nicht zwischen den möglichen Labelfunktionen h_1 und h_2. Wegen $h \sim \mathcal{Q}$ und der Wahl von $\mathcal{Q}$ ist $h(x_2)$ aus Sicht des Lerners ein Zufallsbit. Es sei B_2 das Ereignis $h_S(x_2) \neq h(x_2)$. Wenn B_2 eintritt, dann gilt $L_{\mathcal{D}}(h_S) = 2\varepsilon$. Wir wollen im Folgenden zeigen, dass

$$m \leq \frac{1}{4\varepsilon} \ln\left(\frac{1}{4\delta}\right)$$

eine zu kleine Wahl für die Größe der Trainingssequenz ist, Mit dieser Wahl von m ist die Wahrscheinlichkeit von B_1 gleich $(1 - 2\varepsilon)^m \geq e^{-4\varepsilon m} \geq 4\delta$. Die bedingte Wahrscheinlichkeit von B_2 zu gegebenem B_1 beträgt $1/2$. Also treten mit einer Wahrscheinlichkeit von mindestens 2δ beide Ereignisse (insbesondere B_2) ein. Es gilt dann

$$\Pr_{S|_{\mathcal{X}} \sim \mathcal{P}^m, h \sim \mathcal{Q}}[L_{\mathcal{D}}(h_S)] \geq 2\varepsilon] \geq 2\delta$$

und das Erfolgskriterium (16) wird verfehlt. •

Das folgende Resultat zeigt, dass $m_{\mathcal{H}}(\varepsilon, \delta)$ (sogar unter der Realisierbarkeitsannahme) mindestens linear mit $d = \mathrm{VCD}(\mathcal{H})$ wachsen muss.

Theorem 4.7 *Für alle binären Hypothesenklassen und alle* $0 < \varepsilon \leq 1/8$, $0 < \delta < 1/4$ *gilt (sogar unter der Realisierbarkeitsannahme)*

$$m_{\mathcal{H}}(\varepsilon, \delta) > \frac{d-1}{32\varepsilon} \ .$$

Beweis Es sei $t = d-1$ und $\mathcal{X}_0 = \{x_0, x_1, \ldots, x_t\} \subseteq \mathcal{X}$ sei eine von $\mathcal{H}$ aufspaltbare Menge.[9] Weiter sei $\mathcal{X}_0' = \mathcal{X}_0 \backslash \{x_0\}$. Die Verteilung $\mathcal{P}$ konzentriere ihre ganze Wahrscheinlichkeitsmasse auf $\mathcal{X}_0$ und es gelte

$$\mathcal{P}(x_0) = 1 - 8\varepsilon \ \text{ und } \ \mathcal{P}(x_1) = \ldots = \mathcal{P}(x_t) = \frac{8\varepsilon}{t} \ .$$

Es bezeichne $\mathcal{D}$ wieder die von $\mathcal{P}$ und der fehlerfreien Labelfunktion h induzierte Verteilung auf $\mathcal{X}_0 \times \{0,1\}$. Wähle eine Menge $\mathcal{H}_0$ aus 2^t Hypothesen aus, welche die 2^t Bitmuster $b = (b_0, b_1, \ldots, b_t)$ mit $b_0 = 0$ realisieren. Die Verteilung $\mathcal{Q}$ konzentriere ihre ganze Wahrscheinlichkeitsmasse auf $\mathcal{H}_0$ und sei dort uniform. Im Folgenden identifizieren wir die Hypothesen aus $\mathcal{H}_0$ mit den von ihnen realisierten Bitmustern b. Beachte, dass eine zufällige Wahl von $h \sim \mathcal{Q}$ äquivalent dazu ist, dass das von h realisierte Bitmuster b (abgesehen von $b_0 = 0$) aus t unabhängigen perfekten Zufallsbits besteht. Betrachte nun eine Menge $M = S|_{\mathcal{X}} \sim \mathcal{P}^m$ von unmarkierten Trainingspunkten. Es sei X die Zufallsvariable, welche zählt wieviele Treffer M in $\mathcal{X}_0'$ landet. Es sei B_1 das Ereignis $X \leq t/2$. Aus Sicht des Lerners sind die Labels der nicht in M enthaltenen Instanzen aus $\mathcal{X}_0'$ unabhängige perfekte Zufallsbits. Es sei B_2 das Ereignis, dass h_S auf $\mathcal{X}_0'$ mindestens $t/4$ Fehler macht. Wenn das Ereignis B_2 eintritt, dann gilt offensichtlich $L_{\mathcal{D}}(h_S) \geq 2\varepsilon$ (da $t/4$ Punkte aus $\mathcal{X}_0'$ eine Wahrscheinlichkeitsmasse von insgesamt 2ε besitzen). Wir bestimmen abschließend eine untere Schranke für die Wahrscheinlichkeit von $B_1 \wedge B_2$. Unter der Voraussetzung

$$m \leq \frac{t}{32\varepsilon}$$

gilt $\mathbb{E}[X] \leq t/4$. Aus der Markov Ungleichung folgt, dass die Wahrscheinlichkeit von $X > t/2$ (das Ereignis $\neg B_1$) höchstens $1/2$ beträgt. Die Wahrscheinlichkeit von B_1 ist daher mindestens $1/2$. Die bedingte Wahrscheinlichkeit von B_2 gegeben B_1 beträgt mindestens $1/2$.[10] Es folgt, dass beide Ereignisse (insbesondere B_2) mit einer Wahrscheinlichkeit von mindestens $1/4$ auftreten. Es gilt daher

$$\Pr_{S|_{\mathcal{X}} \sim \mathcal{P}^m, h \sim \mathcal{Q}}[L_{\mathcal{D}}(h_S) \geq 2\varepsilon] \geq \frac{1}{4}$$

und das Erfolgskriterium (16) wird verfehlt. •

Aus Theorem 4.7 können wir unmittelbar die Nichtlernbarkeit von Klassen unendlicher VC-Dimension ableiten:

[9] Der folgende Beweis unterstellt implizit, dass t eine durch 4 teilbare Zahl ist. Er lässt sich aber so modifizieren, dass er auch für andere Werte von t funktioniert.

[10] Genau hier verwenden wir die Teilbarkeit von t durch 4.

Folgerung 4.8 *Eine binäre Hypothesenklasse unendlicher Dimension ist nicht PAC-lernbar, und zwar nicht einmal, wenn wir die Realisierbarkeitsannahme machen und zudem ε konstant auf $1/8$ und δ konstant auf $1/5$ setzen.*

4.4 Sauer's Lemma und die Kapazitätsfunktion

Eine zentrale Definition in der Lerntheorie ist die folgende:

Definition 4.9 (Kapazitätsfunktion) *Die* Kapazitätsfunktion $\tau_{\mathcal{H}}(m)$ *liefert die maximale Anzahl von Bitmustern, die von $\mathcal{H}$ auf einer m-elementigen Teilmenge von $\mathcal{X}$ realisiert werden können. Formal:*

$$\tau_{\mathcal{H}}(m) = \sup\{|\mathcal{H}_M| : M \subseteq \mathcal{X} \wedge |M| = m\} \ .$$

Trivialerweise gilt $\tau_{\mathcal{H}}(m) \leq 2^m$. Dieser Maximalwert wird genau dann erreicht, wenn m die VC-Dimension von $\mathcal{H}$ nicht überschreitet. Eine exponentiell mit m wachsende Zahl von durch $\mathcal{H}$ realisierbaren Bitmustern der Länge m, würde PAC-Lernbarkeit von $\mathcal{H}$ ausschließen. Glücklicherweise — und das wird das Hauptresultat dieses Abschnittes — wächst $\tau_{\mathcal{H}}(m)$ nur polynomiell für Hypothesenklassen $\mathcal{H}$ mit endlicher VC-Dimension d. Genauer: $\tau_{\mathcal{H}}(m) = O(m^d)$, wie wir im Folgenden noch zeigen werden.

Lemma 4.10 *Es sei $\mathcal{H}$ eine binäre Hypothesenklasse über einem Grundbereich M der Größe $m \geq 1$. Dann ist $|\mathcal{H}_M|$ nach oben beschränkt durch die Anzahl der von $\mathcal{H}$ aufspaltbaren Teilmengen von M.*

Beweis Der Beweis erfolgt mit vollständiger Induktion. Für $m = 1$ ist die Aussage offensichtlich. Sei nun $m \geq 2$, x ein beliebiges aus M ausgewähltes Element und $M' = M \setminus \{x\}$. Wir definieren folgende Hypothesenklassen $\mathcal{H}_b$, $b = 0, 1$, über dem Grundbereich M':

$$\mathcal{H}_b = \{h : M' \to \{0,1\} |\ h \in \mathcal{H} \wedge h(x) = b\} \ .$$

Es gilt $|\mathcal{H}| = |\mathcal{H}_0| + |\mathcal{H}_1|$, da $\mathcal{H}$ disjunkt in $\mathcal{H}_0$ und $\mathcal{H}_1$ zerlegt werden kann. Es bezeichne a die Anzahl der von $\mathcal{H}$ aufspaltbaren Teilmengen von M und a_b sei die entsprechende Zahl für $\mathcal{H}_b$. Mit der Induktionsvoraussetzung folgt $|\mathcal{H}_b| \leq a_b$ für $b = 0, 1$. Es gilt $a \geq a_0 + a_1$, und zwar aus folgenden Gründen:

- Jede Menge, die von $\mathcal{H}_0$ oder $\mathcal{H}_1$ aufspaltbar ist, ist auch von $\mathcal{H}$ aufspaltbar.
- Der Term $a_0 + a_1$ zählt zwar jede Mengen $A \subseteq M'$ doppelt, die sowohl von $\mathcal{H}_0$ als auch von $\mathcal{H}_1$ aufspaltbar ist, aber dann ist nicht nur A von $\mathcal{H}$ aufspaltbar sondern auch $A \cup \{x\}$.

Insgesamt hat sich $|\mathcal{H}| = |\mathcal{H}_0| + |\mathcal{H}_1| \leq a_0 + a_1 \leq a$ ergeben, was den induktiven Beweis abschließt. •

Die folgende Funktion liefert (wie wir gleich zeigen werden) eine obere Schranke für die Werte der Kapazitätsfunktion zu einer Hypothesenklasse der VC-Dimension d:

$$\Phi_d(m) = \sum_{i=0}^{d} \binom{m}{d} \leq \left(\frac{em}{d}\right)^d \ .$$

Lemma 4.11 (Sauer's Lemma) *Es sei $\mathcal{H}$ eine binäre Hypothesenklasse der VC-Dimension d. Dann gilt $\tau_{\mathcal{H}}(m) \leq \Phi_d(m)$.*

Beweis Es genügt $|\mathcal{H}_M| \leq \Phi_d(m)$ für eine beliebige Menge $M \subseteq \mathcal{X}$ mit $|M| = m$ nachzuweisen. Gemäß Lemma 4.10 ist $|\mathcal{H}_M|$ nach oben beschränkt durch die Anzahl der von $\mathcal{H}$ aufspaltbaren Teilmengen von M. Wegen $d = \mathrm{VCD}(\mathcal{H})$ hat keine dieser aufspaltbaren Teilmengen mehr als d Elemente. Da $\Phi_d(m)$ mit der Anzahl aller Teilmengen von M mit höchstens d Elementen übereinstimmt, folgt $\tau_{\mathcal{H}}(m) \leq \Phi_d(m)$. •

Die Schranke in Sauer's Lemma ist gelegentlich scharf, zum Beispiel für die Klasse aller Teilmengen von $\mathbb{N}$ mit höchstens d Elementen. Dies ist eine Klasse der VC-Dimension d, deren Kapazitätsfunktion mit $\Phi_d(m)$ übereinstimmt.

4.5 Obere Schranke zur Informationskomplexität

In diesem Abschnitt bezeichnen wir mit Q die Menge der für $\mathcal{H}$ nicht ε-repräsentativen Trainingssequenzen der Größe m. Formal:

$$Q = \{\mathbf{z} \in \mathcal{Z}^m | \ \exists h \in \mathcal{H} : |L_{\mathbf{z}}(h) - L_{\mathcal{D}}(h)| > \varepsilon\} \ .$$

Wir wollen in diesem Abschnitt nachweisen, dass

$$\mathcal{D}^m(Q) \leq 4\tau_{\mathcal{H}}(2m) \exp\left(-\frac{\varepsilon^2 m}{8}\right) \ , \tag{20}$$

vorausgesetzt dass $m \geq 2\ln(4)/\varepsilon^2$. Der Nachweis ist nichttrivial und bedarf einiger begrifflicher Vorbereitungsarbeiten.

Eine Trainingssequenz $\mathbf{z} \in \mathcal{Z}^{2m}$ der Größe $2m$ hat die Form $\mathbf{z} = (z_1, \ldots, z_{2m})$ mit $z_i = (x_i, y_i) \in \mathcal{Z} = \mathcal{X} \times \{0, 1\}$. Wir zerlegen $\mathbf{z}$ in Gedanken in zwei Trainingssequenzen $\mathbf{z}', \mathbf{z}''$ der Größe m, so dass $\mathbf{z} = (\mathbf{z}', \mathbf{z}'')$ mit $\mathbf{z}' = (z_1, \ldots, z_m)$ und $\mathbf{z}'' = (z_{m+1}, \ldots, z_{2m})$. Es wird sich als zweckmäßig herausstellen, die Menge Q in Beziehung zu setzen zu folgender Menge:

$$\begin{aligned} R &= \{\mathbf{z} \in \mathcal{Z}^m \times \mathcal{Z}^m | \ \exists h \in \mathcal{H} : |L_{\mathbf{z}'}(h) - L_{\mathbf{z}''}(h)| \geq \varepsilon/2\} \\ &= \left\{\mathbf{z} \in \mathcal{Z}^{2m} | \ \exists h \in \mathcal{H} : \frac{1}{m} \left| \sum_{i=1}^{m} \mathbb{1}_{[h(x_i) \neq y_i]} - \mathbb{1}_{[h(x_{m+i}) \neq y_{m+i}]} \right| \geq \varepsilon/2 \right\} \ . \end{aligned}$$

Die Definitionsbedingung der Menge R in Worten lautet: es gibt eine Hypothese in $\mathcal{H}$, bei welcher die empirischen Fehlerraten bezüglich $\mathbf{z}'$ und $\mathbf{z}''$ um mindestens $\varepsilon/2$ von einander abweichen. Für $b \in \{\pm 1\}^m$ definieren wir folgende zu R verwandten Ereignisse:

$$\begin{aligned} R_b &= \left\{ \mathbf{z} \in \mathcal{Z}^{2m} |\ \exists h \in \mathcal{H} : \frac{1}{m} \left| \sum_{i=1}^{m} b_i \cdot \left(\mathbb{1}_{[h(x_i) \neq y_i]} - \mathbb{1}_{[h(x_{m+i}) \neq y_{m+i}]} \right) \right| \geq \varepsilon/2 \right\} \\ R_{b,h} &= \left\{ \mathbf{z} \in \mathcal{Z}^{2m} |\ \frac{1}{m} \left| \sum_{i=1}^{m} b_i \cdot \left(\mathbb{1}_{[h(x_i) \neq y_i]} - \mathbb{1}_{[h(x_{m+i}) \neq y_{m+i}]} \right) \right| \geq \varepsilon/2 \right\} \ . \end{aligned}$$

Der Nachweis der oberen Schranke (20) benutzt folgende (noch zu verifizierende) Zwischenschritte:

Behauptung 1: $\mathcal{D}^m[Q] \leq 2\mathcal{D}^{2m}(R)$, sofern $m \geq 2\ln(4)/\varepsilon^2$.

Behauptung 2: Für alle $b \in \{\pm 1\}^m$ gilt: $D^{2m}(R) = D^{2m}(R_b)$.

Behauptung 3: Es bezeichne U die uniforme Verteilung auf $\{\pm 1\}^m$. Dann gilt:

$$\mathcal{D}^{2m}(R) \leq \sup_{\mathbf{z} \in \mathcal{Z}^{2m}} \Pr_{b \sim U}[\mathbf{z} \in R_b] \ .$$

Behauptung 4: Für jede feste Wahl von $\mathbf{z} \in \mathcal{Z}^{2m}$ gilt:

$$\Pr_{b \sim U}[\mathbf{z} \in R_b] \leq \tau_{\mathcal{H}}(2m) \sup_{h \in \mathcal{H}} \Pr_{b \sim U}[\mathbf{z} \in R_{b,h}] \ .$$

Behauptung 5: Für jede feste Wahl von $\mathbf{z} \in \mathcal{Z}^{2m}$ und $h \in \mathcal{H}$ gilt:

$$\Pr_{b \sim U}[\mathbf{z} \in R_{b,h}] \leq 2\exp\left(-\frac{\varepsilon^2 m}{8} \right) \ .$$

Aus diesen fünf Behauptungen ist (20) direkt ersichtlich. Es folgen die Nachweise, dass die aufgestellten Behauptungen korrekt sind.

Beweis von Behauptung 1: Für jedes $\mathbf{z} \in Q$ wählen wir eine Funktion $h_{\mathbf{z}} \in \mathcal{H}$ aus mit $|L_{\mathbf{z}}(h_{\mathbf{z}}) - L_{\mathcal{D}}(h_{\mathbf{z}})| > \varepsilon$. Wir starten mit folgender Rechnung:

$$\begin{aligned} \mathcal{D}^{2m}(R) &= \Pr_{\mathbf{z} \sim \mathcal{D}^{2m}}[\mathbf{z} \in R] \\ &\geq \Pr_{\mathbf{z} \sim \mathcal{D}^{2m}}[\mathbf{z} \in R \wedge \mathbf{z}' \in Q] \\ &= \Pr_{\mathbf{z} \sim \mathcal{D}^{2m}}[(\mathbf{z}', \mathbf{z}'') \in R |\ \mathbf{z}' \in Q] \cdot \Pr_{\mathbf{z}' \sim \mathcal{D}^m}[\mathbf{z}' \in Q] \\ &= \Pr_{\mathbf{z} \sim \mathcal{D}^{2m}}[(\mathbf{z}', \mathbf{z}'') \in R |\ \mathbf{z}' \in Q] \cdot \mathcal{D}^m(Q) \ . \end{aligned}$$

Zum Nachweis von Behauptung 1 ist noch $\Pr_{\mathbf{z}\sim\mathcal{D}^{2m}}[(\mathbf{z}',\mathbf{z}'')\in R|\ \mathbf{z}'\in Q]\geq 1/2$ zu zeigen. Wir stellen folgende Überlegung an. Gegeben dass $\mathbf{z}'\in Q$, also dass $L_{\mathbf{z}'}(h)$ um mehr als ε von $L_{\mathcal{D}}(h)$ abweicht, dann kann das Ereignis $(\mathbf{z}',\mathbf{z}'')\notin R$ nur dann eintreten, wenn für alle Hypothesen $h\in\mathcal{H}$ die Bedingung $|L_{\mathbf{z}'}(h)-L_{\mathbf{z}''}(h)|<\varepsilon/2$ erfüllt ist. Insbesondere muss dies für $h_{\mathbf{z}'}$ gelten. Wegen $|L_{\mathbf{z}'}(h_{\mathbf{z}'})-L_{\mathcal{D}}(h_{\mathbf{z}'})|>\varepsilon$ folgt dann zwangsläufig, dass $|L_{\mathbf{z}''}(h_{\mathbf{z}'})-L_{\mathcal{D}}(h_{\mathbf{z}'})|>\varepsilon/2$. Gemäß der Hoeffding-Schranke beträgt die Wahrscheinlichkeit für letzteres Ereignis höchstens $2\exp(-2m(\varepsilon/2)^2)=2\exp(-m\varepsilon^2/2)\leq 1/2$, wobei die letzte Ungleichung wegen $m\geq 2\ln(4)/\varepsilon^2$ gültig ist. Aus dieser Diskussion folgt, $\Pr_{\mathbf{z}\sim\mathcal{D}^{2m}}[(\mathbf{z}',\mathbf{z}'')\notin R|\ \mathbf{z}'\in Q]\leq 1/2$ und somit $\Pr_{\mathbf{z}\sim\mathcal{D}^{2m}}[(\mathbf{z}',\mathbf{z}'')\in R|\ \mathbf{z}'\in Q]\geq 1/2$. Damit ist der Beweis von Behauptung 1 komplett.

Beweis von Behauptung 2: Für jede Permutation σ von $1,\ldots,2m$ sind die Zufallsvariablen $\mathbf{z}=(z_1,\ldots,z_{2m})\sim\mathcal{D}^{2m}$ und $\mathbf{z}^\sigma=(z_{\sigma(1)},\ldots,z_{\sigma(2m)})$ mit $\mathbf{z}\sim\mathcal{D}^{2m}$ gleichverteilt. Daher gilt $\Pr_{\mathbf{z}\sim\mathcal{D}^{2m}}[\mathbf{z}\in R]=\Pr_{\mathbf{z}\sim\mathcal{D}^{2m}}[\mathbf{z}^\sigma\in R]$. Zu $b\in\{\pm1\}^m$ betrachte nun die Permutation σ_b, die (für $i=1,\ldots,m$) i mit $m+i$ vertauscht, falls $b_i=-1$, und $i,m+i$ als Fixpunkte hat, falls $b_i=1$. Offensichtlich gilt $\mathbf{z}^{\sigma_b}\in R\Leftrightarrow\mathbf{z}\in R_b$. Hieraus ergibt sich die Behauptung 2.

Beweis von Behauptung 3: Die Behauptung ergibt sich aus folgender Rechnung:

$$\begin{aligned}\mathcal{D}^{2m}(R) &= 2^{-m}\sum_{b\in\{\pm1\}^m} D^{2m}(R_b)=\mathbb{E}_{b\sim U,\mathbf{z}\sim\mathcal{D}^{2m}}[\mathbb{1}_{[\mathbf{z}\in R_b]}]\\ &\leq \sup_{\mathbf{z}\in\mathcal{Z}^{2m}}\mathbb{E}_{b\sim U}[\mathbb{1}_{[\mathbf{z}\in R_b]}]=\sup_{\mathbf{z}\in\mathcal{Z}^{2m}}\Pr_{b\sim U}[\mathbf{z}\in R_b]\ .\end{aligned}$$

Beweis von Behauptung 4: Für eine feste Wahl von $\mathbf{z}=(z_1,\ldots,z_{2m})\in\mathcal{Z}^{2m}$ mit $x_i=(x_i,y_i)$ spielen zur Berechnung von $\Pr_{b\sim U}[\mathbf{z}\in R_b]$ bei jeder Hypothese $h\in\mathcal{H}$ nur ihre Werte auf $M:=\{x_1,\ldots,x_{2m}\}$ eine Rolle. Diese Wahrscheinlichkeit würde sich nicht ändern, wenn wir auf der rechten Seite der Definitionsgleichung für R_b "$\exists h\in\mathcal{H}_M$" statt "$\exists h\in\mathcal{H}$" schreiben. Aus der Definition der Kapazitätsfunktion $\tau_{\mathcal{H}}$ folgt $|\mathcal{H}_M|\leq\tau_{\mathcal{H}}(2m)$. Behauptung 4 ergibt sich daher aus der „Union Bound".

Beweis von Behauptung 5: Für jede feste Wahl von $\mathbf{z},h$ und $b\sim U$ betrachte die Zufallsvariablen

$$X_i=b_i\cdot(\mathbb{1}_{[h(x_i)\neq y_i]}-\mathbb{1}_{[h(x_{m+i})\neq y_{m+i}]})\in\{-1,0,1\}\subset[-1,1]$$

für $i=1,\ldots,m$. Dann sind $X_1,\ldots,X_m$ iid-Variablen mit Erwartungswert 0. Die Aussage $\mathbf{z}\in R_{b,h}$ ist äquivalent dazu, dass $(X_1+\ldots+X_m)/m$ um mindestens $\varepsilon/2$ vom Erwartungswert 0 abweicht. Mit der Hoeffding-Ungleichung erhalten wir $\Pr_{b\sim U}[\mathbf{z}\in R_{b,h}]\leq 2\exp(-\varepsilon^2m/8)$, wie man leicht nachrechnen könnte.

Wir wollen rückblickend noch einmal auf wichtige Techniken aufmerksam machen, die beim Nachweis von (20) zur Anwendung kamen. Behauptungen 2 und 3 bringen die Vorzeichenmuster $b\in\{\pm1\}^m$ ins Spiel und nutzen aus, dass $\mathcal{D}^{2m}(R_b)$ für alle Wahlen von b stets den

selben Wert hat (eine Art Symmetrie weswegen man auch von einer „Symmetrisierungstechnik“ spricht). Ab Behauptung 3 haben wir es mit einem endlichen Wahrscheinlichkeitsraum mit den Elementarereignissen $b \in \{\pm 1\}^m$ zu tun. Dieser ist wesentlich einfacher zu analysieren als der ursprüngliche Raum mit der (dem Lerner unbekannten) Verteilung $\mathcal{D}^{2m}$. Beim Übergang von Behauptung 3 zu Behauptung 4 werden wir den Existenzquantor los, der in den Definitionsbedingungen der Ereignisse R und R_b steckt. Um ihn loszuwerden, haben wir Sauer's Lemma und die Union Bound eingesetzt. Danach ist das Problem soweit „runter gekocht“, dass (im Beweis von Behauptung 5) schließlich die Hoeffding Ungleichung zum Einsatz gebracht werden kann.

Wenn wir (20) mit dem Lemma von Sauer kombinieren, erhalten wir:

Theorem 4.12 *Es sei $\mathcal{H}$ eine binäre Hypothesenklasse der VC-Dimension d. Dann gilt für alle $m \geq 2\ln(4)/\varepsilon^2$:*

$$\Pr_{\mathbf{z}\sim\mathcal{D}^m}[\exists h \in \mathcal{H} : |L_{\mathbf{z}}(H) - L_{\mathcal{D}}(\mathcal{H})| > \varepsilon] \leq 4\Phi_d(2m)\exp\left(\frac{-\varepsilon^2 m}{8}\right) \leq 4\left(\frac{2em}{d}\right)^d \exp\left(\frac{-\varepsilon^2 m}{8}\right) \quad . \tag{21}$$

Weiterhin existiert eine universelle Konstante $c > 0$, so dass

$$m_{\mathcal{H}}^{UC}(\varepsilon, \delta) \leq c \cdot \frac{1}{\varepsilon^2}\left(d + \ln\left(\frac{1}{\delta}\right)\right) \quad .$$

Insbesondere erfüllt jede binäre Hypothesenklasse einer endlichen VC-Dimension die uniforme Konvergenzbedingung.

Beweis Die erste Aussage ist klar. Die zweite ergibt sich aus der ersten, indem wir die rechte Seite von (21) kleiner oder gleich δ setzen und dann (geschickt) nach m auflösen. •

4.6 Das Fundamentaltheorem der PAC-Lerntheorie

Die qualitative Version des Fundamentaltheorems liest sich wie folgt:

Theorem 4.13 *Für eine binäre Hypothesenklasse (und die Null-Eins-Verlustfunktion) sind folgende Aussagen äquivalent:*

1. *$\mathcal{H}$ erfüllt die uniforme Konvergenzbedingung.*
2. *$\mathcal{H}$ ist mit ERM agnostisch PAC-lernbar.*
3. *$\mathcal{H}$ ist agnostisch PAC-lernbar.*
4. *$\mathcal{H}$ ist PAC-lernbar unter der Realisierbarkeitsannahme.*
5. *$\mathcal{H}$ ist mit ERM PAC-lernbar unter der Realisierbarkeitsannahme.*

6. Die VC-Dimension von $\mathcal{H}$ ist endlich.

Beweis Aus dem Erfüllen der uniformen Konvergenzbedingung ergeben sich alle Aussagen zur PAC-Lernbarkeit von $\mathcal{H}$, also die Aussagen 2,3,4,5. Aus jedem der PAC-Lernbarkeitsresultate ergibt sich die Endlichkeit der VC-Dimension, da gemäß Folgerung 4.8 Klassen mit unendlicher VC-Dimension nicht einmal unter der Realisierbarkeitsannahme PAC-lernbar sind. Zu einem vollständigen Ringschluss fehlt also nur die Herleitung der ersten Aussage (Erfüllen der uniformen Konvergenzbedingung) aus der sechsten (Endlichkeit der VC-Dimension). Diesen fehlenden Puzzlestein liefert aber gerade Theorem 4.12. •

Es folgt die quantitative Version des Fundamentaltheorems:

Theorem 4.14 *Es gibt universelle Konstanten $c_1, \ldots, c_7 > 0$, so dass für jede binäre Hypothesenklasse und $d = \mathrm{VCD}(\mathcal{H})$ folgendes gilt:*

$$\begin{aligned} c_1 \cdot \frac{1}{\varepsilon^2}\left(d + \ln\left(\frac{1}{\delta}\right)\right) &\leq m_{\mathcal{H}}^{UC}(\varepsilon,\delta) \leq c_2 \cdot \frac{1}{\varepsilon^2}\left(d + \ln\left(\frac{1}{\delta}\right)\right) \\ c_3 \cdot \frac{1}{\varepsilon^2}\left(d + \ln\left(\frac{1}{\delta}\right)\right) &\leq m_{\mathcal{H}}(\varepsilon,\delta) \leq c_4 \cdot \frac{1}{\varepsilon^2}\left(d + \ln\left(\frac{1}{\delta}\right)\right) \end{aligned}$$

Unter der Realisierbarkeitsannahme gilt weiterhin

$$c_5 \cdot \frac{1}{\varepsilon}\left(d + \ln\left(\frac{1}{\delta}\right)\right) \leq m_{\mathcal{H}}(\varepsilon,\delta) \leq c_6 \cdot \frac{1}{\varepsilon}\left(d\log\left(\frac{1}{\varepsilon}\right) + \ln\left(\frac{1}{\delta}\right)\right)$$

bzw. sogar

$$m_{\mathcal{H}}(\varepsilon,\delta) \leq c_7 \cdot \frac{1}{\varepsilon}\left(d + \ln\left(\frac{1}{\delta}\right)\right) \quad ,$$

wenn der Lernalgorithmus eine Hypothese außerhalb von $\mathcal{H}$ ausgeben darf. Alle oberen Schranken mit Ausnahme der letzten werden durch die ERM-Lernstrategie (Ausgabe einer Hypothese aus $\mathcal{H}$ mit minimalem empirischen Verlustwert) erreicht.

Theorem 4.14 haben wir nur teilweise bewiesen:

- Die angegebene obere Schranke für $m_{\mathcal{H}}^{UC}(\varepsilon,\delta)$ wurde im Rahmen von Theorem 4.12 bewiesen.
- Die angegebene obere Schranke für $m_{\mathcal{H}}(\varepsilon,\delta)$ (ohne Realisierbarkeitsannahme) folgt dann aus der allgemeinen Ungleichung $m_{\mathcal{H}}(\varepsilon,\delta) \leq m_{\mathcal{H}}^{UC}(\varepsilon/3,\delta)$.
- Die unter der Realisierbarkeitsannahme angegebene untere Schranke für $m_{\mathcal{H}}(\varepsilon,\delta)$ haben wir in Abschnitt 4.3 hergeleitet.

Wir verzichten auf den vollständigen Beweis.

5 Lernbarkeit im nichtuniformen Modell

Im agnostischen PAC-Lernmodell muss der Lerner imstande sein, eine Hypothese auszugeben, deren Fehlerrate mit hoher Wahrscheinlichkeit nicht wesentlich größer ist, als die beste Fehlerrate, die sich mit einer Vergleichshypothese aus der vorgegebenen Hypothesenklasse erzielen lässt. Wie wir wissen sind Klassen mit unendlicher VC-Dimension in diesem Modell (nicht einmal unter der Realisierbarkeitsannahme) lernbar. Wenn wir jedoch erlauben, die zum Lernen erforderliche Größe der Trainingsmenge nicht nur von ε, δ sondern auch von der Vergleichshypothese h abhängig zu machen, dann ist eine unendliche VC-Dimension i.A. keine unüberwindliche Barriere mehr. Ansätze dieser Art führen zum sogenannten nichtuniformen Lernmodell, dass wir in Abschnitt 5.1 eingehend besprechen. Es wird sich zeigen, dass ERM im nichtuniformen Lernmodell keine brauchbare Lernstrategie ist. Die in diesem Modell erfolgreiche Strategie trägt den Namen strukturierte Risikominimierung, oder auch kurz SRM. Eine spezielle Ausprägung von SRM namens „Minimum Description Length (MDL)“ besprechen wir in Abschnitt 5.2. Schließlich kommen wir in Abschnitt 5.3 wieder auf das „bias-complexity dilemma“ zu sprechen und vergleichen diesbezüglich das uniforme mit dem nichtuniformen Lernmodell.

5.1 Definition und Analyse des nichtuniformen Modells

Definition 5.1 *Eine binäre Hypothesenklasse $\mathcal{H}$ heißt* lernbar im nichtuniformen Modell*, wenn eine Funktion $m_{\mathcal{H}} : (0,1)^2 \times \mathcal{H} \to \mathbb{R}_+$ und ein Lernalgorithmus A existieren, so dass für alle $0 < \varepsilon, \delta < 1$, alle $m \geq 1$ und jede Verteilung $\mathcal{D}$ auf $\mathcal{X} \times \{0,1\}$ folgende Bedingung erfüllt ist: A, gestartet auf einer zufälligen Trainingsmenge $S \sim D^m$, ermittelt eine Hypothese $h_S \in \mathcal{H}$ und es gilt*

$$\Pr_{S \sim D^m} \left[L_{\mathcal{D}}(h_S) \leq \inf_{h \in \mathcal{H} : m_{\mathcal{H}}(\varepsilon, \delta, h) \leq m} L_{\mathcal{D}}(h) + \varepsilon \right] \geq 1 - \delta \ . \tag{22}$$

Ähnlich wie beim agnostischen Lernen kann der Lerner A prinzipiell (im Sinne der Bedingung (22)) mit jeder Hypothese aus $\mathcal{H}$ gut konkurrieren, aber im Unterschied zum herkömmlichen Modell des agnostischen Lernens hängt die Größe der dazu erforderlichen Trainingsmenge nicht nur von ε und δ sondern auch von der Vergleichshypothese h ab. Das Hauptresultat in diesem Kapitel ist das folgende:

Theorem 5.2 *Folgende Aussagen zu einer binären Hypothesenklasse $\mathcal{H}$ sind äquivalent:*

1. *$\mathcal{H}$ ist lernbar im nichtuniformen Modell.*
2. *$\mathcal{H}$ ist eine abzählbare Vereinigung von Klassen endlicher VC-Dimension.*
3. *$\mathcal{H}$ ist eine abzählbare Vereinigung agnostisch PAC-lernbarer Klassen.*
4. *$\mathcal{H}$ ist eine abzählbare Vereinigung von Klassen, welche die uniforme Konvergenzbedingung erfüllen.*

Beweis Wir beweisen die Äquivalenzen durch einen Ringschluss.
1.⇒2. Es sei A der Algorithmus und $m_{\mathcal{H}} : (0,1)^2 \times \mathcal{H} \to \mathbb{R}_+$ die Funktion aus Definition 5.1. Für alle $n \geq 1$ setze

$$\mathcal{H}_n = \left\{ h \in \mathcal{H} |\ m_{\mathcal{H}} \left(\frac{1}{8}, \frac{1}{5}, h \right) \leq n \right\} \ .$$

Dann hat $S \sim \mathcal{D}^n$ mit einer Wahrscheinlichkeit von mindestens $4/5$ die Eigenschaft, dass die von A ermittelte Hypothese h_S die Bedingung $L_{\mathcal{D}}(h_S) \leq \inf_{h \in \mathcal{H}_n} L_{\mathcal{D}}(h) + 1/8$ erfüllt. In anderen Worten: $\mathcal{H}_n$ ist agnostisch PAC-lernbar in dem schwächeren Sinn, dass ε bzw. δ konstant auf $1/8$ bzw. $1/5$ gesetzt werden. Wir wissen aus Folgerung 4.8, dass dies nur möglich ist, wenn $\mathcal{H}_n$ eine endliche VC-Dimension hat. Hieraus ergibt sich 2.
2.⇒3. und 3.⇒4. Diese Implikationen ergeben sich direkt aus dem Fundamentaltheorem der PAC-Lerntheorie.
4.⇒1. Es sei $\mathcal{H} = \cup_{n \geq 1} \mathcal{H}_n$ die Darstellung von $\mathcal{H}$ als Vereinigung von Klassen, welche allesamt die uniforme Konvergenzbedingung erfüllen. Es sei $m_n^{UC} : (0,1)^2 \to \mathbb{R}_+$ eine Funktion, welche die uniforme Konvergenz bezeugt. Dann gilt:

- Für alle $0 < \varepsilon, \delta < 1$ und $m \geq m_n^{UC}(\varepsilon, \delta)$ ist $S \sim \mathcal{D}^m$ mit einer Wahrscheinlichkeit von mindestens $1 - \delta$ für alle Hypothesen aus $\mathcal{H}_n$ ε-repräsentativ.

Die folgende Funktion wird im Rest des Beweises eine wichtige Rolle spielen:

$$\varepsilon_n^{UC}(m, \delta) = \min\{\varepsilon \in (0,1) |\ m_n^{UC}(\varepsilon, \delta) \leq m\} \ . \tag{23}$$

Die obige Aussage zur ε-Repräsentativität von $S \sim \mathcal{D}^m$ kann mit Hilfe von ε_n^{UC} umformuliert werden wie folgt:

- Für alle $m \geq 1$ une $\delta \in (0,1)$ ist $S \sim \mathcal{D}^m$ mit einer Wahrscheinlichkeit von mindestens $1 - \delta$ für alle Hypothesen aus $\mathcal{H}_n$ $\varepsilon_n^{UC}(m, \delta)$-repräsentativ.

Beachte weiterhin, dass gilt:

$$m \geq m_n^{UC}(\varepsilon, \delta) \Rightarrow \varepsilon_n^{UC}(m, \delta) \leq \varepsilon \ .$$

Die Gefahr der Nicht-Repräsentativität von $S \sim \mathcal{D}^m$ besteht für alle Hypothesen aus $\mathcal{H} = \cup_{n \geq 1} \mathcal{H}_n$. Wir werden daher δ (die tolerierte Wahrscheinlichkeit für die Nicht-Repräsentativität von $S \sim \mathcal{D}^m$) auf die Klassen $(\mathcal{H}_n)_{n \geq 1}$ verteilen, d.h., wir wählen eine Folge $(w_n)_{n \geq 1}$ unter den Randbedingungen $w_n > 0$ und $\sum_{n \geq 1} w_n \leq 1$ und reservieren den Konfidenzparameter $\delta_n = w_n \delta$ für $\mathcal{H}_n$. Für alle $h \in \mathcal{H}$ definieren wir:

$$n(h) := \min\{n |\ h \in \mathcal{H}_n\} \ \text{ und } \ m_{\mathcal{H}}(\varepsilon, \delta, h) := m_{n(h)}^{UC}(\varepsilon/2, \delta_n) \ .$$

Offensichtlich treffen dann für jede Wahl von $m \geq 1$ und $\varepsilon, \delta \in (0,1)$ die folgenden Aussagen auf $S \sim \mathcal{D}^m$ mit einer Wahrscheinlichkeit von mindestens $1 - \delta$ zu:

- Für alle $n \geq 1$ gilt: S ist $\varepsilon_n^{UC}(m, \delta_n)$-repräsentativ für alle Hypothesen aus $\mathcal{H}_n$ (und somit sogar $\varepsilon/2$-repräsentativ, falls $m \geq m_n^{UC}(\varepsilon/2, \delta_n)$).

- S ist $\varepsilon/2$-repräsentativ für alle $h \in \mathcal{H}$ mit $m_{\mathcal{H}}(\varepsilon, \delta, h) \leq m$.

Diese Betrachtungen legen die folgende Lernregel nahe.

SRM: Gegeben eine Trainingsmenge $S \in Z^m$ und ein Konfidenzparameter $0 < \delta < 1$, finde eine Hypothese $h_S \in \mathcal{H}$, welche die Kostenfunktion $L_S(h) + \varepsilon^{UC}_{n(h)}(m, \delta_{n(h)})$ minimiert.

Es sei h_S die SRM-Hypothese (d.h. die Hypothese, die sich aus Anwendung der SRM-Lernregel ergibt). Für alle $h \in \mathcal{H}$ mit $m_{\mathcal{H}}(\varepsilon, \delta, h) \leq m$ erfüllt die SRM-Hypothese h_S das Zielkriterium $L\mathcal{D}(h_S) \leq L_{\mathcal{D}}(h) + \varepsilon$ wie sich aus folgender Abschätzung ergeben wird:

$$L_{\mathcal{D}}(h_S) \leq L_S(h_S) + \varepsilon^{UC}_{n(h_S)}(m, \delta_{n(h_S)}) \leq L_S(h) + \varepsilon^{UC}_{n(h)}(m, \delta_{n(h)}) \leq L_{\mathcal{D}}(h) + \varepsilon$$

Die erste Ungleichung gilt, da $|L_S(h_S) - L_{\mathcal{D}}(h_S)|$ durch $\varepsilon^{UC}_{n(h_S)}(m, w_{n(h_S)}\delta)$ begrenzt ist. Die zweite Ungleichung gilt, da h_S die Kostenfunktion $L_S(h) + \varepsilon^{UC}_{n(h)}(m, w_{n(h)}\delta)$ minimiert. Die dritte Ungleichung gilt, da wegen $m \geq m_{\mathcal{H}}(\varepsilon, \delta, h)$ sowohl $\varepsilon^{UC}_{n(h)}(m, w_{n(h)}\delta)$ als auch $|L_S(h) - L_{\mathcal{D}}(h)|$ durch $\varepsilon/2$ begrenzt sind. Damit ist der Beweis für die Lernbarkeit von $\mathcal{H}$ im nichtuniformen Modell abgeschlossen. •

Mögliche Wahlen für die Folge $(w_n)_{n\geq 1}$ sind zum Beispiel $w_n = 2^{-n}$ oder $w_n = \frac{1}{n(n+1)}$. Diesen Wahlen von w_n liegen die geometrische und die streng harmonische Reihe mit

$$\sum_{n\geq 1} 2^{-n} = 1 = \sum_{n\geq 1} \frac{1}{n(n+1)}$$

zugrunde.

Es gibt (sogar viele) Hypothesenklassen, die nicht (uniform) PAC-lernbar sind (d.h. eine unendliche VC-Dimension besitzen) aber ohne Weiteres im nichtuniformen Modell gelernt werden können:

Beispiel 5.3 *Ein Polynom p in einer reellen Variable zerlegt $\mathbb{R}$ in den Bereich $\{x|\ p(x) \geq 0\}$ und $\{x|\ p(x) < 0\}$. Es ist nicht schwer zu zeigen, dass die VC-Dimension der Klasse*

$$\mathcal{H}_d = \{\mathbb{1}_{[p(x)\geq 0]}|\ p \text{ ist Polynom vom Grad } d\}$$

gleich $d+1$ ist, so dass $\mathcal{H} = \cup_{d\geq 1}\mathcal{H}_d$ eine unendliche VC-Dimension besitzt. Nach dem Fundamentalsatz der PAC-Lerntheorie ist $\mathcal{H}$ nicht PAC-lernbar. Hingegen folgt aus Theorem 5.2, dass $\mathcal{H}$ im nichtuniformen Modell lernbar ist.

5.2 Minimum Description Length

Es sei $\mathcal{H} = \{h_n|\ n \geq 1\}$ eine abzählbar unendliche Klasse von binären Hypothesen. Da $\mathcal{H}_n = \{h_n\}$ trivialerweise die uniforme Konvergenzbedingung erfüllt (bezeugt durch die Hoeffding-Ungleichung), ist $\mathcal{H} = \cup_{n\geq 1}\{h_n\}$ eine Darstellung von $\mathcal{H}$ als Vereinigung von Klassen, welche die uniforme Konvergenzbedingung erfüllen. Gemäß Theorem 5.2 ist $\mathcal{H}$ im nichtuniformen Modell lernbar. Es hat sich somit ergeben:

Bemerkung 5.4 *Jede abzählbare (endliche oder unendliche) binäre Hypothesenklasse ist im nichtuniformen Modell lernbar.*

Da es abzählbar unendliche Klassen mit unendlicher VC-Dimension gibt (zum Beispiel die Klasse aller endlichen Teilmengen der natürlichen Zahlen), gibt es abzählbar unendliche Klassen, die im nichtuniformen Modell lernbar sind, aber nicht im herkömmlichen Modell des PAC-Lernens.

Wir wollen im Folgenden untersuchen, wie SRM, angesetzt auf eine abzählbar unendliche Klasse, zu Werke geht. Zunächst einmal liefert die Hoeffding-Ungleichung für jede einzelne Hypothese $h \in \mathcal{H}$:

$$\Pr_{S\sim\mathcal{D}^m}[|L_S(h) - L_{\mathcal{D}}(h)| \geq \varepsilon] \leq 2e^{-2m\varepsilon^2} \ . \tag{24}$$

Auflösen der Ungleichung

$$2e^{-2m\varepsilon^2} \leq \delta$$

nach m bzw. nach ε liefert folgende mögliche Wahlen für die Funktionen m_n^{UC} bzw. ε_n^{UC}:

$$m_n^{UC}(\varepsilon, \delta) = \frac{\ln(2/\delta)}{2\varepsilon^2} \text{ und } \varepsilon_n^{UC}(m, \delta) = \sqrt{\frac{\ln(2/\delta)}{2m}} \ .$$

Wir schreiben im Folgenden $w(h_n)$ anstelle von $w(n)$ und betrachten somit w als eine auf den Hypothesen $h \in \mathcal{H}$ definierte Gewichtsfunktion. Die von SRM minimierte Kostenfunktion mit den Termen $L_S(h)$ und $\varepsilon_{n(h)}^{UC}(m, w(h)\delta)$ hat dann die Form

$$L_S(h) + \sqrt{\frac{-\ln(w(h)) + \ln(2/\delta)}{2m}} \ . \tag{25}$$

Da $\mathcal{H}$ abzählbar unendlich ist, können die Hypothesen $h \in \mathcal{H}$ durch Strings über einem Alphabet beschrieben werden. Die Beschreibung könnte zum Beispiel eine von $x \in \mathcal{X}$ abhängige mathematische Formel sein, die als Ergebnis 0 oder 1 liefert. Da letztendlich alles binär kodiert werden kann, gehen wir der Einfachheit halber davon aus, dass wir h mit einem Binärstring identifizieren können. Die Bitlänge dieses Strings notieren wir als $|h|$. Wir wollen zudem annehmen, dass $\mathcal{H}$ präfixfrei ist, d.h., keine Hypothese in $\mathcal{H}$ ist Präfix (=Anfangswort) einer anderen Hypothese in $\mathcal{H}$. Eine natürliche Wahl der Gewichte wäre nun $w(h) = 2^{-|h|}$. Diese Gewichtswahl ist zulässig, d.h., sie erfüllt die Ungleichung $\sum_{h\in\mathcal{H}} w(h) \leq 1$, wie folgendes Resultat aufzeigt:

Lemma 5.5 (Kraft's Ungleichung) *Es sei $\mathcal{H} \subseteq \{0,1\}^*$ eine präfixfreie Menge von Strings. Dann gilt $\sum_{h\in\mathcal{H}} 2^{-|h|} \leq 1$.*

Beweis Wir verbinden mit den Strings aus $\mathcal{H}$ auf folgende Weise einen Wahrscheinlichkeitsraum. Stelle einen Bitstring durch wiederholtes Werfen einer perfekten Münze her bis — es folgt die Stoppbedingung[11] — der aktuelle Bitstring ein Mitglied der Menge $\mathcal{H}$ ist. Wegen

[11] die nicht notwendig eintreten muss

der Präfixfreiheit von $\mathcal{H}$ hat jeder String $h \in \mathcal{H}$ eine Wahrscheinlichkeit von $2^{-|h|}$ auf diese Weise zufällig generiert zu werden. Da die Generierung verschiedener Strings aus $\mathcal{H}$ disjunkte Ereignisse sind, können sich die Wahrscheinlichkeiten aller $h \in \mathcal{H}$ maximal zu 1 aufaddieren. •

Durch Einsetzen von $w(h) = 2^{-|h|}$ in (25) kann die von SRM minimierte Kostenfunktion in der Form

$$L_S(h) + \sqrt{\frac{\ln(2)|h| + \ln(2/\delta)}{2m}} \tag{26}$$

geschrieben werden. Durch diese Wahl der Kostenfunktion werden im Zweifelsfalle (also bei ähnlicher empirischer Fehlerrate) Hypothesen mit einer kleinen Beschreibungslänge vor denen mit einer großen Beschreibungslänge vorgezogen. Die aus der Gewichtswahl $w(h) = 2^{-|h|}$ resultierende Variante von SRM heißt daher *MDL-Strategie* oder kurz *MDL*. „MDL" steht für „Minimum Description Length".

Wir fassen die bisherigen Überlegungen dieses Abschnittes wie folgt zusammen:
Es sei $\mathcal{H}$ eine abzählbare Hypothesenklasse, deren Hypothesen wir mit binären Beschreibungsstrings identifizieren. Somit gilt $\mathcal{H} \subseteq \{0,1\}^*$. Es wird vorausgesetzt, dass $\mathcal{H}$ präfixfrei ist. Dann führt SRM, parametrisiert mit den Gewichten $w(h) = 2^{-|h|}$, zu folgender Lernstrategie:

MDL: Gegeben eine Trainingsmenge $S \sim \mathcal{D}^m$ und ein Konfidenzparameter δ, ermittle eine Hypothese h_S, welche die Kostenfunktion (26) minimiert.

Ockham's Rasiermesser (Occam's Razor). Wilhelm von Ockham (1288–1347) war ein mittelalterlicher Philosoph in der Epoche der Spätscholastik. Er formulierte eine Art Sparsamkeitsprinzip, welches sinngemäß besagt, dass, auf der Suche nach Erklärungen für die Phänomene dieser Welt, grundsätzlich einfachen Erklärungen der Vorzug über unnötig komplizierten zu geben ist. Es scheint so, als würde die MDL-Strategie exakt nach diesem Prinzip vorgehen. Dabei ist allerdings Vorsicht geboten, da wir bei der Wahl des binären Codes für die Hypothesen erst einmal völlige Narrenfreiheit besitzen. Die Tatsache, dass wir eine Hypothese h kürzer kodieren als eine andere Hypothese h' kann eine völlig willkürliche (und auch revidierbare) Festlegung sein, die nichts mit einer der Hypothese innewohnenden Einfachheit oder Komplexität zu tun haben muss. Wer an einer vertieften Diskussion solcher und verwandter Fragen interessiert ist, sollte sich die Abschnitte 7.3.1 und 7.5 des von Shalev-Shwartz und Ben-David verfassten Lehrbuches durchlesen.

5.3 Hypothesentest, uniformes und nichtuniformes Modell

Vor dem Hintergrund von SRM als Alternative zu ERM kommen wir noch einmal auf das „bias-complexity dilemma" zu sprechen.

Hypothesentest: Ein extrem starkes „Bias" wäre es, sich a-priori (also vor Inspektion der empirischen Daten) auf eine einzige Hypothese h festzulegen. Die empirischen Daten

dienen dann gleichsam nur als Testmenge, um a-posteriori die Fehlerrate von h empirisch zu schätzen. Der Schätzfehler ist dann sehr klein und kann mit der Hoeffding Ungleichung kontrolliert werden. Auf der anderen Seite riskieren wir mit der frühen Festlegung auf eine Hypothese einen extrem hohen Approximationsfehler, da unsere Wahl der Hypothese nicht mit den empirischen Daten abgestimmt wird.

Uniformes Lernen: Ein weniger starkes „Bias“ liegt vor, wenn wir uns a-priori auf eine Hypothesenklasse (anstelle einer einzelnen Hypothese) festlegen. Wenn diese Klasse eine unendliche VC-Dimension besitzt, dann ist das „Bias“ zu schwach, und der Schätzfehler kann nicht mehr kontrolliert werden (wohingegen der Approximationsfehler vermutlich klein ist). Eine Hypothesenklasse mit endlicher VC-Dimension erfüllt, wie wir inzwischen wissen, die uniforme Konvergenzbedingung, und darüber lässt sich der Schätzfehler kontrollieren. Die Klasse muss allerdings clever gewählt sein, damit der Approximationsfehler beherrschbar ist.

Nichtuniformes Lernen: Nichtuniformes Lernen eröffnet die Möglichkeit, eine sehr ausdrucksstarke Hypothesenklasse $\mathcal{H}$ mit unendlicher VC-Dimension zu wählen, welche als eine Vereinigung $\cup_{n\geq 1}\mathcal{H}_n$ von Klassen endlicher VC-Dimension geschrieben werden kann. Um den Schätzfehler zu kontrollieren wird das von $\mathcal{H}$ repräsentierte (eher schwache) „Bias“ dadurch erhöht, dass wir a-priori eine Gewichtung der Klassen $\mathcal{H}_n$ (und damit auch der einzelnen Hypothesen) vornehmen. Wie wir im Beweis von Theorem 5.2 gesehen haben, lässt sich dadurch der Schätzfehler für alle Hypothesen aus $\mathcal{H}$ hinreichend gut kontrollieren. In das „bias-complexity dilemma“ geraten wir nun bei der Wahl der Gewichte. Wenn wir zum Beispiel die gesamte Gewichtsmasse 1 im Wesentlichen auf eine einzelne Hypothese konzentrieren, dann degeneriert das nichtuniforme Lernen zum Hypothesentesten. Nur eine clevere Wahl der Gewichte wird sowohl den Approximations- als auch den Schätzfehler unter Kontrolle halten.

Der Begriff der „cleveren Wahl“ von $\mathcal{H}$ bzw. $(w(n))_{n\geq 1}$ ist anwendungsabhängig. Eine clevere Wahl erfordert also i.A. eine große Vertrautheit mit der Anwendung, in welcher ERM bzw. SRM zum Einsatz kommt.

6 Effizientes PAC-Lernen

Im gesamten Kapitel machen wir die folgenden Voraussetzungen:

1. Die Informationskomplexität von $\mathcal{H}$ ist beherrschbar, d.h., $\mathrm{VCD}(\mathcal{H}_n)$ ist polynomiell in n beschränkt.

2. Die Hypothesen $h \in \mathcal{H}_n$ besitzen eine „natürliche Kodierung“. Wir werden eine Hypothese $h \in \mathcal{H}_n$ oft (der Einfachheit halber) mit ihrem Codewort identifizieren. Die Länge dieses Codewortes wird dann als $|h|$ notiert. Wir setzen voraus, dass $\sup_{h\in\mathcal{H}_n} |h|$ polynomiell in n beschränkt ist.

3. Eine analoge Bemerkung gilt für die Beispielinstanzen aus $\mathcal{X}_n$.

4. Das Auswertungsproblem zu $\mathcal{H}$ — zu gegebenem $h \in \mathcal{H}_n$ und $x \in \mathcal{X}_n$ (mit beliebigem $n \geq 1$), berechne $h(x) \in \{0,1\}$ — ist in Polynomialzeit lösbar.

Wir werden in Abschnitt 6.1 an grundlegende Konzepte der Komplexitätstheorie erinnern. In Abschnitt 6.2 diskutieren wir zwei zu einer Klasse $\mathcal{H}$ assoziierte Probleme: das Konsistenzproblem und das „Minimum Disagreement Problem". Es wird sich, grob gesprochen, ergeben dass $\mathcal{H}$ genau dann unter der Realisierbarkeitsannahme effizient PAC-lernbar ist, wenn das Konsistenzproblem zu $\mathcal{H}$ (evtl. randomisiert) in Polynomialzeit lösbar ist. Ein analoger Zusammenhang gilt für effiziente agnostische PAC-Lernbarkeit und das Minimum Disagreement Problem. In Abschnitt 6.3 stellen wir einige populäre Hypothesenklassen über dem Booleschen Grundbereich $\{0,1\}^n$ vor. Beispiele effizient PAC-lernbarer Klassen lernen wir in den Abschnitten 6.4 und 6.5 kennen. Inhärent harte Lernprobleme behandeln wir im Abschnitt 6.6. Dabei diskutieren wir sowohl über komplexitätstheoretische als auch über kryptographische Barrieren zum Lernen.

6.1 Ein Ausflug in die Komplexitätstheorie

Es bezeichne Σ ein Alphabet (= endliche Menge) und Σ^+ die Menge der nichtleeren über Σ gebildeten Strings. Wenn wir zum Beispiel von einem binären Alphabet $\Sigma = \{0,1\}$ ausgehen, so ergibt sich $\Sigma^+ = \{0,1,00,01,10,11,000,\ldots\}$. Wir definieren $\Sigma^* = \Sigma^+ \cup \{\lambda\}$ als die Erweiterung von Σ^+ um das sogenannte *leere Wort* λ (bestehend aus 0 Buchstaben).

Ein Algorithmus ist ein Rechenverfahren A, das, angesetzt auf einen Eingabestring $w \in \Sigma^*$, eine Ausgabe aus $A(w) \in \Sigma^*$ produziert. Bei einem Algorithmus für ein Entscheidungsproblem ist die Ausgabe entweder 0 (für die Antwort „Nein") oder 1 (für die Antwort „Ja"). $L(A) = \{w \in \Sigma^* |\ A(w) = 1\}$ ist dann die *Sprache der von A akzeptierten Wörter*.

Wir setzen ein intuitives Verständnis für einen *elementaren Rechenschritt* in einem Algorithmus voraus. Eine formale Definition würde ein mathematisches Rechenmodell erfordern wie zum Beispiel die Turing-Maschine (TM) oder die Random Access Maschine (RAM). Solche Rechenmodelle werden in Vorlesungen wie „Theoretische Informatik" oder „Datenstrukturen" eingeführt und verwendet.

Um ein Gefühl für den Begriff eines elementaren Rechenschrittes zu bekommen, erläutern wir im Folgenden, grob und informell, was eine RAM in einem Rechenschritt tun kann:

- eine arithmetisch-logische Operation durchführen, d.h. zwei in Rechen- oder Indexregistern befindliche Zahlen miteinander verknüpfen (etwa mit den Operationen $+, -, *, /$ bzw. $\vee, \wedge, \neg$),

- einen Transportbefehl ausführen (etwa den Inhalt einer Speicherzelle in ein Register laden oder, umgekehrt, den Inhalt eines Registers in einer Speicherzelle ablegen),

- einen bedingten oder unbedingten Sprung zu einer anderen Kommandozeile des Rechenprogrammes durchführen,

- mit direkter oder indirekter Adressierung auf eine Speicherzelle zugreifen.

Speicherzellen können wir uns abstrakt wie eine Kollektion $x_1, x_2, \ldots$ von Variablen vorstellen. Die Variable x_i hat (per Definition) die Adresse i. Bei direkter Adressierung steht die Adresse i explizit im Programmtext. Bei indirekter Adressierung steht die Adresse i in einem Indexregister.[12]

Bei dem sogenannten *uniformen Kostenmaß* wird ein Rechenschritt mit Kosten 1 belegt. Bei dem *logarithmischen Kostenmaß* wird als Kostenterm die Bitlänge der beteiligten Zahlen veranschlagt (etwa Kosten k bei der arithmetisch-logischen Verknüpfung zweier k-Bit Zahlen). Das uniforme Kostenmaß könnte missbraucht werden, indem man riesige kombinatorische Strukturen (zum Beispiel die Adjazenzmatrix eines Graphen) in die Bits einer riesengroßen Zahl hineinkodiert. Das könnte dazu führen, dass ein einziger arithmetisch-logischer Rechenschritt einen großen Graphen durchmustert und modifiziert. In solchen Fällen ist das logarithmische Kostenmaß vorzuziehen. Solange aber nur Zahlen einer polynomiell (in der Eingabelänge n) beschränkten Bitlänge verwendet werden, ist das uniforme Kostenmaß für unsere Zwecke akkurat genug. Manchmal werden wir das uniforme Kostenmaß sogar bei arithmetischen Operationen auf den rellen Zahlen einsetzen (freilich ohne die unendliche Bitlänge mit unsauberen Tricks auszunutzen).

Die Eingabelänge: Formal betrachtet ist eine Eingabeinstanz für einen Algorithmus durch ein Eingabewort w über einem Grundalphabet Σ gegeben und $n = |w|$ ist dann die Eingabelänge. Wenn wir aber nur daran interessiert sind zu überprüfen, ob die Anzahl der zum Lösen des Problems erforderlichen Rechenschritte polynomiell in n beschränkt ist, können wir ebensogut „natürliche Maße“ für die Eingabelänge n verwenden, die sich von $|w|$ nur um ein Polynom unterscheiden. Zum Beispiel ist die natürliche Größe eines Graphen $G = (V, E)$ einfach die Anzahl $n = |V|$ seiner Knoten. Wenn das Codewort für den Graphen ein Bitstring wäre, der seiner Adjazenzmatrix A entspricht, dann wäre die Länge des Codewortes ungefähr n^2 (= Anzahl der Bits in der Matrix A). Da n und n^2 sich nur um ein Polynom (hier vom Grad 2) unterscheiden, wird es für unsere Zwecke keine Rolle spielen, ob wir die Größe der Eingabe mit n oder mit n^2 veranschlagen. Ein Markenzeichen eines polynomiell zeitbeschränkten Algorithmus ist es, dass er die Eingabegrößen (wie zum Beispiel die Knoten und die Kanten eines Graphen) nur beschränkt häufig inspiziert und bei jeder einzelnen Inspektion nur sehr elementare Aktionen ausführt.

Komplexitätsklassen. Als *formale Sprache* über dem Alphabet Σ bezeichnen wir jede Teilmenge von Σ^*. Zu einer formalen Sprache $L \subseteq \Sigma^*$ assoziieren wir folgendes Entscheidungsproblem, genannt das *Wort- oder Mitgliedschaftsproblem zur Sprache L*: gegeben $w \in \Sigma^*$, ist w ein Mitglied von L, d.h., gilt $w \in L$? Ein *Entscheidungsproblem* ist ein Problem, bei dem zu Eingabeinstanzen w nur die Antworten „Ja“ oder „Nein“ (bzw. 1 oder

[12] Da ein Programm aus einem endlichen Text besteht, könnte man ohne indirekte Adressierung nur konstant viele Speicherzellen nutzen. Der Algorithmus muss aber die Möglichkeit haben die Anzahl der verwendeten Speicherzellen von der Länge $|w|$ des Eingabestrings abhängig zu machen.

0) möglich sind. Formale Sprachen und Entscheidungsprobleme sind zwei Seiten der selben Münze:

- Wir können $L \subseteq \Sigma^*$ mit dem Wortproblem zu L identifizieren.
- Wir können umgekehrt ein Entscheidungsproblem identifizieren mit der formalen Sprache bestehend aus allen Strings, welche Ja-Instanzen repräsentieren.

Eine *Komplexitätsklasse* enthält, salopp gesprochen, formale Sprachen bzw. Entscheidungsprobleme die „ungefähr" den gleichen Aufwand an Rechenzeit (oder Speicherplatz) erfordern. In diesem Kapitel benötigen wir die Definition der Komplexitätsklassen P, NP und RP. P steht hierbei für „deterministisch Polynomialzeit", NP für „nichtdeterministisch Polynomialzeit" und RP für „Random P":

die Klasse P: Wir sagen ein Algorithmus arbeitet in *Polynomialzeit*, wenn die Anzahl seiner Rechenschritte durch ein Polynom in der Eingabelänge nach oben beschränkt ist. Wir sagen eine Sprache $L \subseteq \Sigma^*$ gehört zur Klasse P genau dann, wenn ein Algorithmus existiert, der zu einem Eingabewort w in Polynomialzeit entscheidet, ob $w \in L$.

die Klasse NP: Wir sagen eine Sprache $L \subseteq \Sigma^*$ gehört zur Klasse NP genau dann, wenn ein Polynom q und eine Sprache $L' \in P$ existieren mit

$$L = \{w \in \Sigma^* | \ \exists u \in \Sigma^* : (|u| \leq q(|w|) \wedge (\langle u, w\rangle \in L'\} \ .$$

Hierbei steht $\langle u, w\rangle$ für eine (natürliche) Kodierung des Paares (u, w) in einen String über dem Alphabet Σ. Man bezeichnet u dann auch als ein *Zertifikat* für die Aussage $w \in L$. Die Bedingung $|u| \leq q(|w|)$ besagt, salopp ausgedrückt, dass das Zertifikat „kurz" zu sein hat. Die Bedingung $\langle u, w\rangle \in L'$ in Verbindung mit $L' \in P$ besagt, salopp ausgedrückt, dass ein Zertifikat für $w \in L$ „leicht überprüfbar" sein muss. Die Klasse NP besteht, so gesehen, aus allen Sprachen, deren Strings ein kurzes und leicht überprüfbares Zertifikat besitzen.

die Klasse RP: Die Definition der Klasse RP benutzt randomisierte Algorithmen, also Algorithmen, die Zugriff auf perfekte (und voneinander unabhängige) Zufallsbits haben. Dadurch wird bei einem randomisierten Algorithmus für ein Entscheidungsproblem die Ausgabe $A(w) \in \{0, 1\}$ zu einer $0, 1$-wertigen Zufallsvariable (also zu einer Bernoulli-Variable). Die Klasse RP besteht per Definition aus allen Sprachen L, für welche ein randomisierter Polynomialzeit-Algorithmus A existiert, so dass

$$\Pr[A(w) = 1] \begin{cases} = 0 & \text{falls } w \notin L \\ \geq \frac{1}{2} & \text{falls } w \in L \end{cases} \ . \tag{27}$$

Mit anderen Worten: Strings außerhalb von L werden niemals akzeptiert und Strings innerhalb von L werden mit einer Wahrscheinlichkeit von mindestens $1/2$ akzeptiert. Ein Algorithmus A mit den genannten Eigenschaften heißt *Monte-Carlo Algorithmus* für L.

Die Bedingung, eine Eingabe $w \in L$ lediglich mit einer Wahrscheinlichkeit von mindestens $1/2$ zu akzeptieren, ist recht schwach. Sie kann aber verschärft werden wie folgt:

Bemerkung 6.1 *Für jede Konstante $k \geq 1$ gilt folgendes: ein Monte-Carlo Algorithmus A für L kann in einen Polynomialzeit-Algorithmus A_k transformiert werden, der die Bedingung*

$$\Pr[A(w) = 1] \begin{cases} = 0 & \text{falls } w \notin L \\ \geq 1 - 2^{-k} & \text{falls } w \in L \end{cases}$$

an Stelle der schwächeren Bedingung (27) erfüllt.

Beweis Algorithmus A_k setzt A k-mal auf den Eingabestring w an (mit jeweils „frischen“ Zufallsbits) und akzeptiert genau dann, wenn mindestens eine der k Rechnungen von A auf Eingabe w akzeptierend ist. Es ist offfensichtlich, dass A_k die gewünschten Eigenschaften hat. •

Offensichtlich gilt

$$P \subseteq RP \subseteq NP \ .$$

Es wird vermutet, dass diese Inklusionen echt sind. „$P \neq NP$?“ ist eine der berühmtesten offenen Fragen in der theoretischen Informatik.

Schwerste Probleme in NP. Die Theorie der schwersten Probleme in NP beruht auf folgendem Reduktionsbegriff:

Definition 6.2 (polynomielle Reduktion) *Für formale Sprachen L_1, L_2 über einem Alphabet Σ definieren wir: $L_1 \leq_{pol} L_2$ genau dann, wenn eine in Polynomialzeit berechenbare Abbildung $f : \Sigma^* \to \Sigma^*$ existiert mit*

$$\forall w \in \Sigma^* : w \in L_1 \Leftrightarrow f(w) \in L_2 \ .$$

Die Abbildung f wird dann die Reduktionsabbildung *für $L_1 \leq_{pol} L_2$ genannt.*

Bemerkung 6.3 *Die Relation $\leq_{pol}$ hat die folgenden Eigenschaften:*

1. *Sie ist transitiv, d.h., $L_1 \leq_{pol} L_2 \leq_{pol} L_3 \Rightarrow L_1 \leq_{pol} L_3$.*
2. *Aus $L_1 \leq_{pol} L_2$ und $L_2 \in P$ folgt, dass $L_1 \in P$.*
3. *Aus $L_1 \leq_{pol} L_2$ und $L_2 \in RP$ folgt, dass $L_1 \in RP$.*

Beweis

1. Wenn f eine Reduktionsabbildung für $L_1 \leq_{pol} L_2$ und g eine für $L_2 \leq_{pol} L_3$ ist, dann ist $g \circ f$ eine Reduktionsabbildung für $L_1 \leq_{pol} L_3$.
2. Das Problem „$w \in L_1$?“ kann in Polynomialzeit gelöst werden wie folgt:

(a) Berechne $f(w)$.

(b) Setze den Polynomialzeit-Algorithmus für L_2 auf $f(w)$ an und erhalte das Indikatorbit $b = \mathbb{1}_{[w \in L_2]}$.

(c) Gib b als Indikatorbit für „$w \in L_1$“ aus.

Wegen der Äquivalenzen $b = 1 \Leftrightarrow f(w) \in L_2 \Leftrightarrow w \in L_1$ ist der angegebene Algorithmus korrekt. Offensichtlich arbeitet er in Polynomialzeit.

3. Der Beweis der dritten Aussage ist ähnlich zum eben geführten Beweis der zweiten Aussage.

•

Wir kommen nun zum zentralen Begriff der *NP*-Vollständigkeitstheorie:

Definition 6.4 (schwerste Probleme in NP) *Eine formale Sprache L_0 heißt NP-*hart*, falls jede Sprache $L \in NP$ polynomiell auf L_0 reduzierbar ist. Falls zusätzlich $L_0 \in NP$ gilt, dann heißt L_0 NP-*vollständig *oder auch eine* schwerste Sprache *in NP.*

Als unmittelbare Folgerung aus Bemerkung 6.3 und Definition 6.4 erhalten wir:

Bemerkung 6.5 *1. Falls L_1 NP-hart ist und $L_1 \leq_{pol} L_2$, dann ist auch L_2 NP-hart.*

2. Falls L_0 NP-hart ist und $L_0 \in P$, dann gilt $P = NP$.

3. Falls L_0 NP-hart ist und $L_0 \in RP$, dann gilt $RP = NP$.

Beispiel 6.6 (Vertex Cover (VC)) *Es ist bekannt, dass das folgende Problem ein schwerstes Problem in NP ist: gegeben ein Graph $G = (V, E)$ und eine Zahl $k \geq 1$, existiert in G ein „Vertex Cover“ der Größe höchstens k, d.h, existiert eine Teilmenge I von V mit $|I| \leq k$, welche von jeder Kante aus E mindestens einen Randknoten enthält?*

Es ist klar, dass VC ein Mitglied der Klasse NP ist, da die „Ja-Instanzen“ (G, k) ein kurzes leicht überprüfbares Zertifikat besitzen, nämlich das betreffende Vertex Cover I:

- Die natürliche Größe der Eingabe (G, k) ist die Anzahl $n = |V|$ der Knoten im Graphen. Ein Vertex Cover $I \subseteq V$ kann nicht mehr als n Knoten besitzen. Das Zertifikat I ist also kurz.

- Wir können die Menge E kantenweise durchmustern. Für jede Kante $\{i, j\}$ können wir leicht kontrollieren, dass I mindestens einen der Randknoten i, j enthält. Das Zertifikat I ist also leicht überprüfbar.

Warum ist das Problem VC also schwer? Nun die Schwierigkeit besteht darin, ein passendes Zertifikat effizient zu finden. Es gibt $\binom{n}{k}$ mögliche Teilmengen $I \subseteq V$ mit $|I| = k$. Für $k = n/2$ sind das

$$\binom{n}{n/2} \approx \frac{2^n}{\sqrt{n}}$$

(also exponentiell) viele. Ein Verfahren mit einer „exhaustive search“ durch alle Kandidatenmengen würde nicht in Polynomialzeit arbeiten.

6.2 Grundlegende Resultate

Das *Konsistenzproblem* zu $\mathcal{H} = (\mathcal{H}_n)_{n\geq 1}$ ist definiert wie folgt:

als Entscheidungsproblem: Gegeben $S \in \mathcal{Z}_n^m$, existiert eine Hypothese $h \in \mathcal{H}_n$ mit $L_S(h) = 0$?

als Konstruktionsproblem: Gegeben $S \in \mathcal{Z}_n^m$, kläre, ob eine Hypothese $h \in \mathcal{H}_n$ mit $L_S(h) = 0$ existiert und, ggf., konstruiere eine solche.

Der Parameter n ist bei diesem Problem variabel. Eine Hypothese $h \in \mathcal{H}_n$ mit $L_S(h) = 0$ ist eine zu S *konsistente* Hypothese, d.h., eine Hypothese, die auf S fehlerfrei ist. Beachte, dass ein Algorithmus für das Konstruktionsproblem eine ERM-Hypothese konstruiert, sofern eine zu S konsistente Hypothese in $\mathcal{H}_n$ existiert (also etwa für $S \sim \mathcal{D}_n^m$ unter der Realisierbarkeitsannahme). Die Entscheidungsproblem-Variante notieren wir im Folgenden als Kons-E($\mathcal{H}$). Die Konstruktionsproblem-Variante notieren wir als Kons-K($\mathcal{H}$).
Das *Minimum Disagreement Problem* zu $\mathcal{H} = (\mathcal{H}_n)_{n\geq 1}$ ist definiert wie folgt:

als Entscheidungsproblem: Gegeben $S \in \mathcal{Z}_n^m$ und $k \geq 0$, existiert eine Hypothese $h \in \mathcal{H}_n$ mit $L_S(h) \leq k/m$? Mit anderen Worten: existiert eine Hypothese $h \in \mathcal{H}_n$, welche auf S maximal k Fehler macht?

als Konstruktionsproblem: Gegeben $S \in \mathcal{Z}_n^m$ und $k \geq 0$, kläre, ob eine Hypothese $h \in \mathcal{H}_n$ mit $L_S(h) \leq k/m$ existiert und, ggf., konstruiere eine solche.

als Optimierungsproblem: Gegeben $S \in \mathcal{Z}_n^m$, konstruiere eine Hypothese $h \in \mathcal{H}_n$ mit einer kleinstmöglichen empirischen Fehlerrate auf S.

Der Parameter n ist bei diesem Problem wieder variabel. Beachte, dass ein Algorithmus für die Optimierungsvariante eine ERM-Hypothese konstruiert. Die Entscheidungsproblem-Variante notieren wir im Folgenden als MinDis-E($\mathcal{H}$), die Konstruktionsproblem-Variante als MinDis-K($\mathcal{H}$) und die Optimierungsvariante als MinDis-Opt($\mathcal{H}$).

Den Zusammenhang zwischem diesen beiden kombinatorischen Problemen und effizientem PAC-Lernen klären die folgenden beiden Resultate:

Theorem 6.7 *1. Wenn Kons-K($\mathcal{H}$) in Polynomialzeit lösbar ist, dann ist $\mathcal{H}$ unter der Realisierbarkeitsannahme effizient PAC-lernbar.*

2. Wenn $\mathcal{H}$ unter der Realisierbarkeitsannahme effizient PAC-lernbar ist, dann gilt Kons-E($\mathcal{H}$) $\in RP$.

Beweis

1. Der Algorithmus zum Lösen des Problems Kons-K($\mathcal{H}$), angesetzt auf eine Trainingssequenz $S \sim \mathcal{D}_n^m$, konstruiert in Polynomialzeit (unter der Realisierbarkeitsannahme) eine Hypothese h mit $L_S(h) = 0$, also eine ERM-Hypothese. Da ERM, wie wir wissen, zum PAC-Lernen führt, folgt direkt, dass $\mathcal{H}$ unter der Realisierbarkeitsannahme effizient PAC-lernbar ist.

2. Es sei A ein unter der Realisierbarkeitsannahme effizienter PAC-Lernalgorithmus für $\mathcal{H}$. Wir wollen zeigen, dass sich aus A ein Monte-Carlo Algorithmus für das Problem Kons-E($\mathcal{H}$) destillieren lässt. Es sei $S \in \mathcal{Z}_n^m$ die gegebene Menge von markierten Beispielen. Wir spezifizieren einen Algorithmus A' wie folgt:

   (a) Wir setzen $\varepsilon = 1/(m+1)$, $\delta = 1/2$ und wählen $\mathcal{P}_n$ als die uniforme Verteilung auf $S|_{\mathcal{X}}$. $\mathcal{D}_n$ bezeichne die von $\mathcal{P}_n$ und den Markierungen in S induzierte Verteilung auf $S_{\mathcal{X}} \times \{0,1\}$.

   (b) Wir generieren mit Hilfe von S und $\mathcal{P}_n$ eine zufällige Trainingssequenz $T \sim \mathcal{P}_n^{m'}$ der für A erforderlichen Größe m'. Dabei setzen wir zur Generierung von $T|_{\mathcal{X}}$ Zufallsbits ein: jede Instanz in $T|_{\mathcal{X}}$ wird zufällig gemäß $\mathcal{P}_n$ aus $S|_{\mathcal{X}}$ ausgewählt. Die Labels in T werden einfach aus S übernommen.

   (c) Wir setzen den Lernalgorithmus A auf T an und erhalten eine Hypothese h_T.

   (d) Wir testen[13], ob h_T auf S fehlerfrei ist. Falls dem so ist, akzeptieren wir die Eingabeinstanz S. Andernfalls verwerfen wir sie.

   Die zentralen Beobachtungen sind wie folgt:

   (a) Wenn in $\mathcal{H}$ keine zu S konsistente Hypothese existiert, dann wird S niemals (also für keine Wahl der eingesetzten Zufallsbits) akzeptiert.

   (b) Wenn aber in $\mathcal{H}$ eine zu S konsistente Hypothese existiert, dann erfüllen $\mathcal{H}_n$ und $\mathcal{D}_n$ die Realisierbarkeitsannahme und es gilt folgendes:

      i. Mit einer Wahrscheinlichkeit von mindestens $1 - \delta = 1/2$ gilt $L_{\mathcal{D}_n}(h_T) \leq \epsilon = 1/(m+1)$.

      ii. Würde h_T auch nur einen einzigen Fehler auf S machen, dann würde (wegen unserer Wahl von $\mathcal{D}_n$) gelten: $L_{\mathcal{D}_n}(h) \geq 1/m > 1/(m+1)$.

      iii. Insgesamt hat sich also ergeben, dass h_T mit einer Wahrscheinlichkeit von mindestens $1/2$ auf S fehlerfrei ist und daher wird S mit einer Wahrscheinlichkeit von mindestens $1/2$ akzeptiert.

   Aus den Beobachtungen (a) und (b)iii. folgt leicht, dass A' ein Monte-Carlo Algorithmus für Kons-E($\mathcal{H}$) ist.

•

Theorem 6.8 *1. Wenn MinDis-Opt($\mathcal{H}$) in Polynomialzeit lösbar ist, dann ist $\mathcal{H}$ effizient agnostisch PAC-lernbar.*

2. Wenn $\mathcal{H}$ effizient agnostisch PAC-lernbar ist, dann gilt MinDis-E($\mathcal{H}$) $\in RP$.

[13] Hier geht die Voraussetzung ein, dass das Auswertungsproblem zu $\mathcal{H}$ polynomiell lösbar ist.

Den Beweis zu Theorem 6.8, der völlig analog zum Beweis von Theorem 6.7 zu führen ist, lassen wir aus.

Folgerung 6.9 *1. Wenn Kons-E($\mathcal{H}$) NP-hart ist, dann ist $\mathcal{H}$ nicht einmal unter der Realisierbarkeitsannahme effizient PAC-lernbar (außer wenn $RP = NP$).*

2. Wenn MinDis-E($\mathcal{H}$) NP-hart ist, dann ist $\mathcal{H}$ nicht effizient agnostisch PAC-lernbar (außer wenn $RP = NP$).

Beweis Wir beschränken uns auf den Beweis der ersten Aussage, da der Beweis der zweiten Aussage völlig analog geführt werden kann. Es genügt zu zeigen, dass aus der NP-Härte von Kons-E($\mathcal{H}$) und der effizienten PAC-Lernbarkeit von $\mathcal{H}$ unter der Realisierbarkeitsannahme die Aussage $RP = NP$ gefolgert werden kann. Die effiziente PAC-Lernbarkeit von $\mathcal{H}$ unter der Realisierbarkeitsannahme impliziert gemäß Theorem 6.7, dass Kons-E($\mathcal{H}$) $\in RP$. Wenn aber ein einziges NP-hartes Problem, wie zum Beispiel Kons-E($\mathcal{H}$), zur Komplexitätsklasse RP gehört, dann ergibt sich mit der dritten Aussage in Bemerkung 6.5, dass $RP = NP$.

•

Die Resultate in diesem Abschnitt haben ergeben, dass sich die Frage der *effizienten* PAC-Lernbarkeit einer binären Hypothesenklasse anhand der Komplexität zweier rein kombinatorischer Probleme entscheiden lässt, nämlich anhand des zu $\mathcal{H}$ assoziierten Konsistenz- und anhand des zu $\mathcal{H}$ assoziierten Mininimum Disagreement Problems.

6.3 Ein Zoo von Booleschen Hypothesenklassen

Wir definieren zunächst ein paar spezielle Boolesche Formeln, die mit den Operationen $\vee$ (für das logische Oder), $\wedge$ (für das logische Und) über n Booleschen Variablen $v_1, \ldots, v_n$ und ihren Negationen $\overline{v_1}, \ldots, \overline{v_n}$ gebildet werden:

Definition 6.10 *Es seien $v_1, \ldots, v_n$ Boolesche Variablen, die mit den Werten 0 (für FALSE) und 1 (für TRUE) belegt werden können. Ein* Literal *ist per Definition eine negierte oder unnegierte Boolesche Variable, d.h., ein Literal hat die Form $\overline{v_i}$ oder v_i. Eine* Klausel *ist eine „Veroderung“ von Literalen, also ein Ausdruck der Form $\ell_1 \vee \ldots \vee \ell_k$ mit $k \geq 0$, wobei $\ell_1, \ldots, \ell_k$ Literale sind. Ein* Term *ist eine „Verundung“ von Literalen, also ein Ausdruck der Form $\ell_1 \wedge \ldots \wedge \ell_k$ mit $k \geq 0$, wobei $\ell_1, \ldots, \ell_k$ Literale sind. Eine k-*Klausel *ist eine Klausel, in welcher maximal k Literale vorkommen. Eine* monotone Klausel *ist eine Klausel, in welcher ausschließlich unnegierte Variable vorkommen. Die Begriffe k-*Term *und* monotoner Term *sind analog zu verstehen. Eine* CNF-Formel *ist eine Verundung von Klauseln. „CNF“ steht dabei für „Conjunctive Normal Form“. Eine* DNF-Formel *ist eine Veroderung von Termen. „DNF“ steht dabei für „Disjunctive Normal Form“. Eine k-*CNF Formel *ist eine CNF-Formel in der ausschließlich k-Klauseln verundet werden. Eine k-*DNF Formel *ist eine DNF-Formel in der ausschließlich k-Terme verodert werden.*

Diese Typen von Booleschen Formeln führen zu den folgenden binären Hypothesenklassen:

Definition 6.11 *$k-\mathrm{CNF}_n$ ist die Klasse aller Booleschen k-CNF Formeln über den Variablen $v_1, \ldots, v_n$. Weiter sei $k-\mathrm{CNF} = (k-\mathrm{CNF}_n)_{n\geq 1}$. $k-\mathrm{DNF}_n$ ist die Klasse aller Booleschen k-DNF Formeln über den Variablen $v_1, \ldots, v_n$. Weiter sei $k-\mathrm{DNF} = (k-\mathrm{DNF}_n)_{n\geq 1}$. $k-\mathrm{clause}-\mathrm{CNF}_n$ ist die Klasse aller CNF-Formeln, in denen maximal k viele Klauseln verundet werden. $k-\mathrm{term}-\mathrm{DNF}_n$ ist die Klasse aller DNF-Formeln, in denen maximal k viele Terme verodert werden. Weiter sei $k-\mathrm{clause}-\mathrm{CNF} = (k-\mathrm{clause}-\mathrm{CNF}_n)_{n\geq 1}$ und $k-\mathrm{term}-\mathrm{DNF} = (k-\mathrm{term}-\mathrm{DNF}_n)_{n\geq 1}$.*

Die Klasse 1-CNF heißt auch auch die Klasse der *Booleschen Konjunktionen*. Sie enthält Formeln der Form $\ell_1 \wedge \ldots \wedge \ell_k$ für $k \geq 0$ viele Literale $\ell_1, \ldots, \ell_k$.

Jede Belegung der Booleschen Variablen $v_1, \ldots, v_n$ mit 0 oder 1 führt auf die offensichtliche Weise zu einem $0,1$-Wert einer über $v_1, \ldots, v_n$ gebildeten Formel. Dabei sind lediglich folgende Auswertungsregeln zu berücksichtigen:

1. Aus $v_i = 0$ folgt $\overline{v_i} = 1$ und aus $v_i = 1$ folgt $\overline{v_i} = 0$.

2. Eine Formel F der Form $F_1 \vee \ldots \vee F_k$ hat den Wert 1 genau dann, wenn mindestens eine der Formeln $F_1, \ldots, F_k$ den Wert 1 liefert. Dies impliziert $F = 0$ für den pathologischen Grenzfall $k = 0$.

3. Eine Formel F der Form $F_1 \wedge \ldots \wedge F_k$ hat den Wert 1 genau dann, wenn alle Formeln $F_1, \ldots, F_k$ den Wert 1 liefern. Dies impliziert $F = 1$ für den pathologischen Grenzfall $k = 0$.

Somit kann jede Boolesche Formel F über $v_1, \ldots, v_n$ als eine Boolesche Funktion $F : \{0,1\}^n \to \{0,1\}$ interpretiert werden und die in Definition 6.11 aufgeführten Formelklassen repräsentieren binäre Hypothesenklassen über dem Booleschen Grundbereich.

Definition 6.12 *Eine* Boolesche 1-Entscheidungsliste *über den Variablen $v_1, \ldots, v_n$ ist eine Liste der Form*

$$(\ell_1, b_1), \ldots, (\ell_k, b_k), (\ell_{k+1}, b_{k+1}) \ , \tag{28}$$

wobei $k \geq 0$, $b_1, \ldots, b_k, b_{k+1} \in \{0,1\}$, $\ell_1, \ldots, \ell_k$ sind Literale und $\ell_{k+1} = 1$. Die Paare (ℓ_i, b_i) heißen die Einträge *der Liste. Das spezielle Paar (ℓ_{k+1}, b_{k+1}) heißt der* Schlusseintrag. *Eine 1-Entscheidungsliste heißt* negationsfrei, *wenn alle in ihr enthaltenen Literale unnegierte Boolesche Variable sind. Es bezeichne $1-\mathrm{DL}_n$ die Klasse aller 1-Entscheidungslisten über $v_1, \ldots, v_n$ und $1-\mathrm{DL} = (1-\mathrm{DL}_n)_{n\geq 1}$ bezeichne die zugehörige parametrisierte Hypothesenklasse.*

Wir sagen ein Vektor $\mathbf{a} \in \{0,1\}^n$ wird von dem Paar (v_j, b) *absorbiert*, wenn $a_j = 1$. Er wird von dem Paar $(\overline{v_j}, b)$ absorbiert, wenn $a_j = 0$. Er wird also von einem Paar (ℓ, b) mit Literal ℓ absorbiert, wenn ℓ (interpretiert als Boolesche Formel) an der Stelle $\mathbf{a}$ zu 1

ausgewertet wird. Wir legen fest: ein Paar der Form $(1, b)$ mit $b \in \{0, 1\}$ absorbiert jeden Vektor $\mathbf{a} \in \{0, 1\}^n$.

Eine Entscheidungsliste L der Form (28) wird als Boolesche Funktion $L : \{0, 1\}^n \to \{0, 1\}$ interpretiert wie folgt:

1. Zu gegebenem $\mathbf{a} \in \{0, 1\}^n$ bestimme den frühesten Eintrag in L, sagen wir den Eintrag (ℓ_i, b_i), in welchem $\mathbf{a}$ absorbiert wird. (Wegen $\ell_{k+1} = 1$ ist dies spätestens für $i = k+1$ der Fall.)

2. Setze dann $L(\mathbf{a}) = b_i$.

Die Klasse der k-Entscheidungslisten, notiert als $k-$DL, ist analog zu $1 -$ DL definiert mit dem Unterschied, dass die Einträge die Form (M, b) haben, wobei M ein k-Term ist (also ein Term, der maximal k Literale enthält).

6.4 Elementare effizient lösbare Lernprobleme

Eine Familie F von Teilmengen einer Grundmenge M ist durch die Teilmengenrelation $\subseteq$ partiell geordnet. Wenn F abgeschlossen unter der Durchschnittsbildung $\cap$ ist, dann existiert zu jeder Teilmenge $S \subseteq M$, die in mindestens einer Menge aus F enthalten ist, eine eindeutige kleinste S enthaltende Teilmenge, nämlich $\cap_{T \in F: S \subseteq T} T$.

Wie wir wissen können wir jede Teilmenge von M mit ihrer Indikatorfunktion identizieren, d.h., wir können eine Familie F solcher Mengen auch als eine Familie von auf M definierten $0, 1$-wertigen Funktionen auffassen. Die Teilmengenrelation entspricht dann der folgenden Relation $\leq$:

$$f \leq g \Leftrightarrow (\forall x \in M : f(x) = 1 \Rightarrow g(x) = 1) \ .$$

Die Durchschnittsabgeschlossenheit entspricht nun der Abgeschlossenheit bezüglich Verundung $\wedge$.

Wir können diese Beobachtungen auf Hypothesenklassen $\mathcal{H}_n$ über einem Grundbereich $\mathcal{X}_n$ anwenden, sofern $\mathcal{H}_n$ bezüglich $\cap$ bzw. $\wedge$ abgeschlossen ist. Unter der Realisierbarkeitsannahme eröffnet dies die Möglichkeit, zu einer Trainingssequenz $S \in \mathcal{Z}_n^m$ eine sehr spezielle ERM-Hypothese zu assoziieren:

- Es sei $S_+ = \{a |\ (a, 1) \in S\}$ die Menge aller positiven Beispiele in S. Die Menge S_- aller negativen Beispiele aus S ist analog definiert.

- Wähle die kleinste Hypothese aus $\mathcal{H}_n$, im Folgenden notiert als $\langle S_+ \rangle$, die auf S_+ fehlerfrei ist (d.h., die alle Instanzen in S_+ mit 1 markiert).

Beachte, dass $\langle S_+ \rangle$ nicht nur auf S_+, sondern auch auf S_- (und somit auf ganz S) fehlerfrei sein muss:
Unter der Realisierbarkeitsannahme existiert eine zu S konsistente Hypothese $h \in \mathcal{H}$. Da diese insbesondere auf S_+ fehlerfrei sein muss, folgt wegen der Minimalität von $\langle S_+ \rangle$, dass $\langle S_+ \rangle \leq h$. Da h Punkte aus S_- mit 0 klassifiziert, muss dies erst recht für $\langle S_+ \rangle$ gelten.

Algorithmen, die aus S die Hypothese $\langle S_+ \rangle$ berechnen, heißen „Closure Algorithmen“. Unsere kleine Diskussion lässt sich zusammenfassen wie folgt:

- Wenn alle Klassen $\mathcal{H}_n$ von $\mathcal{H} = (\mathcal{H}_n)_{n \geq 1}$ abgeschlossen unter $\cap$ bzw. $\wedge$ sind, und der Closure Algorithmus für $\mathcal{H}$ in Polynomialzeit implementierbar ist, dann ist das Problem Kons-K($\mathcal{H}$) in Polynomialzeit lösbar.

Dieses Argumentationsmuster wird uns gleich helfen, die erste Aussage des folgenden Resultates zu beweisen:

Theorem 6.13 *Das Problem Kons-K($\mathcal{H}$) ist für folgende Klassen $\mathcal{H}$ in Polynomialzeit lösbar:*

- *die Klasse der monotonen Booleschen Konjunktionen (also $\mathcal{H}$ = Monotone 1-CNF),*
- *die Klasse der negationsfreien 1-Entscheidungslisten (also $\mathcal{H}$ = 1-NF-DL).*

Beweis Wir betrachten zunächst das Konsistenzproblem für monotone Boolesche Konjunktionen. Eine monotone Boolesche Konjunktion über den Variablen $v_1, \ldots, v_n$ hat die allgemeine Form $F_I = \wedge_{i \in I} v_i$. Wegen

$$\bigwedge_{i \in I} v_i \wedge \bigwedge_{j \in J} v_j = \bigwedge_{k \in I \cup J} v_k$$

sind monotone Boolesche Konjunktionen abgeschlossen unter $\wedge$. Wir müssen also lediglich zeigen, dass der Closure-Algorithmus für diese Klasse effizient implementiert werden kann. Beachte dazu, dass

$$F_I \leq F_J \Leftrightarrow J \subseteq I \ .$$

Die kleinste auf S_+ konsistente Hypothese aus Monotone 1-CNF ist daher gegeben durch die bezüglich $\subseteq$ größte Indexmenge $J \subseteq [n]$ einer auf S_+ fehlerfreien Hypothese F_J. Diese Menge kann wie folgt berechnet werden:

$$I := \{i \in [n] |\ \exists \mathbf{a} \in S_+ : a_i = 0\} \ ; \ J := [n] \setminus I \ .$$

J entsteht, indem aus $[n]$ alle Indizes i entfernt werden, die zu einem Fehler auf S_+ führen würden. Da Indizes gleichsam nur unter Zugzwang entfernt werden, hat J die gewünschte Maximalitätseigenschaft.
Wir setzen den Beweis nun mit der Klasse der negationsfreien 1-Entscheidungslisten fort. Gegeben sei eine Sequenz $S \in \mathcal{Z}_n^m$ markierter Beispiele. Es sei T eine Teilsequenz von S. Wir sagen T ist *b-rein* mit $b \in \{0, 1\}$, wenn alle Beispiele in T das Label b haben. T heißt *rein*, wenn alle Beispiele in T das gleiche Label haben. Es sei $T_j = \{(\mathbf{a}, b) \in T |\ a_j = 1\}$. Das folgende Verfahren bestimmt (wie wir noch zeigen werden) eine zu S konsistente 1-Entscheidungsliste, sofern eine solche überhaupt existiert:

1. Initialisiere L als leere Entscheidungsliste (ohne Einträge) und die Menge T als S.

2. Solange T nicht rein ist, durchlaufe folgende Schleife:

   (a) Finde, falls möglich, ein Paar (j, b) mit: T_j ist b-rein und $T_j \neq \emptyset$. Falls jedoch kein solches Paar existiert, dann gib die Meldung „Es gibt für S keine konsistente negationsfreie 1-Entscheidungsliste“ aus und stoppe.

   (b) Erweitere L um den Eintrag (v_j, b) und setze $T := T \setminus T_j$.

3. Finde b mit T ist b-rein, erweitere L um den Eintrag $(1, b)$ und gib diese Liste dann aus.

Es ist offensichtlich, dass eine zu S konsistente Liste L ausgegeben wird, falls in Schritt 2(a) stets ein passendes Paar (j, b) gefunden wird. Es ist demnach noch Folgendes zu zeigen:

Behauptung: Falls eine zu S konsistente negationsfreie 1-Entscheidungsliste L^* existiert, dann existiert bei jeder Ausführung von Schritt 2(a) ein passendes Paar (j, b).

Die Variable T des geschilderten Verfahrens enthält zu jedem Zeitpunkt die Beispiele aus S, deren Instanzen bisher nicht von L absorbiert werden. Es bezeichne L^+ die Liste L^* abzüglich ihres Schlusseintrages. Falls jeder Eintrag von L^+ auch in L vorkommen würde, dann würde L mindestens soviele Vektoren aus S absorbieren wie die Liste L^+. Dann müsste aber T zu diesem Zeitpunkt rein sein, so dass die Abbruchbedingung der Schleife erreicht und somit Schritt 2 verlassen würde. Schritt 2(a) würde dann nicht mehr erreicht. Wir dürfen daher annehmen, dass beim Erreichen von Schritt 2(a) die Liste L^+ einen Eintrag besitzt, der nicht bereits in L vorkommt. Es sei (v_j, b) der früheste solche Eintrag in L^+. Es sei T' die Menge der Beispiele aus S, deren Instanzen diesen Eintrag erreichen (also nicht vorher schon von L^+ absorbiert wurden). Dann ist T'_j natürlich b-rein. Da alle Einträge vor (v_j, b) in L^+ auch in L vorkommen, gilt $T \subseteq T'$. Daher ist T_j ebenfalls b-rein. Somit ist (j, b) ein passendes Paar, was den Beweis der Behauptung (und damit auch des Theorems) abschließt. •

Das Minimum Disagreement Problem ist leider schon für einfache Hypothesenklassen NP-hart, es sei denn man friert den Komplexitätsparameter n auf eine Konstante ein. Wir diskutieren im Folgendem ein konkretes Beispiel.

Definition 6.14 (achsenparallele Quader) *Es sei $\mathcal{B}_n$ die Klasse der achsenparallelen Quader, kurz n-*Boxen *genannt, über dem Grundbereich $\mathcal{X}_n = \mathbb{R}^n$. Weiter bezeichne $\mathcal{B}$ die zugehörige parameterisierte Klasse, d.h., $\mathcal{B} = (\mathcal{B}_n)_{n \geq 1}$.*

Beachte, dass die 2-Boxen gerade die achenparallelen Rechtecke in der euklidischen Ebene $\mathbb{R}^2$ sind.

Es ist nicht schwer zu zeigen, dass Kons-E($\mathcal{B}$) NP-hart ist. Hingegen ist sogar MinDIS-Opt($\mathcal{B}_n$) für eine *konstante Wahl des Komplexitätsparameters n* in Polynomialzeit lösbar. Wir illustrieren, das im Falle $n = 2$:

Lemma 6.15 *Das Problem MinDis-Opt($\mathcal{B}_2$) ist in Polynomialzeit lösbar.*

Beweis Gegeben sei eine Sequenz $S \in (\mathbb{R}^2 \times \{0,1\})^m$ markierter Beispiele in der euklidischen Ebene. Es bezeichne S_+ die Menge der positiven (also mit 1 markierten) Beispielinstanzen in S. Es ist leicht zu argumentieren, dass eine 2-Box B, welche keinen Punkt aus S_+ enthält, kein Minimierer von $L_S(B)$ sein kann. Jede 2-Box B, welche mindestens einen Punkt aus S_+ enthält, können wir, ohne $L_S(B)$ zu vergrößern, so schrumpfen lassen, dass jede der vier Begrenzungskanten von B mindestens einen der Punkte aus S_+ enthält (kann im Extremfall 4-mal der selbe Punkt sein). Jede auf diese Weise normalisierte 2-Box ist auf die offensichtliche Art durch ein Quadrupel aus S_+^4 gegeben. Wir erhalten dann eine 2-Box mit kleinstmöglicher empirischer Fehlerrate auf S wie folgt:

1. Zu jedem Quadrupel $Q \in S_+^4$ bestimme die kleinste Q enthaltende 2-Box B_Q sowie ihre empirische Fehlerrate $L_S(B_Q)$.
2. Wähle ein Quadrupel Q^* aus, welches $L_S(B_Q)$ minimiert, und gib Q^* (als Repräsentation von B_{Q^*}) aus.

Die Laufzeit wird dominiert von der Anzahl aller Quadrupel aus S_+^4, maximal m^4 an der Zahl. •

Bemerkung 6.16 *1. Mit einem weniger naiven Algorithmus für das Problem MinDis-Opt($\mathcal{B}_2$) lässt sich (unter Verwendung dynamischer Programmierung) die Laufzeit verbessern von $O(m^4)$ auf $O(m^2 \log(m))$.*

2. Die Laufzeit eines naiven Algorithmus für das Problem MinDis-Opt($\mathcal{B}_n$) würde $O(m^{2n})$ betragen. Für ein konstant gehaltenes n ist dies immer noch ein Polynom in $m = |S|$.

6.5 Komplexere effizient lösbare Lernprobleme

Folgende Reduktionstechnik wird uns helfen, aus der effizienten Lernbarkeit einfacher Basisklassen, zuweilen die Lernbarkeit komplexerer Klassen zu folgern:

Definition 6.17 (polynomielle Lernreduktionen) *Es sei $\mathcal{H} = (\mathcal{H}_n)_{n\geq 1}$ bzw. $\mathcal{H}' = (\mathcal{H}'_n)_{n\geq 1}$ eine binäre Hypothesenklasse über dem Grundbereich $\mathcal{X} = (\mathcal{X}_n)_{n\geq 1}$ bzw. $\mathcal{X}' = (\mathcal{X}'_n)_{n\geq 1}$. Wir sagen $\mathcal{H}$ ist auf $\mathcal{H}'$* polynomiell L-reduzierbar*, notiert als $\mathcal{H} \leq_{pol}^{L} \mathcal{H}'$, wenn eine injektive und polynomiell beschränkte Abbildung $n \mapsto n'$ und für alle $n \geq 1$ Abbildungen $f_n : \mathcal{X}_n \to \mathcal{X}'_{n'}$, $g_n : \mathcal{H}_n \to \mathcal{H}'_{n'}$ und $g'_n : \mathcal{H}'_{n'} \to \mathcal{H}_n$ existieren, so dass folgendes gilt:*

1. *Für alle $n \geq 1$, alle $x \in \mathcal{X}_n$, alle $h \in \mathcal{H}_n$ sowie $x' = f_n(x)$ und $h' = g_n(h)$ gilt: $h(x) = h'(x')$.*
2. *Für alle $n \geq 1$, alle $x \in \mathcal{X}_n$, alle $h' \in \mathcal{H}'_{n'}$ sowie $x' = f_n(x)$ und $h = g'_n(h')$ gilt: $h'(x') = h(x)$.*

3. *Es seien* $f : \cup_{n\geq 1}\mathcal{X}_n \to \cup_{n'\geq 1}\mathcal{X}'_{n'}$, $g : \cup_{n\geq 1}\mathcal{H}_n \to \cup_{n'\geq 1}\mathcal{H}'_{n'}$ *und* $g' : \cup_{n'\geq 1}\mathcal{H}'_{n'} \to \cup_{n\geq 1}\mathcal{H}_n$ *die Abbildungen, deren Restriktionen auf* $\mathcal{X}_n$ *bzw.* $\mathcal{H}_n$ *oder* $\mathcal{H}'_{n'}$ *mit den Abbildungen* f_n, g_n, g'_n *übereinstimmen. Dann ist jede dieser drei Abbildungen in Polynomialzeit berechenbar.*

Die Abbildung f *wird als* Instanzentransformation *und die Abbildungen* g *und* g' *werden als* (erste und zweite) Hypothesentransformationen *bezeichnet.*

Die erste Definitionsbedingung sagt im Prinzip aus, dass eine simultane Anwendung von Instanzen- und erster Hypothesentransformation zur selben Markierung (mit 0 oder 1) führt. Die zweite Definitionsbedingung macht eine analoge Aussage für die zweite Hypothesentransformation an Stelle der ersten. Definition 6.17 ist motiviert durch die folgenden beiden Resultate:

Theorem 6.18 *Unter der Voraussetzung* $\mathcal{H} \leq^L_{pol} \mathcal{H}'$ *gilt folgendes:*

1. *Wenn Kons-E*$(\mathcal{H})$ *NP-hart ist, so gilt dies auch für Kons-E*$(\mathcal{H}')$*.*
2. *Wenn Kons-K*$(\mathcal{H}')$ *in Polynomialzeit lösbar ist, so gilt dies auch für Kons-K*$(\mathcal{H})$*.*

Beweis

1. Die Instanzentransformation f lässt sich wie folgt zu einer Beispielsequenzentransformation f^* fortsetzen:

$$S = [(x_1, b_1), \ldots, (x_m, b_m)] \mapsto S' = f^*(S) = [(x'_1, b_1), \ldots, (x'_m, b_m)] \ .$$

   Hierbei gilt $x'_i = f(x_i)$. Mit anderen Worten: wir wenden die Instanzentransformation f auf alle Beispielinstanzen x_i in S an und übernehmen alle Labels b_i aus S, ohne sie zu ändern. Es sei n die natürliche Zahl mit $S \in \mathcal{Z}^m_n$. Die ersten beiden Bedingungen in Definition 6.17 bewirken, dass es zu jeder Hypothese $h \in \mathcal{H}_n$ eine Hypothese $h' \in \mathcal{H}'_{n'}$ mit $L_{S'}(h') = L_S(h)$ gibt, und umgekehrt. Insbesondere enthält $\mathcal{H}_n$ genau dann eine zu S konsistente Hypothese, wenn $\mathcal{H}'_{n'}$ eine zu S' konsistente Hypothese enthält. Die dritte Bedingung in Definition 6.17 bewirkt, dass sich die Abbildung $S \mapsto f^*(S) = S'$ in Polynomialzeit berechnen lässt. Damit ist f^* eine Reduktionsabbildung für Kons-E$(\mathcal{H}) \leq_{pol}$ Kons-E$(\mathcal{H}')$ und NP-Härte von Kons-E$(\mathcal{H})$ vererbt sich auf Kons-E$(\mathcal{H}')$.

2. Es sei A' ein Algorithmus, der in Polynomialzeit testet, ob zu einer gegebenen Sequenz $S' \in (\mathcal{Z}'_n)^m$ eine konsistente Hypothese $h' \in \mathcal{H}'_n$ existiert und diese ggf. konstruiert. Wir transformieren A' in folgenden Algorithmus A:

   (a) Gegeben eine Sequenz $S \in \mathcal{Z}^m_n$ berechne die Sequenz $S' = f^*(S) \in (\mathcal{Z}'_{n'})^m$.

   (b) Setze A' auf S' an und erhalte entweder (Fall 1) eine Meldung „Es existiert keine zu S' konsistente Hypothese in $\mathcal{H}'_{n'}$“ oder (Fall 2) eine zu S' konsistente Hypothese $h' \in \mathcal{H}'_{n'}$.

(c) Im Fall 1 mache eine entsprechende Meldung für S und $\mathcal{H}_n$. Im Fall 2 ermittle die Hypothese $h = g'_n(h')$.

Beachte, dass h zu S konsistent ist, da h' zu S' konsistent ist. Es ist offensichtlich, dass der Algorithmus A das Problem Kons-K($\mathcal{H}$) in Polynomialzeit löst.

•

Den Beweis für das folgende Resultat, der völlig analog zum Beweis von Theorem 6.18 zu führen ist, lassen wir aus:

Theorem 6.19 *Unter der Voraussetzung $\mathcal{H} \leq^L_{pol} \mathcal{H}'$ gilt folgendes:*

1. *Wenn MinDis-E($\mathcal{H}$) NP-hart ist, so gilt dies auch für MinDis-E($\mathcal{H}'$).*
2. *Wenn MinDis-Opt($\mathcal{H}'$) in Polynomialzeit lösbar ist, so gilt dies auch für MinDis-Opt($\mathcal{H}$).*

Wir wenden unsere Reduktionstechnik jetzt auf zwei konkrete Hypothesenklassen an:

Theorem 6.20 $k-$CNF $\leq^L_{pol}$ *Monotone* $1-$CNF.

Beweis Wir erinnern daran, dass $k-\text{CNF}_n$ die Klasse aller Verundungen von Klauseln mit maximal k Literalen bezeichnet. Die Klasse Monotone 1-CNF_n enthält alle Verundungen von (unnegierten) Booleschen Variablen. In beiden Fällen werden die Formeln über n Booleschen Variablen $v_1, \ldots, v_n$ gebildet. Es sei $C_1, \ldots, C_{n'}$ eine (beliebige aber feste) Aufzählung aller Klauseln mit höchstens k Literalen. Einfache Kombinatorik liefert

$$n' = \sum_{i=0}^{k} \binom{n}{i} 2^i = O(n^k) \ .$$

Offensichtlich ist die Abbildung $n \mapsto n'$ injektiv und polynomiell in n beschränkt. Es seien $v'_1, \ldots, v'_{n'}$ neue Boolesche Variable. Zu $\mathbf{a} \in \{0,1\}^n$ und $C \in \{C_1, \ldots, C_{n'}\}$ bezeichne $C(\mathbf{a}) \in \{0,1\}$ die Auswertung der Klausel C für die Belegung $\mathbf{a}$ der Booleschen Variablen $v_1, \ldots, v_n$. Dann verwenden wir die Abbildung

$$\mathbf{a} \mapsto (C_1(\mathbf{a}), \ldots, C_{n'}(\mathbf{a}))$$

als Instanzentransformation. Als erste Hypothesentransformation verwenden wir die Abbildung

$$\wedge_{j=1}^{r} C_{i_j} \mapsto \wedge_{j=1}^{r} v'_{i_j} \ ,$$

d.h., wir ersetzen jede Klausel C_i, die in einer Formel aus $k-\text{CNF}_n$ vorkommt, durch die Boolesche Variable v'_i. Die umgekehrte Substitution, also Ersetzung von v'_i durch C_i liefert die zweite Hypothesentransformation. Man überzeugt sich leicht davon, dass die Bedingungen der Definition 6.17 erfüllt sind. •

Da wir bereits wissen, dass das Problem Kons-K($\mathcal{H}$) für die Klasse $\mathcal{H}$ der monotonen 1-CNF Formeln in Polynomialzeit lösbar ist, ergibt sich aus Theorem 6.18:

Folgerung 6.21 *1. Kons-K($k-$CNF) ist in Polynomialzeit lösbar.*

2. $k-$CNF ist unter der Realisierbarkeitsannahme effizient PAC-lernbar.

Den Beweis für folgendes Resultat, der sehr ähnlich zum Beweis von Theorem 6.20 ist, lassen wir aus:

Theorem 6.22 $k-\mathrm{DL} \leq_{pol}^{L} NF\ 1-\mathrm{DL}$.

Da wir bereits wissen, dass das Problem Kons-K($\mathcal{H}$) für die Klasse $\mathcal{H} = \mathrm{NF}\ 1-\mathrm{DL}$ in Polynomialzeit lösbar ist, ergibt sich aus Theorem 6.18:

Folgerung 6.23 *1. Kons-K($k-$DL) ist in Polynomialzeit lösbar.*

2. $k-$DL ist unter der Realisierbarkeitsannahme effizient PAC-lernbar.

6.6 Inhärent harte Lernprobleme

Die NP-Härte des Konsistenzproblems bzw. des Minimum Disagreement Problems zu einer binären Hypothesenklasse $\mathcal{H}$ ist, so wie wir es in Folgerung 6.9 präzisiert haben, eine Barriere auf dem Weg zum effizienten PAC-Lernen. Sie wird auch als die „komplexitätstheoretische Barriere“ bezeichnet. Diese Barriere lässt sich manchmal dadurch umgehen, dass wir dem Lerner erlauben, seine Hypothese aus einer Klasse $\mathcal{H}'$ zu wählen, welche ausdrucksstärker ist als die zu lernende Klasse $\mathcal{H}$. In Abschnitt 6.6.1 beschäftigen wir uns mit dieser Thematik. Es gibt jedoch eine gravierendere Barriere ohne diese Umgehungsmöglichkeit. Sie wird die „kryptographische Barriere“ genannt wird. Eine solche Barriere liegt vor, wenn wir zeigen können, dass die Abgabe einer Hypothese mit kleiner Fehlerrate dazu führt, dass ein als sicher geltendes Kryptosystem geknackt werden kann (zum Beispiel, indem wir eine Chiffre mit einem Bit des zugehörigen geheimen Klartextes labeln und diese Klassifikationslabels aus Beispielen lernen). Damit beschäftigen wir uns im Abschnitt 6.6.2.

6.6.1 Komplexitätstheoretische Barrieren

Wie früher bereits erwähnt sind Minimum Disagreement Probleme leider bereits für einfache Hypothesenklassen NP-hart. Wir illustrieren das anhand der Klasse Monotone 1-CNF:

Theorem 6.24 *MinDis-E(Monotone $1-\mathrm{CNF}$) ist NP-hart.*

Beweis Es sei VC das (als NP-hart bekannte) Vertex Cover Problem. Wir geben eine Reduktionsabbildung für VC $\leq_{pol}$MinDis-E(Monotone $1-$CNF) an, woraus sich dann das Theorem unmittelbar erschließt. Eine Eingabeinstanz für VC besteht aus einem Graphen $G = (V, E)$ und einer Zahl $k \geq 1$. Wir setzen $n = |V|$ und setzen oBdA voraus, dass $V = [n]$, d.h., wir identifizieren die n Knoten von G mit den Nummern von 1 bis n. Die Frage lautet, ob eine Teilmenge $K \subseteq [n]$ mit $|K| \leq k$ existiert, die ein Vertex Cover ist, d.h., die von jeder

Kante in E mindestens einen Randknoten enthält. Die Reduktionsabbildung soll daraus ein äquivalentes Minimum Disagreement Problem für die Klasse der monotonen 1-CNF Formeln machen. Für $K \subseteq [n]$ bezeichne $\mathbf{0_K}$ den Vektor aus $\{0,1\}^n$ der in den Positionen aus K Nulleinträge und außerhalb von K 1-Einträge hat. Statt $\mathbf{0_{\{i\}}}$ bzw. $\mathbf{0_{\{i,j\}}}$ schreiben wir aber einfach $\mathbf{0_i}$ bzw. $\mathbf{0_{i,j}}$. Wir bilden dann das Paar (G,k) auf folgende Eingabeinstanz (S,k) unseres Minimum Disagreement Problemes ab:

Knoten-Beispiele: Die Vektoren $\mathbf{0_1}, \ldots, \mathbf{0_n}$ erklären wir zu positiven Beispielen, d.h., sie erhalten das Label 1.

Kanten-Beispiele: Für jede Kante $\{i,j\} \in E$, erklären wir den Vektor $\mathbf{0_{i,j}}$ zu einem negativen Beispiel, d.h., er erhält das Label 0.

Die Menge S bestehe genau aus diesen Knoten- bzw. Kanten-Beispielen. Es genügt zu zeigen, dass folgendes gilt:

Behauptung: Es gibt genau dann ein Vertex Cover der Größe höchstens k in G, wenn es eine monotone 1-CNF Formel mit Booleschen Variablen $v_1, \ldots, v_n$ gibt, die auf S höchstens k Fehler macht.

Eine monotone 1-CNF Formel über den Variablen $v_1, \ldots, v_n$ hat die allgemeine Form $F_K = \wedge_{k \in K} v_k$ für eine Indexmenge $K \subseteq [n]$. Der Beweis der Behauptung basiert auf den folgenden zentralen Beobachtungen:

1. Der Vektor $\mathbf{0_i}$ wird von F_K genau dann korrekt gelabelt (also zu 1 ausgewertet), wenn $i \notin K$. Somit macht F_K auf den positiven Beispielen in S genau $|K|$ Fehler.

2. Ein Kantenbeispiel $\mathbf{0_{i,j}}$ wird von F_K genau dann korrekt gelabelt (also zu 0 ausgewertet), wenn $K \cap \{i,j\} \neq \emptyset$, d.h., wenn K mindestens einen Randknoten der Kante $\{i,j\}$ enthält.

Wir betrachten als erstes die Beweisrichtung „$\Rightarrow$“. Es sei also $K \subseteq [n]$ ein Vertex Cover in G. Aus unseren obigen zentralen Beobachtungen folgt sofort, dass F_K keine Fehler auf den Kantenbeispielen und $|K|$ Fehler auf den Knotenbeispielen macht, d.h., die Anzahl der Fehler von F_K ist nicht größer als die Anzahl $|K|$ der Knoten im Vertex Cover K.
Kommen wir schließlich zu der Beweisrichtung „$\Leftarrow$“. Es sei also eine Formel F_K mit $K \subseteq [n]$ gegeben, die auf S höchstens k Fehler macht. Falls ein Fehler auf einem Kanten-Beispiel $\mathbf{0_{i,j}}$ vorliegt, dann wird dieses von F_K fälschlicherweise mit 1 gelabelt. Dies ist nur dann möglich, wenn $i,j \notin K$. Wenn wir nun i in K aufnehmen, dann wird ein neuer Fehler auf $\mathbf{0_i}$ erzeugt (Label 0 statt Label 1), aber dafür wird $\mathbf{0_{i,j}}$ jetzt richtig mit 0 gelabelt. Die neue Formel hat also keine höhere empirische Fehlerrate auf S als die ursprüngliche. Diese kleine Überlegung zeigt, dass wir oBdA voraussetzen dürfen: erstens, F_K macht keine Fehler auf den Kanten-Beispielen und, zweitens, F_K macht höchstens k Fehler auf den Knoten-Beispielen. In Verbindung mit den obigen zentralen Beobachtungen können wir folgern: erstens, K ist ein Vertex Cover in G und, zweitens, $|K| \leq k$. Damit ist obige Behauptung, wie auch das Theorem, vollständig bewiesen. •

Für etwas komplexere Hypothesenklassen erweist sich auch das Konsistenzproblem zuweilen als NP-hart. Wir illustrieren das anhand der Klasse Monotone 2-Term DNF. Beachte, dass die Klasse Monotone 1-Term DNF übereinstimmt mit der Klasse Monotone 1-CNF: es handel sich beide Male um die Klasse der Verundungen von unnegierten Booleschen Literalen. Beim Übergang von Monotone 1-Term DNF zu Monotone 2-Term DNF geschieht also der Quantensprung von einem effizient lösbaren zu einem NP-harten Problem.

Theorem 6.25 *Kons-E(Monotone 2-Term DNF) ist NP-hart.*

Beweis Es sei 2-HG-Colorability (zu deutsch: 2-Färbbarkeitsproblem für Hypergraphen) das folgende (als NP-hart bekannte) Problem: gegeben $m, n \geq 1$ und $I_1, \ldots, I_m \subseteq [n]$, Können wir die Elemente $1, \ldots, n$ mit zwei Farben so einfärben, dass keine der Mengen $I_1, \ldots, I_m$ monochromatisch ist? Formal: existiert eine Abbildung $f : [n] \to \{1, 2\}$, so dass für alle $i = 1, \ldots, m$ für die Bildmenge $f(I_i)$ gilt $f(I_i) = \{1, 2\}$?[14] Die Struktur $([n], \{I_1, \ldots, I_m\})$ wird auch als Hypergraph mit Knotenmenge $[n]$ und Hyperkanten $I_1, \ldots, I_m$ bezeichnet. Unsere Reduktionsabbildung soll die Frage der 2-Färbbarkeit dieses Hypergraphen in ein Konsistenzproblem für 2-Term DNF Formeln verwandeln. Wir geben eine passende Reduktionsabbildung an und verwenden dabei die im Beweis von Theorem 6.24 eingeführte Notation $\mathbf{0_I}$ mit $I \subseteq [n]$.

Knoten-Beispiele: Die Vektoren $\mathbf{0_1}, \ldots, \mathbf{0_n}$ erklären wir zu positiven Beispielen, d.h., sie erhalten das Label 1.

Hyperkanten-Beispiele: Für alle $i = 1, \ldots, m$ erklären wir den Vektor $\mathbf{0_{I_i}}$ zu einem negativen Beispiel, d.h., er erhält das Label 0.

Die Menge S bestehe genau aus diesen gelabelten Knoten- bzw. Hyperkanten-Beispielen. Es genügt zu zeigen, dass folgendes gilt:

Behauptung: Der Hypergraph $([n], \{I_1, \ldots, I_m\}$ ist genau dann 2-färbbar, wenn es eine zu S konsistente monotone 2-Term DNF Formel mit Booleschen Variablen $v_1, \ldots, v_n$ gibt.

Eine monotone 2-Term DNF über den Variablen $v_1, \ldots, v_n$ hat die allgemeine Form

$$F_{J,K} = \left(\bigwedge_{j \in J} v_j \right) \vee \left(\bigwedge_{k \in K} v_k \right)$$

für zwei Indexmengen $J, K \subseteq [n]$. Die zentralen Beobachtungen zum Beweis der Behauptung sind wie folgt:

1. Der Vektor $\mathbf{0_i}$ wird von $F_{J,K}$ korrekt gelabelt (also zu 1 ausgewertet), wenn $i \notin J \cap K$. Daher ist $F_{J,K}$ auf den Knoten-Beispielen genau dann fehlerfrei, wenn $J \cap K = \emptyset$.

[14] Wenn alle Mengen $I_1, \ldots, I_m$ aus exakt zwei Elementen bestünden, könnten wir sie als Kanten eines Graphen interpretieren, und wir erhielten das 2-Färbungsproblem für Graphen (welches effizient lösbar ist).

2. Ein Vektor der Form $\mathbf{0_I}$ mit $I \subseteq [n]$ wird von $F_{J,K}$ genau dann zu 0 ausgewertet, wenn $I \cap J \neq \emptyset$ und $I \cap K \neq \emptyset$.

Wir betrachten nun die Beweisrichtung „$\Rightarrow$“. Sei also eine 2-Färbung von $1, \ldots, n$ gegeben, so dass keine der Mengen $I_1, \ldots, I_m$ monochromatisch ist. Sagen wir die Knoten aus $J \subseteq [n]$ sind „rot“ und die Knoten aus $K = [n] \setminus J$ sind „blau“ gefärbt. Da $J \cap K = \emptyset$, ist $F_{J,K}$ auf den Knoten-Beispielen fehlerfrei. Da keine der Mengen $I_1, \ldots, I_m$ monochromatisch ist, gilt $I_i \cap J \neq \emptyset$ und $I_i \cap K \neq \emptyset$ für alle $i = 1, \ldots, m$. Somit ist $F_{J,K}$ auch auf den Hyperkanten-Beispielen fehlerfrei.
Betrachten wir abschließend die Beweisrichtung „$\Leftarrow$“. Sei also eine zu S konsistente Formel $F_{J,K}$ mit $J, K \subseteq [n]$ vorgegeben. Da $F_{J,K}$ auf den Knoten-Beispielen fehlerfrei ist, folgt $J \cap K = \emptyset$. Da $F_{J,K}$ auch auf den Hyperkanten-Beispielen fehlerfrei ist, folgt $I_i \cap J \neq \emptyset$ und $I_i \cap K \neq \emptyset$ für alle $i = 1, \ldots, m$. Offensichtlich erhalten wir eine zulässige 2-Färbung, wenn wir alle Knoten aus J rot und alle Knoten aus $[n] \setminus J$ blau einfärben.[15] Damit ist obige Behauptung, wie auch das Theorem, vollständig bewiesen. •

Aus den Theoremen 6.24 und 6.25 können wir unmittelbar folgendes schließen:

Folgerung 6.26 *1. Die Klasse Monotone-1-CNF ist nicht effizient agnostisch PAC-lernbar (außer wenn RP = NP).*

2. Die Klasse Monotone 2-Term DNF ist nicht einmal unter der Realisierbarkeitsannahme effizient PAC-lernbar (außer wenn RP = NP).

Negativresultate dieser Art können manchmal umgangen werden, indem wir einem Algorithmus zum Lernen einer Klasse $\mathcal{H}$ erlauben, eine Hypothese zu wählen, die aus einer „mächtigeren“ Klasse $\mathcal{H}'$ stammt. Dabei heißt $\mathcal{H}'$ *mächtiger* als $\mathcal{H}$, wenn es für jedes $n \geq 1$ und für jede Hypothese $h \in \mathcal{H}_n$ eine Hypothese $h' \in \mathcal{H}'_n$ gibt, welche die selbe binäre Funktion repräsentiert wie die Hypothese h. Die *effiziente PAC-Lernbarkeit von $\mathcal{H}$ mit Hypothesen aus $\mathcal{H}'$* ist dann auf die offensichtliche Weise definiert. Es gilt folgendes:

Lemma 6.27 *Wenn $\mathcal{H}'$ mächtiger ist als $\mathcal{H}$ und $\mathcal{H}'$ ist effizient agnostisch (bzw. effizient unter der Realisierbarkeitsannahme) PAC-lernbar, dann ist $\mathcal{H}$ effizient agnostisch (bzw. effzient unter der Realisierbarkeitsannahme) PAC-lernbar mit Hypothesen aus $\mathcal{H}'$.*

Beweis Wir führen den Beweis für PAC-Lernen unter der Realisierbarkeitsannahme. Da $\mathcal{H}'$ mächtiger ist als $\mathcal{H}$, ist eine Trainingssequenz $S \in \mathcal{Z}_n^m$, deren Labels einer (dem Lerner unbekannten) Funktion $h \in \mathcal{H}_n$ entsprechen zugleich auch eine Trainingssequenz, deren Labels einer (dem Lerner unbekannten) Funktion $h' \in \mathcal{H}'_n$ entsprechen. Daher können wir einfach den PAC-Lernalgorithmus für $\mathcal{H}'$ auf S ansetzen.
Die Beweisführung für den Fall des effizienten agnostischen PAC-Lernens ist analog. •

Wir diskutieren ein Anwendungsbeispiel:

[15] Die Knoten aus $[n] \setminus (J \cup K)$ können natürlich beliebig eingefärbt werden, da bereits die Einfärbung der Knoten aus $J \cup K$ monochromatische Mengen I_i verhindert.

Lemma 6.28 *k-CNF ist mächtiger als k-Term DNF.*

Beweis Wir nutzen das für Boolesche Algebren gültige Distributivitätsgesetz aus:

$$a \vee (b \wedge c) = (a \vee b) \wedge (a \vee c) \quad \text{und} \quad a \wedge (b \vee c) = (a \wedge b) \vee (a \wedge c) \ .$$

Diese Regel kann benutzt werden, um eine k-Term DNF Formel in eine äquivalente k-CNF Formel umzuformen. •

Beispiel 6.29 *Die 2-Term DNF Formel*

$$(v_1 \wedge \overline{v_3}) \vee (\overline{v_1} \wedge v_2 \wedge \overline{v_4})$$

repräsentiert die selbe Boolesche Funktion wie die 2-CNF Formel

$$(v_1 \vee \overline{v_1}) \wedge (v_1 \vee v_2) \wedge (v_1 \vee \overline{v_4}) \wedge (\overline{v_3} \vee \overline{v_1}) \wedge (\overline{v_3} \vee v_2) \wedge (\overline{v_3} \vee \overline{v_4}) \ .$$

Die Umformung beruht auf wiederholter Anwendung des Distibutivitätsgesetzes.

Da k-CNF unter der Realisierbarkeitsannahme effizient PAC-lernbar ist, erhalten wir unmittelbar folgendes Resultat:

Folgerung 6.30 *Die Klasse k-Term DNF ist unter der Realisierbarkeitsannahme effizient PAC-lernbar mit Hypothesen aus k-CNF.*

6.6.2 Kryptographische Barrieren

Wir starten mit der kleinen, aber feinen, Beobachtung, dass PAC-Lernalgorithmen (auch wenn sie Hypothesen außerhalb von $\mathcal{H}$ abliefern) verwendet werden können, um das unbekannte Label einer neuen Testinstanz (die im Training i.A. nicht gesehen wurde) mit kleiner erwarteter Fehlerrate vorherzusagen:

Lemma 6.31 *Wenn eine binäre Hypothesenklasse $\mathcal{H}$ unter der Realisierbarkeitsannahme effizient PAC-lernbar ist mit einer Klasse $\mathcal{H}'$ (deren Hypothesen in Polynomialzeit auswertbar sind), dann kann der zugrunde liegende Algorithmus (trainiert auf einer hinreichend großen Menge markierter Beispiele) benutzt werden, um in Polynomialzeit das Label zu einer bezüglich $\mathcal{D}$ zufälligen Testinstanz x mit einer Erfolgswahrscheinlichkeit von mindestens $1 - \alpha$ korrekt vorherzusagen. Dabei ist $\alpha > 0$ eine vorgegebene beliebig kleine positive Zahl.*

Beweis Es sei A der entsprechende Lernalgorithmus. Wir setzen $\epsilon = \delta = \alpha/2$, setzen A auf eine hinreichend große Trainingssequenz an und erhalten mit einer Wahrscheinlichkeit von mindestens $1 - \delta$ eine Hypothese h' mit einer Fehlerrate von maximal ε. Dann labeln wir die vorgegebene Testinstanz x mit $h'(x)$. Das korrekte Label sei b. Das Label $h'(x)$ ist korrekt, sofern nicht folgendes „schlechte Ereignis“ B eintritt:

$$L_{\mathcal{D}}(h') > \varepsilon \quad \text{oder} \quad (L_{\mathcal{D}}(h') \leq \varepsilon \quad \text{und} \quad h'(x) \neq b) \ .$$

Die Wahrscheinlichkeit von B ist beschränkt durch $\delta + \varepsilon = \alpha/2 + \alpha/2 = \alpha$. •

Ein Ausflug in die elementare Zahlentheorie. Es sei $N \geq 2$ eine natürliche Zahl und $\mathbb{Z}_N = \{0, 1, \ldots, N-1\}$ seien die kleinsten Reste bei ganzzahliger Division durch N. Aus einer der Grundvorlesungen sollte bekannt sein, dass $(\mathbb{Z}_N, +, \cdot)$ ein Ring ist. Dabei wird Addition „$+$“ und Multiplikation „$\cdot$“ (sofern nicht ausdrücklich anders gesagt) immer modulo N ausgeführt. Weiterhin enthält

$$\mathbb{Z}_N^* = \{a \in \mathbb{Z}_N | \ \mathrm{ggT}(a, N) = 1\}$$

exakt die Menge der Elemente in $\mathbb{Z}_N$, welche ein multiplikatives Inverses besitzen. Mit Hilfe des erweiteren Euklidischen Algorithmus

- können wir effizient testen, ob $a \in \mathbb{Z}_N^*$
- und ggf. können wir effizient das Inverse b mit $a \cdot b = 1$ bestimmen.

$(\mathbb{Z}_N^*, \cdot)$ ist eine multiplikative Gruppe, die sogenannte Gruppe der primen Reste modulo N. Die Anzahl der Elemente in $\mathbb{Z}_N^*$ ist durch die Eulersche Funktion φ gegeben, d.h., $|\mathbb{Z}_N^*| = \varphi(N)$. Für alle $a \in \mathbb{Z}_N^*$ gilt $a^{\varphi(N)} = 1$. Hieraus folgt, dass man beim Rechnen in $(\mathbb{Z}_N^*, \cdot)$ in dem Exponenten einer Potenz modulo $\varphi(N)$ rechnen darf. Wir nehmen im Folgenden an, dass $\varphi(N)$ größer als 3, aber kein Vielfaches von 3, ist, so dass $\mathrm{ggT}(3, \varphi(N)) = 1$. Somit hat 3 modulo $\varphi(N)$ ein multiplikatives Inverses d mit $3 \cdot d = 1 \pmod{\varphi(N)}$. In diesem Fall ist die Abbildung $f_N : \mathbb{Z}_N^* \to \mathbb{Z}_N^*$ mit $f_N(x) = x^3$ eine Bijektion mit Umkehrabbildung $f_N^{-1}(x) = x^d$. Es gilt nämlich

$$f_N(f_N^{-1}(x)) = f_N^{-1}(f_N(x)) = x^{3d} = x \ ,$$

weil wir im Exponenten modulo $\varphi(N)$ rechnen dürfen und weil $3 \cdot d = 1 \pmod{\varphi(N)}$ gültig ist. Die Zahl $x = y^d \in \mathbb{Z}_N^*$ erfüllt demnach die Gleichung $x^3 = y$ und wird daher auch als die *diskrete Kubikwurzel* von y bezeichnet. Wenn N, y, d gegeben sind, dann ist $y^d \in \mathbb{Z}_N^*$ (mit der Methode des iterierten Quadrierens) effizient berechenbar. Es wird aber vermutet, dass die diskrete Kubikwurzel von y schwer zu berechnen ist, wenn lediglich N und y gegeben sind und wenn N das Produkt zweier großer Primzahlen ist:

Diskrete-Kubikwurzel-Vermutung (DKW-Vermutung) Es seien p, q zwei zufällig gewählte $n/2$-Bit Primzahlen, so dass für die n-Bit Zahl $N = pq$ gilt: 3 ist kein Teiler von $\varphi(N) = (p-1)(q-1)$. Es sei $y \in \mathbb{Z}_N^*$ zufällig gewählt. Die Vermutung lautet: es gibt keinen Algorithmus A mit einer in n polynomiell beschränkten Laufzeit, der die diskrete Kubikwurzel von y zu gegebenem (N, y) mit einer nicht vernachlässigbaren Erfolgswahrscheinlichkeit[16] korrekt berechnet. Die Wahrscheinlichkeit wird dabei genommen über die zufällige Wahl von p, q, y und (sofern A randomisiert ist) über die von A verwendeten Zufallsbits.

In dieser Vermutung bedeutet „zufällig“ stets „zufällig bezüglich der uniformen Verteilung“. Die DKW-Vermutung beruht auf der weitergehenden Vermutung, dass die Primfaktorzerlegung von $N = pq$ schwer zu berechnen ist. Wenn nämlich die Teiler p, q von N bekannt wären, dann könnte die diskrete Kubikwurzel von y effizient berechnet werden wie folgt:

[16] d.h. mit einer Wahrscheinlichkeit von $1/p(n)$ für ein geeignetes Polynom p

1. Berechne $\varphi(N) = (p-1)(q-1)$.

2. Berechne das Inverse d von 3 modulo $\varphi(N)$ mit dem erweiterten Euklidischen Algorithmus.

3. Berechne $x = y^d$ mit der sogenannten Methode des iterierten Quadrierens.

Wir können im Rahmen dieser Vorlesung nicht auf den erweiterten Euklidischen Algorithmus und die Methode des iterierten Quadrierens eingehen. Es handelt sich um bekannte effiziente Methoden zur Bestimmung des multiplikativen Inversen und zum Potenzieren im Rahmen der modulo-Rechnung.

Berechnung diskreter Kubikwurzeln als Lernproblem Es sei N eine n-Bit-Zahl und $x \in \mathbb{Z}_N^*$. Wir identifizieren dann x mit seinem Binärcode aus $\{0,1\}^n$. Das i-te Bit von x wird notiert als $x[i]$. Wir werden im Folgenden Funktionen mit Definitionsbereich $\mathbb{Z}_N^*$ als Funktionen mit Definitionsbereich $\{0,1\}^n$ ansehen mit dem Verständnis, dass wir sie nur auf Binärcodes von Elementen aus $\mathbb{Z}_N^*$ anzuwenden gedenken. Wir sagen N ist *n-zulässig*, wenn Folgendes gilt: $N = pq$ für zwei $n/2$-Bit Primzahlen p, q, so dass 3 kein Teiler von $\varphi(N) = (p-1)(q-1)$ ist. Für eine n-zulässige Zahl N und $i \in [n]$, definieren wir die Funktion $\mathrm{DKW}_{N,i} : \{0,1\}^n \to \{0,1\}$ wie folgt:

$$\mathrm{DKW}_{N,i}(y) = x[i] \text{ mit } x^3 = y \text{ in } \mathbb{Z}_N^* \ .$$

Weiter sei

$$\mathrm{DKW}_n = \{\mathrm{DKW}_{N,i} |\ N \text{ ist } n\text{-zulässig und } 1 \leq i \leq n\ \}$$

und $\mathrm{DKW} = (\mathrm{DKW}_n)_{n \geq 1}$.

Theorem 6.32 *Unter der DKW-Vermutung gilt folgendes: die Klasse DKW ist nicht effizient PAC-lernbar, und zwar selbst dann nicht, wenn die Realisierbarkeitsannahme gemacht wird und der Lerner Hypothesen aus einer von DKW verschiedenen Klasse abliefern darf.*

Beweis Es genügt zu zeigen, dass die DKW-Vermutung falsch ist, falls die Aussage dieses Theorems falsch ist (weil das im Umkehrschluss heißt, dass das Theorem unter der DKW-Vermutung korrekt ist). Angenommen wir verfügen über einen in Polynomialzeit arbeitenden Lernalgorithmus A. Wir verwenden diesen im Sinne von Lemma 6.31 als Hilfsmittel zum effizienten Voraussagen von Labels auf einer Testinstanz und können die diskrete Kubikwurzel x^* von $y^* \in \mathbb{Z}_N^*$ zu gegebenem N, y^* berechnen wie folgt:

1. Es sei n die Bitlänge von N. Wir setzen $\alpha := 1/n^2$.

2. Für jedes $i \in [n]$ präparieren wir eine hinreichend große Trainingssequenz S^i. Diese enthält Beipiele der Form $(y, x[i])$, wobei $x \in \mathbb{Z}_N^*$ zufällig gewählt und $y := x^3$ modulo N gesetzt wird (so dass x die diskrete Kubikwurzel von y modulo N ist).

3. Für jedes $i \in [n]$ setzen wir A auf S^i sowie auf die Test-Instanz y^* an und erhalten ein Voraussagebit $b[i]$ für $x^*[i]$.

4. Wir geben den Bitvektor $b = (b[1], \ldots, b[n])$ aus.

Wir beobachten zunächst, dass $y = x^3$ uniform auf $\mathbb{Z}_N^*$ verteilt ist, sofern x dies ist. Daher ist y im Trainingsbeispiel $(y, x[i])$ von Schritt 2 uniform zufällig aus $\mathbb{Z}_N^*$ (ebenso wie das Testbeispiel im Szenario der DKW-Vermutung)). Gemäß Lemma 6.31 ist für jedes $i \in [n]$ die Wahrscheinlichkeit für $b[i] \neq x^*[i]$ durch $\alpha = 1/n^2$ beschränkt. Gemäß der Union Bound ist dann die Wahrscheinlichkeit von $b \neq x^*$ durch $n\alpha = 1/n$ beschränkt. Somit wäre die Erfolgswahrscheinlichkeit sogar größer oder gleich $1 - 1/n$ (also bei Weitem nicht vernachlässigbar) und die DKW-Vermutung wäre widerlegt. •

Kryptographisch harte natürliche Lernprobleme. Die Klasse DKW ist ein „Kunstprodukt", das mit der Absicht entworfen wurde, ein hartes Lernproblem zu erzeugen. Wir wollen aber jetzt aufzeigen, dass sich das in Theorem 6.32 formulierte negative Ergebnis auf natürlicher gewählte Hypothesenklassen ausdehnen lässt. Zunächst beobachten wir, dass das Lernproblem zu DKW schwer bleibt, wenn wir es leicht modifizieren, indem wir die Beipielinstanzen nicht in der Form y präsentieren sondern in der redundanten Form

$$(z_0, z_1, \ldots, z_{n-1}) \text{ mit } z_i = y^{2^i} \text{ für } i = 0, 1, \ldots, n-1 \ . \tag{29}$$

Der Beweis von Theorem 6.32 muss dazu nur in Schritt 2 des Kubikwurzelberechnungsverfahrens entsprechend modifiziert werden. Der Sinn dieses Manövers ist, dass sich die Kubikwurzel $x = y^d$ jetzt in einer sehr speziellen Form darstellen lässt: für $d = \sum_{i=0}^{n-1} d_i 2^i$ gilt

$$y^d = y^{\sum_{i:d_i=1} 2^i} = \prod_{i:d_i=1} y^{2^i} = \prod_{i:d_i=1} z_i \ . \tag{30}$$

Eine Funktion der Form

$$h_{N,d}(z) = \prod_{i:d_i=1} z_i \text{ in } \mathbb{Z}_N^*$$

mit n-Bit Zahlen N, d sowie $z = (z_0, z_1, \ldots, z_{n-1})$ und $z_0, z_1, \ldots, z_{n-1} \in \mathbb{Z}_N^*$ wird als *iteriertes Produkt* in $\mathbb{Z}_N^*$ bezeichnet. Weiter sei $h_{N,d,i}$ die Funktion, die z auf das i-te Bit von $h_{N,d}(z)$ abbildet. Wir sagen eine Hypothesenklasse $\mathcal{H} = (\mathcal{H}_n)_{n\geq 1}$ *kann das iterierte Produkt darstellen*, wenn jede Funktion der Form $h_{N,d,i}$ in $\mathcal{H}_{n^2}$ mit Grundbereich $\{0,1\}^{n^2}$ enthalten ist. Eine solche Klasse kann dann auch die Funktionen der Form $\mathrm{DKW}_{N,i}$ darstellen und is daher nicht einfacher zu lernen sein als die Klasse DKW. Damit ergibt sich unter der DKW-Vermutung Folgendes:

Bemerkung 6.33 *Wenn eine Hypothesenklasse $\mathcal{H}$ das iterierte Produkt darstellen kann, dann ist $\mathcal{H}$ nicht effizient PAC-lernbar, und zwar selbst dann nicht, wenn die Realisierbarkeitsannahme gemacht wird und der Lerner Hypothesen aus einer von $\mathcal{H}$ verschiedenen Klasse abliefern darf.*

Aus der Komplexitätstheorie ist bekannt, dass die folgenden Klassen das iterierte Produkt darstellen können (ohne Beweis):

- Boolesche Schaltkreise (über der Standardbasis $\wedge, \vee, \neg$) einer polynomiell beschränkten Größe
- Boolesche Schaltkreise über der Basis der linearen Schwellenfunktionen (sogenannte Schwellen-Schaltkreise) einer polynomiell beschränkten Größe und einer konstanten Tiefe
- Deterministische Turing-Maschinen (DTM) mit einer logarithmischen Platzschranke (was, technisch gesehen, daraus folgt, dass Boolesche Schaltkreise von einer logarithmisch platzbeschränkten DTM auswertbar sind)

Schwache L-Reduktion und weitere Hiobsbotschaften. Eine *polynomielle schwache L-Reduktion* von einer Hypothesenklasse $\mathcal{H}$ auf eine Hypothesenklasse $\mathcal{H}'$ entsteht aus einer gewöhnlichen L-Reduktion durch Weglassen der zweiten Hypothesentransformation und durch Weglassen der Forderung, dass die erste Hypothesentransformation in Polynomialzeit berechenbar sein muss. Es ist nicht schwer, Folgendes zu zeigen:

Lemma 6.34 *Es sei $\mathcal{H}$ polynomiell schwach auf $\mathcal{H}'$ L-reduzierbar. Wenn $\mathcal{H}'$ mit Hilfe einer anderen Hypothesenklasse effizient (unter der Realisierbarkeitsannahme bzw. agnostisch) PAC-lernbar ist, dann trifft dies auch auf $\mathcal{H}$ zu.*

Im Umkehrschluss heißt das natürlich: Wenn $\mathcal{H}$ nicht effizient PAC-lernbar ist, dann trifft dies auch auf $\mathcal{H}'$ zu.

Lemma 6.35 *Die Klasse der logarithmisch platzbeschränkten DTM ist polynomiell schwach L-reduzierbar auf die Klasse der deterministischen endlichen Automaten (englisch: Deterministic Finite Automata oder kurz DFA) einer polynomiell beschränkten Größe.*

Beweis Wir skizzieren den Beweis lediglich. Wir nutzen aus, dass eine DTM M mit einer logarithmischen Platzschranke, angesetzt auf ein Eingabewort der Länge n, nur polynomiell in n viele verschiedene Konfigurationen haben kann. Die erste Hypothesentransformation bildet M ab auf einen DFA M', dessen Zustände 1–zu–1 den Konfigurationen von M für Eingabewörter w der Länge n entsprechen. Wir können aber nicht die triviale Instanzentransformation $w \mapsto w$ verwenden: der Kopf auf dem Eingabeband von M darf sich nämlich sowohl nach rechts als auch nach links bewegen, wohingegen der Kopf des DFA M' in jedem Rechenschritt eine Position nach rechts rücken muss. Wir lösen dieses Problem, indem wir die Eingabetransformation $w \mapsto w \dots w$ für eine ausreichende Anzahl von Duplikaten von w verwenden: wenn der Kopf auf dem Eingabeband von M eine Position nach links rückt, dann rückt der Kopf von M' um $n-1$ Positionen nach rechts. Auf diese Weise kann M' die Rechnung von M auf w simulieren. •

Wir ziehen ein deprimierendes Résumée:

Folgerung 6.36 *Die folgenden Klassen $\mathcal{H}$ sind nicht effizient PAC-lernbar, und zwar selbst dann nicht, wenn die Realisierbarkeitsannahme gemacht wird und wenn der Lerner Hypothesen aus einer von $\mathcal{H}$ verschiedenen Klasse abliefern darf:*

- *Boolesche Schaltkreise einer polynomiell beschränkten Größe*
- *Schwellen-Schaltkreise einer polynomiell beschränkten Größe und einer konstanten Tiefe*
- *logarithmisch platzbeschränkte deterministische Turing-Maschinen*
- *Deterministische endliche Automaten einer polynomiell beschränkten Größe*

7 Lineare Voraussagefunktionen

Lineare Voraussagefunktionen sind wichtig, und zwar aus folgenden Gründen:

- Sie werden in Anwendungen oft benutzt.
- Sie führen zu effizient lösbaren Lernproblemen (mit gewissen Einschränkungen im agnostischen Fall).
- Sie sind der Intuition zugänglich und einfach zu interpretieren.
- Wenn sie geeignet eingesetzt werden, beschreiben sie oftmals die in Anwendungen vorliegenden Daten auf eine befriedigende Weise.

Im Folgenden fassen wir Vektoren $\mathbf{w} \in \mathbb{R}^d$ grundsätzlich als Spaltenvektoren auf. Der betreffende Zeilenvektor wird als $\mathbf{w}^\top$ notiert. Das Skalarprodukt von $\mathbf{w}, \mathbf{x} \in \mathbb{R}^d$ schreiben wir entweder in der Form $\mathbf{w}^\top \mathbf{x}$ oder (als inneres Produkt) in der Form $\langle \mathbf{w}, \mathbf{x} \rangle$. Weiterhin sei

$$\mathrm{LIN}_d := \{x \mapsto h_{\mathbf{w},b}(x) : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\} \tag{31}$$

mit

$$h_{\mathbf{w},b} : \mathbb{R}^d \to \mathbb{R} \ , \ h_{\mathbf{w},b}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = \sum_{i=1}^{d} w_i x_i + b \ .$$

die Klasse der *affinen Funktionen*. Wir bezeichnen $\mathbf{w}$ als den *Gewichtsvektor* und b als das *Bias* der Abbildung $h_{\mathbf{w},b}$. Eine *lineare Voraussagefunktion* mit Werten in $\mathcal{Y}$ ergibt sich aus der Komposition einer Funktion $\phi : \mathbb{R} \to \mathcal{Y}$ mit einer affinen Funktion aus LIN_d:

$$\phi \circ h_{\mathbf{w},b} : \mathbb{R}^d \to \mathcal{Y} \ , \ \phi(h_{\mathbf{w},b}(\mathbf{x})) = \phi(\langle \mathbf{w}, \mathbf{x} \rangle + b) \ .$$

Mögliche Wahlen von ϕ:

1. $\phi = \text{sign} : \mathbb{R} \to \{-1, 1\}$ mit

$$\text{sign}(a) = \begin{cases} +1 & \text{if } a \geq 0 \\ -1 & \text{if } a < 0 \end{cases}$$

für alle $a \in \mathbb{R}$. Diese Wahl erfolgt im Zusammenhang mit binären Klassifikationsproblemen.

2. $\phi = \text{id} : \mathbb{R} \to \mathbb{R}$ mit $\text{id}(a) = a$ für alle $a \in \mathbb{R}$. Diese Wahl erfolgt im Zusammenhang mit Regressionsproblemen.

Wir vereinbaren, dass die Anwendung der Funktion sign auf einen Vektor komponentenweise durchzuführen ist.

Es bezeichne $\text{LIN}_d^0 = \{h_{\mathbf{w},0} : \mathbf{w} \in \mathbb{R}^d\}$ die Klasse der *linearen Funktionen* (affine Funktionen mit Bias 0). Es gilt einerseits $\text{LIN}_d^0 \subset \text{LIN}_d$; andererseits können wir eine Funktion $h_{\mathbf{w},b} \in \text{LIN}_d$ auch als Funktion in LIN_{d+1}^0 auffassen, indem wir folgende Einbettungen von $\mathbf{x}, \mathbf{w} \in \mathbb{R}^d$ in den $(d+1)$-dimensionalen Euklidischen Raum vornehmen:

$$\mathbf{x}' = (1, \mathbf{x}) = (1, x_1, \ldots, x_d) \neq \mathbf{0} \quad \text{und} \quad \mathbf{w}' = (b, \mathbf{w}) = (b, w_1, \ldots, w_d) \ . \tag{32}$$

Offensichtlich gilt dann:

$$h_{\mathbf{w},b}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle + b = \langle \mathbf{w}', \mathbf{x}' \rangle = h_{\mathbf{w}',0}(\mathbf{x}') \ .$$

Wann immer wir das bequem finden, werden wir daher lineare Funktionen (angewendet auf vom Nullvektor verschiedene Vektoren) anstelle von affinen Funktionen betrachten, d.h., wir werden mitunter oBdA $b = 0$ setzen und $\mathbf{x} \neq \mathbf{0}$ voraussetzen. Man spricht dann auch vom "homogenen Fall" (im Unterschied zum sogenannten"inhomogenen Fall" bei Betrachtung von allgemeinen affinen Funktionen).

7.1 Halbräume

Es bezeichne

$$H_{\mathbf{w},b} = \{\mathbf{x} \in \mathbb{R}^d : h_{\mathbf{w},b}(\mathbf{x}) = 0\} = \{\mathbf{x} \in \mathbb{R}^d : \langle \mathbf{w}, \mathbf{x} \rangle + b = 0\}$$

die durch $\mathbf{w}, b$ gegebene affine Hyperebene im $\mathbb{R}^d$. Diese zerteilt $\mathbb{R}^d$ in zwei Halbräume:

$$H_{\mathbf{w},b}^+ = \{\mathbf{x} \in \mathbb{R}^d : \langle \mathbf{w}, \mathbf{x} \rangle + b \geq 0\} \quad \text{(positiver Halbraum)} \ .$$
$$H_{\mathbf{w},b}^- = \{\mathbf{x} \in \mathbb{R}^d : \langle \mathbf{w}, \mathbf{x} \rangle + b < 0\} \quad \text{(negativer Halbraum)} \ .$$

Wir bezeichnen dann

$$\text{HS}_d = \{H_{\mathbf{w},b}^+ : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\}$$

als die Klasse der *(affinen) Halbräume*. Alternativ können wir einen positiven Halbraum $H_{\mathbf{w},b}^+$ auch identifizieren mit der Abbildung $\mathbf{x} \mapsto \text{sign}(h_{\mathbf{w},b}(\mathbf{x}))$, welche die Punkte im positiven

Halbraum mit 1 und die im negativen Halbraum mit -1 markiert. Die Klasse der affinen Halbräume, aufgefasst als Klasse von ± 1-wertigen Funktionen, wäre dann gegeben durch

$$\mathrm{HS}_d = \mathrm{sign} \circ \mathrm{LIN}_d = \{\mathbf{x} \mapsto \mathrm{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b) : \mathbf{w} \in \mathbb{R}^d, b \in \mathbb{R}\} \ .$$

Wir werden Halbräume meist mit $H^+_{\mathbf{w},b}$ identifizieren (gelegentlich aber auch mit $\mathbf{x} \mapsto \mathrm{sign}(h_{\mathbf{w},b}(\mathbf{x}))$).

Eine affine Hyperebene $H_{\mathbf{w},b}$ heißt *homogen*, falls $b = 0$ und *inhomogen* falls $b \neq 0$. Der Gewichtsvektor $\mathbf{w} \in \mathbb{R}^d$ wird auch als der *Normalenvektor* von $H_{\mathbf{w},b}$ bezeichnet. Aus der linearen Algebra ist bekannt, dass folgendes gilt:

Homogener Fall: $\mathbf{x} \in H_{\mathbf{w},0}$ (bzw. $\mathbf{x} \in H^+_{\mathbf{w},0} \setminus H_{\mathbf{w},0}$ oder $\mathbf{x} \in H^-_{\mathbf{w},0}$) genau dann, wenn $\mathbf{x}$ und $\mathbf{w}$ einen rechten Winkel (bzw. einen spitzen oder einen stumpfen) Winkel miteinander bilden.

Inhomogener Fall: Es sei $\mathbf{a} \in H_{\mathbf{w},b}$ ein beliebiger Punkt auf der affinen Hyperebene $H_{\mathbf{w},b}$. Dann gilt

$$H_{\mathbf{w},b} = a + H_{\mathbf{w},0} \ , \ H^+_{\mathbf{w},b} = a + H^+_{\mathbf{w},0} \ \text{ und } H^-_{\mathbf{w},b} = a + H^-_{\mathbf{w},0} \ .$$

Somit gilt $\mathbf{a} + \mathbf{x} \in H_{\mathbf{w},b}$ (bzw. $\mathbf{a} + \mathbf{x} \in H^+_{\mathbf{w},b} \setminus H_{\mathbf{w},b}$ oder $\mathbf{a} + \mathbf{x} \in H^-_{\mathbf{w},b}$) genau dann, wenn $\mathbf{x}$ und $\mathbf{w}$ einen rechten Winkel (bzw. einen spitzen oder einen stumpfen) Winkel miteinander bilden.

7.1.1 Informationskomplexität der Klasse der Halbräume

Wir wissen, dass die Anzahl der zum PAC-Lernen einer binären Hypothesenklasse $\mathcal{H}$ benötigten Trainigsbeispiele proportional zur VC-Dimension von $\mathcal{H}$ ist. In diesem Abschnitt werden wir zeigen, dass die VC-Dimension von HS_d exakt $d + 1$ beträgt und somit eine lediglich lineare Abhängigkeit in dem „Komplexitätsparameter“ d aufweist.

Es bezeichne $\mathrm{HS}^0_d = \{H^+_{\mathbf{w},0} : \mathbf{w} \in \mathbb{R}^d\}$ die Klasse der homogenen Halbräume. Wir notieren mit $\mathbf{e_i}$ den Vektor mit einer 1 in Position i und Nullen in allen anderen Positionen. Im Zusammenhang mit Halbräumen benutzen wir im Folgenden die Binärlabels $-1, 1$ (anstelle der Labels $0, 1$).

Lemma 7.1 *Eine Punktmenge $S \subseteq \mathbb{R}^d$ ist aufspaltbar durch HS^0_d genau dann, wenn S aus linear unabhängigen Vektoren besteht.*

Beweis Wir beginnen mit der Beweisrichtung „$\Leftarrow$“. Wir dürfen annehmen, dass S aus d linear unabhängigen Vektoren $\mathbf{x_1}, \ldots, \mathbf{x_d} \in \mathbb{R}^d$ besteht (da wir kleinere Mengen S zu einer Basis ergänzen können). Es sei A die $(d \times d)$-Matrix mit $\mathbf{x_1}^\top, \ldots, \mathbf{x_d}^\top$ als Zeilenvektoren. Beachte, dass $A\mathbf{w}$ dann der Vektor mit den Komponenten $\mathbf{x_i}^\top \mathbf{w} = \mathbf{w}^\top \mathbf{x_i}$ ist, d.h., $\mathrm{sign}(A\mathbf{w})$ ist das Binärmuster, dass w auf den Vektoren $\mathbf{x_1}, \ldots, \mathbf{x_d}$ erzeugt. Da A den Rang d hat, existiert die inverse Matrix A^{-1}. Dann ist das Gleichungssystem $A\mathbf{w} = \mathbf{y}$ für jede Wahl von

$\mathbf{y} \in \mathbb{R}^d$ lösbar, nämlich mit $\mathbf{w} = A^{-1}\mathbf{y}$. Dies gilt insbesondere für alle $\mathbf{y} \in \{-1,1\}^d$. Daher ist S durch HS_d^0 aufspaltbar.
Die Beweisrichtung „$\Rightarrow$“ zeigen wir indirekt. Wir nehmen an, dass S aus linear abhängigen Vektoren $\mathbf{x_1}, \ldots, \mathbf{x_m}$ besteht und zeigen, dass dann nicht alle Binärmuster durch HS_d^0 realisierbar sind. Wegen der linearen Abhängigkeit existiert ein Vektor $\mathbf{a} \in \mathbb{R}^m \setminus \{\mathbf{0}\}$ mit $\sum_{i=1}^m a_i \mathbf{x_i} = 0$. Betrachte die zwei Indexmengen $I = \{i \in [m] : a_i > 0\}$ und $J = \{j \in [m] : a_j < 0\}$, von denen mindestens eine nichtleer ist.

Fall 1: $J = \emptyset$.

Dann folgt $I \neq \emptyset$ und $\sum_{i \in I} a_i \mathbf{x_i} = 0$. Nimm an, dass das Bitmuster $y_i = -1$ für alle $i \in I$ realisierbar ist, sagen wir mit dem Gewichtsvektor $\mathbf{w}$. Wir erhalten einen Widerspruch wie folgt:

$$0 > \sum_{i \in I} a_i \langle \mathbf{w}, \mathbf{x_i} \rangle = \langle \mathbf{w}, \sum_{i \in I} a_i \mathbf{x_i} \rangle = \langle \mathbf{w}, \mathbf{0} \rangle = 0 \ .$$

Fall 2: $J \neq \emptyset$.

Die Gleichung $\sum_{i=1}^m a_i \mathbf{x_i} = 0$ lässt sich umschreiben zu $\sum_{i \in I} a_i \mathbf{x_i} = \sum_{j \in J} |a_j| \mathbf{x_j}$. Nimm an, dass das Binärmuster $y_i = 1$ für alle $i \in I$ und $y_j = -1$ für alle $j \in J$ realisierbar ist, sagen wir durch den Gewichtsvektor $\mathbf{w}$. Wir erhalten einen Widerspruch wie folgt:

$$0 \leq \sum_{i \in I} a_i \langle \mathbf{w}, \mathbf{x_i} \rangle = \langle w, \sum_{i \in I} a_i \mathbf{x_i} \rangle = \langle w, \sum_{j \in J} |a_j| \mathbf{x_j} \rangle = \sum_{j \in J} |a_j| \langle \mathbf{w}, \mathbf{x_j} \rangle < 0 \ . \qquad (33)$$

Damit ist der Beweis des Lemmas abgeschlossen. •

Lemma 7.2 *Eine Punktmenge $S \subseteq \mathbb{R}^d$ ergänzt um den Nullvektor $\mathbf{0}$ ist durch HS_d aufspaltbar genau dann, wenn S aus linear unabhängigen Vektoren besteht.*

Beweis Aus der in (32) beschriebenen Darstellung affiner Funktionen aus LIN_d als lineare Funktionen aus LIN_{d+1}^0 folgt leicht: $S \cup \{\mathbf{0}\}$ ist aufspaltbar durch HS_d genau dann, wenn $S' := \{(1, \mathbf{x}) : \mathbf{x} \in S\} \cup \{(1, \mathbf{0})\}$ aufspaltbar durch HS_{d+1}^0 ist. Gemäß Lemma 7.1 ist dies der Fall genau dann, wenn S' aus linear unabhängigen Vektoren besteht. Dies wiederum gilt genau dann (wie man sich leicht überlegt), wenn die ursprüngliche Menge S aus linear unabhängigen Vektoren besteht. •

Theorem 7.3 $\mathrm{VCD}(\mathrm{HS}_d^0) = d$ *und* $\mathrm{VCD}(\mathrm{HS}_d) = d + 1$.

Beweis $\mathrm{VCD}(\mathrm{HS}_d^0) = d$ ergibt sich unmittelbar aus Lemma 7.1. Wir wissen bereits, dass HS_d sich als eine Unterklasse von HS_{d+1}^0 auffassen lässt. Daraus folgt $\mathrm{VCD}(\mathrm{HS}_d) \leq \mathrm{VCD}(\mathrm{HS}_{d+1}^0) = d + 1$. Aus Lemma 7.2 folgt, dass $\mathrm{VCD}(\mathrm{HS}_d) \geq d + 1$, da jede Basis von $\mathbb{R}^d$ ergänzt um den Nullvektor durch HS_d aufspaltbar ist. •

Punkte $\mathbf{x}$ mit $\langle \mathbf{w}, \mathbf{x} \rangle + b = 0$ erhalten vom Halbraum $H_{\mathbf{w},b}$ das Label 1, da $\text{sign}(0) = 1$. Es wäre befriedigender, wenn wir garantieren könnten, dass positive Labels nur aus positiven Werten von $\langle \mathbf{w}, \mathbf{x} \rangle + b$ resultieren. Dies kann man in der Tat auf endlichen Punktmengen immer erreichen, wie die folgenden Ausführungen zeigen werden.

Definition 7.4 *Es seien $\mathbf{w}, \mathbf{x} \in \mathbb{R}^d$ und $b \in \mathbb{R}$. Die Größe $|\langle \mathbf{w}, \mathbf{x} \rangle + b|$ heißt der* (vorzeichenlose) funktionale Randabstand *der affinen Hyperebene $H_{\mathbf{w},b}$ vom Punkt $\mathbf{x}$.*

Beachte, dass jeder von Null verschiedene funktionale Randabstand beliebig vergrößert werden kann, indem $\mathbf{w}$ und b um den gleichen positiven Faktor $\lambda > 0$ hochskaliert werden. Dabei bleibt das Vorzeichen unverändert: $\text{sign}(\langle \mathbf{w}, \mathbf{x} \rangle + b) = \text{sign}(\langle \lambda \mathbf{w}, \mathbf{x} \rangle + \lambda b)$.

Lemma 7.5 *Für eine endliche Punktmenge S in $\mathbb{R}^d$ gilt folgendes. Wenn ein Binärmuster auf den Punkten aus S mit einem affinen Halbraum realisierbar ist, dann ist dies bei geeigneter Wahl von (w, b) auch so möglich, dass $H_{w,b}$ einen funktionalen Randabstand von mindestens 1 zu jedem der Punkte aus S aufweist.*

Beweis Betrachte eine affine Hyperebene H, die ein vorgegebenes Binärmuster realisiert. Sie hat zu allen nicht auf ihr liegenden Punkten aus S einen echt positiven Randabstand. Durch hinreichend kleine Parallelverschiebung können wir erreichen, dass einerseits zu allen Punkten aus S ein positiver funktionaler Randabstand entsteht, aber andererseits das realisierte Binärmuster nicht verändert wird. Sind alle funktionalen Randabstände verschieden von 0, dann können wir sie durch Skalierung beliebig vergrößern. •

7.1.2 Die Berechnungskomplexität der Klasse der Halbräume

Beim *uniformen Kostenmäß* wird eine Zahl als eine Einheit gesehen und eine arithmetisch-logische Grundoperation auf zwei Zahlen zählt als 1 Rechenschritt. Beim *logarithmischen Kostenmaß* wird die Länge einer Zahl mit der Anzahl der Bits in ihrer Binärkodierung gemessen (was auch bei der Bestimmung der Eingabelänge eine Rolle spielt) und die Kosten einer arithmetisch-logischen Operation auf zwei Zahlen werden gleich gesetzt mit der Summe ihrer Bitlängen (d.h. im Wesentlichen: es wird die Anzahl der benötigten Bitoperationen gezählt).

Wir werden in diesem Abschnitt zeigen, dass das Konsistenzproblem für die Klasse $\mathcal{H}_d$ effizient (in Anhängigkeit vom Komplexitätsparameter d) lösbar ist, wenn wir $\mathbb{Q}^d$ (statt $\mathbb{R}^d$) als den Grundbereich ansehen und das logarithmische Kostenmaß zugrunde legen. Wir nutzen dabei aus, dass das Problem der linearen Programmierung — Optimierung einer linearen Zielfunktion unter linearen Randbedingungen vom Typ „$\geq$“ oder „$\leq$“ — effizient lösbar ist (zumindest bei Zugrundelegung des logarithmischen Kostenmaßes).[17]

[17] Die Frage, ob lineare Programmierung auch bezüglich uniformem Kostenmaß in Polynomialzeit lösbar ist, ist völlig offen. Bei logarithmischem Kostenmaß gibt es Polynomialzeitalgorithmen (zum Beispiel der Ellipsoid-Algorithmus oder „Interior-Point“-Methoden).

Ein Spezialfall von linearer Programmierung ist das Problem zu entscheiden, ob vorgegebene lineare Randbedingungen (in Variablen mit Werten in $\mathbb{Q}$) simultan erfüllt werden können. Wir werden im Beweis zu folgendem Satz zeigen, dass sich das Konsistenzproblem für affine Halbräume als ein solches Erfüllbarkeitsproblem darstellen lässt.

Theorem 7.6 *Das Konsistenzproblem zur Klasse* $\mathrm{HS} = (\mathrm{HS}_d)_{d\geq 1}$ *der affinen Halbräume (über dem Grundbereich* $(\mathbb{Q}^d)_{d\geq 1}$*) ist ein Teilproblem von linearer Programmierung und daher (bezüglich des logarithmischen Kostenmaßes) effizient lösbar.*

Beweis Das Konsistenzproblem zu $\mathrm{HS} = (\mathrm{HS}_d)_{d\geq 1}$ ist, wie wir wissen, folgendes Problem:

Eingabeinstanz: zwei endliche Punktmengen $S^+, S^- \subset \mathbb{Q}^d$ (wobei der obere Index „+“ bzw. „−“ die binäre Markierung anzeigt und $d \geq 1$ variabel ist)

Frage: Existieren $\mathbf{w} \in \mathbb{Q}^d$ und $b \in \mathbb{Q}$, so dass $S^+ \subseteq \mathrm{HS}^+_{\mathbf{w},b}$ und $S^- \subseteq \mathrm{HS}^-_{\mathbf{w},b}$?

Mit Lemma 7.5 ergibt sich, dass folgende Aussagen äquivalent sind:

$$\begin{aligned}
\exists \mathbf{w} \in \mathbb{Q}^d, b \in \mathbb{Q}: &\quad (S^+ \subseteq \mathrm{HS}^+_{\mathbf{w},b}) \wedge (S^- \subseteq \mathrm{HS}^-_{\mathbf{w},b}) \\
\exists \mathbf{w} \in \mathbb{Q}^d, b \in \mathbb{Q}: &\quad (\forall \mathbf{x} \in S^+ : \langle \mathbf{w}, \mathbf{x}\rangle + b \geq 0) \wedge (\forall \mathbf{x} \in S^- : \langle \mathbf{w}, \mathbf{x}\rangle + b < 0) \\
\exists \mathbf{w}' \in \mathbb{Q}^d, b' \in \mathbb{Q}: &\quad (\forall \mathbf{x} \in S^+ : \langle \mathbf{w}', \mathbf{x}\rangle + b' \geq 1) \wedge (\forall \mathbf{x} \in S^- : \langle \mathbf{w}', \mathbf{x}\rangle + b' \leq -1)
\end{aligned}$$

Wir landen somit bei der Frage, ob wir Werte an die Variablen $\mathbf{w} = (w_1, \ldots, w_d)$ und b so zuweisen können, dass die linearen Randbedingungen

$$(\forall \mathbf{x} \in S^+ : \langle \mathbf{w}, \mathbf{x}\rangle + b \geq 1) \wedge (\forall \mathbf{x} \in S^- : \langle \mathbf{w}, \mathbf{x}\rangle + b \leq -1)$$

simultan erfüllt sind (Spezialfall von linearer Programmierung). •

Wir merken kurz an, dass die Punktmengen S^+, S^- *linear separierbar* heißen, falls eine sie trennende Hyperebene $H_{\mathbf{w},b}$ im Sinne von

$$(\forall \mathbf{x} \in S^+ : \langle \mathbf{w}, \mathbf{x}\rangle + b > 0) \wedge (\forall \mathbf{x} \in S^- : \langle \mathbf{w}, \mathbf{x}\rangle + b < 0) \tag{34}$$

existiert. Unter der Realisierbarkeitsannahme des PAC-Lernmodells gilt, dass die durch $S \sim \mathcal{D}^m$ gegebenen Mengen S^+, S^- linear separierbar sind. Beim agnostischen PAC-Lernmodell befinden wir uns dagegen im nicht-separierbaren Fall. Es lässt sich zeigen (hier ohne Beweis), dass das Problem MinDisE(HS) NP-hart ist.

Effiziente Lösbarkeit des Konsistenzproblems und NP-Härte von „Minimum Disagreement“ haben Konsequenzen:

Theorem 7.7 *1. Die Klasse der affinen Halbräume ist unter der Realisierbarkeitsannahme effizient PAC-lernbar.*

2. Die Klasse der affinen Halbräume ist nicht effizient agnostisch PAC-lernbar (außer wenn RP = NP).

7.2 Das Rosenblatt'sche Perzeptron

Es sei $(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_m}, y_m) \in \mathbb{R}^d \times \{-1, 1\}$ eine Folge von markierten Trainigsbeispielen. Die Menge $\{\mathbf{x_1}, \ldots, \mathbf{x_m}\}$ zerfällt dann in $S^+ = \{\mathbf{x_i} : y_i = 1\}$ und $S^- = \{\mathbf{x_i} : y_i = -1\}$. Die Bedingung (34) für eine S^+ und S^- linear separierende Hyperebene lässt sich mit Hilfe der Labels y_i auch schreiben wie folgt:

$$\forall i = 1, \ldots, m : y_i(\langle \mathbf{w}, x_i \rangle + b) > 0 \ . \tag{35}$$

Im Jahre 1958 publizierte Rosenblatt einen Algorithmus, genannt Perzeptron, welcher $\mathbf{w}, b$ iterativ modifiziert, solange die Bedingung (35) verletzt ist. Wir beschreiben diesen Algorithmus der Einfachheit halber für den homogenen Fall (mit $b = 0$), wobei wir voraussetzen dürfen, dass alle $\mathbf{x_i}$ vom Nullvektor verschieden sind.[18]

Initialisierung: Setze $\mathbf{w^{(1)}} = \mathbf{0}$.

Hauptschleife: Für $t = 1, 2, 3, \ldots$ mache folgendes:
Falls ein $i \in [m]$ mit $y_i \langle \mathbf{w^{(t)}}, \mathbf{x_i} \rangle \leq 0$ existiert, dann setze

$$\mathbf{w^{(t+1)}} = \mathbf{w^{(t)}} + y_i \mathbf{x_i} \ . \tag{36}$$

Falls nicht, dann gibt $\mathbf{w^{(t)}}$ aus und stoppe.

Die Aktualisierung (36) arbeitet (im Sinne der Bedingung (35)) in die richtige Richtung, da

$$y_i \langle \mathbf{w^{(t+1)}}, \mathbf{x_i} \rangle = y_i \langle \mathbf{w^{(t)}} + y_i \mathbf{x_i}, \mathbf{x_i} \rangle = y_i \langle \mathbf{w^{(t)}}, \mathbf{x_i} \rangle + \langle \mathbf{x_i}, \mathbf{x_i} \rangle$$

und für $\mathbf{x_i} \neq \mathbf{0}$ gilt $\langle \mathbf{x_i}, \mathbf{x_i} \rangle > 0$.

Das folgende Resultat zeigt, dass das Perzeptron nach endlich vielen Iterationen eine separierende Hyperebene findet (sofern sie existiert):

Theorem 7.8 *Wir setzen den separablen Fall voraus. Es sei*

$$\begin{aligned} B &= \min\{\|\mathbf{w}\| : y_i \langle \mathbf{w}, \mathbf{x_i} \rangle \geq 1 \text{ für alle } i \in [m]\} \ , \\ R &= \max\{\|\mathbf{x_i}\| : i \in [m]\} \ . \end{aligned}$$

Sei T die Anzahl der Iterationen, die (36) durchlaufen. Mit diesen Notationen gilt: $T \leq R^2 B^2$ und der ausgegebene Gewichtsvektor $\mathbf{w^{(T+1)}}$ erfüllt die Separabilitätsbedingung (35).

Beweis Das Perzeptron steigt aus der Hauptschleife nur dann aus, wenn die Separabilitätsbedingung erfüllt ist. Es genügt also zu zeigen, dass die Anzahl T der Aktualisierungen des Gewichtsvektors durch $R^2 B^2$ nach oben beschränkt ist. Der Parameter B (s. oben) gibt die

[18] Vergleiche mit der Reduktion des d-dimensionalen inhomogenen Falles auf den $(d+1)$-dimensionalen homogenen Fall.

kleinstmögliche Euklidische Länge eines Gewichtsvektors an, der die Bedingung (35) mit einem funktionalen Randabstand von mindestens 1 erfüllt. Es sei $\mathbf{w}^*$ ein Gewichtsvektor mit $\|\mathbf{w}^*\| = B$, der das leistet. Unser Ziel ist der Nachweis von folgender Bedingung:

$$1 \geq \underbrace{\frac{\langle \mathbf{w}^*, \mathbf{w}^{(\mathbf{T+1})} \rangle}{\|\mathbf{w}^*\| \cdot \|\mathbf{w}^{(\mathbf{T+1})}\|}}_{=\cos(\mathbf{w}^*, \mathbf{w}^{(\mathbf{T+1})})} \geq \frac{\sqrt{T}}{RB} \ . \tag{37}$$

Wenn wir die Ungleichungen in (37) nach T auflösen, ergibt sich sofort $T \leq R^2B^2$. Die erste Ungleichung in (37) ergibt sich aus „Cauchy-Schwartz“. Der Nachweis der zweiten Ungleichung in (37) ergibt sich aus folgenden Rechnungen (mit den Notationen wie in der Hauptschleife des Perzeptron-Algorithmus):

1. $\langle \mathbf{w}^*, \mathbf{w}^{(\mathbf{t+1})} \rangle - \langle \mathbf{w}^*, \mathbf{w}^{(\mathbf{t})} \rangle = \langle \mathbf{w}^*, y_i \mathbf{x_i} \rangle = y_i \langle \mathbf{w}^*, \mathbf{x_i} \rangle \geq 1$.
   Mit $\langle \mathbf{w}^*, \mathbf{w}^{(\mathbf{1})} \rangle = \langle \mathbf{w}^*, \mathbf{0} \rangle = 0$ ergibt sich induktiv $\langle \mathbf{w}^*, \mathbf{w}^{(\mathbf{T+1})} \rangle \geq T$.
2. $\|\mathbf{w}^{(\mathbf{t+1})}\|^2 = \|\mathbf{w}^{(\mathbf{t})} + y_i \mathbf{x_i}\|^2 = \|\mathbf{w}^{(\mathbf{t})}\|^2 + \underbrace{2 y_i \langle \mathbf{w}^{(\mathbf{t})}, \mathbf{x_i} \rangle}_{\leq 0} + \|\mathbf{x_i}\|^2 \leq \|\mathbf{w}^{(\mathbf{t})}\|^2 + R^2$.
   Mit $\|\mathbf{w}^{(\mathbf{1})}\|^2 = \|\mathbf{0}\|^2 = 0$ ergibt sich hieraus induktiv $\|\mathbf{w}^{(\mathbf{T+1})}\|^2 \leq TR^2$ und somit $\|\mathbf{w}^{(\mathbf{T+1})}\| \leq \sqrt{T}R$.

Aus $\langle \mathbf{w}^*, \mathbf{w}^{(\mathbf{T+1})} \rangle \geq T$, $\|\mathbf{w}^*\| = B$ und $\|\mathbf{w}^{(\mathbf{T+1})}\| \leq \sqrt{T}R$ folgt unmittelbar

$$\frac{\langle \mathbf{w}^*, \mathbf{w}^{(\mathbf{T+1})} \rangle}{\|\mathbf{w}^*\| \cdot \|\mathbf{w}^{(\mathbf{T+1})}\|} \geq \frac{T}{B\sqrt{T}R} = \frac{\sqrt{T}}{RB} \ ,$$

was den Beweis abschließt. •

Im „worstcase“ können B und R exponentiell von der Eingabelänge (in binärer Kodierung) abhängen. I.A. ist also das Perzeptron kein effizienter Algorithmus. Wenn jedoch die Parameter R und B nur moderat groß sind (was bei vielen Anwendungen der Fall sein mag), dann ist die obere Schranke R^2B^2 gar nicht so übel.

7.3 Lineare Regression

In diesem Abschnitt operieren wir mit dem Grundbereich $\mathcal{X} = \mathbb{R}^d$ und der Labelmenge $\mathcal{Y} = \mathbb{R}$. Eine Trainingssequenz hat daher die Form

$$S = [(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_m}, y_m)] \in (\mathbb{R}^d \times \mathbb{R})^m \ . \tag{38}$$

Wir betrachten die Hypothesenklasse LIN_d der affinen Funktionen gemäß (31) und die quadratische Verlustfunktion

$$\ell(h, (\mathbf{x}, y)) = (h(\mathbf{x}) - y)^2 \ .$$

Es sei $\mathcal{D}$ eine Verteilung auf $\mathbb{R}^d \times \mathbb{R}$. Wie üblich setzen wir uns das Ziel, eine Hypothese $h \in \mathrm{LIN}_d$ mit möglichst kleinem Wert von $L_{\mathcal{D}}(h)$ (dem erwarteten quadratischen Fehler von h bezüglich $\mathcal{D}$) zu finden.

Warnung: Die auf VC-Dimension aufbauende Theorie ist nicht unmittelbar auf nicht-binäre Voraussageprobleme anwendbar. Evtl. werden wir zu einem späteren Zeitpunkt der Vorlesung klären, wie groß Trainingssequenzen bei Regressionsproblemen (oder allgemeiner bei nicht-binären Voraussageproblemen) zu sein haben.

Beispiel 7.9 *Das Erraten des Gewichtes eines Babys anhand seines Geburtsgewichtes und seines Alters (auf Basis einer gegebenen Trainingssequenz) ist ein 2-dimensionales Regressionsproblem.*

Wir betrachten im Folgenden die ERM-Lernregel: gegeben eine Trainingssequenz S, finde eine Hypothese $h \in \mathrm{LIN}_d$ mit möglichst kleinem Wert von $\hat{L}_S(h)$ (dem mittleren quadratischen Fehler von h auf der Trainingssequenz S). Im Falle $d = 1$ wird eine der ERM-Lernregel entsprechende Hypothese als „Ausgleichsgrade“ bezeichnet. Die Methode zum Berechnen einer Hypothese h gemäß der ERM-Lernregel bei beliebig hoher Dimension d ist unter dem Namen „Least Squares“ bekannt.

7.3.1 Die „Least Squares“-Methode

Wir beschränken uns der Einfachheit halber wieder auf den homogenen Fall (mit $b = 0$) und betrachten das Problem

$$\mathrm{argmin}_{\mathbf{w}\in\mathbb{R}^d} \hat{L}_S(h_{\mathbf{w},0}) = \mathrm{argmin}_{\mathbf{w}\in\mathbb{R}^d} \left\{ \frac{1}{m} \cdot \sum_{i=1}^{m} \left(\langle \mathbf{w}, \mathbf{x_i} \rangle - y_i\right)^2 \right\} \ .$$

Wenn wir den Gradienten der Zielfunktion auf Null setzen, erhalten wir die Gleichung

$$\frac{2}{m} \sum_{i=1}^{m} (\langle \mathbf{w}, \mathbf{x_i} \rangle - y_i)\mathbf{x_i} = 0 \ .$$

Dies ist äquivalent zu

$$0 = \sum_{i=1}^{m} \mathbf{x_i}(\mathbf{x_i}^\top \mathbf{w} - y_i) = \left(\sum_{i=1}^{m} \mathbf{x_i}\mathbf{x_i}^\top \right) \mathbf{w} - \sum_{i=1}^{m} y_i \mathbf{x_i} \ .$$

Wenn wir

$$A = \sum_{i=1}^{m} \mathbf{x_i}\mathbf{x_i}^\top \ \text{ und } \ \mathbf{b} = \sum_{i=1}^{m} y_i \mathbf{x_i} \tag{39}$$

setzen, haben wir also das lineare Gleichungssystem $A\mathbf{w} = \mathbf{b}$ zu lösen.

Beachte: $\mathbf{x_i} \in \mathbb{R}^d$ ist ein Spaltenvektor und, als Matrix aufgefasst, eine $(d \times 1)$-Matrix. Somit ist $\mathbf{x_i}^\top$ eine $(1 \times d)$-Matrix und $\mathbf{x_i}\mathbf{x_i}^\top$ (sowie die Matrix A) eine $(d \times d)$-Matrix. Der Eintrag in Zeile j und Spalte k von $\mathbf{x_i}\mathbf{x_i}^\top$ ist gleich $x_i(j)x_i(k)$, also die j-te Komponente des Vektors $\mathbf{x_i}$ multipliziert mit seiner k-ten Komponente.

Setze $r = \text{rank}(A)$. Ein sympathisch einfacher Fall wäre $r = d$. Dann besitzt A eine Inverse A^{-1} und $\mathbf{w} = A^{-1}\mathbf{b}$ wäre eine optimale Wahl von $\mathbf{w}$. Wir wollen uns im Folgenden aber mit dem interessanteren Fall $r < d$ beschäftigen.

Mit $\langle A \rangle$ bezeichnen wir den Vektorraum, der von den Spaltenvektoren von A aufgespannt wird. Das Gleichungssystem $A\mathbf{w} = \mathbf{b}$ ist genau dann lösbar, wenn $\mathbf{b} \in \langle A \rangle$. Dies ist bei unserer Anwendung glücklicherweise der Fall:

Lemma 7.10 *Wenn A und $\mathbf{b}$ gemäß (39) gewählt werden, dann gilt $\mathbf{b} \in \langle A \rangle$.*

Den Beweis des Lemmas holen wir weiter unten nach.

Das folgende Resultat klärt, wie wir im Falle der Lösbarkeit an eine Lösung $\mathbf{w}$ des Gleichungssystems $A\mathbf{w} = \mathbf{b}$ gelangen.

Lemma 7.11 *Falls $\mathbf{b} \in \langle A \rangle$, dann existiert eine Matrix $A^\dagger$ (genannt das* Pseudo-Inverse *zu A) mit*

$$A(A^\dagger \mathbf{b}) = (AA^\dagger)\mathbf{b} = \mathbf{b} \ .$$

Weiterhin läßt sich $A^\dagger$ aus A und $\mathbf{b}$ effizient berechnen.

Auch den Beweis dieses Lemmas holen wir weiter unten nach.

Im Falle $r = d$ stimmt das Pseudo-Inverse $A^\dagger$ von A mit der Inversen A^{-1} überein. Wir erhalten somit das Hauptresultat dieses Abschnittes:

Theorem 7.12 *S sei gemäß (38) und $A, \mathbf{b}$ seien gemäß (39) gegeben. Dann ist $\mathbf{w} = A^\dagger \mathbf{b}$ ein der ERM-Lernregel (bezüglich quadratischem Fehler) entsprechender Gewichtsvektor. Er lässt sich effizient aus S ermitteln.*

Am Ende dieses Abschnittes holen wir die zwei fehlenden Beweise nach.

Beweis[Lemma 7.10] Der Beweis benutzt die aus der linearen Algebra bekannte Tatsache, dass jede (reelle) Matrix M die Gleichung $\text{rank}(MM^\top) = \text{rank}(M)$ erfüllt. Es ist leicht nachzurechnen, dass für $M = [\mathbf{x_1} \dots \mathbf{x_m}]$ (also die Matrix mit den Spaltenvektoren $\mathbf{x_1}, \dots, \mathbf{x_m}$) die Gleichung

$$\sum_{i=1}^{m} \mathbf{x_i}\mathbf{x_i}^\top = MM^\top$$

gilt. Da $A = \sum_{i=1}^{m} \mathbf{x_i}\mathbf{x_i}^\top = MM^\top$ und da die Spaltenvektoren der Matrix $MM^\top$ Linearkombinationen von M's Spaltenvektoren sind, können wir $\langle A \rangle \subseteq \langle M \rangle$ folgern. Wegen $A = MM^\top$ haben A und M den gleichen Rang. Daher gilt sogar $\langle A \rangle = \langle M \rangle$. Wegen $\mathbf{b} = \sum_{i=1}^{m} y_i \mathbf{x_i}$ und $M = [\mathbf{x_1} \dots \mathbf{x_m}]$ gilt $\mathbf{b} \in \langle M \rangle$ und damit auch $\mathbf{b} \in \langle A \rangle$. •

Beweis[Lemma 7.11] Es gibt viele Möglichkeiten, sich ein Pseudo-Inverses einer symmetrischen Matrix A zu besorgen. Wir konstruieren im Folgenden die sogenannte Moore-Penrose

Inverse von A. Zu diesem Zweck benutzen wir das aus der linearen Algebra bekannte Spektraltheorem, welches folgendes besagt. Jede symmetrische $(d \times d)$-Matrix A vom Range r (wie zum Beispiel $A = \sum_{i=1}^{m} \mathbf{x_i}\mathbf{x_i}^\top$) besitzt eine (effizient berechenbare) Zerlegung der Form

$$A = VDV^\top \quad \text{mit} \quad D = \text{diag}(\lambda_1, \ldots, \lambda_r, 0, \ldots, 0) \ .$$

Hierbei gilt:

1. $V = [\mathbf{v_1}, \ldots \mathbf{v_d}]$ bezeichnet eine orthogonale Matrix (d.h., $VV^\top = V^\top V = I$ mit $I =$ Einheitsmatrix) und es gilt $\langle A \rangle = \langle \mathbf{v_1}, \ldots, \mathbf{v_r} \rangle$.

2. diag(...) bezeichnet eine Diagonalmatrix mit den in „..." aufgelisteten Werten auf der Hauptdiagonalen und $\lambda_1, \ldots, \lambda_r$ sind die r nicht-trivialen (also von 0 verschiedenen) reellen Eigenwerte von A.

Wir setzen nun $D^\dagger = \text{diag}(\lambda_1^{-1}, \ldots, \lambda_r^{-1}, 0, \ldots, 0)$, $A^\dagger = VD^\dagger V^\top$ und rechnen nach:

$$\begin{aligned} AA^\dagger \mathbf{b} &= VDV^\top VD^\dagger V^\top \mathbf{b} \\ &= VDD^\dagger V^\top \mathbf{b} = V\text{diag}(1, \ldots, 1, 0, \ldots, 0)V^\top \mathbf{b} \\ &= \sum_{i=1}^{r} \mathbf{v_i}\mathbf{v_i}^\top \mathbf{b} = \sum_{i=1}^{r} \langle \mathbf{v_i}, \mathbf{b} \rangle \mathbf{v_i} \ . \end{aligned}$$

Der letzte Ausdruck beschreibt die Projektion von $\mathbf{b}$ auf $\langle \mathbf{v_1}, \ldots, \mathbf{v_r} \rangle = \langle A \rangle$. Da wir $\mathbf{b} \in \langle A \rangle$ vorausgesetzt haben, muss diese Projektion mit $\mathbf{b}$ übereinstimmen. •

7.3.2 Reduktion von polynomieller auf lineare Regression

Es seien $\mathcal{H} = (\mathcal{H}_n)_{n \geq 1}$ und $\mathcal{H}' = (\mathcal{H}'_n)_{n \geq 1}$ zwei parameterisierte Hypothesenklassen reellwertiger Funktionen; $\mathcal{X} = (\mathcal{X}_n)_{n \geq 1}$ und $\mathcal{X}' = (\mathcal{X}'_n)_{n \geq 1}$ seien die zugehörigen Grundbereiche. Eine *polynomielle L-Reduktion* von $\mathcal{H}$ auf $\mathcal{H}'$, notiert als $\mathcal{H} \leq_{pol}^{L} \mathcal{H}'$, ist so definiert (mit einer Instanzen- und zwei Hypothesentransformationen) wie wir das von binären Hypothesenklassen kennen. Der einzige Unterschied besteht darin, dass die beteiligten Klassen jetzt reellwertige Funktionen enthalten.

Theorem 7.13 *Unter der Voraussetzung $\mathcal{H} \leq_{pol}^{L} \mathcal{H}'$ gilt folgendes:*

1. *Wenn das Regressionsproblem zur Klasse $\mathcal{H}'$ effizient lösbar ist (im Sinne des üblichen (ε, δ)-Kriteriums), so gilt dies auch für das Regressionsproblem zur Klasse $\mathcal{H}$.*

2. *Wenn die ERM-Lernregel für $\mathcal{H}'$ effizient implementiert werden kann (Auffinden einer Hypothese aus $\mathcal{H}'$ mit kleinstmöglichem quadratischen Fehler auf den Trainingsdaten), so gilt dies entsprechend auch für die Klasse $\mathcal{H}$.*

Beweis Wir beschränken uns auf den Beweis der zweiten Aussage. Es sei

$$S = [(x_1, y_1), \ldots, (x_m, y_m)] \in (\mathcal{X}_n \times \mathbb{R})^m$$

eine gegebene Trainingssequenz. Zum Auffinden einer ERM-Hypothese aus $\mathcal{H}$ gehe vor wie folgt:

1. Berechne $S' = [(x'_1, y_1), \ldots, (x'_m, y_m)] \in (\mathcal{X}'_{n'} \times \mathbb{R})^m$. Dies ist eine m-fache Anwendung der Instanzentransformation $x \mapsto x'$.

2. Berechne eine ERM-Hypothese $h' \in \mathcal{H}'_{n'}$ zu der Trainingssequenz S'.

3. Berechne die Hypothese $h \in \mathcal{H}_n$, die sich aus h' gemäß der zweiten Hypothesentransformation ergibt.

4. Gib h als ERM-Hypothese zur Trainingssequenz S aus.

Fixiere eine beliebige Hypothese $h_* \in \mathcal{H}_n$. Wir haben $\hat{L}_S(h) \leq \hat{L}_S(h_*)$ zu zeigen. Dies geht mit einem „Schach Matt“ in drei Zügen:

- Es sei h'_* die Hypothese, die sich aus h_* gemäß der ersten Hpothesentransformation ergibt. Da $h_*(x_i) = h'_*(x'_i)$ und da S' die reellwertigen Labels von S übernommen hat, folgt $\hat{L}_{S'}(h'_*) = \hat{L}_S(h_*)$.

- Da h' eine ERM-Hypothese zu S' ist, folgt $\hat{L}_{S'}(h') \leq \hat{L}_{S'}(h'_*)$.

- Da h sich aus h' über $h' \mapsto h$ ergibt, gilt wieder $h(x_i) = h'(x'_i)$, was $\hat{L}_S(h) = \hat{L}_{S'}(h')$ zur Folge hat.

Wenn wir die drei Beobachtungen zusammensetzen, erhalten wir $\hat{L}_S(h) = \hat{L}_{S'}(h') \leq \hat{L}_{S'}(h'_*) = \hat{L}_S(h_*)$, was den Beweis abschließt. •

Es bezeichne POL_d die Klasse der Polynome in d Variablen über $\mathbb{R}$, d.h.,

$$POL_d = \{\mathbf{x} \mapsto p(\mathbf{x}) : p \in \mathbb{R}[X_1, \ldots, X_d]\} \ .$$

Weiter bezeichne POL_d^n die Teilklasse der Polynome vom Maximalgrad n. Der Grad eines Monoms der Form $X_1^{n_1} \ldots X_d^{n_d}$ ist per Definition gleich $n_1 + \ldots + n_d$ (und der Grad eines Polynoms ist das Maximum der Grade seiner Monome).

Lemma 7.14 *Es gibt* $\binom{n+d}{d}$ *Monome vom Maximalgrad* n *in* d *Variablen.*

Beweis Es gibt genau $\binom{n+d}{d}$ geordnete Partitionen der Zahl n in $d+1$ Summanden aus $\mathbb{N}_0$ (eine aus der Vorlesung über „Diskrete Mathematik“ bekannte Tatsache). Da diese Zahlpartitionen und die Monome vom Maximalgrad n in d Variablen sich 1-zu-1 entsprechen, ergibt sich die Aussage des Lemmas. •

Beispiel 7.15 *Es sei $n = 6$ und $d = 3$. Das Monom $X_1X_3^2 = 1^3X_1^1X_2^0X_3^2$ entspricht der geordneten Partition von $n = 6$ in die Summanden $(3, 1, 0, 2)$, und umgekehrt.*

Theorem 7.16 *Es sei d eine beliebige aber fest gewählte Konstante, $\mathrm{LIN} = (\mathrm{LIN}_n)_{n\geq 1}$ und $POL_d = (POL_d^n)_{n\geq 1}$. Mit diesen Bezeichnungen gilt: $POL_d \leq_{pol}^L \mathrm{LIN}$.*

Beweis Wir setzen $n' = \binom{n+d}{d} = O(n^d)$. Die Abbildung $n \mapsto n'$ ist injektiv und (da d eine Konstante ist) polynomiell in n beschränkt. Es sei $M_1, \ldots, M_{n'}$ eine beliebige aber feste Auflistung aller Monome vom Maximalgrad n in d Variablen. Es sei $X'_1, \ldots, X'_{n'}$ eine entsprechende Auflistung von neuen Variablen. Wenn wir in einem Polynom $h \in POL_d^n$ jedes Monom M_i durch die Variable X'_i ersetzen, erhalten wir eine lineare Funktion h'. Dies liefert die Hypothesentransformation $h \mapsto h'$. Die Rücksubstitution von M_i für X'_i ergibt die Hypothesentransformation $h' \mapsto h$. Es sei $\mathbf{x} = (x_1, \ldots, x_d) \in \mathbb{R}^d$. Es bezeichne $M_i(\mathbf{x})$ die Auswertung von M_i an der Stelle $\mathbf{x}$. Wir setzen $\mathbf{x}' = (M_1(\mathbf{x}), \ldots, M_{n'}(\mathbf{x}))$. Dies liefert die Instanzentransformation $\mathbf{x} \mapsto \mathbf{x}'$. Es ist leicht zu verifizieren, dass diese Transformationen alle Definitionskriterien einer polynomiellen L-Reduktion erfüllen. •

Indem wir die Theoreme 7.12, 7.13 und 7.16 kombinieren, erhalten wir die

Folgerung 7.17 *Eine ERM-Hypothese (bezüglich quadratischem Fehler) ist für die Klasse POL_d in Polynomialzeit berechenbar.*

7.4 Logistische Regression

Es bezeichne

$$S(z) = \frac{1}{1 + \exp(-z)}$$

die sogenannte *Sigmoid-Funktion*. Sie hat folgende (leicht zu verifizierende) Eigenschaften:

1. S is streng monoton wachsend.
2. $\lim_{z\to-\infty} S(z) = 0$, $\lim_{z\to\infty} S(z) = 1$ und $S(0) = 1/2$.

Wir können S als eine glatte Version einer von 0 auf 1 wechselnden Schwellenfunktion ansehen. Wir definieren die Klasse SIG_d^0 wie folgt:

$$\mathrm{SIG}_d^0 = S \circ \mathrm{LIN}_d^0 = \{\mathbf{x} \mapsto S(\langle \mathbf{w}, \mathbf{x}\rangle) : \mathbf{w} \in \mathbb{R}^d\} \ .$$

Wir setzen der Kürze halber

$$h_{\mathbf{w}}(\mathbf{x}) = S(\langle \mathbf{w}, \mathbf{x}\rangle) = \frac{1}{1 + \exp(-\langle \mathbf{w}, \mathbf{x}\rangle)} \ .$$

Lemma 7.18 *Wenn wir $h_{\mathbf{w}}(\mathbf{x})$ (bzw. $1 - h_{\mathbf{w}}(\mathbf{x})$) interpretieren als die Wahrscheinlichkeit, eine Instanz $\mathbf{x}$ mit dem Label 1 (bzw. -1) zu versehen, dann ist die Wahrscheinlichkeit $\mathbf{x}$ mit dem Label $y \in \{-1, 1\}$ zu versehen gegeben durch $(1 + \exp(-y\langle \mathbf{w}, \mathbf{x}\rangle))^{-1}$.*

Beweis Die Aussage ist für $y = 1$ offensichtlich richtig. Eine einfache Rechnung zeigt, dass

$$1 - h_{\mathbf{w}}(\mathbf{x}) = \frac{1}{1 + \exp(\langle \mathbf{w}, \mathbf{x} \rangle)} \ .$$

Somit stimmt die Aussage auch für $y = -1$. •

Die *logistische Verlustfunktion* ist definiert wie folgt:

$$\ell(h_{\mathbf{w}}, (\mathbf{x}, y)) = \ln(1 + \exp(-y\langle \mathbf{w}, \mathbf{x} \rangle)) \ .$$

Sie bestraft eine Hypothese, die das Label y für $\mathbf{x}$ mit Wahrscheinlichkeit p korrekt voraussagt, mit $\ln(1/p)$. Dieser Strafterm ist gleich 0 für $p = 1$ und konvergiert mit logarithmischer Geschwindigkeit gegen ∞ für $p \to 0$.

Wir sagen eine Verlustfunktion ℓ *entspricht dem „Maximum Likelihood (ML)"-Prinzip*, wenn die ERM-Lernregel bezüglich ℓ zu einer Hypothese führt, welche die Wahrscheinlichkeit einer gegebenen Trainingssequenz maximiert.

Theorem 7.19 *Wir interpretieren $h_{\mathbf{w}}(\mathbf{x})$ (bzw. $1 - h_{\mathbf{w}}(\mathbf{x})$) als die Wahrscheinlichkeit, eine Instanz $\mathbf{x}$ mit dem Label 1 (bzw. -1) zu versehen. Weiterhin unterstellen wir, dass die zufälligen Markierungen verschiedener Instanzen unabhängig von einander erfolgen. Unter diesen Voraussetzungen entspricht die logistische Verlustfunktion dem ML-Prinzip.*

Beweis Es sei $S = [(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_m}, y_m)] \in (\mathbb{R}^d \times \{-1, 1\})^m$ eine gegebene Trainingssequenz. Die S durch $h_{\mathbf{w}}$ zugewiesene Wahrscheinlichkeit ist dann gegeben durch

$$p_S(\mathbf{w}) = \prod_{i=1}^{m} \frac{1}{1 + \exp(-y_i \langle \mathbf{w}, \mathbf{x_i} \rangle)} \ .$$

Offensichtlich sind die folgenden Optimierungsprobleme äquivalent:

- Maximieren von $p_S(\mathbf{w})$.
- Minimieren von $\prod_{i=1}^{m}(1 + \exp(-y_i \langle \mathbf{w}, \mathbf{x_i} \rangle))$
- Minimieren von $\frac{1}{m} \sum_{i=1}^{m} \ln(1 + \exp(-y_i \langle \mathbf{w}, \mathbf{x_i} \rangle))$.

Die letzte Zielfunktion ist identisch zu $\hat{L}_S(h_{\mathbf{w}})$, wenn wir die logistische Verlustfunktion zugrunde legen. •

Wir merken kurz an, dass die Zielfunktion

$$\hat{L}_S(h_{\mathbf{w}}) = \frac{1}{m} \sum_{i=1}^{m} \ln(1 + \exp(-y_i \langle \mathbf{w}, \mathbf{x_i} \rangle))$$

eine in $\mathbf{w}$ konvexe Funktion ist. Zur Minimierung konvexer Funktionen existieren effiziente Standardwerkzeuge (zum Beispiel „Interior Point"-Methoden). Die ERM-Lernregel bei Zugrundelegung der logistischen Verlustfunktion ist daher effizient implementierbar.

8 Boosting

In diesem Kapitel gehen wir der Frage nach: „Kann man aus den Ratschlägen eines Narren Weisheit schöpfen?“ Die Rolle des „Narren“ übernimmt ein Algorithmus, den wir als „schwachen Lerner“ bezeichnen. Seine Voraussagen sind nur etwas besser als zufälliges Raten, d.h., er sagt ein binäres Label mit einer etwas unter 1/2 liegenden Fehlerrate voraus. Den Part der „Weisheit“ übernimmt eine Voraussagefunktion, die ein gewichtetes Majoritätsvotum über eine Kollektion von schwachen Hypothesen durchführt. Kern dieses Kapitels ist eine auf Freund und Schapire zurückgehende Methode namens AdaBoost, welche einen schwachen Lerner in einen (starken) PAC-Lerner (mit einer Fehlerrate nahe bei 0) transformiert.

In Abschnitt 8.1 beschäftigen wir uns mit dem Konzept eines „schwachen Lerners“. Abschnitt 8.2 liefert eine obere Schranke für die VC-Dimension der Klasse der Linearkombinationen von Basishypothesen. Dies wird uns später helfen, den Schätzfehler von AdaBoost zu kontrollieren. Der das Kapitel abschließende Abschnitt 8.3 ist AdaBoost und seiner Analyse gewidmet.

8.1 Schwache Lernbarkeit

Definition 8.1 *Es sei $0 < \gamma < 1/2$. $\mathcal{H}$ und $\mathcal{B}$ seien Hypothesenklassen mit Grundbereich $\mathcal{X}$ und Wertebereich $\mathcal{Y} = \{\pm 1\}$.*

- *Wir sagen ein Lernalgorithmus A* lernt *$\mathcal{H}$ γ-*schwach mit Hypothesen aus *$\mathcal{B}$, wenn eine Funktion $m_{\mathcal{H}} : (0,1) \to \mathbb{N}$ existiert, so dass für jedes $\delta \in (0,1)$, jede Verteilung $\mathcal{D}$ über $\mathcal{X} \times \{0,1\}$, welche bezüglich $\mathcal{H}$ die Realisierbarkeitsannahme erfüllt, folgendes gilt. Wenn A auf einer Trainingssequenz $S \sim \mathcal{D}^m$ mit $m \geq m_{\mathcal{H}}(\delta)$ gestartet wird, dann ist mit einer Erfolgswahrscheinlichkeit von mindestens $1 - \delta$ die Ausgabe von A eine Hypothese $h \in \mathcal{B}$ mit $L_{\mathcal{D}}(h) \leq 1/2 - \gamma$.*

- *$\mathcal{H}$ heißt* γ-schwach lernbar *mit Hypothesen aus $\mathcal{B}$, wenn (unter der Realisierbarkeitsannahme) ein entsprechender γ-schwacher Lerner existiert.*

Beispiel 8.2 *Es sei $\mathcal{X} = \mathbb{R}$. Weiter sei $\mathcal{G}_T$ die Klasse aller Funktionen von $\mathbb{R}$ nach $\{-1, 1\}$ mit höchstens T Vorzeichenwechseln, wenn wir die reelle Zahlengerade von links nach rechts durchlaufen (= Klasse der ± 1-wertigen $(T+1)$-stückweise konstanten Funktionen). Zum Beispiel ist jede Funktion aus $\mathcal{G}_2$ gegeben durch eine Zerlegung der reellen Zahlengeraden in drei (oder weniger) Intervalle I_1, I_2, I_3, geordnet von links nach rechts, auf denen entweder das Binärmuster $1, -1, 1$ oder das Binärmuster $-1, 1, -1$ angenommen wird. Eine* Schwellenfunktion (mit Vorzeichen) *ist eine Funktion der Form $x \mapsto b \cdot sign(x - \theta)$ mit $\theta \in \mathbb{R}$ und $b \in \{-1, 1\}$. Sie ordnet x den Wert b zu, falls $x \geq \theta$ und den Wert $-b$ andernfalls. Es sei* DS_1 *die Klasse aller Funktionen dieser Form. Wir zeigen, dass ein ERM-Algorithmus für die Klasse* DS_1 *zu einem 1/12-schwachen Lerner für die Klasse $\mathcal{G}_2$ aufgebaut werden kann. Es sei S eine Trainingssequenz, die oBdA so markiert ist, dass Punkte auf den Intervallen I_1, I_2, I_3 jeweils die Werte $-1, 1, -1$ annehmen. (Der Fall mit vertauschten ± 1-Labels oder mit weniger als drei Intervallen ist analog zu behandeln.) Es ist leicht zu sehen, dass es zu*

jedem Intervall $I \in \{I_1, I_2, I_3\}$ eine Hypothese aus DS_1 *gibt, die nur die Trainingspunkte in I falsch markiert. Daher gilt für eine ERM-Hypothese $h_S \in \mathrm{DS}_1$: $L_S(h_S) \leq 1/3 = 1/2 - 1/6$. Die VC-Dimension von* DS_1 *ist gleich 2 (einfach zu sehen). Daher hat* DS_1 *die „Uniform Convergence Property (UCP)". Es sei $\mathcal{D}$ die auf $\mathbb{R} \times \{\pm 1\}$ zugrunde liegende Verteilung. Wir können die Größe der Trainingssequenz S so bemessen, dass mit einer Wahrscheinlichkeit von mindestens $1 - \delta$ die absolute Differenz von $L_{\mathcal{D}}(h)$ und $L_S(h)$ für alle $h \in \mathrm{DS}_1$ durch $1/12$ beschränkt ist. Dann gilt: $L_{\mathcal{D}}(h_S) \leq 1/2 - 1/6 + 1/12 = 1/2 - 1/12$.*

Die Funktionen der Klasse DS_1 werden auch *Entscheidungsstümpfe (decision stumps)* über $\mathbb{R}$ genannt (hauptsächlich weil sie als rudimentäre Entscheidungsbäume gesehen werden können).
Es sei

$$S = [(x_1, y_1), \ldots, (x_m, y_m)] \in (\mathbb{R} \times (\mathbb{R} \setminus \{0\}))^m \quad \text{mit} \quad x_1 < x_2 < \ldots < x_m \tag{40}$$

eine „sortierte gewichtete Trainingssequenz". Die Gewichte können wahlweise Mehrfachvorkommen einer Instanz oder auch Wahrscheinlichkeiten repräsentieren. Zum Beispiel könnte $(x_i, -3)$ das 3-fache Vorkommen einer mit -1 markierten Instanz x_i repräsentieren.

Wir wollen im Folgenden überlegen, welcher Entscheidungsstumpf auf S die kleinste (gewichtete) Anzahl von Fehlern macht. Es ist leicht zu sehen, dass wir nur Hypothesen aus DS_1 mit $\theta \in \{x_1, \ldots, x_m\}$ in Betracht ziehen müssen. Für $j = 1, \ldots, m$ sei

$$W^+(j) = \sum_{i \in [j]: y_i > 0} y_i \quad \text{und} \quad W^-(j) = \sum_{i \in [j]: y_i < 0} |y_j| \ .$$

Weiter sei $W^+ = W^+(m)$ bzw. $W^- = W^-(m)$ die Summe aller positiven Gewichte bzw. aller Absolutbeträge von negativen Gewichten. Die folgende Funktion wird sich als nützlich erweisen:

$$F(j) = W^+(j) - W^-(j) = \sum_{i \in [j]} y_i \ ,$$

d.h., $F(j)$ ist die Summe der ersten j Gewichte, wobei $F(0) = 0$. Es bezeichne DS_1^+ (bzw. DS_1^-) die Menge der Hypothesen aus DS_1 mit $b = 1$ (bzw. $b = -1$).

Lemma 8.3 *Es sei $j \in [m]$. Dann gilt folgendes. Die Hypothese $h_j^+(x) = sign(x - x_j)$ macht auf S insgesamt $W^- + F(j-1)$ Fehler und die Hypothese $h_j^-(x) = -sign(x - x_j)$ macht auf S insgesamt $W^+ - F(j-1)$ Fehler.*

Beweis Wir beweisen die erste Aussage. (Der Beweis für die zweite Aussage ist analog zu führen.) Die Hypothese h_j^+ macht $W^+(j-1)$ Fehler auf $(x_1, y_1), \ldots, (x_{j-1}, y_{j-1})$ und $W^- - W^-(j-1)$ Fehler auf $(x_j, y_j), \ldots, (x_m, y_m)$. Die gesamte Fehlerzahl von h_j^+ beträgt demnach $W^+(j-1) + (W^- - W^-(j-1)) = W^- + F(j-1)$, wie behauptet. •

Dies führt zu dem folgenden Algorithmus A_{DS}, der die ERM-Lernregel für die Klasse DS_1 implementiert:

1. Transformiere eine Trainingssequenz in die zugehörige sortierte gewichtete Trainingssequenz S der Form (40).

2. Berechne die Größen W^+, W^- und $F(j)$ für $j = 0, 1, \ldots, m$.

3. Bestimme $j^+ = \text{argmin}_{j\in[m]} F(j-1)$ und $j^- = \text{argmax}_{j\in[m]} F(j-1)$.

4. Falls $W^- + F(j^+ - 1) \leq W^+ - F(j^- - 1)$, dann gib die Hypothese $\text{sign}(x - x_{j^+})$ aus, andernfalls die Hypothese $-\text{sign}(x - x_{j^-})$.

Die Laufzeit für den ersten Schritt wird vom Sortieren dominiert und beträgt somit $O(|S| \log |S|)$. Die weiteren Schritte lassen sich in Linearzeit ausführen. Wir fassen diese Diskussionen in folgendem Resultat zusammen:

Lemma 8.4 *Die ERM-Lernregel ist für die Hypothesenklasse* DS_1 *mit Laufzeit* $O(|S| \log |S|)$ *implementierbar (bzw. sogar Linearzeit, falls die Trainingssequenz bereits sortiert gegeben ist).*

Wir kommen nun zu einer d-dimensionalen Verallgemeinerung der Klasse DS_1. Die *Klasse der Entscheidungsstümpfe* über $\mathbb{R}^d$ ist gegeben durch

$$\text{DS}_d = \{\mathbf{x} \mapsto b \cdot \text{sign}(x_i - \theta) : b \in \{-1, 1\}, i \in [d], \theta \in \mathbb{R}\} \ .$$

Wir können also eine Hypothese aus DS_d wählen, indem wir uns für eine Dimension $i \in [d]$ entscheiden und dann einen 1-dimensionalen Entscheidungsstumpf auswählen, welcher auf x_i angewendet wird.

Der obige Algorithmus zur Implementierung der ERM-Lernregel für die Klasse DS_1 lässt sich auf die offensichtliche Weise benutzen, um die ERM-Lernregel auf DS_d zu implementieren:

- Wende A_{DS} d-mal an (eine Anwendung pro Dimension) und erhalte die jeweils besten Hypothesen, sagen wir $h_1, \ldots, h_d$, sowie die Anzahl der Fehler, die sie jeweils auf der Trainingssequenz machen.

- Wähle unter $h_1, \ldots, h_d$ die Hypothese mit der kleinsten Fehleranzahl aus.

Diese Vorgehensweise führt zu folgendem Ergebnis:

Theorem 8.5 *Die ERM-Lernregel für die Hypothesenklasse* DS_d *ist mit Laufzeitschranke* $O(d \cdot |S| \log |S|)$ *implementierbar.*

Wie man sieht wird die Laufzeit dominiert durch das d-malige Sortieren der Trainingssequenz (eine Sortierung für jede der d Dimensionen).

8.2 Linearkombination von Basishypothesen

Wir werden sehen, dass geeignete Linearkombinationen von Hypothesen, welche von einem schwachen Lerner produziert wurden, eine sehr kleine empirische Fehlerrate auf den Trainingsdaten haben. Mit anderen Worten: der Approximationsfehler wird sich als kontrollierbar erweisen. Dies wirft die Frage nach der Kontrollierbarkeit des Schätzfehlers und somit die Frage nach der VC-Dimension auf.

Es sei nun $\mathcal{B}$ eine Klasse von ± 1-wertigen Basishypothesen über einem Grundbereich $\mathcal{X}$ und $T \in \mathbb{N}$. Wir definieren

$$\mathrm{LIN}(\mathcal{B}, T) = \left\{ x \mapsto \mathrm{sign}\left(\sum_{t=1}^{T} w_t h_t(x) \right) : \ \mathbf{w} \in \mathbb{R}^T, h_1, \ldots, h_T \in \mathcal{B} \right\} \ .$$

Der Einfachheit halber nennen wir die Elemente von $\mathrm{LIN}(\mathcal{B}, T)$ „Linearkombinationen“ von T Basisfunktionen aus $\mathcal{B}$ (obwohl es sich, streng genommen, um die sign-Funktion angewendet auf eine solche Linearkombination handelt).

Wie mächtig kann die Klasse $\mathrm{LIN}(\mathcal{B}, T)$ im Vergleich zu $\mathcal{B}$ werden? Wir betrachten zunächst ein

Beispiel 8.6 *Es sei $\mathcal{G}_T$ die bereits in Beispiel 8.2 eingeführte Hypothesenklasse. Offensichtlich gilt $\mathrm{VCD}(\mathcal{G}_T) = T + 1$. Es ist nicht schwer zu zeigen, dass $\mathcal{G}_T$ eine Teilklasse von $\mathrm{LIN}(\mathrm{DS}_1, T)$ ist. S. Übung. Es folgt, dass $\mathrm{VCD}(\mathrm{LIN}(\mathrm{DS}_1, T)) \geq \mathrm{VCD}(\mathcal{G}_T) = T + 1$, wohingegen $\mathrm{VCD}(\mathrm{DS}_1) = 2$.*

Allgemeinere Einsichten ergeben sich aus folgendem

Theorem 8.7 *Es sei $\mathcal{B}$ eine Klasse von ± 1-wertigen Basishypothesen der VC-Dimension $d \geq 3$ und $3 \leq T \in \mathbb{N}$. Dann gilt:*

$$\mathrm{VCD}(\mathrm{LIN}(\mathcal{B}, T)) \leq \underbrace{T(d+1)(2 + 3\log(d+1))}_{=D(T,d)} \ .$$

Beweis Es seien $x_1, \ldots, x_m \in \mathcal{X}$. Wir sagen eine Matrix $B \in \{-1, 1\}^{T \times m}$ ist *realisierbar* durch $\mathcal{B}$, falls $h_1, \ldots, h_T \in \mathcal{B}$ existieren mit $h_t(x_i) = B[t, i]$ für alle $t \in [T]$ und für alle $i \in [m]$. Mit Sauer's Lemma folgt, dass in jeder Zeile $[h_t(x_1), \ldots, h_t(x_m)]$ von B höchstens $(em/d)^d$ Binärmuster durch $\mathcal{B}$ realisierbar sind. Somit sind höchstens $(em/d)^{dT}$ verschiedene Matrizen $B \in \{-1, 1\}^{T \times m}$ durch $\mathcal{B}$ realisierbar. Die Realisierung eines Binärmusters auf $x_1, \ldots, x_m$ durch die Klasse $\mathrm{LIN}(\mathcal{B}, T)$ können wir uns zweistufig vorstellen wie folgt:

1. Wir wählen $h_1, \ldots, h_T$ und bilden x_i auf den Spaltenvektor $B_i = (h_1(x_i), \ldots, h_T(x_i))^\top$ ab (für alle $i \in [m]$).

2. Zu jeder festen (durch $\mathcal{B}$ realisierbaren) Wahl von $B = [B_1 \ldots B_m]$ wählen wir einen Gewichtsvektor $\mathbf{w} \in \mathbb{R}^T$ und bilden B_i auf $\mathrm{sign}(\langle \mathbf{w}, B_i \rangle)$ ab (für alle $i \in [m]$).

Wie bereits oben festgestellt gibt es höchstens $(em/d)^{dT}$ mögliche Wahlen von B. Da die VC-Dimension von $\text{sign} \circ \text{LIN}_T^0$ gleich T ist, gibt es für jede feste Wahl von B höchstens $(em/T)^T$ Binärmuster, die auf $B_1, \ldots, B_m$ durch Wahl eines Gewichtsvektors $\mathbf{w}$ realisierbar sind. Insgesamt sind in diesem zweistufigen Prozess also maximal

$$(em/d)^{dT} \cdot (em/T)^T < m^{(d+1)T}$$

Binärmuster realisierbar. Falls $x_1, \ldots, x_m$ durch $\text{LIN}(\mathcal{B}, T)$ aufspaltbar wären, dann müsste

$$2^m \leq m^{(d+1)T} \tag{41}$$

gelten. Dies ist für „große“ Werte von m sicherlich falsch, da die linke Seite von (41) exponentiell, die rechte jedoch nur polynomiell, in Abhängigkeit von m wächst. Eine genauere Rechnung (s. Lemma A.1 im Appendix A des Lehrbuches) zeigt, dass (41) für alle $m > T(d+1)(2+3\log(d+1))$ falsch ist. Daher muss die VC-Dimension von $\text{LIN}(\mathcal{B}, T)$ durch $T(d+1)(2+3\log(d+1))$ nach oben beschränkt sein. •

Wir merken kurz an, dass wir die Voraussetzung $T, d \geq 3$ in Theorem 8.7 eliminieren können, wenn wir in der oberen Schranke des Theorems T bzw. d durch $\max\{3, T\}$ bzw. $\max\{3, d\}$ ersetzen.

8.3 AdaBoost

„AdaBoost“ steht für „Adaptive Boosting“ und bezeichnet ein von Freund und Schapire entworfenes Verfahren, einen schwachen Lerner WL für eine Hypothesenklasse $\mathcal{H}$ in einen (starken) PAC-Lerner für $\mathcal{H}$ zu transformieren. Falls WL Hypothesen aus einer Basisklasse $\mathcal{B}$ verwendet, dann verwendet AdaBoost Hypothesen aus $\cup_{T \geq 1} \text{LIN}(\mathcal{B}, T)$. AdaBoost wird WL als Hilfsprozedur einsetzen. Ein Aufruf von WL hat die allgemeine Form $\text{WL}(P, S, \delta')$. Hierbei bezeichnet $S \in (\mathcal{X} \times \{-1, 1\})^m$ eine Trainingssequenz, P eine Verteilung auf $[m]$ (und damit auch auf $S_{\mathcal{X}}$) und $0 < \delta' < 1$ den Zuverlässigkeitsparameter. Die Parameter P, S repräsentieren eine auf $S_{\mathcal{X}}$ konzentrierte Verteilung auf $\mathcal{X}$ und WL wird mit einer Erfolgswahrscheinlichkeit von mindestens $1 - \delta'$ eine Hypothese $h \in \mathcal{B}$ mit $L_P(h) \leq 1/2 - \gamma$ abliefern.

Bemerkung 8.8 *Da wir* WL *über die Parameter P, S die Verteilung auf $\mathcal{X}$ mitteilen, kann (eine entsprechende Erweiterung von)* WL *sich die von ihm benötigte Trainigssequenz selber generieren. Die markierten Beipiele werden dabei P-zufällig aus S gezogen.*
Falls WL *die ERM-Lernregel für $\mathcal{B}$ auf* gewichteten *Trainingssequenzen implementiert (wie es zum Beispiel für $\mathcal{B} = \text{DS}_d$ auf effiziente Weise möglich ist) so vereinfacht sich die Vorgehensweise: die Prozedur* WL *kann dann unmittelbar auf die mit P gewichtete Trainingssequenz S angewendet werden.*

AdaBoost wird rundenweise vorgehen und in jeder Runde t folgende Objekte generieren:

- eine Basishypothese h_t (mit Hilfe von WL)

- die Fehlerrate ε_t von h_t bezüglich der in dieser Runde gültigen Verteilung $D^{(t)}$ auf S
- einen Gewichtsparameter w_t
- die für die nächste Runde gültige Verteilung $D^{(t+1)}$

Am Ende wird eine Art gewichtetes Majoritätvotum der Basishypothesen h_t ausgegeben.

Der Schlüssel zum Verständnis von AdaBoost liegt in der Wahl der Verteilungen $D^{(t)}$. Sie legen ein vergleichsweise hohes Gewicht auf Beispiele, die bei einem gewichteten Majoritätsvotum (zum gegenwärtigen Zeitpunkt) falsch behandelt würden. Dies zwingt den schwachen Lerner, den „problematischen" Beispielen verstärkte Aufmerksamkeit zu schenken.
Es folgt eine kompakte Beschreibung des Algorithmus' AdaBoost.

Eingabedaten: eine zufällige und zu einer Hypothese aus $\mathcal{H}$ konsistente Trainingssequenz $S = [(x_1, y_1), \ldots, (x_m, y_m)] \in (\mathcal{X} \times \{-1, 1\})^m$, ein Parameter $T \in \mathbb{N}$ für die Anzahl der Runden sowie ein Zuverlässigkeitsparameter δ

Hilfsprozedur: ein γ-schwacher Lerner WL für $\mathcal{H}$, welcher Hypothesen aus der Basisklasse $\mathcal{B}$ produziert

Initialisierung: $D^{(1)} = (1/m, \ldots, 1/m)$.

Hauptschleife: Für $t = 1, \ldots, T$ mache folgendes:

$$h_t = \mathrm{WL}\left(D^{(t)}, S, \frac{\delta}{2T}\right) \ , \tag{42}$$

$$\varepsilon_t = \sum_{i=1}^{m} D_i^{(t)} \mathbb{1}_{[h_t(x_i) \neq y_i]} \ , \tag{43}$$

$$w_t = \frac{1}{2} \ln\left(\frac{1}{\varepsilon_t} - 1\right) \ , \tag{44}$$

$$D_i^{(t+1)} = \frac{D_i^{(t)} \exp(-w_t y_i h_t(x_i))}{\sum_{j=1}^{m} D_j^{(t)} \exp(-w_t y_j h_t(x_j))} \text{ für } i = 1, \ldots, m \tag{45}$$

Ausgabe: $h_S(x) = \mathrm{sign}\left(\sum_{t=1}^{T} w_t h_t(x)\right)$.

Der folgende Satz beschreibt, wie der Approximationsfehler von AdaBoost kontrolliert werden kann.

Theorem 8.9 *Mit den im Algorithmus AdaBoost verwendeten Notationen gilt folgendes. Falls* WL *in jeder Runde $t \in [T]$ eine Hypothese h_t mit*

$$\varepsilon_t \leq 1/2 - \gamma \tag{46}$$

abliefert, dann gilt für die von AdaBoost ausgegebene Hypothese

$$L_S(h_S) = \frac{1}{m} \sum_{i=1}^{m} \mathbb{1}_{[h_S(x_i) \neq y_i]} \leq e^{-2\gamma^2 T} \ .$$

Beweis Für $t = 0, 1 \ldots, T$ definieren wir

$$f_t = \sum_{p=1}^{t} w_p h_p \text{ und } Z_t = \frac{1}{m} \sum_{i=1}^{m} \exp(-y_i f_t(x_i)) \ . \tag{47}$$

Es gilt dann $f_0 = 0$ und $Z_0 = 1$. Expandieren der rekursiven Gleichung (45) für $D_i^{(t+1)}$ liefert

$$D_i^{(t+1)} = \frac{e^{-y_i f_t(x_i)}}{\sum_{j=1}^{m} e^{-y_j f_t(x_j)}} \ . \tag{48}$$

Die von AdaBoost ausgegebene Hypothese h_S hat die Form $H_S = \text{sign} \circ f_T$. Für eine beliebige reellwertige Funktion f und $(x, y) \in \mathcal{X} \times \{-1, 1\}$ gilt $\mathbb{1}_{[\text{sign}(f(x)) \neq y]} \leq e^{-yf(x)}$. Folglich ist Z_T eine obere Schranke für $L_S(h_S) = L_S(\text{sign} \circ f_T)$. Es genügt daher $Z_T \leq e^{-2\gamma^2 T}$ nachzuweisen. Wegen $Z_0 = 1$ kann Z_T als Teleskopprodukt geschrieben werden:

$$Z_T = \frac{Z_T}{Z_0} = \prod_{t=0}^{T-1} \frac{Z_{t+1}}{Z_t} \ .$$

Also brauchen wir für jedes t lediglich die Ungleichung $Z_{t+1}/Z_t \leq e^{-2\gamma^2}$ nachzuweisen. Dies leistet die folgende Rechnung:

$$\begin{aligned}
\frac{Z_{t+1}}{Z_t} &\overset{(47)}{=} \frac{\sum_{i=1}^{m} \exp(-y_i f_{t+1}(x_i))}{\sum_{j=1}^{m} \exp(-y_j f_t(x_j))} \\
&\overset{(47)}{=} \frac{\sum_{i=1}^{m} \exp(-y_i f_t(x_i)) \exp(-y_i w_{t+1} h_{t+1}(x_i))}{\sum_{j=1}^{m} \exp(-y_j f_t(x_j))} \\
&\overset{(48)}{=} \sum_{i=1}^{m} D_i^{(t+1)} \exp(-y_i w_{t+1} h_{t+1}(x_i)) \\
&= e^{-w_{t+1}} \sum_{i: y_i h_{t+1}(x_i) = 1} D_i^{(t+1)} + e^{w_{t+1}} \sum_{i: y_i h_{t+1}(x_i) = -1} D_i^{(t+1)} \\
&= e^{-w_{t+1}}(1 - \varepsilon_{t+1}) + e^{w_{t+1}} \varepsilon_{t+1} \\
&\overset{(44)}{=} \frac{1}{\sqrt{1/\varepsilon_{t+1} - 1}}(1 - \varepsilon_{t+1}) + \sqrt{1/\varepsilon_{t+1} - 1}\,\varepsilon_{t+1} \\
&= \sqrt{\frac{\varepsilon_{t+1}}{1 - \varepsilon_{t+1}}}(1 - \varepsilon_{t+1}) + \sqrt{\frac{1 - \varepsilon_{t+1}}{\varepsilon_{t+1}}}\varepsilon_{t+1} \\
&= 2\sqrt{\varepsilon_{t+1}(1 - \varepsilon_{t+1})} \\
&\overset{(46)}{\leq} 2\sqrt{\left(\frac{1}{2} - \gamma\right)\left(\frac{1}{2} + \gamma\right)} = (1 - 4\gamma^2)^{1/2} \leq e^{-2\gamma^2}
\end{aligned}$$

Im letzten Schritt haben wir die bekannte Ungleichung $1 + a \leq e^a$ verwendet. •

Wir merken an, dass die Voraussetzung (46) in Theorem 8.9 mit einer Wahrscheinlichkeit von mindestens $1 - \delta/2$ für alle $t \in [T]$ erfüllt ist, da AdaBoost die Prozedur WL mit dem Parameter $\delta/(2T)$ aufruft.

Zusammensetzen der Puzzleteile. Es bezeichne

$$m_D(\varepsilon, \delta) = O\left(\frac{1}{\varepsilon^2}\left(D + \log\frac{1}{\delta}\right)\right)$$

eine Anzahl von Trainingsbeispielen, die den Schätzfehler für eine Hypothesenklasse der VC-Dimension D mit einer Wahrscheinlichkeit von mindestens $1-\delta$ auf ε begrenzt. Die Basisklasse $\mathcal{B}$ habe die VC-Dimension d. Es sei $D(T,d)$ die aus Theorem 8.7 bekannte obere Schranke von $\mathrm{VCD}(\mathrm{LIN}(\mathcal{B},T))$.

Wir wollen klären, wie die Parameter T und m gewählt werden müssen, damit AdaBoost mit einer Wahrscheinlichkeit von mindestens $1-\delta$ eine Hypothese mit einer Fehlerrate von höchstens ε abliefert. Es genügt zu diesem Zweck folgende Ziele durchzusetzen:

- Die empirische Fehlerrate von AdaBoost soll (unter der Voraussetzung (46)) durch $\varepsilon/2$ beschränkt sein. Hierzu genügt es,

  $$T = \left\lceil \frac{\ln(2/\varepsilon)}{2\gamma^2} \right\rceil$$

  zu setzen, da dann $e^{-2\gamma^2 T} \leq \varepsilon/2$.

- Mit einer Wahrscheinlichkeit von mindestens $1-\delta/2$ soll $|L_S(h) - L_{\mathcal{D}}(h)| \leq \varepsilon/2$ für alle $h \in \mathrm{LIN}(\mathcal{B},T)$ gelten. Hierzu genügt es,

  $$m = m^{UC}_{\mathrm{LIN}(\mathcal{B},T)}(\varepsilon/2, \delta/2)$$

  zu setzen.

Theorem 8.10 *Unter der Realisierbarkeitsannahme für $\mathcal{D}$ und $\mathcal{H}$ gilt folgendes. Es sei* WL *ein Algorithmus, der $\mathcal{H}$ mit Hypothesen aus $\mathcal{B}$ γ-schwach lernt. Dann ist AdaBoost (mit der Hilfsprozedur* WL *und mit den obigen Wahlen der Parameter T, m) ein PAC-Lerner für $\mathcal{H}$ mit Hypothesen aus $\cup_{T\geq 1}\mathrm{LIN}(\mathcal{B},T)$.*

Beachte dabei folgendes: wenn $\mathcal{D}$ bezüglich $\mathcal{H}$ die Realisierbarkeitsannahme erfüllt, so gilt dies auch für alle von AdaBoost verwendeten Verteilungen $D^{(t)}$, da diese auf einer Trainingssequenz $S \sim \mathcal{D}^m$ konzentriert sind.

9 Modellselektion und Validierung

Zur Lösung eines Lernproblems haben wir i.A. mehrere Optionen wie, zum Beispiel, die Auswahl zwischen verschiedenen Lernmethoden, zwischen verschiedenen Hypothesenklassen oder auch zwischen verschiedenen Wertzuweisungen an programmierbare Parameter eines Lernverfahrens (wie etwa die Festlegung des Parameters T für die Anzahl der Iterationen bei AdaBoost). Das Problem, sich auf eine dieser Optionen festzulegen, wird als das Modellselektionsproblem bezeichnet. Tendenziell neigen zu primitive Modelle dazu, einen hohen

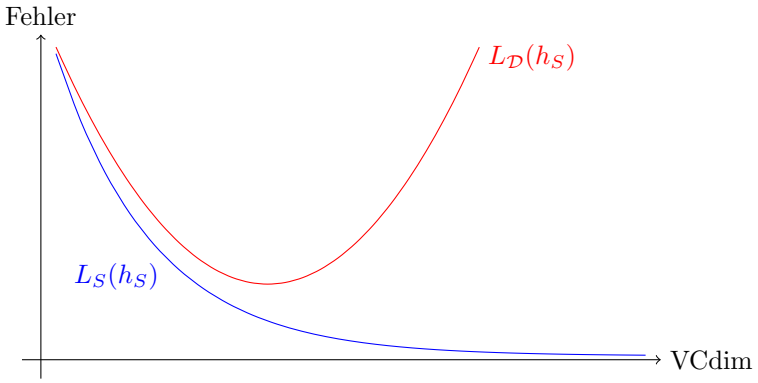


Abbildung 1: Schematische Darstellung des „bias-complexity“ Dilemmas.

Approximationsfehler zu produzieren (underfitting), wohingegen zu komplexe Modelle die Gefahr eines hohen Schätzfehlers (overfitting) bergen. Das resultierende Dilemma haben wir in einem früheren Kapitel als das „bias-complexity“ Dilemma bezeichnet.

Abbildung 1 veranschaulicht das Problem der Modellselektion:

- Die $L_S(h_S)$-Kurve fällt monoton mit wachsender Komplexität des Modells, da sich das Modell immer besser an die Daten in S anpassen kann.
- Die (uns eigentlich interessierende) $L_{\mathcal{D}}(h_S)$-Kurve fällt zunächst bei wachsender Komplexität des Modells, da der Approximationsfehler abnimmt. Wird ein bestimmter Grad der Komplexität überschritten, steigt der $L_{\mathcal{D}}(h_S)$-Wert jedoch wieder an, da h_S sich zu stark an die Daten in S anpasst.

Die Kunst besteht darin, die Komplexität des Modells so zu justieren, dass wir dem Minimalwert der $L_{\mathcal{D}}(h_S)$-Kurve nahekommen.

Im Abschnitt 9.1 besprechen wir den SRM-basierten Ansatz zur Lösung des Modellselektionsproblems. Er ist theoretisch gut fundiert, leidet aber an zu pessimistischen oberen Schranken für die Fehlerraten. Praktikabler ist die validierungsbasierte Methode, die wir in Abschnitt 9.2 besprechen. Was ist zu tun, wenn die von uns präferierte Methode zu schlechten Ergebnissen führt? Um klug zu reagieren, muss man zunächst herausfinden, ob „over-“ oder „underfitting“ (oder beides) vorliegt. In Abschnitt 9.3 machen wir entsprechende Diagnose- und Therapievorschläge.

9.1 SRM-basierte Modellselektion

Wir betrachten den Fall, dass $\mathcal{H}$ eine abzählbare Vereinigung von binären Hypothesenklassen ist, welche die uniforme Konvergenzbedingung erfüllen. Konkreter: es sei $\mathcal{H} = \cup_{d \geq 1} \mathcal{H}_d$ und, für alle $d \geq 1$, sei $\mathcal{H}_d$ eine binäre Hypothesenklasse der VC-Dimension d. Gemäß dem Fundamentaltheorem der PAC-Lerntheorie gibt es eine Konstante $c > 0$, so dass

$$m^{UC}_{\mathcal{H}_d}(\varepsilon, \delta) \leq \frac{c}{\varepsilon^2} \left(d + \ln\left(\frac{1}{\delta}\right) \right) \quad . \tag{49}$$

Wie im Kapitel über Lernbarkeit im nichtuniformen Modell sei

$$\varepsilon_d^{UC}(m,\delta) = \min\{\varepsilon \in (0,1)|\ m_{\mathcal{H}_d}^{UC}(\varepsilon,\delta) \leq m\}\ .$$

Aus (49) lässt sich mit einer einfachen Rechnung ableiten:

$$\varepsilon_d^{UC}(m,\delta) = \sqrt{\frac{c}{m}\left(d + \ln\left(\frac{1}{\delta}\right)\right)}\ . \tag{50}$$

Die SRM-Lernregel wählt zu einer gegebenen Trainingssequenz $S \in \mathcal{Z}^m$ eine Hypothese $h_S \in \mathcal{H}$ aus, welche die Kostenfunktion $L_S(h) + \varepsilon_{d(h)}^{UC}(m, w_{d(h)}\delta)$ minimiert. Hierbei sei $d(h)$ der kleinste Index d mit $h \in \mathcal{H}_d$ und $w_1, w_2, \ldots > 0$ seien Gewichtsparameter, die sich zu 1 aufaddieren. Wir setzen $w_d = \frac{1}{d(d+1)}$. Folglich ist $h_S \in \mathcal{H}$ eine Hypothese, welche die Kostenfunktion

$$\begin{aligned} f_S(h) &:= L_S(h) + \varepsilon_{d(h)}^{UC}\left(m, \frac{\delta}{d(h)(d(h)+1)}\right) \\ &\overset{(50)}{=} L_S(h) + \sqrt{\frac{c}{m}\left(d + \ln(d(h)(d(h)+1)) + \ln\left(\frac{1}{\delta}\right)\right)} \end{aligned}$$

minimiert. Aus dem Hauptresultat in dem Kapitel über Lernbarkeit im nichtuniformen Modell wissen wir, dass mit einer Wahrscheinlichkeit von mindestens $1-\delta$ eine zufällige Trainingssequenz $S \sim \mathcal{D}^m$ generiert wird, so dass die Ungleichung $L_{\mathcal{D}}(h) \leq f_S(h)$ für alle $h \in \mathcal{H}$ erfüllt ist. Die SRM-Lernregel wählt also eine Hypothese $h_S \in \mathcal{H}$ (und damit ein „Modell“ $\mathcal{H}_{d(h_S)}$) aus, die diese (mit hoher Wahrscheinlichkeit geltende) obere Schranke für die Fehlerrate $L_{\mathcal{D}}(h)$ minimiert. Der Term

$$\sqrt{\frac{c}{m}\left(d + \ln(d(d+1)) + \ln\left(\frac{1}{\delta}\right)\right)}$$

kann als Strafterm für die Wahl des „Modells“ $\mathcal{H}_d$ (also für die Selektion einer Hypothese aus $\mathcal{H}_d$) angesehen werden. Modelle einer höheren VC-Dimension werden stärker „bestraft“ als Modelle einer vergleichsweise kleinen VC-Dimension. Damit soll die Wahl eines zu komplexen Modells, das sich zu stark an die Trainingsdaten anpasst (overfitting), verhindert werden. Andererseits sorgt der Term $L_S(h)$ dafür, dass auch zu primitive Modelle, welche die empirisch ermittelten Daten nicht gut beschreiben können (underfitting), diskreditiert werden. Das Problem bei dieser Hypothesen- und Modellselektion ist, dass die obere Schranke $f_S(h)$ für $L_{\mathcal{D}}(h)$ i.A. viel zu pessimistisch ist. Daher wird das eigentliche Ziel, eine Hypothese mit minimaler Fehlerrate auszuwählen, möglicherweise verfehlt. Im nächsten Abschnitt lernen wir ein praktikableres Verfahren für Modell- und Hypothesenselektion kennen.

9.2 Validierungsbasierte Modellselektion

In diesem Abschnitt sei $\mathcal{H}$ eine (nicht notwendig binäre) Hypothesenklasse und $\ell : \mathcal{H} \times \mathcal{Z} \to [0,1]$ eine Verlustfunktion mit dem beschränkten Wertebereich $[0,1]$. Es sei $(S,V) \sim \mathcal{D}^m \times$

$\mathcal{D}^{m'}$ eine Sequenz aus insgesamt $m+m'$ Trainingsbeispielen. Dabei wird $S \sim \mathcal{D}^m$ wie üblich als Trainingssequenz bezeichnet. $V \sim \mathcal{D}^{m'}$ nennen wir die *Validierungssequenz*. Bei der validierungsbasierten Modellselektion mit einer beschränkten Anzahl r an Modellen (wie etwa r verschiedenen Hypothesenklassen oder r möglichen Werten für einen programmierbaren Parameter) gehen wir vor wie folgt:

1. Für jedes Modell $i \in [r]$ bestimmen wir in Abhängigkeit von der Trainingssequenz S (aber ohne Ansehen von V) eine Hypothese h_i und erhalten auf diese Weise r miteinander konkurierende Hypothesen $h_1, \ldots, h_r$.

2. Wir bestimmen für jede Hypothese h_i ihren mittleren Verlustwert $L_V(h_i)$ auf V und entscheiden uns am Ende für die Hypothese h_{i^*} mit dem kleinsten L_V-Wert.

Da die Hypothesen h_i ohne Ansehen von V ermittelt wurden, benötigen wir zur Kontrolle von $|L_{\mathcal{D}}(h_i) - L_V(h_i)|$ — also der Kontrolle der absoluten Differenz zwischen dem erwarteten Verlustwert und dem mittleren Verlustwert auf V — kein uniformes Konvergenzkriterium, sondern können die Hoeffding-Ungleichung (in Verbindung mit der „Union Bound“) bemühen. Auf diese Weise erhält man leicht[19] das folgende Resultat:

$$\Pr_{V \sim \mathcal{D}^{m'}} \left[\forall i \in [r] : |L_{\mathcal{D}}(h_i) - L_V(h_i)| \leq \sqrt{\frac{\ln(2r/\delta)}{2m'}} \right] \geq 1 - \delta \ .$$

Den in dieser Schranke vorkommenden Parameter r (Anzahl der in Betracht gezogenen Modelle) können wir als Strafterm dafür ansehen, dass wir uns nicht schon zu Anfang auf ein Modell festlegen. In Betracht ziehen von vielen Modellen birgt die Gefahr des „overfitting“.

Eine Variante der validierungsbasierten Modellselektion ist die sogenannte *Kreuzvalidierung*. Hierbei wird die Trainingssequenz $S \sim \mathcal{D}^m$ in k ungefähr gleich große Blöcke $S_1, \ldots, S_k$ zerlegt. Bei insgesamt k Runden spielt in Runde i der Block S_i die Rolle der Validierungssequenz und $S \setminus S_i$ die Rolle der Trainingssequenz. Der Mittelwert der Verlustwerte $L_{S_i}(\cdot)$ dient dann als Schätzwert für den erwarteten Verlustwert $L_{\mathcal{D}}(\cdot)$.

Stellen wir uns vor es sei ein Lernverfahren A gegeben, das einen programmierbaren Parameter besitzt, welcher Werte θ aus einer endlichen Menge Θ annehmen kann. Modellselektion bedeutet dann Festlegung auf einen Wert $\theta^* \in \Theta$. Unter Verwendung von Kreuzvalidierung würden wir uns für den Wert θ^* entscheiden, der zu einer Hypothese mit dem kleinsten Mittelwert der L_{S_i}-Werte führt. Danach setzen wir den Algorithmus A, parametrisiert mit θ^*, noch einmal auf die gesamte Trainingssequenz S an, und erhalten die finale Hypothese $h_{\theta^*} = A(S; \theta^*)$. Der Pseudocode für diese Methode liest sich wie folgt:

Eingabe: $S \in \mathcal{Z}^m$, $k \geq 1$, Algorithmus A, Wertemenge Θ

Methode:

1. Zerlege S in k etwa gleich große Blöcke $S_1, \ldots, S_k$.

[19] Die mathematische Herleitung ist analog zur Analyse des PAC-Lernens endlicher Hypothesenklassen.

2. Für alle $\theta \in \Theta$ mache folgendes:

   (a) Für alle $i = 1, \ldots, k$ setze $h_{i,\theta} := A(S \setminus S_i; \theta)$.

   (b) Setze $\bar{L}(\theta) := \frac{1}{k} \sum_{i=1}^{k} L_{S_i}(h_{i,\theta})$.

3. Setze $\theta^* := \operatorname{argmin}_\theta[\bar{L}(\theta)]$ und $h_{\theta^*} := A(S; \theta^*)$.

Ausgabe: h_{θ^*}

Die Methode der Kreuzvalidierung ist theoretisch noch nicht gut verstanden. Mathematische Analysen existieren nur für einige Spezialfälle (wie zum Beispiel „Nearest Neighbor“).

Trainings-, Validierungs- und Testsequenz. Es ist eine naheliegende und häufig praktizierte Idee, die Trainings- und die Validierungssequenz um eine weitere Sequenz T von *Testbeispielen* zu ergänzen, wobei $(S, V, T) \sim \mathcal{D}^m \times \mathcal{D}^{m'} \times \mathcal{D}^{m''}$. Mit Hilfe der Trainingssequenz S werden, bei r miteinander konkurrierenden Modellen, insgesamt r verschiedene Hypothesen erzeugt. Mit Hilfe der Validierungssequenz ermittelt man das „Siegermodell“ θ^* und die „Siegerhypothese“ h_{θ^*}. Schließlich bestimmt man den mittleren Verlustwert $L_T(h_{\theta^*})$ der Siegerhypothese auf der Testsequenz. Da h_{θ^*} ohne Ansehen von T ermittelt wurde, lässt sich der Fehlerterm $|L_{\mathcal{D}}(h_{\theta^*}) - L_T(h_{\theta^*})|$ über die Hoeffding-Ungleichung kontrollieren und es gilt

$$\Pr\left[|L_{\mathcal{D}}(h_{\theta^*}) - L_T(h_{\theta^*})| \leq \sqrt{\frac{\ln(2/\delta)}{2m''}}\right] \geq 1 - \delta \ .$$

9.3 Hilfsmaßnahmen bei hohem Testfehler

In diesem Abschnitt bezeichne h_S eine ERM-Hypothese zu einer Trainingssequenz $S \sim \mathcal{D}^m$ (und zu einem per Modellselektion vorher bestimmten Modell θ^*). Wenn der Testfehler $L_T(h_S)$ „groß“ ist, dann gilt dies vermutlich auch für den erwarteten Verlustwert $L_{\mathcal{D}}(h_S)$. Was ist dann zu tun? Es gibt eine ganze Reihe von möglichen Maßnahmen:

Im Falle von „overfitting“: Vergrößere die Trainingssequenz oder wähle ein weniger komplexes Modell.

Im Falle von „underfitting“: Wähle ein komplexeres Modell.

Perfiderweise können „over-“ und „underfitting“ sogar kombiniert auftreten. Zum Beispiel könnte das Modell θ^*

einerseits so komplex sein, dass es eine Überanpassung an die Daten in S ermöglicht,

andererseits so inadäquat für das vorliegende Lernproblem sein, dass es keine Hypothese mit kleinem erwarteten Verlustwert bereithält.

In diesem Fall sind radikalere Maßnahmen erforderlich. Beispielsweise:

- Ziehe andere Merkmalsvektoren zur Darstellung der Daten in Betracht.
- Ziehe andersartige Hypothesenklassen bzw. andersartige Lernalgorithmen in Betracht.

Maßnahmen dieser Art könnten auch bei reinem „overfitting“ in Betracht kommen, wenn sich einerseits keine weiteren Trainingsdaten beschaffen lassen, aber andererseits der Übergang zu einem weniger komplexen Modell schlagartig zu „underfitting“ führt. Eine entsprechende Bemerkung gilt für reines „underfitting“, falls der Übergang zu einem komplexeren Modell wegen eintretenden „overfittings“ versperrt ist.

Die genannten Ratschläge sind gut und schön, aber wie stelle ich fest, ob der hohe Testfehler auf „over-“ oder auf „underfitting“ beruht? Es wird sich im Verlaufe dieses Abschnittes zeigen: unser Hauptinstrument für Diagnose und Einleiten entsprechender „Therapiemaßnahmen“ ist die *Verlustwertzerlegung*.

Es bezeichne $h^* \in \mathcal{H}$ die Hypothese aus $\mathcal{H}$ mit dem kleinstmöglichen erwarteten Verlustwert $L_{\mathcal{D}}(h)$. Wir erinnern an die folgende Zerlegung aus einem früheren Kapitel:

$$L_{\mathcal{D}}(h_S) = \underbrace{L_{\mathcal{D}}(h^*)}_{=:\varepsilon_{app}} + \underbrace{(L_{\mathcal{D}}(h_S) - L_{\mathcal{D}}(h^*))}_{=:\varepsilon_{est}} \ .$$

Wir hatten seinerzeit ε_{app} als den *Approximationsfehler* und ε_{est} als den *Schätzfehler* bezeichnet. Ein hoher Approximationsfehler zeigt „underfitting“ und ein hoher Schätzfehler zeigt „overfitting“ an. Das Problem ist nur: da wir $\mathcal{D}$ nicht kennen, kennen wir weder ε_{app} noch ε_{est}. Wir betrachten daher besser die folgende Verlustwertzerlegung:[20]

$$L_{\mathcal{D}}(h_S) = \underbrace{(L_{\mathcal{D}}(h_S) - L_T(h_S))}_{\text{unbek. aber klein}} + \underbrace{(L_T(h_S) - L_S(h_S))}_{\text{bekannt}} + \underbrace{L_S(h_S)}_{\text{bekannt}} \ .$$

Wir verwenden hier und im Folgenden die Daumenregel, dass auf der Hoeffding-Ungleichung beruhende Schätzungen „gut“ sind. Die betreffenden Schätzfehler, wie zum Beispiel $L_{\mathcal{D}}(h_S) - L_T(h_S)$, bezeichnen wir salopp einfach als „klein“. Wenn $L_T(h_S)$ „klein“ ist, so ist auch $L_{\mathcal{D}}(h_S)$ „klein“ und wir können uns entspannt zurücklehnen. Alle folgenden Überlegungen setzen einen unbefriedigend hohen Wert von $L_T(h_S)$ voraus, so dass Handlungsbedarf besteht.

Da $L_T(h_S)$ „groß“ ist, muss auch mindestens einer der beiden Terme $L_T(h_S) - L_S(h_S)$ und $L_S(h_S)$ ebenfalls „groß“ sein. Wenn $L_T(h_S) - L_S(h_S)$ „groß“ ist, dann ist auch $L_{\mathcal{D}}(h_S) - L_S(h_S)$ „groß“, d.h., die Hypothese ist an die Trainingssequenz S überangepasst und es liegt „overfitting“ vor. Um eine entsprechende Überlegung für den Term $L_S(h_S)$ anzustellen, betrachten wir die Zerlegung

$$L_S(h_S) = \underbrace{\overbrace{(L_S(h_S) - L_S(h^*))}^{\leq 0}}_{\text{unbekannt}} + \underbrace{(L_S(h^*) - L_{\mathcal{D}}(h^*))}_{\text{unbek. aber klein}} + L_{\mathcal{D}}(h^*) \ . \tag{51}$$

[20] Wenn h_S ohne Ansehen von V ermittelt wurde, kann bei dieser Zerlegung T durch V ersetzt werden.

Beachte, dass die Differenz $L_S(h^*) - L_{\mathcal{D}}(h^*)$ darum (betragsmäßig) klein ist, weil die obige Definition von h^* (als Hypothese aus $\mathcal{H}$ mit dem kleinstmöglichen erwarteten Verlustwert) nicht von S abhängt und $L_S(h^*) - L_{\mathcal{D}}(h^*)$ daher mit der Hoeffding-Ungleichung kontrolliert werden kann. $L_S(h_S) - L_S(h^*) \leq 0$ gilt, da h_S als ERM-Hypothese zu gegebener Trainingssequenz S vorausgesetzt wurde. Aus (51) folgern wir: wenn $L_S(h_S)$ „groß“ ist, dann ist $\varepsilon_{app} = L_{\mathcal{D}}(h^*)$ ebenfalls groß und es liegt „underfitting“ vor. Wenn $L_S(h_S)$ „klein“ ist, so ist (wie weiter oben bereits schon einmal angemerkt) $L_T(h_S) - L_S(h_S)$ „groß“ und es liegt „overfitting“ vor. Allerdings ist es durchaus denkbar, dass zusätzlich auch noch $L_{\mathcal{D}}(h^*)$ „groß“ ist (was „underfitting“ impliziert): nämlich dann, wenn $L_S(h_S) - L_S(h^*)$ ein betragsmäßig großer negativer Wert ist. Um

$$L_S(h_S) \text{ „klein“ und } L_{\mathcal{D}}(h^*) \text{ „groß“ (Fall 1)}$$

zu unterscheiden von

$$L_S(h_S) \text{ „klein“ und } L_{\mathcal{D}}(h^*) \text{ „klein“ (Fall 2)} \quad ,$$

könnte man beobachten, wie sich der mittlere Verlustwert $L_S(h_S)$ von h_S auf S und der mittlere Verlustwert $L_T(h_S)$ von h_S auf T bei wachsender Anzahl von Beispielen (also wachsendem m bzw. wachsendem m'') verhalten:

1. Im Fall 1 müsste $L_T(h_S)$ bei wachsendem m'' auf einem hohen Niveau verweilen (da $L_{\mathcal{D}}(h_S)$ nicht kleiner als $L_{\mathcal{D}}(h^*)$ werden kann).[21]

2. Im Fall 2 ist zu erwarten, dass $L_T(h_S)$ auf ein niedriges Niveau absinkt und $L_S(h_S)$ sich dem Niveau von $L_T(h_S)$ annähert.

Wir fassen diese informellen Überlegungen zu folgenden Daumenregeln zusammen:

$$\begin{aligned}
L_T(h_S) - L_S(h_S) \text{ „groß“} &\Rightarrow \text{„overfitting“} \\
L_S(h_S) \text{ „groß“} &\Rightarrow \text{„underfitting“} \\
L_S(h_S) \text{ „klein“ und Indizien für } L_{\mathcal{D}}(h^*) \text{ „groß“} &\Rightarrow \text{„over-“ und „underfitting“} \\
L_S(h_S) \text{ „klein“ und Indizien für } L_{\mathcal{D}}(h^*) \text{ „klein“} &\Rightarrow \text{„overfitting“}
\end{aligned}$$

Es können dann, wie zu Anfang dieses Abschnittes besprochen, die entsprechenden Maßnahmen eingeleitet werden.

10 Support-Vector Maschinen (SVM)

SVM sind ein wichtiges Werkzeug zum Lernen linearer Voraussagefunktionen in Räumen einer hohen (evtl. sogar unendlich großen) Dimension. Dies wirft die Frage nach der Beherrschbarkeit der Informations- und der Berechnungskomplexität auf. In diesem Kapitel wird aufgezeigt, wie sich die Informationskomplexität mit Hilfe von „Large Margin“-Separatoren kontrollieren lässt. Grob gesagt ist ein „Large Margin“-Separator eine affine Hyperebene,

[21] $L_S(h_S)$ wird bei wachsendem m solange „klein“ bleiben, wie die Situation des „overfitting“ noch nicht beseitigt ist.

welche die Datenpunkte nicht nur auf der richtigen Seite (Stichwort: positiver versus negativer Halbraum) liegen hat sondern zusätzlich zu diesen einen Sicherheitsabstand, genannt „Margin“, einhält. Im folgenden Kapitel besprechen wir den sogenannten „Kernel-Trick“, mit welchem sich die Berechnungskomplexität kontrollieren lässt.

Die SVM-Theorie basiert auf der von Vapnik und Chervonenkis in den 1970er Jahren entworfenen Theorie der empirischen und strukturellen Risikominimierung. Sie wurde in den 1990er Jahren (vor allem) von Vapnik perfektioniert. SVM zählten in der Spanne von 1990-2010 zu den erfolgreichsten Werkzeugen des maschinellen Lernens. In der letzten Dekade erleben die neuronalen Netzwerke (oder vergleichbare Architekturen) eine Renaissance und laufen den SVM in verschiedenen Bereichen (insbesondere bei lernbasierter Bild- und Sprachverarbeitung) den Rang ab (Stichwort: „deep neural networks“). Nichtsdestotrotz sind SVM theoretisch so gut fundiert und (mit Hilfe von effizienter „public domain“-Software so bequem anwendbar), dass wir ihnen das folgende ausführliche Kapitel widmen.

10.1 Die harte SVM-Lernregel

Es sei $S = [(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_m}, y_m)] \in (\mathbb{R}^d \times \{-1, 1\})^m$ eine Trainingssequenz. S heißt *linear separierbar*, wenn eine separierende Hyperebene für S existiert, d.h., wenn $\mathbf{w} \in \mathbb{R}^d$ und $b \in \mathbb{R}$ existieren, so dass

$$\forall i \in [m] : y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle + b) > 0 \ . \tag{52}$$

Wir setzen im gesamten Abschnitt 10.1 voraus, dass S linear separierbar ist. Betrachten wir zwei separierende Hyperebenen, von denen eine dicht an einigen der Datenpunkte vorbeischrammt, während die andere zu den Datenpunkten einen „ordentlichen“ Sicherheitsabstand einhält. Die Intuition scheint einem zu sagen, dass die zweite separierende Hyperebene der ersten vorzuziehen ist.

In diesem Abschnitt führen wir den Begriff des geometrischen Randabstandes (= Margin) einer Hyperebene zu den Datenpunkten in S ein und beschäftigen uns dann mit „Large Margin“-Separatoren, welche versuchen den Margin zu maximieren. Zunächst aber benötigen wir eine algebraische Formel, welche den Abstand eines Datenpunktes von einer Hyperebene angibt:

Lemma 10.1 *Sei $\mathbf{w} \in \mathbb{R}^d$ ein Einheitsvektor (d.h. $\|\mathbf{w}\| = 1$) und $b \in \mathbb{R}$. Dann gilt: die Distanz zwischen einem Punkt $\mathbf{x} \in \mathbb{R}^d$ und der Hyperebene $H_{\mathbf{w},b}$ beträgt $|\langle \mathbf{w}, \mathbf{x} \rangle + b|$.*

Beweis Von $\mathbf{x}$ aus erreichen wir die Hyperebene $H_{\mathbf{w},b}$ auf dem kürzesten Weg, wenn wir im rechten Winkel auf sie zusteuern. Das wäre für Punkte im positiven (bzw. negativen) Halbraum die Richtung $-\mathbf{w}$ (bzw. $\mathbf{w}$). Die Distanz zwischen $\mathbf{x}$ und $H_{\mathbf{w},b}$ ist demnach der Absolutbetrag des Skalars λ mit $\mathbf{x} - \lambda\mathbf{w} \in H_{\mathbf{w},b}$. Die Bedingung $\mathbf{x} - \lambda\mathbf{w} \in H_{w,b}$ ist äquivalent zu $\langle \mathbf{w}, \mathbf{x} - \lambda\mathbf{w} \rangle + b = 0$. Die folgende Rechnung zeigt, dass $\lambda = \langle \mathbf{w}, \mathbf{x} \rangle + b$ der gesuchte Skalar ist:

$$\langle \mathbf{w}, \mathbf{x} - (\langle \mathbf{w}, \mathbf{x} \rangle + b)\mathbf{w} \rangle + b = \langle \mathbf{w}, \mathbf{x} \rangle - (\langle \mathbf{w}, \mathbf{x} \rangle + b)\langle \mathbf{w}, \mathbf{w} \rangle + b = 0 \ .$$

Die letzte Gleichung nutzt aus, dass $1 = \|\mathbf{w}\| = \|\mathbf{w}\|^2 = \langle \mathbf{w}, \mathbf{w} \rangle$. •

Es sei $\mathbf{w} \in \mathbb{R}^d$ ein Einheitsvektor. Die Distanz zwischen $S_{\mathcal{X}} = \{\mathbf{x_1}, \ldots, \mathbf{x_m}\}$ und $H_{\mathbf{w},b}$ wird auch der *geometrische Randabstand (Margin)* von $H_{\mathbf{w},b}$ zu S genannt. Gemäß Lemma 10.1 ist der Margin gegeben durch

$$\gamma_{\mathbf{w},b}(S) = \min_{i \in [m]} |\langle \mathbf{w}, \mathbf{x_i} \rangle + b| \ . \tag{53}$$

Wenn $\mathbf{w}$ nicht notwendig normiert ist, dann heißt die durch (53) gegebene Größe $\gamma_{\mathbf{w},b}(S)$ der *funktionale Randabstand (Functional Margin)* von $H_{\mathbf{w},b}$ zu S.

Die *harte SVM-Lernregel* besagt: wähle $\mathbf{w}^*, b^*$ so, dass $H_{\mathbf{w}^*,b^*}$ eine separierende Hyperebene für S mit dem größtmöglichen geometrischen Randabstand zu S ist. Formal:

$$(\mathbf{w}^*, b^*) = \text{argmax}_{\mathbf{w},b} \gamma_{\mathbf{w},b}(S) \ \text{ unter NB } \ (\|\mathbf{w}\| = 1) \wedge (52) \ .$$

„NB“ steht hier und im Folgenden für „Nebenbedingung“. Der maximale Wert bei diesem Optimierungsproblem heißt der *größtmögliche geometrische Randabstand (Maximum Margin)* für S. Er wird im Folgenden mit $\gamma_{max}(S)$ notiert.

Bemerkung 10.2 *Eine Hyperebene $H_{\mathbf{w},b}$ hat genau dann einen funktionalen Randabstand von mindestens 1 zu S, wenn mit dem normierten Vektor $\mathbf{w}/\|\mathbf{w}\|$ anstelle von $\mathbf{w}$ und $b/\|\mathbf{w}\|$ anstelle von b ein geometrischer Randabstand von mindestens $1/\|\mathbf{w}\|$ erzielt wird. Die für S separierende Hyperebene mit dem größtmöglichen Randabstand zu S ist also identisch mit der für S separierenden Hyperebene, die mit einem Vektor möglichst kleiner Länge einen funktionalen Randabstand von mindestens 1 zu S erzielt.*

Aus Bemerkung 10.2 ergibt sich folgendes Resultat:

Lemma 10.3 *Die harte SVM-Lernregel lässt sich auf äquivalente Weise umformulieren wie folgt: wähle $(\mathbf{w}^*, b^*)$ so, dass $H_{\mathbf{w}^*,b^*}$ eine separierende Hyperebene für S mit einem funktionalen Randabstand von mindestens 1 zu S ist und $\|\mathbf{w}^*\|$ unter dieser Randbedingung minimiert wird. Formal:*

$$(\mathbf{w}^*, b^*) = \text{argmin}_{\mathbf{w},b} \|\mathbf{w}\|^2 \ \ \textit{unter NB} \ \ \forall i \in [m] : y_i(\langle \mathbf{w}, \mathbf{x_i} \rangle + b) \geq 1 \ . \tag{54}$$

Es gilt dann:

$$\gamma_{max}(S) = \frac{1}{\|\mathbf{w}^*\|}$$

und dieser Margin wird von $(\mathbf{w}^*/\|\mathbf{w}^*\|, b^*/\|\mathbf{w}^*\|)$ erzielt.

Beachte, dass das Minimieren der Kostenfunktion $\|\mathbf{w}\|^2$ äquivalent zum Minimieren der Kostenfunktion $\|\mathbf{w}\|$ ist. Die Wahl von $\|\mathbf{w}\|^2$ statt $\|\mathbf{w}\|$ in (54) hat den Vorteil, dass ein *konvexes* quadratisches Optimierungsproblem entsteht, das mit Standardwerkzeugen effizient gelöst werden kann.

Wir merken kurz an, dass die harte SVM-Lernregel im homogenen Fall (also mit der Festlegung $b = 0$) die folgende Form hat:

$$\mathbf{w}^* = \text{argmin}_{\mathbf{w}} \|\mathbf{w}\|^2 \ \text{ unter NB } \ \forall i \in [m] : y_i \langle \mathbf{w}, \mathbf{x_i} \rangle \geq 1 \ . \tag{55}$$

Reduktion auf den homogenen Fall. Wie wir wissen ist $S = [(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_m}, y_m)] \in (\mathbb{R}^d \times \{-1, 1\})^m$ genau dann durch eine affine Hyperebene im $\mathbb{R}^d$ separierbar, wenn $S' = [(\mathbf{x'_1}, y_1), \ldots, (\mathbf{x'_m}, y_m)]$ mit $\mathbf{x'_i} = (1, \mathbf{x_i})$ durch eine homogene Hyperebene (mit Bias $b = 0$) im $\mathbb{R}^{d+1}$ separierbar ist, sagen wir durch $H_{\mathbf{w}',0}$ mit $\mathbf{w}' = (w_0, \mathbf{w})$ und $\mathbf{w} = (w_1, \ldots, w_d)$. Die entsprechende affine Hyperebene im $\mathbb{R}^d$ wäre dann $H_{\mathbf{w},w_0}$, d.h., w_0 spielt die Rolle des Bias. Allerdings liefert die harte SVM-Lernregel für das Szenario im $\mathbb{R}^d$ i.A. *nicht* das selbe Resultat wie für das entsprechende Szenario im $\mathbb{R}^{d+1}$. Der Grund ist, dass die Einbettung in den $\mathbb{R}^{d+1}$ die geometrischen Randabstände verändert: die Distanz zwischen $H_{\mathbf{w},w_0}$ und $\mathbf{x_i}$ beträgt

$$\frac{1}{\|\mathbf{w}\|}|\langle \mathbf{w}, \mathbf{x_i}\rangle + w_0| \ ,$$

wohingegen die Distanz zwischen $H_{\mathbf{w}',0}$ und $\mathbf{x'_i}$ gegeben ist durch

$$\frac{1}{\|\mathbf{w}'\|}|\langle \mathbf{w}', \mathbf{x'_i}\rangle| = \frac{1}{\|\mathbf{w}'\|}|\langle \mathbf{w}, \mathbf{x_i}\rangle + w_0| \ .$$

Beim Optimierungsproblem (55) (mit $\mathbf{w}'$ in der Rolle von $\mathbf{w}$) macht sich der Unterschied dadurch bemerkbar, dass wir die Kostenfunktion $\|\mathbf{w}'\|^2$ verwenden und nicht $\|\mathbf{w}\|^2$. In der Sprechweise der Lerntheorie: das Bias $b = w_0$ wird als Bestandteil des Gewichtsvektors $\mathbf{w}'$ im homogenen Fall „mitregularisiert". Die Erfahrung in der Praxis scheint aber dahin zu gehen, dass es keinen dramatischen Unterschied macht, ob wir die Reduktion auf den homogenen Fall vornehmen oder nicht.

10.2 Weiche SVM-Lernregeln

In Anwendungen sind Trainingssequenzen selten perfekt linear separierbar. Es ist daher ratsam, die harte SVM-Lernregel etwas aufzuweichen, so dass funktionale Randabstände unterhalb von 1 grundsätzlich zugelassen sind. Das kann zum Beispiel umgesetzt werden, indem für jeden Datenpunkt $(\mathbf{x_i}, y_i)$, $i = 1, \ldots, m$, eine nicht-negative Schlupfvariable ξ_i eingeführt und die Bedingung $y_i(\langle \mathbf{w}, \mathbf{x_i}\rangle + b) \geq 1$ in (54) durch $y_i(\langle \mathbf{w}, \mathbf{x_i}\rangle + b) \geq 1 - \xi_i$ ersetzt wird. Sogar $\xi_i > 1$, d.h. x_i ist auf der falschen Seite der Hyperebene, ist nicht kategorisch ausgeschlossen. Um den exzessiven Gebrauch der ξ_i zu drosseln, muss im Gegenzug ein Strafterm in die Kostenfunktion eingeführt werden, zum Beispiel der Term $\sum_{i=1}^m \xi_i/m$. Das Verhältnis dieses Strafterms zum Regularisierungsterm $\|\mathbf{w}\|^2$ kann über einen Skalar $\lambda > 0$ geregelt werden.
Diese Ideen führen zu der folgenden *weichen SVM-Lernregel*: wähle $(\mathbf{w}^*, b^*)$ gemäß

$$(\mathbf{w}^*, b^*) = \mathrm{argmin}_{\mathbf{w},b,\xi} \left(\lambda \|\mathbf{w}\|^2 + \frac{1}{m} \sum_{i=1}^m \xi_i \right) \ \text{ unter NB } \ \forall i \in [m] : (y_i(\langle \mathbf{w}, \mathbf{x_i}\rangle + b) \geq 1 - \xi_i) \wedge (\xi_i \geq 0) \ . \tag{56}$$

Die *Hinge-Verlustfunktion (hinge loss)* ist gegeben durch

$$\ell^{hinge}((\mathbf{w}, b), (\mathbf{x}, y)) = \max\{0, 1 - y(\langle \mathbf{w}, \mathbf{x}\rangle + b)\} \ .$$

Beachte, dass $\ell^{hinge}((\mathbf{w},b),(\mathbf{x},y)) \geq 1$ gilt, falls $\mathbf{x}$ auf der falschen Seite der Hyperebene $H_{\mathbf{w},b}$ liegt (also falls $y(\langle \mathbf{w},\mathbf{x}\rangle + b) \leq 0$). Daher gilt stets $\ell^{0-1}((\mathbf{w},b),(\mathbf{x},y)) \leq \ell^{hinge}((\mathbf{w},b),(\mathbf{x},y))$. Der mittlere Hinge-Verlust von $(\mathbf{w},b)$ auf S wird dann als $L_S^{hinge}(\mathbf{w},b)$ notiert, d.h.,

$$L_S^{hinge}(\mathbf{w},b) = \frac{1}{m}\sum_{i=1}^{m} \ell^{hinge}((\mathbf{w},b),(\mathbf{x_i},y_i)) \ .$$

Es ist offensichtlich, dass $\ell^{hinge}((\mathbf{w},b),(\mathbf{x_i},y_i))$ exakt dem nicht-negativen Strafterm ξ_i in (56) entspricht. Daher ist die weiche SVM-Lernregel (56) äquivalent zu

$$(\mathbf{w}^*,b^*) = \text{argmin}_{\mathbf{w},b}\left(\lambda\|\mathbf{w}\|^2 + L_S^{hinge}(\mathbf{w},b)\right) \ .$$

Es ist somit die Summe aus einem Regularisierungsterm, $\lambda\|\mathbf{w}\|^2$, und einem empirischen Verlust, $L_S^{hinge}(\mathbf{w},b)$, zu minimieren. Optimierungsprobleme dieser Form werden in der Lerntheorie auch als *regularisierte Verlustminimierungsprobleme* bezeichnet.

Wir merken kurz an, dass die weiche SVM-Lernregel im homogenen Fall (also mit der Festlegung $b = 0$) die folgende Form hat:

$$\mathbf{w}^* = \text{argmin}_{\mathbf{w}}\left(\lambda\|\mathbf{w}\|^2 + L_S^{hinge}(\mathbf{w})\right) \ , \tag{57}$$

wobei $L_S^{hinge}(\mathbf{w}) = L_S^{hinge}(\mathbf{w},0)$ gesetzt wird.

10.3 Optimalitätskriterien und Support-Vektoren

Betrachten wir nochmals das Optimierungsproblem (55). Es sei $\mathbf{w}^*$ eine optimale Lösung. Ein Vektor $\mathbf{x_i}$ aus der Trainingssequenz S heißt *Support-Vektor* für $\mathbf{w}^*$, wenn $y_i\langle \mathbf{w}^*,\mathbf{x_i}\rangle = 1$. Dies bedeutet, dass $\mathbf{x_i}$ einer der Punkte aus S mit der geringsten Distanz zu der homogenen Hyperebene $H_{\mathbf{w}^*} = H_{\mathbf{w}^*,0}$ ist. Der Name „Support-Vektor“ ist motiviert durch das folgende Resultat:

Theorem 10.4 *Eine optimale Lösung des Problems (55) liegt im Vektorraum, der von ihren Support-Vektoren aufgespannt wird.*

Dieser Satz ist eine einfache Folgerung aus der folgenden allgemeinen „Fritz John Optimalitätsbedingung“:

Lemma 10.5 *Es seien $f, g_1, \ldots, g_m$ auf $\mathbb{R}^d$ definierte differenzierbare Funktionen. Weiter sei*

$$\mathbf{w}^* = \text{argmin}_{\mathbf{w}} f(\mathbf{w}) \quad \textit{unter NB} \quad \forall i \in [m] : g_i(\mathbf{w}) \leq 0 \ .$$

Dann existiert $\mathbf{a} \in \mathbb{R}^m$ mit

$$\nabla f(\mathbf{w}^*) + \sum_{i\in[m]:g_i(\mathbf{w}^*)=0} a_i \nabla g_i(\mathbf{w}^*) = \mathbf{0} \ .$$

Eine analoge Aussage gilt auch im inhomogenen Fall. Es wäre eine gute Übung, Theorem 10.4 mit Hilfe von Lemma 10.5 zu beweisen. (Den Beweis des Lemmas lassen wir aus.)

Ein alternativer Beweis von Theorem 10.4 könnte unter Ausnutzen von Dualität und Anwendung der sogenannten „complementary slackness"-Bedingungen geführt werden. Auf Details können wir im Rahmen dieser Vorlesung nicht eingehen.

10.4 Kontrolle der Informationskomplexität

Definition 10.6 *Es sei $\mathcal{D}$ eine Verteilung über $\mathbb{R}^d \times \{-1, 1\}$ und es seien $\gamma, \rho > 0$. Wir sagen $\mathcal{D}$ ist* separierbar mit (γ, ρ)-Margin, *falls ein Paar $(\mathbf{w}, b) \in \mathbb{R}^d \times \mathbb{R}$ mit $\|\mathbf{w}\| = 1$ existiert, so dass mit Wahrscheinlichkeit 1 für $(\mathbf{x}, y) \sim \mathcal{D}$ folgendes gilt: $\|\mathbf{x}\| \leq \rho$ und $y(\langle \mathbf{w}, \mathbf{x} \rangle + b) \geq \gamma$. In Worten: ein mit $\mathcal{D}$ zufällig gezogenes Trainingsbeispiel $(\mathbf{x}, y)$ liegt fast sicher mit einem Margin von mindestens γ auf der richtigen Seite der Hyperebene $H_{\mathbf{w},b}$ und die (Euklidische) Länge von $\mathbf{x}$ ist fast sicher durch ρ nach oben beschränkt.*

Das folgende Resultat liefert eine obere Schranke für die Fehlerrate von Hypothesen, die mit der harten SVM-Lernregel ermittelt werden:

Theorem 10.7 *Es sei $\mathcal{D}$ eine mit (γ, ρ)-Margin separierbare Verteilung über $\mathbb{R}^d \times \{-1, 1\}$. Weiter sei $(\mathbf{w}^*, b^*) \in \mathbb{R}^d \times \mathbb{R}$ die Ausgabe der harten SVM-Lernregel zu einer gegebenen Trainingssequenz der Größe m. Dann gilt mit einer Wahrscheinlichkeit von mindestens $1-\delta$ (und den üblichen Notationen):*

$$L_{\mathcal{D}}^{0-1}(\mathbf{w}^*, b^*) \leq \frac{1}{\sqrt{m}} \left(\frac{2\rho}{\gamma} + \sqrt{2 \log \frac{2}{\delta}} \right) \ .$$

Den Beweis, der auf theoretischen Resultaten zur sogenannten Rademacher-Komplexität beruht, können wir an dieser Stelle nicht führen. Das entsprechende Resultat für die weiche SVM-Lernregel (hier ebenfalls ohne Beweis) liest sich wie folgt:

Theorem 10.8 *Es sei $\rho > 0$, $\mathcal{X} = \{\mathbf{x} \in \mathbb{R}^d : \|\mathbf{x}\| \leq \rho\}$ und $\mathcal{D}$ eine Verteilung über $\mathcal{X} \times \{-1, 1\}$. Weiter sei $\mathbf{w}^*$ die Ausgabe der weichen SVM-Lernregel (im homogenen Fall) zu einer gegebenen Trainingssequenz S der Größe m. Dann gilt:*

$$\mathbb{E}_{S \sim \mathcal{D}^m}[L_{\mathcal{D}}^{hinge}(\mathbf{w}^*)] \leq \min_{\mathbf{w} \in \mathbb{R}^d} \left\{ L_{\mathcal{D}}^{hinge}(\mathbf{w}) + \lambda \|\mathbf{w}\|^2 \right\} + \frac{2\rho^2}{\lambda m} \ .$$

Darüberhinaus gilt für jedes $B > 0$ und der Parmeterwahl $\lambda = \frac{\rho}{B}\sqrt{\frac{2}{m}}$ folgendes:

$$\mathbb{E}_{S \sim \mathcal{D}^m}[L_{\mathcal{D}}^{hinge}(\mathbf{w}^*)] \leq \min_{\mathbf{w} \in \mathbb{R}^d : \|\mathbf{w}\| \leq B} L_{\mathcal{D}}^{hinge}(\mathbf{w}) + 2\rho B \sqrt{\frac{2}{m}} \ .$$

Da die Hinge-Verlustfunktion eine obere Schranke für die 0-1-Verlustfunktion darstellt, gelten die oberen Schranken aus Theorem 10.8 auch für $L_{\mathcal{D}}^{0-1}$ anstelle von $L_{\mathcal{D}}^{hinge}$.

Das Bemerkenswerte an den Fehlerschranken in den Theoremen 10.7 und 10.8 ist, dass sie gänzlich unabhängig von der Dimension d des zugrunde liegenden Euklidischen Raumes sind. Das bedeutet, dass die Informationskomplexität Margin-basiert auch in extrem hochdimensionalen Räumen beherrschbar ist.

Die Beschränkung auf Vektoren x mit $\|\mathbf{x}\| \leq \rho$ ist notwendig: Margins ließen sich sonst durch Skalierung der Punkte $\mathbf{x}$ beliebig vergrößern, hätten dann aber keine Aussagekraft mehr.

11 Kernel-Methoden

Dieses Kapitel ist dem sogenannten „Kernel-Trick“ gewidmet, einer Methode,

- welche es erlaubt lineare Voraussagefunktionen mit Erfolg auf hochgradig nicht-lineare Daten anzuwenden
- und welche eine Kontrolle über die Berechnungskomplexität herstellt selbst bei (impliziter) Verwendung extrem hochdimensionaler Räume.

11.1 Daten- und Merkmalsraum

Anwendungsdaten weisen in der Regel Nicht-Linearitäten auf. Eine unmittelbare Anwendung von linearen Voraussagefunktionen ist daher nur selten erfolgreich. Stattdessen ist es besser, die Daten aus $\mathcal{X}$ in einen Merkmalsraum (feature space) $\mathcal{F}$ so abzubilden, dass die Merkmalsvektoren sich (zumindest annähernd) mit Hilfe linearer Voraussagefunktionen klassifizieren lassen. Die in den Daten vorhandenen Nicht-Linearitäten werden dabei, so gut es geht, in die Abbildung von $\mathcal{X}$ in $\mathcal{F}$ eingekapselt. Wir betrachten ein künstliches, aber illustratives,

Beispiel 11.1 *Eine ganze Zahl x erhalte das Label „1“, falls $|x| > 2$ und andernfalls das Label „-1“. Wir betten $\mathcal{X} = \mathbb{Z}$ durch die Abbildung $x \mapsto \psi(\mathbf{x}) = (x, x^2)$ in $\mathcal{F} = \mathbb{Z} \times \mathbb{Z}$ ein. Über $\mathcal{F}$ gibt es eine lineare Voraussagefunktion, welche alle Daten perfekt beschreibt. Zum Beispiel gilt für $(\mathbf{w}, b) = ((0, 1), -6)$:*

$$sign(\langle \mathbf{w}, \psi(\mathbf{x}) \rangle + b) = sign(x^2 - 6)$$

und diese Funktion ordnet allen ganzen Zahlen das korrekte Label zu.

Die allgemeine Vorgehensweise ist wie folgt:

1. Wähle eine Abbildung $\psi : \mathcal{X} \to \mathcal{F}$, die jedem Datenpunkt $x \in \mathcal{X}$ einen sogenannten *Merkmalsvektor* $\psi(\mathbf{x}) \in \mathcal{F}$ zuordnet. Wir werden häufig $\mathcal{F} = \mathbb{R}^d$ wählen, aber wir dürfen an Stelle von $\mathbb{R}^d$ einen beliebigen *Hilbert-Raum*[22] (mit evtl. unendlicher Dimension) verwenden.

[22] Zur Definition von Hilbert-Räumen s. Abschnitt 11.2.

2. Gegeben eine Trainingssequenz $S = [(x_1, y_1), \ldots, (x_m, y_m)]$, bestimme die zugehörige Bildsequenz $S = [(\mathbf{x'_1}, y_1), \ldots, (\mathbf{x'_m}, y_m)]$ mit $\mathbf{x'_i} = \psi(\mathbf{x_i})$ für $i = 1, \ldots, m$.

3. Ermittle eine lineare Voraussagefunktion $h : \mathcal{F} \to \{-1, 1\}$.

4. Sage das Label eines Testpunktes $x \in \mathcal{X}$ mit $h(\psi(\mathbf{x}))$ voraus.

Falls $\mathcal{D}$ die zugrunde liegende Verteilung auf $\mathcal{X}$ bezeichnet, so dass $S \sim \mathcal{D}^m$, dann ist die Bildverteilung $\mathcal{D}^\psi$ auf $\mathcal{F}$ gegeben durch

$$\mathcal{D}^\psi(A) = \mathcal{D}(\psi^{-1}(A))$$

und diese Gleichung gilt für alle „messbaren“ Mengen $A \subseteq \mathcal{F}$. Die Abbildung ψ muss selber „messbar“ sein, d.h., aus der Messbarkeit von $A \subseteq \mathcal{F}$ muss die Messbarkeit von $\psi^{-1}(A) \subseteq \mathcal{X}$ folgen. Bei den Abbildungen, die wir betrachten werden, werden diese „Messbarkeitsanforderungen“ immer erfüllt sein (ohne dass wir explizit darauf eingehen werden).
Es folgt eine naheliegende Verallgemeinerung von Beispiel 11.1:

Beispiel 11.2 *Für alle $x \in \mathcal{X} = \mathbb{R}$ setze $\psi(\mathbf{x}) = (1, x, x^2, \ldots, x^k) \in \mathcal{F} = \mathbb{R}^{k+1}$. Wenn die Daten aus $\mathbb{R} \times \{-1, 1\}$ mit einem Polynom vom Grad $\leq k$ separierbar sind, dann sind die zugehörigen markierten Merkmalsvektoren in $\mathbb{R}^{k+1} \times \{-1, 1\}$ linear separierbar.*
Allgemeiner sei nun $\mathcal{X} = \mathbb{R}^n$ und wir nehmen an, dass die Daten aus $\mathcal{X} \times \{-1, 1\}$ durch ein multivariates Polynom (über $x_1, \ldots, x_n$) vom Grad $\leq k$ separierbar sind. Wir setzen der Einfachheit halber $x_0 = 1$. Dann sind die zugehörigen Merkmalsvektoren linear separierbar, wenn wir $\psi(\mathbf{x})$ so definieren, dass für jede Wahl von $(i_1, \ldots, i_k) \in \{0, 1, \ldots, n\}^k$ eine Komponente

$$\psi_{i_1,\ldots,i_k}(\mathbf{x}) = \prod_{j=1}^{k} x_{i_j} \tag{58}$$

von $\psi(\mathbf{x})$ vorgesehen ist (so dass $\psi(\mathbf{x}) \in \mathbb{R}^{(n+1)^k}$).

11.2 Mathematische Grundlagen

Definition 11.3 *Eine symmetrische Matrix $A \in \mathbb{R}^{m \times m}$ heißt* positiv semidefinit*, falls*

$$\forall \mathbf{u} \in \mathbb{R}^m : \mathbf{u}^\top A \mathbf{u} = \sum_{i,j=1}^{m} A_{i,j} u_i u_j \geq 0 \ .$$

Falls dabei die Gleichheit $\mathbf{u}^\top A \mathbf{u} = 0$ nur für den Fall $\mathbf{u} = \mathbf{0}$ eintritt, so heißt A positiv definit*.*

Offensichtlich sind positive Linearkombinationen positiv semidefiniter Matrizen ebenfalls positiv semidefinit, da für $\lambda_k > 0$ und positiv semidefinite Matrizen A_k folgendes gilt:

$$\mathbf{u}^\top \left(\sum_k \lambda_k A_k \right) \mathbf{u} = \sum_k \lambda_k (\mathbf{u}^\top A_k \mathbf{u}) \geq 0 \ .$$

Beispiel 11.4 *Es sei* $\mathbf{z} \in \mathbb{R}^m$ *und* $A = \mathbf{z}\mathbf{z}^\top$, *d.h.,* $A_{i,j} = z_i z_j$ *für alle* $1 \leq i, j \leq m$. *A ist offensichtlich symmetrisch. Weiterhin gilt für jeden Vektor* $\mathbf{u} \in \mathbb{R}^m$*:*

$$\mathbf{u}^\top(\mathbf{z}\mathbf{z}^\top)\mathbf{u} = (\mathbf{u}^\top\mathbf{z})(\mathbf{z}^\top\mathbf{u}) = (\mathbf{u}^\top\mathbf{z})^2 \geq 0 \ .$$

Daher ist die Matrix $A = \mathbf{z}\mathbf{z}^\top$ *positiv semidefinit.*

Beispiel 11.5 *Es seien* $\mathbf{z_1}, \ldots, \mathbf{z_N} \in \mathbb{R}^m$ *und* $M = [\mathbf{z_1} \ldots \mathbf{z_N}] \in \mathbb{R}^{m \times N}$. *Dann ist* $M^\top M \in \mathbb{R}^{N \times N}$ *die Matrix mit* $\mathbf{z_i}^\top \mathbf{z_j}$ *in Zeile i und Spalte j. Die Matrix* $M^\top M$ *ist offensichtlich symmetrisch und wegen*

$$\mathbf{u}^\top(M^\top M)\mathbf{u} = (\mathbf{u}^\top M^\top)(M\mathbf{u}) = (M\mathbf{u})^\top(M\mathbf{u}) = \|M\mathbf{u}\|^2 \geq 0 \ .$$

außerdem positiv semidefinit.

Definition 11.6 *Ein Vektorraum* $\mathcal{F}$ *heißt* Semi-Innerer-Produkt-Raum*, falls er mit einer bilinearen Abbildung* $\langle \cdot, \cdot \rangle$ *versehen ist, die zudem für alle Wahlen von* $\mathbf{z}, \mathbf{z}' \in \mathcal{F}$ *folgende Bedingungen erfüllt:*

$$\langle \mathbf{z}, \mathbf{z}' \rangle = \langle \mathbf{z}', \mathbf{z} \rangle \text{ und } \langle \mathbf{z}, \mathbf{z} \rangle \geq 0 \ .$$

Falls $\langle \mathbf{z}, \mathbf{z} \rangle = 0$ *nur für den Nullvektor* $\mathbf{z} = \mathbf{0}$ *gilt, dann heißt* $\mathcal{F}$ *ein* Innerer-Produkt-Raum.

In einem Semi-Inneren-Produkt-Raum $(\mathcal{F}, \langle \cdot, \cdot \rangle)$ erhalten wir durch

$$\|\mathbf{x}\| := \sqrt{\langle \mathbf{x}, \mathbf{x} \rangle} \ \text{ und } \ d(\mathbf{x}, \mathbf{y}) := \|\mathbf{x} - \mathbf{y}\| \tag{59}$$

eine *Pseudonorm* $\| \cdot \|$ und eine Pseudometrik $d(\cdot, \cdot)$. Beachte, dass bei einer Pseudonorm (im Unterschied zu einer Norm) $\|\mathbf{x}\| = 0$ auch für $\mathbf{x} \neq \mathbf{0}$ möglich ist. Entsprechend kann bei einer Pseudometrik $d(\mathbf{x}, \mathbf{y}) = 0$ auch für Vektoren $\mathbf{x} \neq \mathbf{y}$ gelten. Falls jedoch $(\mathcal{F}, \langle \cdot, \cdot \rangle)$ ein (herkömmlicher) Innerer-Produkt-Raum ist, dann ist auch $\| \cdot \|$ bzw. $d(\cdot, \cdot)$ eine Norm bzw. eine Metrik im herkömmlichen Sinne.

Wie oben schon mal erwähnt kann die Rolle des Merkmalsraumes $\mathcal{F}$ von einem beliebigen „Hilbert-Raum“ gespielt werden. Letztere sind definiert wie folgt:

Definition 11.7 *Ein* Hilbert-Raum $\mathcal{F}$ *ist ein Semi-Innerer-Produkt-Raum, der (bezüglich der von* $\langle \cdot, \cdot \rangle$ *induzierten Pseudo-Metrik) topologisch vollständig ist. Letzteres bedeutet, dass jede Cauchy-Sequenz in* $\mathcal{F}$ *auch einen Grenzwert in* $\mathcal{F}$ *besitzt.*

Beispiel 11.8

1. *Für jede Wahl von* $d \geq 1$ *ist* $\mathbb{R}^d$ *ein Hilbert-Raum.*
2. *Es sei* ℓ_2 *die Menge aller Folgen* $\mathbf{z} = (z_n)_{n \geq 1}$ *über* $\mathbb{R}$ *so dass*

$$\|\mathbf{z}\|^2 = \sum_{n=1}^{\infty} z_n^2 < \infty \ .$$

Wir definieren ein inneres Produkt wie folgt:

$$\langle \mathbf{z}, \mathbf{z}' \rangle = \sum_{n=1}^{\infty} z_n z'_n \ .$$

Es ist nicht schwer zu zeigen, dass $\langle \cdot, \cdot \rangle$ ein inneres Produkt und dass ℓ_2, mit diesem inneren Produkt versehen, ein (unendlich-dimensionaler) Hilbert-Raum ist.

Ein Untervektorraum $\mathcal{F}'$ eines Hilbert-Raumes $\mathcal{F}$ heißt *Unter-Hilbert-Raum*, falls er selber ein Hilbert-Raum ist. Jeder endlich-dimensionale Untervektorraum $\mathcal{F}'$ von $\mathcal{F}$ ist ein Unter-Hilbert-Raum, da er isomorph zu dem Hilbert-Raum $\mathbb{R}^d$ mit $d = \dim(\mathcal{F}')$ ist.

Wir werden hauptsächlich ausnutzen, dass „Projektionen“ auf Unter-Hilbert-Räume wohldefiniert sind, d.h., wenn $\mathcal{F}'$ ein Unter-Hilbert-Raum eines Hilbert-Raumes $\mathcal{F}$ ist, dann gibt es zu jedem Vektor $\mathbf{z} \in \mathcal{F}$ eine eindeutige Zerlegung $\mathbf{z} = \mathbf{u} + \mathbf{v}$ mit $\mathbf{u} \in \mathcal{F}'$ und für alle $\mathbf{u}' \in \mathcal{F}'$ gilt $\langle \mathbf{u}', \mathbf{v} \rangle = 0$. Der Vektor $\mathbf{u}$ ist dann die Projektion von $\mathbf{z}$ auf $\mathcal{F}'$ (also der Vektor in $\mathcal{F}'$ mit der kürzesten Distanz zu $\mathbf{z}$).

11.3 Der Kernel-Trick

Wie wir in Kapitel 10 gesehen haben, ist die Verwendung von hochdimensionalen Räumen akzeptabel, sofern Margin-basierte Schranken für den Generalisierungsfehler zum Einsatz kommen. In diesem Abschnitt beschäftigen wir uns mit der Berechnungskomplexität. Wenn die Dimension d sehr groß ist, könnte allein das Aufschreiben eines Vektors mit d Komponenten zuviel Zeit erfordern (ganz zu schweigen von der Zeit, die wir benötigen, um die betreffenden konvexen Optimierungsprobleme im Raum $\mathbb{R}^d$ zu lösen).

An dieser Stelle kommt der sogenannte „Kernel-Trick“ zum Einsatz. Eine Funktion $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ heißt *Kernel* für den Datenraum $\mathcal{X}$, falls ein Hilbert-Raum $\mathcal{F}$ und eine Abbildung $\psi : \mathcal{X} \to \mathcal{F}$ existieren, so dass

$$\forall x, x' \in \mathcal{X} : K(x, x') = \langle \psi(x), \psi(x') \rangle \ . \tag{60}$$

Die Abbildung ψ wird auch *Merkmalsabbildung (feature map)* genannt. Ein Algorithmus heißt *kernbasiert*, wenn er über ein „Orakel“ verfügt, welches auf Anfrage (x, x') in einem Schritt den Wert $K(x, x')$ zurückliefert. Ein effizienter kernbasierter Algorithmus wird zu einem „richtigen“ effizienten Algorithmus, wenn wir das Orakel durch eine effiziente Prozedur zur Berechnung von $K(x, x')$ ersetzen. Es wird sich zeigen, dass $K(x, x')$ häufig unmittelbar auf dem Datenraum (also ohne den Umweg über die Gleichung (60)) effizient berechnet werden kann. Weder muss dabei ein Vektor $\psi(x)$ jemals explizit konstruiert werden, noch ist es notwendig die Abbildung ψ auch nur zu kennen! Der Rest dieses Abschnittes ist den kernbasierten Lernalgorithmen gewidmet.

Wenn wir uns die Merkmalsvektoren $\psi(x_i)$ an die Stelle der Datenpunkte x_i denken, dann haben die in (55) und (57) beschriebenen Optimierungsprobleme die folgende allgemeine Form:[23]

$$\mathbf{w}^* = \text{argmin}_{\mathbf{w}} \left(R(\|\mathbf{w}\|) + f(\langle \mathbf{w}, \psi(x_1) \rangle, \ldots, \langle \mathbf{w}, \psi(x_m) \rangle) \right) \tag{61}$$

[23]Eine analoge Aussage gilt für die entsprechenden Optimierungsprobleme im inhomogenen Fall.

Dabei ist $f : \mathbb{R}^m \to \mathbb{R} \cup \{\infty\}$ eine beliebige und $R : \mathbb{R}_+ \to \mathbb{R}$ eine monoton wachsende Funktion. Zum Beispiel erhalten wir (55), indem wir $R(a) = a^2$ und

$$f(a_1, \ldots, a_m) = \begin{cases} 0 & \text{falls } y_i a_i \geq 1 \text{ für alle } i \in [m] \\ \infty & \text{sonst} \end{cases}$$

setzen. Wir erhalten (57), indem wir $R(a) = \lambda a^2$ und

$$f(a_1, \ldots, a_m) = \frac{1}{m} \sum_{i=1}^{m} \max\{0, 1 - y_i a_i\}$$

setzen.

Der folgende Satz besagt, dass eine optimale Lösung für (61) existiert, die von den Merkmalsvektoren $\psi(x_1), \ldots, \psi(x_m)$ aufgespannt wird:

Theorem 11.9 (Representer Theorem) *Es sei $\psi : \mathcal{X} \to \mathcal{F}$ für einen Hilbert-Raum $\mathcal{F}$. Dann existiert ein $\alpha \in \mathbb{R}^m$, so dass*

$$\mathbf{w} = \sum_{i=1}^{m} \alpha_i \psi(x_i)$$

das Problem (61) optimal löst.

Beweis Es sei $\mathcal{F}' = \langle \psi(x_1), \ldots, \psi(x_m) \rangle$. Es sei $\mathbf{w}^* \in \mathcal{F}$ eine optimale Lösung von (61). Da $\mathcal{F}$ ein Hilbert-Raum ist, können wir $\mathbf{w}^*$ zerlegen gemäß $\mathbf{w}^* = \mathbf{w} + \mathbf{v}$, wobei $\mathbf{w} = \sum_{i=1}^{m} \alpha_i \psi(x_i)$ die Projektion von $\mathbf{w}^*$ auf $\mathcal{F}'$ ist und es gilt $\langle \mathbf{w}', \mathbf{v} \rangle = 0$ für alle $\mathbf{w}' \in \mathcal{F}'$. Insbesondere gilt $\langle \mathbf{w}, \mathbf{v} \rangle = 0 = \langle \psi(x_i), \mathbf{v} \rangle$. Es genügt zu zeigen, dass auch $\mathbf{w}$ eine optimale Lösung von (61) ist. Wegen $\langle \mathbf{w}, \mathbf{v} \rangle = 0$ gilt $\|\mathbf{w}^*\|^2 = \|\mathbf{w}\|^2 + \|\mathbf{v}\|^2$ und somit $\|\mathbf{w}\| \leq \|\mathbf{w}^*\|$ sowie auch $R(\|\mathbf{w}\|) \leq R(\|\mathbf{w}^*\|)$. Weiter gilt

$$\langle \mathbf{w}^*, \psi(x_i) \rangle = \langle \mathbf{w}, \psi(x_i) \rangle + \langle \mathbf{v}, \psi(x_i) \rangle = \langle \mathbf{w}, \psi(x_i) \rangle$$

für alle $i \in [m]$ und somit auch

$$f(\langle \mathbf{w}^*, \psi(x_1) \rangle, \ldots, \langle \mathbf{w}^*, \psi(x_m) \rangle) = f(\langle \mathbf{w}, \psi(x_1) \rangle, \ldots, \langle \mathbf{w}, \psi(x_m) \rangle) \ .$$

Zusammen mit $R(\|\mathbf{w}\|) \leq R(\|\mathbf{w}^*\|)$ ergibt sich, dass die Kosten der Lösung $\mathbf{w}$ nicht höher sind als die der Lösung $\mathbf{w}^*$. Daher ist auch $\mathbf{w}$ eine optimale Lösung von (61). •

Wir bezeichnen die Matrix $K \in \mathbb{R}^{m \times m}$ mit dem Eintrag $K(x_i, x_j) = \langle \psi(x_i), \psi(x_j) \rangle$ in Zeile i und Spalte j als die *Kernel-Matrix*[24] zur Sequenz $x_1, \ldots, x_m$. Auf Grundlage des Representer-Theorems können wir nach einer optimalen Wahl $\mathbf{w}^*$ für $\mathbf{w}$ suchen, indem wir eine optimale Wahl α^* für α treffen und

$$\mathbf{w}^* = \sum_{i=1}^{m} \alpha_i^* \psi(x_i) \tag{62}$$

setzen. Dabei ergibt sich folgendes Optimierungsproblem:

[24] mitunter auch *Gram-Matrix* genannt

Theorem 11.10 *Wir erhalten eine optimale Lösung von (61) gemäß (62), wobei*

$$\alpha^* = \text{argmin}_\alpha \left(R(\sqrt{\alpha^T K \alpha}) + f(K\alpha) \right) \quad . \tag{63}$$

Beweis Ein Vergleich von (63) und (61) zeigt, dass wir für $\mathbf{w} = \sum_{i=1}^m \alpha_i \psi(x_i)$ folgendes verifizieren müssen:

$$\|\mathbf{w}\|^2 = \alpha^\top K \alpha \text{ und } \langle \mathbf{w}, \psi(x_i) \rangle = (K\alpha)_i \text{ für } i = 1, \ldots, m \quad .$$

Die erste Gleichung ergibt sich aus

$$\|\mathbf{w}\|^2 = \langle \sum_{i=1}^m \alpha_i \psi(x_i), \sum_{j=1}^m \alpha_j \psi(x_j) \rangle = \sum_{i,j=1}^m \alpha_i \alpha_j \langle \psi(x_i), \psi(x_j) \rangle = \sum_{i,j=1}^m \alpha_i \alpha_j K(x_i, x_j) = \alpha^\top K \alpha \quad .$$

Die zweite gewünschte Gleichung ergibt sich aus

$$\langle \mathbf{w}, \psi(x_i) \rangle = \langle \sum_{j=1}^m \alpha_j \psi(x_j), \psi(x_i) \rangle = \sum_{j=1}^m \alpha_j \langle \psi(x_i), \psi(x_j) \rangle = \sum_{j=1}^m \alpha_j K(x_i, x_j) = (K\alpha)_i \quad .$$

•

Mit Hilfe sturen Einsetzens ergibt sich aus Theorem 11.10 der folgende Spezialfall:

Folgerung 11.11 *Wir erhalten eine optimale Lösung von (57) gemäß (62), wobei*

$$\alpha^* = \text{argmin}_\alpha \left(\lambda \alpha^\top K \alpha + \frac{1}{m} \sum_{i=1}^m \max\{0, 1 - y_i (K\alpha)_i\} \right) \quad . \tag{64}$$

(64) ist ein konvexes quadratisches Optimierungsproblem, zu dessen Lösung effiziente (und kernbasierte!) Standardwerkzeuge existieren.

Wir merken kurz an, dass die Vorhersage $\text{sign}(\langle \mathbf{w}, \psi(x) \rangle)$, die eine Hypothese $H_{\mathbf{w}}$ auf einer Instanz x tätigt, auch kernbasiert und einfach zu berechnen ist, da

$$\langle \mathbf{w}, \psi(x) \rangle = \sum_{j=1}^m \alpha_j \langle \psi(x_j), \psi(x) \rangle = \sum_{j=1}^m \alpha_j K(x_j, x) \quad .$$

Die analoge Bemerkung gilt für die Vorhersage $\text{sign}(\langle \mathbf{w}, \psi(x) \rangle + b)$ einer Hypothese $H_{\mathbf{w},b}$ im inhomogenen Fall.

11.4 Effizient auswertbare Kernels

Wir stellen in diesem Abschnitt einige der populärsten (und bei Anwendungen erfolgreichsten) Kernels vor und demonstrieren jeweils, dass sie effizient ausgewertet werden können (ohne dass die zugehörige Abbildung in einen Hilbert-Raum bekannt sein muss).

Polynomieller Kernel. Wir kommen zurück auf die $\mathbf{x} \in \mathbb{R}^n$ zugeordneten Merkmalsvektoren $\psi(\mathbf{x})$ aus Beispiel 11.2 mit einer Komponente der Form (58) für jede Wahl von $(i_1, \ldots, i_k) \in \{0, 1 \ldots, n\}^k$. Die folgende Rechnung, bei der wir der Einfachheit halber $x_0 = x'_0 = 1$ gesetzt haben, zeigt, dass

$$K(\mathbf{x}, \mathbf{x}') = (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^k$$

der zu ψ passende Kernel mit $K(\mathbf{x}, \mathbf{x}') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle$ ist:

$$\begin{aligned}
K(\mathbf{x}, \mathbf{x}') &= (1 + \langle \mathbf{x}, \mathbf{x}' \rangle)^k = \left(\sum_{i=0}^{n} x_i x'_i \right)^k \\
&\overset{DG}{=} \sum_{i_1, \ldots, i_k = 0}^{n} \prod_{j=1}^{k} x_{i_j} x'_{i_j} \\
&= \sum_{i_1, \ldots, i_k = 0}^{n} \left(\prod_{j=1}^{k} x_{i_j} \right) \left(\prod_{j=1}^{k} x'_{i_j} \right) \\
&\overset{(58)}{=} \sum_{i_1, \ldots, i_k = 0}^{n} \psi_{i_1, \ldots, i_k}(x) \psi_{i_1, \ldots, i_k}(x') = \langle \psi(\mathbf{x}), \psi(\mathbf{x}') \rangle \ .
\end{aligned}$$

Bei der mit „DG“ markierten Gleichung wurde das Distributivitätsgesetz angewendet.

Gauss–Kernel. Zu $x \in \mathbb{R}$ definieren wir einen (unendlich-dimensionalen) Merkmalsvektor $\psi(x) = (\psi_n(x))_{n \geq 0} \in \ell_2$, wobei

$$\psi_n(x) = \frac{x^n}{\sqrt{n!}} e^{-x^2/2} \ .$$

Für das innere Produkt zweier solcher Merkmalsvektoren gilt dann:

$$\begin{aligned}
\langle \psi(x), \psi(x') \rangle &= \sum_{n=0}^{\infty} \left(\frac{x^n}{\sqrt{n!}} e^{-x^2/2} \frac{(x')^n}{\sqrt{n!}} e^{-(x')^2/2} \right) \\
&= \exp\left(-\frac{x^2 + (x')^2}{2} \right) \sum_{n \geq 0} \frac{(xx')^n}{n!} \\
&= \exp\left(-\frac{1}{2}(x^2 + (x')^2 - 2xx') \right) = \exp\left(-\frac{(x - x')^2}{2} \right)
\end{aligned}$$

Der zu ψ passende Kernel ist demnach

$$K(x, x') = \exp\left(-\frac{(x - x')^2}{2} \right)$$

Dies ist der normierte 1-dimensionale Gauss-Kernel. Der allgemeine *d-dimensionale Gauss-Kernel mit Varianz* σ^2, definiert auf $(\mathbf{x}, \mathbf{x}') \in \mathbb{R}^d \times \mathbb{R}^d$, ist gegeben durch

$$\exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma} \right) \ .$$

Dieser Kernel hat Werte in der Nähe von 0 (bzw. 1), wenn $\mathbf{x}$ und $\mathbf{x}'$ weit von einander weg (bzw. nah bei einander) liegen. Mit dem Parameter σ können wir genauer skalieren, was wir unter „nah“ und „weit“ verstehen wollen. Es ist nicht schwer zu zeigen, dass der zum d-dimensionalen Gauss-Kernel passende Merkmalsvektor $\psi(\mathbf{x})$ für jedes $n \geq 0$ und für jeden Vektor $(i_1, \ldots, i_n) \in [d]^n$ eine Komponente

$$\psi_{i_1,\ldots,i_n}(\mathbf{x}) = \frac{1}{\sqrt{n!}} \exp\left(-\frac{\|\mathbf{x}\|^2}{2\sigma}\right) \prod_{j=1}^{n} x_{i_j}$$

vorsieht. Jede polynomielle Voraussagefunktion kann demnach auch mit dem Gauss-Kernel dargestellt werden. Der Gauss-Kernel wird mitunter auch RBF-Kernel genannt. „RBF“ steht für „Radial Basis Functions“.

Die nächsten Kernels, die wir diskutieren, gehören zur Familie der „String-Kernels“. Wir schaffen aber zunächst ein paar begriffliche Voraussetzungen.

Es A ein Alphabet (= endliche Menge). A^* bezeichne die Menge aller Wörter (strings) über A (inklusive dem leeren Wort λ). Das i-te Zeichen eines Wortes $x \in A^*$ notieren wir als x_i. Für ein Wort $x = x_1 \ldots x_n \in A^*$ bezeichne $|x| = n$ seine Länge. Es gilt $|\lambda| = 0$, d.h., das leere Wort hat die Länge 0. Es sei $x \in A^*$ ein Wort der Länge n und $I \subseteq [n]$. Dann bezeichnet x_I das Wort, das wir erhalten, wenn wir in x die Buchstaben x_i mit $i \in I$ von links nach rechts aneinander reihen. Für $I = \emptyset$ setzen wir $x_I = \lambda$. Wort $u \in A^*$ heißt *Teilsequenz* von x, falls $u = x_I$ für eine geeignete Wahl von I. Eine Teilmenge $I \subset \mathbb{N}$ der Form $I = [r : s] = \{r, r+1, \ldots, s\}$ mit $s \geq r - 1$ nennen wir *kohärent*. Die Menge $I = [r : r-1]$ identifizieren wir dabei mit $\emptyset$. Ein Wort $u \in A^*$ heißt *Teilwort* von x, falls $u = x_I$ für eine geeignete Wahl einer kohärenten Menge I. Ein Teilwort u von x heißt *Präfix* (bzw. *Suffix*) von x, falls das Wort x mit u anfängt (bzw. endet). Beachte, dass das leere Wort λ sowohl Präfix als auch Suffix (und damit auch Teilwort und erst recht Teilsequenz) eines jeden Wortes x ist.

Teilsequenz-Kernel. Zu $x \in A^*$ definieren wir einen (unendlich-dimensionalen) Merkmalsvektor $\psi(x) = (\psi_u(x))_{u \in A^*} \in \ell_2$, wobei

$$\psi_u(x) = \sum_I \mathbb{1}_{[x_I = u]} \tag{65}$$

und I rangiert über alle Teilmengen von $\mathbb{N}$, für welche x_I definiert ist (also über alle Teilmengen von $[|x|]$). Anders ausgedrückt: $\psi_u(x)$ ist die Anzahl der Vorkommen von u als Teilsequenz von x. Für das innere Produkt zweier solcher Merkmalsvektoren gilt dann:

$$\begin{aligned}
\langle \psi(x), \psi(x') \rangle &= \sum_{u \in A^*} \psi_u(x) \psi_u(x') \\
&= \sum_{u \in A^*} \left(\sum_I \mathbb{1}_{[x_I = u]} \right) \left(\sum_{I'} \mathbb{1}_{[x'_{I'} = u]} \right) \\
&= \sum_{I, I'} \sum_{u \in A^*} \mathbb{1}_{[x_I = u \wedge x'_{I'} = u]} = \sum_{I, I'} \mathbb{1}_{[x_I = x'_{I'}]} \ .
\end{aligned}$$

Der zu ψ passende Kernel ist demnach

$$K(x, x') = \sum_{I,I'} \mathbb{1}_{[x_I = x'_{I'}]} \ . \tag{66}$$

In Worten: $K(x, x')$ ist die Anzahl der Vorkommen identischer Teilsequenzpaare in (x, x'). Die Berechnung von $K(x, x')$ gemäß (66) würde exponentiell mit $|x| + |x'|$ wachsende Rechenzeit erfordern. Um zu einem effizienten Auswertungsverfahren zu gelangen, ist folgende (leicht zu verifizierende) rekursive Formel hilfreich:

$$K(x, \lambda) = 1 \text{ und } K(x, x'a) = K(x, x') + \sum_{k:x_k = a} K(x_{[1:k-1]}, x') \ . \tag{67}$$

Aus Effizienzgründen darf man aber die Berechnung von $K(x, x')$ *nicht* rekursiv programmieren (weil dann Einträge der K-Tabelle mehrfach berechnet würden), sondern muss die obige Rekursion verwenden, um eine K-Tabelle anzulegen, welche die K-Werte für alle Präfixe von x und x' bereit hält. Kürzere Präfixe kommen dabei vor längeren zum Einsatz. Die Tabelle kann iterativ effizient ausgefüllt werden: wenn $K(u, u'a)$ gemäß (67) berechnet wird, befindet sich der Eintrag $K(u, u')$ bereits in der Tabelle. S. auch das folgende Beispiel 11.12.

Beispiel 11.12 *Wir legen eine K-Tabelle an, um den Teilsequenz-Kernel an den Worten BEERE und BERT auszuwerten:*

K	λ	B	BE	BEE	$BEER$	$BEERE$
λ	1	1	1	1	1	1
B	1	2	2	2	2	2
BE	1	2	4	6	6	8
BER	1	2	4	6	12	14
$BERT$	1	2	4	6	12	14

Der Eintrag K(BE,BEERE) = K(BEERE,BE), zum Beispiel, wird berechnet gemäß

$$K(BEERE,BE) = K(BEERE,B) + K(B,B) + K(BE,B) + K(BEER,B) = 2 + 2 + 2 + 2 = 8$$

oder, alternativ, gemäß

$$K(BE,BEERE) = K(BE,BEER) + K(B,BEER) = 6 + 2 = 8 \ .$$

Die 14 *Vorkommen identischer Teilsequenzpaare in (BERT,BEERE) lassen sich durch folgende Paare (I, I') spezifizieren:*

$$\begin{array}{ccccc} (\emptyset,\emptyset) & (\{1\},\{1\}) & (\{2\},\{2\}) & (\{2\},\{3\}) & (\{2\},\{5\}) \\ (\{3\},\{4\}) & (\{1,2\},\{1,2\}) & (\{1,2\},\{1,3\}) & (\{1,2\},\{1,5\}) & (\{1,3\},\{1,4\}) \\ (\{2,3\},\{2,4\}) & (\{2,3\},\{3,4\}) & (\{1,2,3\},\{1,2,4\}) & (\{1,2,3\},\{1,3,4\}) & \end{array}$$

Teilwort-Kernel. Zu $x \in A^*$ definieren wir einen (unendlich-dimensionalen) Merkmalsvektor $\psi(x) = (\psi_u(x))_{u \in A^*} \in \ell_2$. Das Merkmal $\psi_u(x)$ ist wieder durch die Gleichung (65) gegeben, aber diesmal rangiert I nur über alle *kohärenten* Teilmengen von $\mathbb{N}$, für welche x_I definiert ist (also über alle kohärenten Teilmengen von $[|x|]$). Anders ausgedrückt: $\psi_u(x)$ ist die Anzahl der Vorkommen von u als Teilwort von x. Der dazu passende Kernel (mit $K(x,x') = \langle \psi(x), \psi(x') \rangle$) ist wieder durch die Gleichung (66) gegeben, aber freilich rangieren I und I' hier nur über *kohärente* Mengen. In Worten: $K(x,x')$ ist die Anzahl der Vorkommen identischer Teilwortpaare in (x,x'). Es bezeichne $K_0(x,x')$ die Anzahl der gemeinsamen Suffixe von x und x'. Um zu einem effizienten Auswertungsverfahren von $K(x,x')$ zu gelangen, werden folgende (leicht zu verifizierenden) rekursiven Formeln für K_0 und K verwendet:

$$K_0(x,\lambda) = 1 \quad \text{und} \quad K_0(xa,x'b) = \begin{cases} 1 + K_0(x,x') & \text{falls } a = b \\ 1 & \text{falls } a \neq b \end{cases}$$

$$K(x,\lambda) = 1 \quad \text{und} \quad K(x,x'a) = K(x,x') + \sum_{k:x_k=a} K_0(x_{[1:k-1]},x') \ .$$

Diese Rekursion muss (ähnlich wie im oben diskutierten Fall von Teilsequenzen) verwendet werden, um eine K_0- und eine K-Tabelle anzulegen, welche die betreffenden Funktionswerte für alle Präfixe von x und x' bereit hält. S. auch das folgende Beispiel 11.13.

Beispiel 11.13 *Wir legen folgende Tabellen an, um den Teilwort-Kernel an den Worten BEERE und BERT auszuwerten:*

K_0	λ	B	BE	BEE	$BEER$	$BEERE$
λ	1	1	1	1	1	1
B	1	2	1	1	1	1
BE	1	1	3	2	1	2
BER	1	1	1	1	3	1
$BERT$	1	1	1	1	1	1

K	λ	B	BE	BEE	$BEER$	$BEERE$
λ	1	1	1	1	1	1
B	1	2	2	2	2	2
BE	1	2	4	5	5	6
BER	1	2	4	5	7	8
$BERT$	1	2	4	5	7	8

Der Eintrag $K(BE,BEERE) = K(BEERE,BE)$, zum Beispiel, wird berechnet gemäß

$$K(BEERE,BE) = K(BEERE,B) + K_0(B,B) + K_0(BE,B) + K_0(BEER,B) = 2+2+1+1 = 6$$

oder, alternativ, gemäß

$$K(BE,BEERE) = K(BE,BEER) + K_0(B,BEER) = 5 + 1 = 6 \ .$$

Die 8 *Vorkommen identischer Teilwortpaare in (BERT,BEERE) lassen sich durch folgende Paare* (I, I') *spezifizieren:*

$$\begin{array}{cccc} (\emptyset,\emptyset) & (\{1\},\{1\}) & (\{2\},\{2\}) & (\{2\},\{3\}) \\ (\{2\},\{5\}) & (\{3\},\{4\}) & (\{1,2\},\{1,2\}) & (\{2,3\},\{3,4\}) \end{array}$$

Bei einem Teilwort u von x pochen wir darauf, dass die Buchstaben von u in x unmittelbar aufeinander folgen, während bei einer Teilsequenz beliebig große Abstände zwischen dem Vorkommen von u_i und dem Vorkommen von u_{i+1} in x liegen dürfen. Die Intuition sagt einem, dass ein großer Wert von $K(x, x')$ bei dem Teilwort-Kernel aussagekräftiger ist als bei dem Teilsequenz-Kernel. Allerdings ist das Teilwortkriterium tendenziell zu strikt und durchaus vorhandene Ähnlichkeiten könnten mit dem Teilwort-Kernel evtl. übersehen werden. Es ist daher nicht verwunderlich, dass in der Literatur viele Hybride zwischen dem Teilsequenz- und dem Teilwort-Kernel zu finden sind. Diese werden mathematisch hergestellt, indem Teilsequenzen zwar grundsätzlich mitgezählt werden, aber mit einem Gewicht, welches umso höher ist je näher die Buchstaben des Wortes u in x aufeinander folgen. Die interessierte Leserin und der interessierte Leser seien auf Kapitel 11 im Buch „Kernel Methods for Pattern Analysis“ von John Shawe-Taylor und Nello Cristianini verwiesen.

11.5 Charakterisierung von Kernels

Das folgende Resultat erlaubt es, Kernels K zu designen und zu verwenden, ohne sich Gedanken um den Hilbert-Raum zu machen, in welchem die Gleichung $K(x, x') = \langle \psi(x), \psi(x') \rangle$ gilt.

Theorem 11.14 *Es sei* $\mathcal{X} = \{x_1, \ldots, x_N\}$ *eine endliche Menge und* $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ *eine Funktion. Dann gilt folgendes:* K *ist ein Kernel genau dann, wenn* K*, aufgefasst als Matrix* $K \in \mathbb{R}^{\mathcal{X}\times\mathcal{X}}$*, symmetrisch und positiv semidefinit ist.*

Beweis Wir beweisen zunächst die Richtung „$\Rightarrow$“ und setzen also voraus, dass K ein Kernel ist. Sei dann $\psi : \mathcal{X} \to \mathcal{F}$ die Abbildung von $\mathcal{X}$ in einen (oBdA endlich-dimensionalen) Hilbert-Raum $\mathcal{F}$ mit $K(x_i, x_j) = \langle \psi(x_i), \psi(x_j) \rangle$. Eine Inspektion des Beispiels 11.5 ergibt, dass K dann symmetrisch und positiv semidefinit ist.
Den Beweis in der Richtung „$\Leftarrow$“ skizzieren wir lediglich. Wir definieren $\psi(x_j)$ als die Funktion von $\mathcal{X}$ nach $\mathbb{R}$, welche x_i auf $K(x_i, x_j)$ abbildet was wir in der Form $\psi(x_j) = K(\cdot, x_j)$ notieren. Es sei $\mathcal{F}_0$ der von $K(\cdot, x_1), \ldots, K(\cdot, x_N)$ aufgespannte Untervektorraum von $\mathbb{R}^{\mathcal{X}}$. Wir definieren folgende Abbildung:

$$\langle \sum_i \alpha_i K(\cdot, x_i), \sum_j \beta_j K(\cdot, x_j) \rangle = \sum_{i,j} \alpha_i \beta_j K(x_i, x_j) \ . \tag{68}$$

Es ist leicht nachzurechnen, dass die so definierte Abbildung $\langle \cdot, \cdot \rangle$ ein semi-inneres Produkt auf $\mathcal{F}_0$ ist. Zum Beispiel ergibt sich die Forderung $\langle z, z \rangle \geq 0$ für alle $z \in \mathcal{F}_0$ aus

$$\langle \sum_i \alpha_i K(\cdot, x_i), \sum_j \alpha_j K(\cdot, x_j) \rangle = \sum_{i,j} \alpha_i \alpha_j K(x_i, x_j) = \alpha^T K \alpha$$

und der (vorausgesetzten) positiven Semidefinitheit von K. Der Beweis kann dadurch abgeschlossen werden, dass $\mathcal{F}_0$ in einen *topologisch vollständigen* semi-inneren Produkt-Raum (also einen Hilbert-Raum) $\mathcal{F}$ eingebettet wird. Hierzu müssen alle Grenzwerte von Cauchy-Sequenzen in $\mathcal{F}_0$ in $\mathcal{F}$ aufgenommen werden. Wir verzichten auf die formale Durchführung dieser Konstruktion. •

Es sei $f = \sum_i \alpha_i K(\cdot, x_i)$ ein beliebiges Element des semi-inneren Produkt-Raums $\mathcal{F}_0$ aus dem Beweis von Theorem 11.14. Es folgt, dass $f(x) = \sum_i \alpha_i K(x, x_i)$. Dann gilt für alle $x \in \mathcal{X}$

$$\langle f, K(\cdot, x)\rangle = f(x) \ ,$$

wie sich durch folgende Rechnung bestätigen lässt:

$$\langle f, K(\cdot, x)\rangle = \langle \sum_i \alpha_i K(\cdot, x_i), K(\cdot, x)\rangle \stackrel{(68)}{=} \sum_i \alpha_i K(x_i, x) = f(x) \ .$$

Die Gleichung $\langle f, K(\cdot, x)\rangle = f(x)$ gilt sogar für alle f aus der (topologisch vollständigen) Erweiterung $\mathcal{F}$ von $\mathcal{F}_0$. Sie wird auch als die „reproducing property“ (reproduzierende Eigenschaft) des semi-inneren Produktes bezeichnet (da das semi-innere Produkt von f mit $K(x)$ den Funktionswert von f an der Stelle x reproduziert).

Der Fall eines unendlichen Grundbereiches. Eine Abbildung $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ (für eine evtl. unendliche Menge $\mathcal{X}$ heiße *positiv semidefinit*, falls folgendes gilt: für jede endliche Teilmenge $\{x_1, \ldots, x_N\}$ von $\mathcal{X}$ ist die entsprechende Einschränkung von K, aufgefasst als $(N \times N)$-Matrix $(K(x_i, x_j))_{1 \le i,j \le N}$, positiv semidefinit. Ein metrischer Raum $\mathcal{X}$ heißt *separabel*, falls eine abzählbare Teilmenge $\mathcal{X}_0$ von $\mathcal{X}$ existiert, die dicht in $\mathcal{X}$ liegt (d.h., dass sich jeder Punkt in $\mathcal{X}$ als Limes einer Cauchy-Sequenz in $\mathcal{X}_0$ darstellen lässt). Wir merken ohne Beweis kurz an, dass die Charakterisierung von Theorem 11.14 unter folgenden Voraussetzungen auch für unendliche Mengen $\mathcal{X}$ gilt:

- $\mathcal{X}$ ist ein separabler metrischer Raum.
- Die Funktion $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ ist stetig.

Die Separabilität von $\mathcal{X}$ überträgt sich dann sogar auf den im Beweis konstruierten Hilbert-Raum $\mathcal{F}$.

Theorem 11.14 kann als einfacher Spezialfall dieser allgemeineren Aussage gesehen werden: wenn wir eine endliche Menge $\mathcal{X}$ mit der diskreten Metrik

$$d(x, x') = \begin{cases} 1 & \text{falls } x \neq x' \\ 0 & \text{falls } x = x' \end{cases}$$

versehen, dann ist $\mathcal{X}$ trivialerweise separabel und *jede* Abbildung $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ ist stetig.

Der im Beweis von Theorem 11.14 (bzw. von der Verallgemeinerung des Theorems auf evtl. unendliche Mengen $\mathcal{X}$) konstruierte Hilbert-Raum $\mathcal{F}$ wird auch „Reproducing Kernel Hilbert Space (RHKS)“ zum Kernel K genannt.

Zur Wahl eines Kernels. Bei einem konkreten Lernproblem sollte die Wahl des Kernels einen formalen, anwendungsunspezifischen und einen intuitiven, anwendungsspezifischen Aspekt berücksichtigen:

- Auf der formalen Seite ist es wichtig, dass K positiv semidefinit ist, damit es sich (gemäß Theorem 11.14 bzw. dessen Verallgemeinerung) überhaupt um einen Kernel handelt.
- Auf der intuitiven Seite sollte $K(x, x')$ ein Maß für die Ähnlichkeit von x und x' sein, wobei hohe positive (bzw. negative) Werte eine große Ähnlichkeit (bzw. Unähnlichkeit) ausdrücken. Zum Beispiel könnten zwei Texte als ähnlich gelten, wenn sie viele Vorkommen von gemeinsamen Teilwortpaaren haben.

12 Multiklassen-Kategorisierung und Rangordnungsprobleme

Ein Lernproblem mit einer Klasse von Hypothesen der Form $h : \mathcal{X} \to \mathcal{Y}$ heißt *Multiklassen-Kategorisierungsproblem*, wenn $\mathcal{Y}$ eine endliche Menge mit $|\mathcal{Y}| \geq 2$ ist. Die Elemente von $\mathcal{Y}$ heißen dann *Kategorien* oder auch *Labels*. Ein binäres Klassifikationsproblem ist gewissermaßen der Spezialfall mit $|\mathcal{Y}| = 2$. Beispiele für Multiklassen-Kategorisierung wären etwa:

Thematische Kategorisierung von Dokumenten: Hier wäre $\mathcal{X}$ eine Menge von Dokumenten und $\mathcal{Y}$ eine Menge möglicher Themen.

Bilderkennung: Hier wäre $\mathcal{X}$ eine Menge von Bildern und $\mathcal{Y}$ eine Menge möglicher auf den Bildern dargestellter Objekte.

Ein *Rangordnungsproblem* besteht (informell gesprochen) darin, eine gegebene Folge von Instanzen gemäß ihrer „Relevanz“ zu ordnen. Beispiele für diese Art von Problemen wären etwa:

Suchmaschinen: Ordne die Resultate einer Suchmaschine bezüglich ihrer Relevanz für die Suchanfrage.

Betrugsaufdeckung: Ordne mitprotokollierte finanzielle online-Transaktionen gemäß ihres Verdachtsgrades auf Betrug.

Wir werden uns in den ersten drei Abschnitten dieses Kapitels mit Multiklassen-Kategorisierung und in den letzten beiden Abschnitten mit dem Rangordnungsproblem befassen.

12.1 Multiklassen-Kategorisierung vermöge Binärklassifikation

In diesem Abschnitt besprechen wir zwei mögliche Arten, Multiklassen-Kategorisierung auf binäre Klassifikation zu reduzieren. Im Folgenden nummerieren wir die $k = |\mathcal{Y}|$ Labels durch und identifizieren ein Label mit seiner Nummer. D.h., wir setzen $\mathcal{Y} = [k]$.

Die „Einer–gegen–Alle“-Reduktion. Für alle $i, j \in [k]$ sei

$$B_j(i) = \begin{cases} 1 & \text{falls } i = j \\ -1 & \text{sonst} \end{cases}$$

die Abbildung, die das Label j durch das Label 1 und alle anderen Labels durch das Label -1 ersetzt. Es sei A ein Algorithmus für binäre Klassifikation der, angesetzt auf eine Trainingssequenz S, eine Voraussagefunktion $A(S) : \mathcal{X} \to \mathbb{R}$ berechnet, so dass $\operatorname{sign} \circ A(S) : \mathcal{X} \to \{\pm 1\}$ eine binäre Voraussagefunktion darstellt.

Einschub: Natürlich ist nicht verboten, dass bereits $A(S)$ nur die Werte ± 1 annimmt. Falls aber beliebige reelle Werte angenommen werden können, dann kann (intuitiv) das Vorzeichen als das vorausgesagte Label und der Absolutbetrag als der Grad an Zuversicht in diese Voraussage interpretiert werden.

Für eine Trainingssequenz $S = [(x_1, y_1), \ldots, (x_m, y_m)] \in (\mathcal{X} \times [k])^m$ und $B : \mathcal{Y} \to \{\pm 1\}$ definieren wir $B(S)$ als die Trainingssequenz, die aus S durch die Labelsubstitution B hervorgeht, d.h.:

$$B(S) = [(x_1, B(y_1)), \ldots, (x_m, B(y_m))] \in (\mathcal{X} \times \{\pm 1\})^m \ .$$

Im Rahmen der Methode „Einer–gegen–Alle“ verarbeiten wir (mit Hilfe einer Prozedur A für Binärklassifikation) eine gegebene Trainingssequenz $S = [(x_1, y_1), \ldots, (x_m, y_m)] \in (\mathcal{X} \times [k])^m$ wie folgt:

1. Berechne $h_j = A(B_j(S))$ für $j = 1, \ldots, k$.
2. Wähle als Hypothese für die Multiklassen-Kategorisierung die folgende Abbildung $h : \mathcal{X} \to [k]$:
$$\forall x \in \mathcal{X} : h(x) = \operatorname{argmax}_{j \in [k]} h_j(x) \ . \tag{69}$$

Salopp gesprochen entscheidet sich die Hypothese h bei einer gegebenen Instanz $x \in \mathcal{X}$ für das Label j, dessen Binärhypothese h_j die höchste Zuversicht ausstrahlt, dass j im Vergleich zu allen anderen Labels die bessere Wahl darstellt. Falls das beste Label in (69) nicht eindeutig ist, dann legen wir fest, dass unter den optimalen Labels dasjenige mit der kleinsten Nummer gewählt wird.

Das folgende Beispiel zeigt, dass die Methode „Einer–gegen–Alle“ versagen kann.

Beispiel 12.1 *Die Grundmenge $\mathcal{X}$ bestehe aus den drei Punkten $\mathbf{x_1}, \mathbf{x_2}, \mathbf{x_3}$ der Ebene mit $\mathbf{x_1} = (-1, 1)$, $\mathbf{x_2} = (0, 1)$ und $\mathbf{x_3} = (1, 1)$. Es sei $\mathcal{Y} = [3]$, d.h., es gibt drei Labels. Die Wahrscheinlichkeitsverteilung $\mathcal{D}$ habe die Werte $\mathcal{D}(\mathbf{x_1}, 1) = \mathcal{D}(\mathbf{x_3}, 3) = 0.4$ und $\mathcal{D}(\mathbf{x_2}, 2) = 0.2$. Insbesondere wird also dem Punkt $\mathbf{x_i}$ fast sicher das Label i zugeteilt. Die Prozedur A liefere eine lineare Voraussagefunktionen $h_{\mathbf{w}}$ mit $h_{\mathbf{w}}(\mathbf{x}) = \langle \mathbf{w}, \mathbf{x} \rangle$. Wir ziehen dann für die Multiklassen-Kategorisierung Hypothesen von der Form*

$$h(\mathbf{x}) = \operatorname{argmax}_{i \in [3]} \langle \mathbf{w_i}, \mathbf{x} \rangle$$

für Gewichtsvektoren $\mathbf{w_i}$ *in Betracht. Man überprüft leicht, dass die Hypothese* h *für die folgende Wahl der drei (normierten) Gewichtsvektoren fehlerfrei ist:*

$$\mathbf{w_1} = \frac{1}{\sqrt{2}}(-1,1) \ , \ \mathbf{w_2} = (0,1) \ , \ \mathbf{w_3} = \frac{1}{\sqrt{2}}(1,1) \ .$$

Wir behaupten, dass die Methode „Einer–gegen–Alle“ nicht *zu einer fehlerfreien Hypothese gelangen kann. Wenn wir nämlich die Prozedur* A *die Hypothese* h_2 *berechnen lassen, bei welcher das Label* 2 *alleine gegen die Labels* 1 *und* 3 *antritt, dann ist eine beste Wahl für* h_2 *eine Hypothese, deren Vorzeichen alle Instanzen auf* -1 *setzt (realisiert zum Beispiel durch den Gewichtsvektor* $(0,-1)$*).*
Begründung: *Aus Konvexitätsgründen ist es unmöglich mit einer linearen Voraussagefunktion* $\mathbf{x_2}$ *auf Label* 1 *und* $\mathbf{x_1}, \mathbf{x_3}$ *beide auf Label* -1 *zu setzen. Wenn aber auf mindestens einem Punkt ein Fehler gemacht werden muss, dann am besten auf dem, der über die kleinste Wahrscheinlichkeitsmasse verfügt.*
Wenn nun Mehrdeutigkeiten bei argmax *zu Gunsten des Labels mit der kleinsten Nummer aufgelöst werden, dann wird die von „Einer–gegen-Alle“ ermittelte Hypothese* h *dem Punkt* $\mathbf{x_2}$ *das Label 1 zuweisen. Damit gilt* $L_{\mathcal{D}}(h) \geq 0.2$*.*

Die „Alle–Paare“-Reduktion. Für alle $1 \leq i < j \leq k$ sei

$$S'_{i,j} = ((x,y))_{(x,y) \in S \wedge y \in \{i,j\}}$$

die Trainingssequenz, welche aus S entsteht, indem wir alle Paare (x,y) mit $y \notin \{i,j\}$ aus S entfernen. Dann ist $S_{i,j} = B_i(S'_{i,j})$ die Trainingssequenz, die aus $S'_{i,j}$ entsteht, indem wir das Label i durch 1 und das Label j durch -1 ersetzen. Im Rahmen der Methode „Alle–Paare“ verarbeiten wir (mit Hilfe einer Prozedur A für Binärklassifikation) eine gegebene Trainingssequenz $S = [(x_1,y_1),\ldots,(x_m,y_m)] \in (\mathcal{X} \times [k])^m$ wie folgt:

1. Berechne $h_{\{i,j\}} = A(S_{i,j})$ für alle $1 \leq i < j \leq k$.

2. Wähle als Hypothese für die Multiklassen-Kategorisierung die folgende Abbildung $h : \mathcal{X} \to [k]$:
$$\forall x \in \mathcal{X} : h(x) = \text{argmax}_{i \in [k]} \sum_{j \in [k] \setminus \{i\}} \text{sign}(j-i) h_{\{i,j\}}(x) \ .$$

Beachte, dass $h_{\{i,j\}}$ die Instanz x lieber mit i als mit j labelt, wenn **entweder** $i < j$ und $h_{\{i,j\}}(x) > 0$ **oder** $i > j$ und $h_{\{i,j\}}(x) < 0$ (was sich mit $\text{sign}(j-i)h_{\{i,j\}} > 0$ zusammenfassen lässt). Der Absolutbetrag von $h_{\{i,j\}}(x)$ ist ein Maß für die Zuversicht in diese Labelpräferenz. Die Summe $\sum_{j \in [k] \setminus \{i\}} \text{sign}(j-i) h_{\{i,j\}}(x)$ liefert ein Gesamtmaß für die Zuversicht, dass i im Vergleich zu von i verschiedenen Labels eine bessere Wahl darstellt.

12.2 Multiklassen-Kategorisierung mit linearen Voraussagefunktionen

Wir beschränken uns in diesem Abschnitt auf den homogenen Fall, d.h., eine lineare Funktion ist durch einen Gewichtsvektor $\mathbf{w}$ gegeben (und das Bias b wird auf 0 gesetzt). Bei einem binären Klassifikationsproblem (mit Labels ± 1) und der Abbildung $\mathcal{X} \ni x \mapsto \psi(x) \in \mathcal{F}$ von einem Datenpunkt x auf den Merkmalsvektor $\psi(x)$ liefert ein Gewichtsvektor $\mathbf{w}$ die Klassifikation $h_{\mathbf{w}}(x) = \text{sign}(\langle \mathbf{w}, \psi(x) \rangle)$. Beachte, dass

$$\text{sign}(\langle \mathbf{w}, \psi(x) \rangle) = \text{argmax}_{y \in \{\pm 1\}} \langle \mathbf{w}, y \cdot \psi(x) \rangle \ .$$

Man kann sich also alternativ vorstellen, dass bei der Berechnung von $h_{\mathbf{w}}(x)$ zwei Bewertungsfunktionen gegeneinander antreten, nämlich $\langle \mathbf{w}, y \cdot \psi(x) \rangle$ für $y = -1, 1$, und das Label, das zu einer höheren Bewertung führt, gewinnt. Das Schöne an dieser alternativen Vorstellung ist, dass sie sich mit Hilfe von „Label-sensitiven Feature Maps“ auf Multiklassen-Kategorisierung verallgemeinern lässt. Die Vorgehensweise ist dabei analog zur Vorgehensweise beim Einsatz von Support-Vektor-Maschinen (SVM) für die Zwecke der Binärklassifikation. Wir listen kurz die wichtigsten Konzepte auf, die im Kapitel über SVM und Kernels eine tragende Rolle spielten:

- „feature map“ $\psi : \mathcal{X} \to \mathcal{F}$
- Verlustfunktion ℓ und empirischer Verlust L_S
- Lineare Voraussagefunktionen $h_{\mathbf{w}}$ (Gewichtsvektor $\mathbf{w}$)
- Separierende Hyperebene
- Hinge-Verlust
- Weiche SVM-Lernregel
- Kontrolle der Informationskomplexität
- Kernels und kernbasierte Verfahren

Alle diese Konzepte werden im Folgenden an die Spezifika der Multiklassen-Kategorisierung angepasst. Es wird sich zeigen, dass sowohl „feature maps“ als auch Verlustfunktionen in einer „Label-sensitiven“ Variante benötigt werden.

Label-sensitive Abbildungen in den Merkmalsraum. Das Zusammenspiel von Merkmals- und Gewichtsvektoren lässt sich im Falle der Multiklassen-Kategorisierung beschreiben wie folgt:

- Wir benutzen eine *Label-sensitive* Abbildung

$$\mathcal{X} \times \mathcal{Y} \ni (x, y) \mapsto \psi(x, y) \in \mathcal{F} \ .$$

- Die Hypothese zu einem Gewichtsvektor $\mathbf{w}$ wertet $\langle \mathbf{w}, \psi(x,y)\rangle$ für alle $y \in \mathcal{Y}$ aus und entscheidet sich für das Label, das zu der höchsten Bewertung führt. D.h.,

$$h_{\mathbf{w}}(x) = \text{argmax}_{y\in\mathcal{Y}} \langle \mathbf{w}, \psi(x,y)\rangle \ . \tag{70}$$

Unsere übliche Herangehensweise bei binären Klassifikationsproblemen entspricht dann dem Spezialfall $\mathcal{Y} = \{\pm 1\}$ und $\psi(x,y) = y \cdot \psi(x)$. Wir wollen anhand zweier Beispiele mögliche Wahlen von ψ ins Spiel bringen.

Beispiel 12.2 (Multivektor-Konstruktion) *Es sei $\mathcal{Y} = [k]$, $\mathcal{F} = \mathbb{R}^d$ und $\psi : \mathcal{X} \to \mathcal{F}$ eine herkömmliche (Label-insensitive) Abbildung. Wir wollen die Freiheit haben, das Paar (x,j) für jede Wahl von $j \in [k]$ mit einem separaten Vektor $\mathbf{w_j} \in \mathbb{R}^d$ zu bewerten, um uns dann für das Label j mit einem möglichst großen Wert von $\langle \mathbf{w_j}, \psi(x)\rangle$ zu entscheiden. Dies lässt sich mit einem einzigen Gewichtsvektor $\mathbf{w} \in \mathbb{R}^{dk}$ und einer Label-sensitiven Abbildung Ψ erreichen wie folgt:*

$$\mathbf{w} = \begin{bmatrix} \mathbf{w_1} \\ \vdots \\ \mathbf{w_k} \end{bmatrix} \text{ und } \Psi(x,j) = \begin{bmatrix} \mathbf{z_1} \\ \vdots \\ \mathbf{z_k} \end{bmatrix} \ ,$$

wobei

$$\mathbf{z_i} = \begin{cases} \psi(x) & \text{falls } i = j \\ \mathbf{0} & \text{sonst} \end{cases} \ .$$

Offensichtlich gilt $\langle \mathbf{w}, \Psi(x,j)\rangle = \langle \mathbf{w_j}, \psi(x)\rangle$. Das ist der gewünschte Effekt.

In Beispiel 12.2 kann ψ anwendungsspezifisch gewählt sein, aber die zusammengesetzte Abbildung Ψ bringt kein zusätzliches Anwendungswissen ein. Die Dimension dk des Merkmalsraumes $\mathcal{F}$ wächst linear mit der Anzahl k der Labels. Im folgenden Beispiel geht Anwendungswissen in einer expliziteren Weise in die Konstruktion der Abbildung ψ ein und die Dimension der Merkmalsvektoren ist gänzlich unabhängig von der Anzahl der Labels.

Beispiel 12.3 (TF-IDF-Konstruktion) *Es sei $\mathcal{X}$ eine Menge von Dokumenten und $\mathcal{Y}$ eine Menge möglicher Themen. Es liege eine Kollektion $w_1, \ldots, w_d$ von Schlüsselwörtern vor. Es bezeichne $\text{TF}(j,x)$ die Anzahl der Vorkommen des Wortes w_j im Dokument x. „TF“ steht für „Term Frequency“. Weiter sei $\text{DF}(j,y)$ eine Schätzung für die relative Häufigkeit, mit der das Wort w_j in Dokumenten zu einem von y verschiedenen Thema vorkommt. „DF“ steht für „Document Frequency“. Es sei dann schließlich $\psi = (\psi_j)_{j\in[d]}$ die durch*

$$\psi_j(x,y) = \text{TF}(j,x) \cdot \log \frac{1}{\text{DF}(j,y)}$$

gegebene Abbildung. Der Term $\psi_j(x,y)$ wird als „Term-Frequency-Inverse Document-Frequency (TF-IDF)“ bezeichnet. Er ordnet dem Paar (x,y) einen hohen Wert zu, wenn w_j in dem Dokument x oft vorkommt, aber vergleichsweise selten in Dokumenten zu einem von y verschiedenen Thema.

Label-sensitive Kosten- und Verlustfunktion. Die 0–1-Verlustfunktion ist bei Multiklassen-Kategorisierung nicht immer adäquat. Nehmen wir als Beispiel das Problem, in einem Bild das dargestellte Objekt zu erkennen. Die Verwechslung von „Tiger“ und „Katze“ ist weniger gravierend als die Verwechslung von „Tiger“ und „Delphin“. Allgemein lässt sich das modellieren durch eine Kostenfunktion $\Delta : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_+$ und eine Verlustfunktion $\ell^\Delta(h,(x,y)) = \Delta(h(x),y)$. Der empirische Verlust einer Hypothese $h : \mathcal{X} \to \mathcal{Y}$ auf einer Trainingssequenz

$$S = [(x_1,y_1),\ldots,(x_m,y_m)] \in (\mathcal{X}\times\mathcal{Y})^m \tag{71}$$

ergibt sich dann gemäß der Formel

$$L_S^\Delta(h) = \frac{1}{m}\sum_{i=1}^m \ell^\Delta(h,(x_i,y_i)) = \frac{1}{m}\sum_{i=1}^m \Delta(h(x_i),y_i) \ .$$

Die 0–1-Verlustfunktion ist der Spezialfall, der sich durch die Setzung $\Delta(y,y') = \mathbb{1}_{[y'\neq y]}$ ergibt.

Lineare Multiklassen-Voraussagefunktionen. Es sei $\psi : \mathcal{X}\times\mathcal{Y} \to \mathcal{F}$ eine Label-sensitive Abbildung und $W \subseteq \mathcal{F}$ eine Teilmenge aller Merkmalsvektoren aus $\mathcal{F}$. Das Paar (ψ, W) spezifiziert dann die folgende Familie von Multiklassen-Voraussagefunktionen:

$$\mathcal{H}_{\psi,W} = \{x \mapsto h_{\mathbf{w}}(x) = \mathrm{argmax}_{y\in\mathcal{Y}}\langle \mathbf{w}, \psi(x,y)\rangle : \mathbf{w}\in W\} \ .$$

Das Pendant zur separierenden Hyperebene. Es sei S gemäß (71) eine Trainingssequenz und $\Delta : \mathcal{Y}\times\mathcal{Y} \to \mathbb{R}_+$ eine Kostenfunktion. Dem Gewichtsvektor, der eine separierende Hyperebene repräsentiert, entspricht bei Multiklassen-Kategorisierung die folgende Wahl von $\mathbf{w}^*$:

$$\mathbf{w}^* = \mathrm{argmin}_{\mathbf{w}\in W} L_S^\Delta(h_{\mathbf{w}}) \ . \tag{72}$$

In Worten: wähle einen Gewichtsvektor mit dem kleinstmöglichen empirischen Δ-Verlust auf S. Angenommen $\mathcal{F} = W = \mathbb{R}^d$ und wir sind im realisierbaren Fall, dann lässt sich ein Gewichtsvektors $\mathbf{w}$ mit $L_S^\Delta(h_{\mathbf{w}}) = 0$ mit Hilfe von linearer Programmierung finden. Wir suchen nämlich einen Vektor $\mathbf{w}$, der folgende linearen Randbedingungen erfüllt:

$$\forall i \in [m], \forall y \in \mathcal{Y}\setminus\{y_i\} : \langle \mathbf{w}, \psi(x_i,y_i)\rangle > \langle \mathbf{w}, \psi(x_i,y)\rangle \ .$$

Da wir $\mathbf{w}$ hochskalieren können, dürfen die Ungleichungen $\langle \mathbf{w}, \psi(x_i,y_i)\rangle > \langle \mathbf{w}, \psi(x_i,y)\rangle$ durch $\langle \mathbf{w}, \psi(x_i,y_i)\rangle \geq \langle \mathbf{w}, \psi(x_i,y)\rangle + 1$ ersetzt werden, so dass wir lineare Randbedingungen in der zulässigen $\geq$-Form erhalten..

Verallgemeinerter Hinge-Verlust. Im nicht realisierbaren Fall ist die Lernregel (72) nicht effizient implementierbar (außer wenn P = NP). Wir suchen einen ähnlichen Ausweg

wie seinerzeit beim Übergang von der harten zur weichen SVM-Lernregel. Als Surrogat-Kostenfunktion dient uns die *verallgemeinerte Hinge-Verlustfunktion*, die definiert ist wie folgt:

$$\ell^{hinge}(h_{\mathbf{w}},(x,y)) = \max_{y'\in\mathcal{Y}} \left(\Delta(y',y) + \langle \mathbf{w}, \psi(x,y') - \psi(x,y)\rangle\right) \ .$$

Der empirische Hinge-Verlust auf einer Trainingsmenge S ergibt sich demnach aus

$$L_S^{hinge}(h_{\mathbf{w}}) = \frac{1}{m}\sum_{i=1}^{m} \ell^{hinge}(h_{\mathbf{w}},(x_i,y_i)) = \frac{1}{m}\sum_{i=1}^{m} \max_{y'\in\mathcal{Y}} \left(\Delta(y',y_i) + \langle \mathbf{w}, \psi(x_i,y') - \psi(x_i,y_i)\rangle\right) \ .$$

Wir machen folgende Beobachtung:

Lemma 12.4 *Es gilt*

$$\ell^{\Delta}(h_{\mathbf{w}},(x,y)) = \Delta(h_{\mathbf{w}}(x),y) \leq \ell^{hinge}(h_{\mathbf{w}},(x,y)) \ ,$$

d.h., der Δ-Verlust von $h_{\mathbf{w}}$ auf (x,y) lässt sich mit dem verallgemeinerten Hinge-Verlust von h auf (x,y) nach oben beschränken.

Beweis Aus der Definition von $h_{\mathbf{w}}(x)$ ergibt sich

$$\langle \mathbf{w}, \psi(x,y)\rangle \leq \langle \mathbf{w}, \psi(x,h_{\mathbf{w}}(x))\rangle \ .$$

Es folgt

$$\Delta(h_{\mathbf{w}}(x),y) \leq \Delta(h_{\mathbf{w}}(x),y) + \langle \mathbf{w}, \psi(x,h_{\mathbf{w}}(x)) - \psi(x,y)\rangle \leq \ell^{hinge}(h_{\mathbf{w}},(x,y)) \ .$$

•

Aus Lemma 12.4 folgt natürlich direkt die Ungleichung $L_S^{\Delta}(h_{\mathbf{w}}) \leq L_S^{hinge}(h_{\mathbf{w}})$.

Das Pendant zur weichen SVM-Lernregel. Die weiche SVM-Lernregel entspricht im Falle von Multiklassen-Kategorisierung der folgenden Wahl des Gewichtsvektors:

$$\mathbf{w}^* = \text{argmin}_{\mathbf{w}\in\mathcal{F}} \left(\lambda\|\mathbf{w}\|^2 + L_S^{hinge}(h_{\mathbf{w}})\right) \ . \tag{73}$$

Es lässt sich leicht nachrechnen, dass im Spezialfall von Binärklassifikation, $\Delta(y,y') = \mathbb{1}_{[y'\neq y]}$ und $\psi(x,y) = y\psi(x)/2$, die allgemeine Hinge-Verlustfunktion das uns vertraute Aussehen hat, d.h., $\ell^{hinge}(h_{\mathbf{w}},(x,y)) = \max\{0, 1 - y\langle \mathbf{w}, \psi(x)\rangle\}$. Die Lernregel (73) stimmt in diesem Spezialfall dann mit der uns bekannten weichen SVM-Lernregel überein.

Kontrolle der Informationskomplexität. Ähnlich wie im Fall der Binärklassifikation lässt sich der Generalisierungsfehler auch im Falle von Multiklassen-Kategorisierung unabhängig von der Dimension des Hilbert-Raumes $\mathcal{F}$ nach oben beschränken. S. etwa Korollar 17.1 im Lehrbuch. Auf Beweistechniken für solche Resultate können wir im Rahmen dieser Vorlesung nicht näher eingehen.

Kernbasierte Verfahren. Ähnlich wie bei der Binärklassifikation ist auch im Fall der Multiklassen-Kategorisierung der Einsatz von Kernels und von kernbasierten Verfahren nötig, um die Berechnungskomplexität zu beherrschen (insbesondere wenn die Merkmalsvektoren in einem extrem hoch-dimensionalen Hilbert-Raum leben). Im Rahmen dieser Vorlesung können wir auf Details hierzu nicht eingehen.

12.3 Multiklassen-Kategorisierung mit strukturierter Objekten

Spezielle Probleme entstehen bei Multiklassen-Kategorisierung, wenn die Labelmenge $\mathcal{Y}$ strukturierte Objekte enthält und $|\mathcal{Y}|$ exponentiell mit einem Komplexitätsparameter r wächst. Als Beispiel mag das Problem der Erkennung handgeschriebener Zeichen[25] dienen. Jedes einzelne Zeichen eines handgeschriebenen Textes stehe als Bild zur Verfügung. Der Datenraum $\mathcal{X}$ besteht somit aus Sequenzen von Bildern. Die Labelmenge $\mathcal{Y}$ besteht aus Zeichensequenzen, also aus Wörtern über einem Grundalphabet A. Wir nehmen der Einfachheit halber an, dass alle Sequenzen die gleiche Länge r haben. Somit gilt $\mathcal{Y} = A^r$ und $|\mathcal{Y}| = |A|^r$ (eine Zahl, die exponentiell mit der Wortlänge r wächst). Eine fehlerfreie Hypothese h würde jedes Bild der Bildersequenz auf das in ihm dargestellte Zeichen abbilden.

Die Kontrolle der Informationskomplexität wird durch große Labelmengen $\mathcal{Y}$ nicht beeinträchtigt, weil die Schranken für den Generalisierungsfehler i.A. nicht von $|\mathcal{Y}|$ abhängen. Es stellt sich aber die Frage, ob die Berechnungskomplexität noch beherrschbar ist. Zum Beispiel könnte alleine schon die Berechnung von $h_{\mathbf{w}}(x)$ gemäß (70) mindestens $|\mathcal{Y}|$ Rechenschritte kosten.[26]

Wir werden im Folgenden am OCR-Beispiel zeigen, wie (zumindest in konkreten Anwendungsszenarien) das Problem der effizienten Berechnung von $h_{\mathbf{w}}(x)$ gelöst werden kann. Wir setzen $q = |A|$, d.h., unser Grundalphabet A hat q Zeichen. Wir nehmen an, dass jedes Bild einer Bildersequenz aus n Pixeln besteht. Eine Bildersequenz x habe die Form einer $(n \times r)$-Matrix, d.h., $x \in \mathbb{R}^{n \times r}$. Der Eintrag $x_{i,j} \in \mathbb{R}$ repräsentiere den Grauton des i-Pixels vom j-ten Bild der Sequenz. Es folgt die Beschreibung einer denkbaren Label-sensitiven Abbildung $\mathcal{X} \times \mathcal{Y} \to \mathbb{R}$. Die Merkmale eines Merkmalsvektors $\psi(x)$ zerfallen in qn Merkmale vom Typ 1 und q^2 Merkmale vom Typ 2.

Typ 1: Für jedes $i \in [n]$ und jedes $a \in A$ definieren wir das Merkmal

$$\psi_{i,a,1}(x, y) = \frac{1}{r} \sum_{j=1}^{r} x_{i,j} \mathbb{1}_{[y_j = a]} \ .$$

Dieses Merkmal könnte nützlich sein, wenn bestimmte Grauwerte des i-ten Pixels das abgebildete Zeichen a von anderen Zeichen unterscheiden.

Typ 2: Für jedes Paar $a, b \in A$ definieren wir das Merkmal

$$\psi_{a,b,2}(x, y) = \frac{1}{r} \sum_{j=2}^{r} \mathbb{1}_{[y_{j-1} = a]} \mathbb{1}_{[y_j = b]} \ .$$

[25] auch OCR genannt, wobei „OCR“ für „Optical Character Recognition“ steht

[26] Ähnliche Probleme stellen sich beim Lösen des Optimierungsproblems (73).

Hohe Werte eines solchen Merkmals zeigen an, dass der Buchstabe a häufig von einem b gefolgt wird. Zum Beispiel folgt im Deutschen auf den Buchstaben „q“ häufig ein „u“.

Von diesen insgesamt $qn+q^2$ Merkmalen wird sich vermutlich nur ein kleiner Teil als nützlich erweisen, aber dies bekommen Algorithmen zum Lernen linearer Voraussagefunktionen mit der Wahl des Gewichtsvektors w in den Griff.
Wir definieren jetzt einen Vektor Φ, dessen Komponenten in einem 1–zu–1 Verhältnis zu den Komponenten von ψ stehen:

$$\Phi_{i,a,1}(x_{i,j}, y_j) = x_{i,j}\mathbb{1}_{[y_j=a]} \text{ und } \Phi_{a,b,2}(y_{j-1}, y_j) = \mathbb{1}_{[y_{j-1}=a]}\mathbb{1}_{[y_j=b]}$$

und beobachten, dass

$$h_{\mathbf{w}}(x) = y^* = \text{argmax}_{y\in\mathcal{Y}}\langle \mathbf{w}, \psi(x,y)\rangle = \text{argmax}_{y\in\mathcal{Y}} \sum_{j=1}^{r} \langle \mathbf{w}, \Phi(x_j, y_{j-1}, y_j)\rangle \ .$$

Dabei bezeichnet x_j die j-te Spalte der Matrix x. Die Variable y_{j-1} wird nur für $j \geq 2$ wirklich benötigt. Es sei $g_{\mathbf{w}}(x)$ der durch das Label $h_{\mathbf{w}}(x)$ erzielte „Profitwert“, d.h.,

$$g_{\mathbf{w}}(x) = \max_{y\in\mathcal{Y}} \langle \mathbf{w}, \psi(x,y)\rangle = \max_{y\in\mathcal{Y}} \sum_{j=1}^{r} \langle \mathbf{w}, \Phi(x_j, y_{j-1}, y_j)\rangle \ .$$

Wir beschreiben im Folgenden, wie sich $g_{\mathbf{w}}(x)$ effizient berechnen lässt. Wir verwenden dynamische Programmierung und benutzen die folgende 2-dimensionale Tabelle:

$$\forall b \in A, \forall k \in [r] : M_{b,k}(x|\mathbf{w}) = \max_{y_1 \ldots y_k : y_k = b} \sum_{j=1}^{k} \langle \mathbf{w}, \Phi(x_j, y_{j-1}, y_j)\rangle \ .$$

Auch bei dieser Tabelle kommt eine Abhängigkeit von y_{j-1} erst ab $j \geq 2$ ins Spiel. Offensichtlich gilt

$$g_{\mathbf{w}}(x) = \max_{b\in A} M_{b,r}(x|\mathbf{w}) \ .$$

Die Tabelle $M(x|\mathbf{w})$ lässt sich spaltenweise mit Hilfe der folgenden Rekursionsformel leicht ausfüllen:

$$M_{b,1}(x|\mathbf{w}) = \langle \mathbf{w}, \Phi(x_1, y_0, b)\rangle \text{ und } M_{b,k+1}(x|\mathbf{w}) = \max_a \left(M_{a,k}(x|\mathbf{w}) + \langle \mathbf{w}, \Phi(x_{k+1}, a, b)\rangle\right) \ .$$

Es gibt zwei Möglichkeiten $h_{\mathbf{w}}(x)$ effizient zu berechnen:

- Wenn beim Ausfüllen der Tabelle $M(x|\mathbf{w})$ ein paar Hilfsgrößen mitgeschleppt werden, dann kann man so nebenbei die beste Zeichensequenz $y^* = (y_1^*, \ldots, y_r^*)$ ausrechnen. S. das Lehrbuch zu den Details dieser Variante.

- Man kann mit Hilfe der Tabelle $M(x|\mathbf{w})$ und „Backtracking“ die beste Sequenz y^* ermitteln. „Backtracking“ bedeutet hier, dass die Tabelle von hinten nach vorn (Spalte r zuerst und Spalte 1 zuletzt) durchmustert wird. Ebenso wird die Sequenz y^* von hinten nach vorn berechnet. Details dazu könnten in den Übungen diskutiert werden.

12.4 Das allgemeine Rangordnungsproblem

Ein Rangordnungsproblem besteht darin, eine gegebene Folge von Instanzen nach ihrer Relevanz zu ordnen. S. dazu auch die zwei eingangs in diesem Kapitel aufgeführten Beispiele (Stichworte: Suchmaschinen, finanzielle online-Transaktionen).

Wir kommen nun zu einer formalen Beschreibung dieser Art von Problemen. Es bezeichne $\mathcal{X}^* = \cup_{r\geq 0}\mathcal{X}^r$ die Menge aller endlichen Sequenzen über einer Grundmenge $\mathcal{X}$. Somit ist $\mathbb{R}^* = \cup_{r\geq 0}\mathbb{R}^r$ die Menge aller endlichen Sequenzen über der Menge der reellen Zahlen. Wir betrachten Hypothesen der Form $h : \mathcal{X}^* \to \mathbb{R}^*$, welche Sequenzen $\mathbf{x} = (x_1, \ldots, x_r)$ der Länge r stets auf Sequenzen $h(\mathbf{x}) = (y_1, \ldots, y_r) \in \mathbb{R}^r$ abbilden. Eine Ungleichung $y_j > y_i$ bedeutet intuitiv, dass die Hypothese h die Instanz x_j in der Sequenz $\mathbf{x}$ für relevanter hält als die Instanz x_i. Falls die Komponenten in $\mathbf{y}$ paarweise verschieden sind, ergibt sich dadurch eine lineare Ordnung auf den Instanzen. Andernfalls erhalten wir lediglich eine partielle Ordnung.[27] Markierte Trainingsbeispiele stammen bei Rangordnungsproblemen über der Grundmenge $\mathcal{X}$ aus der Menge $\cup_{r\geq 0}(\mathcal{X}^r \times \mathbb{R}^r)$, d.h., ein markiertes Beispiel besteht aus einer Sequenz $\mathbf{x}$ und einem Vektor $\mathbf{y}$, welcher numerische Bewertungen der in $\mathbf{x}$ vorkommenden Instanzen enthält. Wenn es nur darum ginge, die Komponenten von $\mathbf{x} = (x_1, \ldots, x_r)$ nach ihrer Relevanz zu ordnen, dann könnten wir an Stelle der numerischen Bewertungen $\mathbf{y} = (y_1, \ldots, y_r)$ Rangnummern von 1 bis r einführen:

Definition 12.5 *Es sei $\mathbf{y} = (y_1, \ldots, y_r) \in \mathbb{R}^r$. Wenn y_i die k-kleinste Zahl in $\mathbf{y}$ ist, dann setzen wir $\bar{y}_i = k$. Wir nennen $\bar{y}_i$ die* Rangnummer *von x_i gemäß $\mathbf{y}$. Wenn mehrere k-kleinste Zahlen $y_{i(1)} = y_{i(2)} = \ldots = y_{i(s)}$ existieren und es gilt $i(1) < i(2) < \ldots < i(s)$, dann vergeben wir die Rangnummern $k, k+1, \ldots, k+s-1$ (in dieser Reihenfolge an die Komponenten $i(1), i(2), \ldots, i(s)$, d.h., wir setzen $\overline{y}_{i(j)} = k + j - 1$ für $j = 1, \ldots, s$.*[28]

Zum Beispiel erhalten wir für $\mathbf{y} = (2, 1, 6, -1, 1/2)$ die Gleichung $\bar{\mathbf{y}} = (4, 3, 5, 1, 2)$. Intuitiv können wir uns vorstellen, dass höhere Rangnummern eine höhere Relevanz anzeigen. Die numerischen Bewertungen in $\mathbf{y}$ enthalten aber mehr Information als die Rangnummern in $\bar{\mathbf{y}}$. Zum Beispiel würde auch der Vektor $\mathbf{y}' = (50, 1, 100, -1, 1/2)$ auf $\overline{\mathbf{y}'} = (4, 3, 5, 1, 2) = \bar{\mathbf{y}}$ abgebildet, die numerischen Werte 50 und 100 an Stelle von 2 und 6 könnten aber als nachdrücklichere Voten für die Vergabe der zwei höchsten Rangnummern gewertet werden.

Übungsaufgabe. Entwerfe einen Algorithmus, der zur Berechnung von $\bar{\mathbf{y}}$ aus $\mathbf{y} \in \mathbb{R}^r$ Zeit $O(r \log r)$ benötigt.

Es bezeichne V_r die Menge aller Tupel aus $[r]^r$, welche paarweise verschiedene Komponenten besitzen (also die Menge der Rangnummern-Vektoren mit r Komponenten). Offensichtlich ist $\mathbf{y} \mapsto \bar{\mathbf{y}}$ eine Abbildung von $\mathbb{R}^*$ nach $\cup_{r\geq 0}V_r$, die Sequenzen aus $\mathbb{R}^r$ stets auf Elemente von V_r abbildet. Da jede Zahl $k \in [r]$ trivialerweise die k-kleinste Nummer in $[r]$ ist, folgt

[27] In diesem Abschnitt können wir annehmen, dass Gleichheiten der Form $y_i = y_j$ nur selten auftreten. Dies wird sich in Abschnitt 12.5 aber ändern.

[28] Dies ist eine völlig willkürliche Festlegung, die nur dazu dient, die Abbildung $\mathbf{y} \mapsto \bar{\mathbf{y}}$ auf eindeutige Weise zu definieren.

$\bar{\mathbf{v}} = \mathbf{v}$ für alle $\mathbf{v} \in V_r$, d.h., der Rangnummern-Vektor zu einem Rangnummern-Vektor $\mathbf{v}$ ist $\mathbf{v}$ selber. Wir schreiben im Folgenden meist einfach V statt V_r (in dem Verständnis, dass die Anzahl der Komponenten in $\mathbf{y}$ bzw. $\bar{\mathbf{y}}$ immer mit der Länge der zugrunde liegenden Sequenz $\mathbf{x}$ übereinstimmt).

12.4.1 Kosten- und Verlustfunktionen

Es sei $\Delta : \cup_{r\geq 0}(\mathbb{R}^r \times \mathbb{R}^r) \to \mathbb{R}_0^+$ eine Funktion, die einem Paar $(\mathbf{y}', \mathbf{y})$ nicht-negative Kosten zuweist. Wir erinnern daran, dass eine Kostenfunktion Δ eine Verlustfunktion induziert:

$$\ell^{\Delta}(h, (\mathbf{x}, \mathbf{y})) = \Delta(h(\mathbf{x}), \mathbf{y}) \ .$$

Wir schreiben meist einfach ℓ statt ℓ^{Δ}, da die zugrunde liegende Kostenfunktion Δ in der Regel aus dem Kontext ersichtlich ist. Im Folgenden diskutieren wir einige Kostenfunktionen. Dabei ist jeweils nur der Fall $r \geq 2$ interessant (da sich für $r \leq 1$ das Rangordnungsproblem noch nicht in voller Schärfe stellt).

0–1 Kostenfunktion. Sie ist gegeben durch $\Delta(\mathbf{y}', \mathbf{y}) = \mathbb{1}_{[\overline{\mathbf{y}'} \neq \bar{\mathbf{y}}]}$, d.h., sie weist dem Paar $(\mathbf{y}', \mathbf{y})$ Kosten 0 zu, falls die von $\mathbf{y}'$ und $\mathbf{y}$ induzierten Rangnummern-Vektoren übereinstimmen, und Kosten 1 andernfalls. Offensichtlich gilt $\Delta(\mathbf{y}', \mathbf{y}) = \Delta(\overline{\mathbf{y}'}, \bar{\mathbf{y}})$, d.h., die numerischen Bewertungen gehen in die Kosten nur indirekt über die betreffenden Rangnummern ein. Die 0–1 Kostenfunktion ist *keine* gute Wahl, da sie nicht zwischen extrem abweichenden und fast identischen Rangordnungen unterscheidet.

Kendall–Tau Kostenfunktion. Sie zählt[29] die mittlere Anzahl der Paarvergleiche, die bei $\mathbf{y}'$ anders ausgehen als bei $\mathbf{y}$:

$$\Delta(\mathbf{y}', \mathbf{y}) = \frac{2}{(r-1)r} \sum_{1 \leq i < j \leq r} \mathbb{1}_{[\mathrm{sign}(y'_j - y'_i) \neq \mathrm{sign}(y_j - y_i)]} \ .$$

Auch hier gilt $\Delta(\mathbf{y}', \mathbf{y}) = \Delta(\overline{\mathbf{y}'}, \bar{\mathbf{y}})$.

NDCG Kostenfunktion. Wir setzen voraus, dass alle numerischen Bewertungsvektoren $\mathbf{y}$ nicht-negativ und verschieden vom Nullvektor sind. Weiter sei $D : \mathbb{N} \to \mathbb{R}^+$ eine monoton steigende Funktion. Die folgende Funktion repräsentiert einen kumulativen Gewinn (Cumulative Gain = CG), den $\mathbf{y}'$ bezüglich $\mathbf{y}$ erzielt:

$$G(\mathbf{y}', \mathbf{y}) = \sum_{i=1}^{r} D(\overline{y'}_i) y_i \in \mathbb{R}^+ \ . \tag{74}$$

[29] zumindest bei Vektoren $\mathbf{y}, \mathbf{y}'$, die jeweils paarweise verschiedene Komponenten besitzen

Es ist nicht schwer zu zeigen[30], dass $G(\mathbf{y}', \mathbf{y})$ zu gegebenem $\mathbf{y}$ maximal wird, wenn $\overline{\mathbf{y}'} = \bar{\mathbf{y}}$ (insbesondere also, wenn $\mathbf{y}' = \mathbf{y}$). Daher repräsentiert die Größe

$$0 < \frac{G(\mathbf{y}', \mathbf{y})}{G(\mathbf{y}, \mathbf{y})} \leq 1$$

so etwas wie einen normierten kumulativen Gewinn (Normalized Cumulative Gain = NCG). Durch Negieren dieser Größe erhalten wir die NDCG Kostenfunktion:

$$\Delta(\mathbf{y}', \mathbf{y}) = 1 - \frac{G(\mathbf{y}', \mathbf{y})}{G(\mathbf{y}, \mathbf{y})} = \frac{1}{G(\mathbf{y}, \mathbf{y})} \left(\sum_{i=1}^{r} y_i (D(\overline{y}_i) - D(\overline{y'}_i)) \right) \quad . \tag{75}$$

Es gilt $\Delta(\mathbf{y}', \mathbf{y}) = \Delta(\overline{\mathbf{y}'}, \mathbf{y})$, d.h., die numerischen Bewertungen in $\mathbf{y}'$ gehen in die Kosten nur indirekt über die Rangnummern in $\overline{\mathbf{y}'}$ ein. Die numerischen Bewertungen in $\mathbf{y}$ hingegen haben einen direkten Einfluss auf $\Delta(\mathbf{y}', \mathbf{y})$.
Die Aufgabe der Funktion D ist im Übrigen, den großen Rangnummern in $\overline{\mathbf{y}'}$ einen (etwas) höheren Einfluss einzuräumen als den kleinen. Dies wird der Tatsache gerecht, dass bei Anwendungen in der Regel nur die Instanzen von hohem Rang wirklich relevant sind. Eine typische Wahl von D wäre etwa

$$D(i) = \begin{cases} \frac{1}{\log_2(r-i+2)} & \text{falls } i \in \{r-k+1, \ldots, r\} \\ 0 & \text{falls } i \in \{1, \ldots, r-k\} \end{cases} \quad .$$

Die resultierende Kostenfunktion berücksichtigt nur die k Spitzenränge. Für den Spitzenrang r gilt $D(r) = 1$. Für den kleinsten noch berücksichtigten Rang $r - k + 1$ gilt $D(r - k + 1) = 1/\log_2(k + 1)$.

12.4.2 Lineare Voraussagefunktionen und konvexe Surrogat-Zielfunktionen

Wir setzen $\mathcal{X} = \mathbb{R}^d$ voraus. Eine Sequenz $X = (\mathbf{x_1}, \ldots, \mathbf{x_r}) \in (\mathbb{R}^d)^r$ setzt sich dann aus r Vektoren $\mathbf{x_1}, \ldots, \mathbf{x_r}$ des Euklidischen Raumes $\mathbb{R}^d$ zusammen. Die folgende Hypothese ermittelt ihre numerischen Bewertungen der Instanzen $\mathbf{x_1}, \ldots, \mathbf{x_r}$ mit einer linearen Funktion (gegeben durch einen Gewichtsvektor $\mathbf{w} \in \mathbb{R}^d$):

$$h_{\mathbf{w}}(X) = (\langle \mathbf{w}, \mathbf{x_1} \rangle, \ldots, \langle \mathbf{w}, \mathbf{x_r} \rangle) \quad .$$

Das Auffinden eines Gewichtsvektors $\mathbf{w}$ mit einem minimalen empirischen Verlust

$$L_S^{\Delta}(h_{\mathbf{w}}) = \frac{1}{m} \sum_{i=1}^{m} \Delta(h_{\mathbf{w}}(X_i), y_i)$$

auf einer gegebenen Trainingsmenge

$$S = [(X_1, y_1), \ldots, (X_m, y_m)]$$

[30] Das wäre eine gute Übungsaufgabe.

(mit $(X_i, y_i) \in (\mathbb{R}^d)^r \times \mathbb{R}^r$) ist für die meisten nicht-trivialen Kostenfunktionen Δ ein NP-hartes Problem. Einen Ausweg bieten (wie schon oft in der Vorlesung) konvexe obere Schranken einer nicht konvexen Kostenfunktion, die zu konvexen Surrogat-Zielfunktionen für die eigentliche Zielfunktion $L_S(h_{\mathbf{w}})$ führen:

Bemerkung 12.6 *Falls $\Delta'(h_{\mathbf{w}}(X), \mathbf{y})$ eine in $\mathbf{w}$ konvexe Funktion ist und falls $\Delta'(h_{\mathbf{w}}(X), \mathbf{y}) \geq \Delta(h_{\mathbf{w}}(X), \mathbf{y})$, dann ist auch $L_S^{\Delta'}(h_{\mathbf{w}})$ eine in w konvexe Funktion mit $L_S^{\Delta'}(h_{\mathbf{w}}) \geq L_S^{\Delta}(h_{\mathbf{w}})$.*

Es genügt daher im Folgenden, eine konvexe obere Schranke Δ' von Δ zu finden. Das führen wir für die Kendall–Tau und für die NDCG Kostenfunktion vor. In beiden Fällen ist Δ' eine Variante der Hinge Kostenfunktion.

Konvexe obere Schranke für die Kendall–Tau Kostenfunktion. Wir beobachten zunächst, dass

$$\mathbb{1}_{[\text{sign}(y_j - y_i) \neq \text{sign}(y'_j - y'_i)]} \leq \mathbb{1}_{[\text{sign}(y_j - y_i) \cdot (y'_j - y'_i) \leq 0]} \ .$$

Für $\mathbf{y}' = h_{\mathbf{w}}(X)$ ergibt sich

$$\mathbb{1}_{[\text{sign}(y_j - y_i)(y'_j - y'_i) \leq 0]} = \mathbb{1}_{[\text{sign}(y_j - y_i) \cdot \langle \mathbf{w}, \mathbf{x_j} - \mathbf{x_i} \rangle \leq 0]} \leq \max\{0, 1 - \text{sign}(y_j - y_i) \cdot \langle \mathbf{w}, \mathbf{x_j} - \mathbf{x_i} \rangle\} \ .$$

Für die Kendall–Tau Kostenfunktion Δ gilt daher

$$\Delta(h_{\mathbf{w}}(X), \mathbf{y}) \leq \frac{2}{(r-1)r} \sum_{1 \leq i < j \leq r} \max\{0, 1 - \text{sign}(y_j - y_i) \cdot \langle \mathbf{w}, \mathbf{x_j} - \mathbf{x_i} \rangle\} =: \Delta'(h_{\mathbf{w}}(X), \mathbf{y}) \ .$$

$\Delta'(h_{\mathbf{w}}(X), \mathbf{y})$ ist die gesuchte in $\mathbf{w}$ konvexe obere Schranke der Kendall–Tau Kostenfunktion.

Konvexe obere Schranke für die NDCG Kostenfunktion. Es sei $D : \mathbb{N} \to \mathbb{R}^+$ eine monoton wachsende Funktion und $G(\mathbf{y}', \mathbf{y})$ die in (74) definierte kumulative Gewinnfunktion. Aus der uns bereits bekannten Ungleichung

$$G(\mathbf{y}', \mathbf{y}) = \sum_{i=1}^{r} D(\overline{y'}_i) y_i \leq G(\mathbf{y}, \mathbf{y})$$

lässt sich leicht folgende Gleichung für $\bar{\mathbf{y}}$ ableiten[31]:

$$\bar{\mathbf{y}} = \text{argmax}_{\mathbf{v} \in V} \sum_{i=1}^{r} v_i y_i \ . \tag{76}$$

Im Spezialfall $X = (\mathbf{x_1}, \ldots, \mathbf{x_r}) \in (\mathbb{R}^d)^r$ und $\mathbf{y} = h_{\mathbf{w}}(X) = (\langle \mathbf{w}, \mathbf{x_1} \rangle, \ldots, \langle \mathbf{w}, \mathbf{x_r} \rangle)$ und mit der Definition $\psi(X, v) = \sum_{i=1}^{r} v_i \mathbf{x_i}$ ergibt sich aus (76) die Gleichung

$$\overline{h_{\mathbf{w}}(X)} = \text{argmax}_{\mathbf{v} \in V} \sum_{i=1}^{r} v_i \langle \mathbf{w}, \mathbf{x_i} \rangle = \text{argmax}_{\mathbf{v} \in V} \langle \mathbf{w}, \sum_{i=1}^{r} v_i \mathbf{x_i} \rangle = \text{argmax}_{\mathbf{v} \in V} \langle \mathbf{w}, \psi(X, \mathbf{v}) \rangle \ . \tag{77}$$

[31] Das wäre eine gute Übungsaufgabe.

Für eine beliebige Kostenfunktion Δ mit $\Delta(\mathbf{y}', \mathbf{y}) = \Delta(\overline{\mathbf{y}'}, \mathbf{y})$ ergibt sich folgende Rechnung:

$$\begin{aligned}
\Delta(h_{\mathbf{w}}(X), \mathbf{y}) &= \Delta(\overline{h_{\mathbf{w}}(X)}, \mathbf{y}) \\
&\overset{(77)}{\leq} \Delta(\overline{h_{\mathbf{w}}(X)}, \mathbf{y}) + \langle \mathbf{w}, \psi\left(X, \overline{h_{\mathbf{w}}(X)}\right)\rangle - \langle \mathbf{w}, \psi(X, \bar{\mathbf{y}})\rangle \\
&\leq \max_{\mathbf{v} \in V} \left(\Delta(\mathbf{v}, \mathbf{y}) + \langle \mathbf{w}, \psi(X, \mathbf{v})\rangle - \langle \mathbf{w}, \psi(X, \bar{\mathbf{y}})\rangle\right) \\
&\overset{*}{=} \max_{\mathbf{v} \in V} \left(\Delta(\mathbf{v}, \mathbf{y}) + \sum_{i=1}^{r} (v_i - \bar{y}_i)\langle \mathbf{w}, \mathbf{x_i}\rangle\right) =: \Delta'(h_{\mathbf{w}}(X), \mathbf{y})
\end{aligned}$$

In der mit "$*$" markierten Gleichung wurde einfach die Definition von $\psi(X, v)$ bzw. $\psi(X, \bar{\mathbf{y}})$ expandiert. $\Delta'(h_{\mathbf{w}}(X), \mathbf{y})$ ist die gesuchte in w konvexe obere Schranke für die NDCG Kostenfunktion.

Wir stellen uns in diesem Abschnitt abschließend die Aufgabe, die Funktion Δ', die sich als konvexe obere Schranke der NDCG Kostenfunktion ergeben hat, zu berechnen. Dies führt uns zu folgendem Optimierungsproblem. Zu gegebenen Parametern $X, \mathbf{y}, w$ finde

$$\mathbf{v}^* = \text{argmax}_{\mathbf{v} \in V} \left(\Delta(\mathbf{v}, \mathbf{y}) + \sum_{i=1}^{r} (v_i - \overline{y_i})\langle \mathbf{w}, \mathbf{x_i}\rangle\right) \quad . \tag{78}$$

Ein Durchmustern aller $\mathbf{v} \in V = V_r$ (exhaustive search) wäre wegen $|V_r| = r!$ hochgradig ineffizient. Wir werden das durch (78) gegebene Optimierungsproblem effizient auf das sogenannte „Zuweisungsproblem" (assignment problem) reduzieren. Für letzteres existieren effiziente Algorithmen. Details folgen.

Aus der Gleichung (75) für die NDCG Kostenfunktion $\Delta(\mathbf{y}', \mathbf{y})$ ergibt sich im Spezialfall $\mathbf{y}' = \mathbf{v} \in V$ wegen $\bar{\mathbf{v}} = \mathbf{v}$ die folgende Gleichung:

$$\Delta(\mathbf{v}, \mathbf{y}) = \frac{1}{G(\mathbf{y}, \mathbf{y})} \sum_{i=1}^{r} (D(\overline{y}_i) - D(v_i)) y_i \quad .$$

Wenn wir dies in (78) einsetzen und ausnutzen, dass die Terme $\overline{y}_i$ und $D(\overline{y}_i)$ nur einen konstanten (von v unabhängigen) Beitrag zur Zielfunktion leisten, erhalten wir folgendes:

$$\mathbf{v}^* = \text{argmax}_{\mathbf{v} \in V} \sum_{i=1}^{r} \left(v_i \langle \mathbf{w}, \mathbf{x_i}\rangle - \frac{D(v_i) y_i}{G(\mathbf{y}, \mathbf{y})}\right) = \text{argmin}_{v \in V} \sum_{i=1}^{r} \left(\frac{D(v_i) y_i}{G(\mathbf{y}, \mathbf{y})} - v_i \langle \mathbf{w}, \mathbf{x_i}\rangle\right) \quad . \tag{79}$$

Es bezeichne $K_{r,r}$ den vollständigen bipartiten Graphen mit den zwei Knotenmengen $A = \{a_1, \ldots, a_r\}$ und $B = \{b_1, \ldots, b_r\}$ und den r^2 Kanten $\{a_i, b_j\}$ mit Kantenkosten $c_{i,j}$ für $1 \leq i, j \leq r$. Wir können uns $\mathbf{v} \in V$ vorstellen als ein perfektes Heiratssystem (perfect matching), wobei a_i mit b_{v_i} „verheiratet" wird.[32] Die Gesamtkosten des Heiratssystems $\mathbf{v}$ betragen dann

$$C(\mathbf{v}) = \sum_{i=1}^{r} c_{i,v_i} \quad .$$

[32]Die Definition von $V = V_r$ impliziert, dass jeder der $2r$ Knoten über die „Heiratsvorschrift" $\mathbf{v}$ exakt einen Partner erhält.

Setzen wir

$$c_{i,j} = \frac{D(j)y_i}{G(\mathbf{y},\mathbf{y})} - j\langle \mathbf{w}, \mathbf{x_i} \rangle \ ,$$

so können wir das Minimierungsproblem in (79) interpretieren als das Problem, im bipartiten Graphen $K_{r,r}$ mit Kantenkosten $(c_{i,j})_{1\leq i,j\leq r}$ ein perfektes Heiratssystem $\mathbf{v}$ mit minimalen Gesamtkosten zu bestimmen. Im Englischen spricht man von einem „Minimum Weight Perfect Matching“ oder auch von einem „Assignment“ Problem, im Deutschen von einem Zuweisungsproblem. Der schnellste bekannte Algorithmus reduziert das Zuweisungsproblem auf das sogenannte „Minimum Cost Flow“ Problem und verwendet für letzteres den sogenannten „Successive Shortest Path“ Algorithmus. Die Darstellung dieses Verfahrens würde den Rahmen unserer Vorlesung sprengen. Wir behandeln aber im nächsten Abschnitt eine Reduktion des Zuweisungsproblems auf „Lineare Programmierung“.

12.4.3 Reduktion des Zuweisungsproblems auf Lineare Programmierung

Eine Matrix $B \in \{0,1\}^{r\times r}$ heißt *Permutationsmatrix*, wenn in jeder Zeile und jeder Spalte exakt ein Eintrag den Wert 1 hat. Beachte, dass die Permutationsmatrizen 1–zu–1 den perfekten Heiratssystemen für den Graphen $K_{r,r}$ entsprechen: a_i wird mit b_j verheiratet, wenn der eindeutige 1-Eintrag von Zeile i in Spalte j zu finden ist. Eine Matrix $B \in [0,1]^{r\times r}$ heißt *doppelt stochastisch*, wenn jede Zeilensumme und jede Spaltensumme in B exakt den Wert 1 liefert. Wir zitieren ohne Beweis das folgende Resultat:

Theorem 12.7 (Birkhoff, 1946) *Jede doppelt stochastische Matrix lässt sich als konvexe Kombination von Permutationsmatrizen darstellen.*

Offensichtlich entspricht das Zuweisungsproblem für den Graphen $K_{r,r}$ mit Kantenkosten $(c_{i,j})$ dem folgenden Optimierungsproblem. Finde eine $(r \times r)$-Matrix B, die die Kostenfunktion

$$C(B) = \sum_{i,j=1}^{r} c_{i,j} b_{i,j}$$

minimiert und dabei folgende Randbedingungen einhält:

1. Für alle $1 \leq i,j \leq r$: $0 \leq b_{i,j} \leq 1$.
2. Für alle $i \in [r]$ gilt $\sum_{j=1}^{r} b_{i,j} = 1$.
3. Für alle $j \in [r]$ gilt $\sum_{i=1}^{r} b_{i,j} = 1$.
4. Für alle $1 \leq i,j \leq r$ gilt $b_{i,j} \in \{0,1\}$.

Die zweite (bzw. dritte) Randbedingung kontrolliert, dass alle Zeilensummen (bzw. Spaltensummen) den Wert 1 ergeben. Die ersten drei Bedingungen erzwingen demnach gerade, dass B eine doppelt stochastische Matrix ist. Zusammen mit der dritten vierten wird dann erzwungen, dass B eine Permutationsmatrix ist. Ohne die vierte Randbedingung hätten wir ein lineares Programmierungsproblem vorliegen. Wir können das Zuweisungsproblem nun lösen wie folgt:

- Finde eine optimale Basislösung zur Minimierung von $C(B)$ unter den ersten beiden Randbedingungen (zum Beispiel mit dem Simplexverfahren).

Eine optimale Basislösung entspricht nämlich den Ecken des Polytops, das durch die linearen Randbedingungen gegeben ist. In unserem konkreten Problem entsprechen die Ecken des Polytops gerade den Permutationsmatrizen.[33] Mit anderen Worten: wenn wir nicht irgendeine optimale Lösung berechnen, sondern eine optimale *Basislösung*, dann wird die vierte (und problematische) Randbedingung automatisch eingehalten!

12.5 Das bipartite Rangordnungsproblem

In diesem Abschnitt betrachten wir Rangordnungsprobleme mit markierten Beispielen der Form $(\mathbf{x}, \mathbf{y}) \in \mathcal{X}^r \times \{\pm 1\}^r$, d.h., die Instanzen in $\mathbf{x}$ werden entweder mit 1 oder mit -1 bewertet. Rangordnungsprobleme mit dieser Einschränkung werden *bipartit* genannt.

Intuitiv können wir die Bewertung „1“ (bzw. „-1“) als Indikator für „Relevanz“ (bzw. „Irrelevanz“) auffassen. Das Erlernen einer bipartiten Rangordnung sieht auf den ersten Blick aus wie ein binäres Klassifikationsproblem. Während aber die 0–1 Kostenfunktion eine gute Wahl für binäre Klassifikationsprobleme ist, ist sie i.A. ungeeignet, um die bei binären Rangordnungsproblemen vorliegende Zielsetzung zu erfassen.

Beispiel 12.8 (Aufdeckung von Betrug) *Bei der Überwachung von finanziellen online-Transaktionen werden betrügerische Aktivitäten i.A. nur einen kleinen Bruchteil ausmachen, sagen wir* $1/1000$. *Die konstante* -1*-Hypothese, die jede Transaktion als „kein Betrug“ bewertet, hätte dann die Fehlerrate* $1/1000$. *Dies ist für ein binäres Klassifikationsproblem kein schlechtes Ergebnis! Jedoch würde eine solche Hypothese* keinen *Beitrag zum Aufspüren betrügerischer Transaktionen leisten.*

12.5.1 Kostenfunktionen

Wir wollen nun erarbeiten, welche Kostenfunktionen statt der 0–1 Kosten verwendet werden sollten. Wenn wir $\mathbf{y} \in \{\pm 1\}^r$ als den Vektor mit den korrekten ± 1-Werten ansehen und $\mathbf{y}' = h(\mathbf{x}) \in \mathbb{R}^r$ bzw. $(\operatorname{sign}(y'_1), \ldots, \operatorname{sign}(y'_r)) \in \{\pm 1\}^r$ als die von einer Hypothese h vorgenommenen Bewertungen, dann können folgende Fälle auftreten:

Korrekte Positivbewertungen (true positives): $y_i = 1$ und $\operatorname{sign}(y'_i) = 1$. Wir setzen

$$a = |\{i \in [r] : y_i = 1 \wedge \operatorname{sign}(y'_i) = 1\}| \ .$$

Falsche Positivbewertungen (false positives): $y_i = -1$ und $\operatorname{sign}(y'_i) = 1$. Wir setzen

$$b = |\{i \in [r] : y_i = -1 \wedge \operatorname{sign}(y'_i) = 1\}| \ .$$

[33] Der Beweis dieser Behauptungen soll hier nicht geführt werden, da wir im Rahmen dieser Vorlesung nicht tiefer in die Grundlagen zur Theorie der linearen Programmierung einsteigen wollen.

Falsche Negativbewertungen (false negatives): $y_i = 1$ und $\text{sign}(y'_i) = -1$. Wir setzen

$$c = |\{i \in [r] : y_i = 1 \wedge \text{sign}(y'_i) = -1\}| \ .$$

Korrekte Negativbewertungen (true negatives): $y_i = -1$ und $\text{sign}(y'_i) = -1$. Wir setzen

$$d = |\{i \in [r] : y_i = -1 \wedge \text{sign}(y'_i) = -1\}| \ .$$

Definition 12.9 *Es seien a, b, c, d die Zählparameter für die oben genannnten vier Fälle, die sich beim Abgleich von $\mathbf{y}' \in \mathbb{R}^r$ mit $\mathbf{y} \in \{\pm 1\}^r$ ergeben. Dann heißt der Quotient $\frac{a}{a+c}$ (bzw. der Quotient $\frac{a}{a+b}$) die* Sensitivität (sensitivity) *(bzw. die* Präzision (precision)*) von $\mathbf{y}'$ bezüglich $\mathbf{y}$. Der Quotient $\frac{d}{d+b}$ heißt im Englischen „specificity“.*

Erwünscht sind Hypothesen h, die auf Beispielen $(\mathbf{x}, \mathbf{y})$ Voraussagen $\mathbf{y}' = h(\mathbf{x})$ treffen, welche bezüglich $\mathbf{y}$ hohe Präzisions- und Sensitivitätswerte erzielen.

Beispiel 12.10 (Aufdeckung von Betrug — fortgesetzt) *Nehmen wir an, dass die* 1*-Einträge in $\mathbf{y}$ Betrugsfälle markieren. Dann liefert $\frac{a}{a+c}$ den Bruchteil der Betrugsfälle, welche durch entsprechende* 1*-Einträge in $\mathbf{y}' = h(\mathbf{x})$ erfolgreich angezeigt werden. Die Hypothese h muss in diesem Sinne* sensitiv *auf relevante Instanzen (hier: Betrugsfälle) reagieren. Eine Sensitivität von* 1 *lässt sich auf triviale Weise erzielen, indem $\mathbf{y}'$ als die konstante* 1*-Hypothese gewählt wird, die jedes Mal auf „Betrug“ wettet. In diesem Fall wäre die Präzision $\frac{a}{a+b}$ sehr gering (insbesondere wenn Betrugsfälle relativ selten vorkommen).*

Wir führen die folgende Kurzschreibweise ein:

$$\text{sensitivity} = \frac{a}{a+c} \ , \ \text{precision} = \frac{a}{a+b} \text{ und specificity} = \frac{d}{d+b} \ .$$

Wir behalten dabei im Kopf, dass die Parameter a, b, c, d Funktionen in $\mathbf{y}'$ und $\mathbf{y}$ sind. Die Parameter sensitivity, precision und specificity sind Funktionen in a, b, c, d (und können daher indirekt als Funktionen in $\mathbf{y}', \mathbf{y}$ aufgefasst werden).

Wir betrachten Kostenfunktionen Δ von der Form $\Delta(\mathbf{y}', \mathbf{y}) = 1 - F$ wobei F durch eine Formel in den Variablen sensitivity, precision und specificity (oder alternativ als Formel in den Variablen a, b, c, d) ausgedrückt werden kann. Wir listen ein paar populäre Wahlen von F auf:

1. Setze F gleich dem arithmetischen Mittel von sensitivity und specificity:
$$\bar{F} = \frac{1}{2}(\text{sensitivity} + \text{specificity}) = \frac{1}{2}\left(\frac{a}{a+c} + \frac{d}{d+b}\right) \ .$$
2. Setze F_1 gleich dem harmonischen Mittelwert von sensitivity und precision:
$$F_1 = \frac{2}{\text{sensitivity}^{-1} + \text{precision}^{-1}} = \frac{2a}{2a+b+c} \ .$$
3. Es sei $\beta > 0$. Wir erhalten eine Verallgemeinerung von F_1, wobei der Beitrag von sensitivity^{-1} zum harmonischen Mittel mit β^2 gewichtet wird:
$$F_\beta = \frac{1+\beta^2}{\beta^2 \cdot \text{sensitivity}^{-1} + \text{precision}^{-1}} = \frac{(1+\beta^2)a}{(1+\beta^2)a + b + \beta^2 c} \ .$$

12.5.2 Lineare Voraussagefunktionen und konvexe Surrogat-Zielfunktionen

Wir setzen wieder $\mathcal{X} = \mathbb{R}^d$ voraus. Für $w \in \mathbb{R}^d$, $\theta \in \mathbb{R}$ und $X = (\mathbf{x_1}, \ldots, \mathbf{x_r}) \in (\mathbb{R}^d)^r$ definieren wir

$$h_{\mathbf{w},\theta}(X) = (\langle \mathbf{w}, \mathbf{x_1} \rangle + \theta, \ldots, \langle \mathbf{w}, \mathbf{x_r} \rangle + \theta) \ .$$

Statt $h_{\mathbf{w},0}$ schreiben wir einfach $h_{\mathbf{w}}$ (und diese Definition von $h_{\mathbf{w}}$ deckt sich mit der in Abschnitt 12.4.2 verwendeten Definition). Die Funktion $b_\theta : \mathbb{R}^* \to \{\pm 1\}^*$ ist definiert wie folgt:

$$b_\theta(a_1, \ldots, a_r) = (\text{sign}(a_1 + \theta), \ldots, \text{sign}(a_r + \theta)) \ .$$

Statt b_0 schreiben wir einfach b. Es gilt

$$b_\theta(h_{\mathbf{w}}(X)) = b_\theta(\langle \mathbf{w}, \mathbf{x_1} \rangle, \ldots, \langle \mathbf{w}, \mathbf{x_r} \rangle) = (\text{sign}(\langle \mathbf{w}, \mathbf{x_1} \rangle + \theta), \ldots, \text{sign}(\langle \mathbf{w}, \mathbf{x_r} \rangle + \theta)) \ .$$

Somit liefert $b_\theta(h_{\mathbf{w}}(X))$ die von $h_{\mathbf{w},\theta}$ induzierten ± 1-Bewertungen der Instanzen $\mathbf{x_1}, \ldots, \mathbf{x_r}$.

Bei Verwendung der im vorigen Abschnitt genannten Kostenfunktionen wird die Klasse der linearen Voraussagefunktionen meistens durch die Festlegung $\theta = 0$ eingeschränkt. Die Verwendung der zusätzlichen Variable θ kann aber sinnvoll sein in folgenden Fällen:

- Die Hypothese $b_\theta \circ h_{\mathbf{w}}$ soll höchstens k Instanzen in $X \in (\mathbb{R}^d)^r$ mit 1 bewerten. Dann kann nach Ermittlung eines Gewichtsvektors w der Parameter θ solange verkleinert werden, bis höchstens k innere Produkte $\langle \mathbf{w}, \mathbf{x_i} \rangle$ oberhalb der Schwelle $-\theta$ liegen.
- Die Hypothese $b_\theta \circ h_{\mathbf{w}}$ soll höchstens k Instanzen in $X \in (\mathbb{R}^d)^r$ mit -1 bewerten. Dann kann nach Ermittlung eines Gewichtsvektors w der Parameter θ solange vergrößert werden, bis höchstens k innere Produkte $\langle \mathbf{w}, \mathbf{x_i} \rangle$ unterhalb der Schwelle $-\theta$ liegen.

Eine Beschränkung auf höchstens k viele 1-Bewertungen kann zum Beispiel sinnvoll sein, wenn jeder 1-Eintrag einer aufwändigen Nachbearbeitung bedarf (wie es zum Beispiel bei verdächtigen finanziellen online-Transaktionen der Fall wäre). Eine analoge Bemerkung gilt für die Beschränkung der Anzahl der -1-Bewertungen.

Aus Effizienzgründen bietet es sich wieder an die eigentliche Kostenfunktion $\Delta(h_{\mathbf{w},\theta}(X), \mathbf{y})$ durch eine in $\mathbf{w}$ konvexe obere Schranke $\Delta'(h_{\mathbf{w},\theta}(X), \mathbf{y})$ zu ersetzen. Wir setzen dabei voraus, dass Δ von $h_{\mathbf{w},\theta}(X)$ nur indirekt über $b_\theta(h_{\mathbf{w}}(X))$ abhängt, d.h., es gilt die Gleichung

$$\Delta(h_{\mathbf{w},\theta}(X), \mathbf{y}) = \Delta(b_\theta(h_{\mathbf{w}}(X)), \mathbf{y}) \ .$$

Offensichtlich gilt

$$b(h_{\mathbf{w}}(X)) = \text{argmax}_{\mathbf{v} \in V} \sum_{i=1}^{r} v_i \langle \mathbf{w}, \mathbf{x_i} \rangle \ .$$

Es gibt eine ähnliche Formel für b_θ:

Übungsaufgabe. Es sei $V = \{\pm 1\}^r$ und $V_{\leq k}$ (bzw. $V_{\geq k}$) sei die Menge der Vektoren aus V mit höchstens (bzw. mindestens) k 1-Komponenten. Zeige, dass es zu einem gegebenen Gewichtsvektor $\mathbf{w}$ und zu jeder Wahl von $\theta \in \mathbb{R}$ ein $k \geq 0$ gibt, so dass für $V' = V_{\leq k}$ oder für $V' = V_{\geq k}$ gilt:

$$b_\theta(h_{\mathbf{w}}(X)) = \text{argmax}_{\mathbf{v}\in V'} \sum_{i=1}^{r} v_i \langle \mathbf{w}, \mathbf{x_i} \rangle \ . \tag{80}$$

Die folgende Rechnung liefert unter der Voraussetzung $y \in V'$ als Ergebnis die gesuchte in w konvexe obere Schranke Δ' von Δ:

$$\begin{aligned}
\Delta(h_{\mathbf{w},\theta}(X), \mathbf{y}) &= \Delta(b_\theta(h_{\mathbf{w}}(X)), \mathbf{y}) \\
&\overset{(80)}{\leq} \Delta(b_\theta(h_{\mathbf{w}}(X)), \mathbf{y}) + \left(\sum_{i=1}^{r} (b_\theta(h_{\mathbf{w}}(X))_i - y_i) \langle \mathbf{w}, \mathbf{x_i} \rangle \right) \\
&\leq \max_{\mathbf{v}\in V'} \left(\Delta(\mathbf{v}, \mathbf{y}) + \sum_{i=1}^{r} (v_i - y_i) \langle \mathbf{w}, \mathbf{x_i} \rangle \right) =: \Delta'(h_{\mathbf{w},\theta}(X), \mathbf{y}) \ .
\end{aligned}$$

Wir stellen uns in diesem Abschnitt abschließend die Aufgabe, die Funktion Δ', die sich als konvexe obere Schranke der Kostenfunktion Δ ergeben hat, zu berechnen. Zu diesem Zweck machen wir zwei Voraussetzungen:

1. Der Wert der Kostenfunktion ist durch die Parameter sensitivity, precision und specificity vollständig bestimmt.

2. Es bezeichne $P = P(\mathbf{y})$ (bzw. $N = N(\mathbf{y})$) die Anzahl der Komponenten $i \in [r]$ von $\mathbf{y}$ mit $y_i = 1$ (bzw. $y_i = -1$). Für $0 \leq a \leq P$ und $0 \leq b \leq N$ bezeichne $V_{\mathbf{y}}[a, b]$ die Menge aller Vektoren aus $V = \{\pm 1\}^r$ mit genau a 1-Komponenten im Bereich $\{i \in [r] : y_i = 1\}$ und genau $b \leq N$ 1-Komponenten im Bereich $\{i \in [r] : y_i = -1\}$. Wir setzen voraus, dass V' sich mit einer geeigneten Auswahl der Paare (a, b) darstellen lässt als eine Vereinigung von Mengen der Form $V_{\mathbf{y}}[a, b]$.

Die erste Voraussetzung ist für alle von uns im Abschnitt 12.5.1 genannten Kostenfunktionen erfüllt. Die zweite Voraussetzung ist zum Beispiel erfüllt für die wichtigen Fälle $V' \in \{V, V_{\leq k}, V_{\geq k}\}$. Für die Menge $V = \{\pm 1\}^r$ ist das offensichtlich. $V_{\leq k}$ lässt sich (für jede mögliche Wahl von $\mathbf{y}$) darstellen als

$$V_{\leq k} = \bigcup_{(a,b):a+b\leq k, 0\leq a\leq P(\mathbf{y}), 0\leq b\leq N(\mathbf{y})} V_{\mathbf{y}}[a, b] \ .$$

Eine analoge Darstellung gilt für $V_{\geq k}$. Die Ungleichungen $0 \leq a \leq P(y)$ und $0 \leq b \leq N(y)$ können bei jeder (sinnvollen) Wahl von V' vorausgesetzt werden.

Widmen wir uns jetzt dem Optimierungsproblem, das sich bei der Auswertung der Kostenfunktion Δ' stellt: Zu gegebenen Parametern $X, \mathbf{y}, \mathbf{w}$ finde

$$\mathbf{v}^* = \text{argmax}_{\mathbf{v}\in V'} \left(\Delta(\mathbf{v}, \mathbf{y}) + \sum_{i=1}^{r} (v_i - y_i) \langle \mathbf{w}, \mathbf{x_i} \rangle \right) = \text{argmax}_{\mathbf{v}\in V'} \left(\Delta(\mathbf{v}, \mathbf{y}) + \sum_{i=1}^{r} v_i \langle \mathbf{w}, \mathbf{x_i} \rangle \right) \ . \tag{81}$$

Hier sind die entscheidenden Beobachtungen:

- Da V' sich als Vereinigung von Mengen der Form $V_{\mathbf{y}}[a,b]$ darstellen lässt, können wir für jede Wahl von a, b den besten Vektor $\mathbf{v}^*_{\mathbf{a},\mathbf{b}}$ in $V_{\mathbf{y}}[a,b]$ ermitteln.
- Für eine feste Wahl von a und b gilt (unabhängig von der Auswahl von $\mathbf{v} \in V_{\mathbf{y}}[a,b]$)

$$\text{sensitivity} = \frac{a}{P} \ , \ \text{precision} = \frac{a}{a+b} \text{ und specificity} = \frac{b}{N} \ .$$

  Daher liefert der Kostenterm $\Delta(\mathbf{v}, \mathbf{y})$ in (81) für alle Wahlen von $\mathbf{v} \in V_{\mathbf{y}}[a,b]$ den gleichen Beitrag. Dadurch wird das Optimierungsproblem runtergekocht auf

$$\mathbf{v}^*_{\mathbf{a},\mathbf{b}} = \text{argmax}_{\mathbf{v} \in V_{\mathbf{y}}[a,b]} \left(\sum_{i=1}^{r} v_i \langle \mathbf{w}, \mathbf{x_i} \rangle \right) \ .$$

- Die Berechnung von $\mathbf{v}^*_{\mathbf{a},\mathbf{b}}$ ist einfach:
  - Sortiere die Paare $(i, \langle \mathbf{w}, \mathbf{x_i} \rangle)$ absteigend nach der zweiten Komponente.
  - Ermittle daraus die Folge $i_1, \ldots, i_P$ der Indizes i mit $y_i = 1$, so dass $\langle \mathbf{w}, \mathbf{x_{i_k}} \rangle \geq \langle \mathbf{w}, \mathbf{x_{i_{k+1}}} \rangle$ für $k = 1, \ldots, P-1$, und ebenso die Folge $j_1, \ldots, j_N$ der Indizes j mit $y_j = -1$, so dass $\langle \mathbf{w}, \mathbf{x_{j_k}} \rangle \geq \langle \mathbf{w}, \mathbf{x_{j_{k+1}}} \rangle$ für $k = 1, \ldots, N-1$.
  - Setze $v_{i_1}, \ldots, v_{i_a}$ und $v_{j_1}, \ldots, v_{j_b}$ auf den Wert 1 und die restlichen Komponenten von v auf den Wert -1.
- Aus der Kollektion der Vektoren $\mathbf{v}^*_{\mathbf{a},\mathbf{b}}$ wählen wir den „Champion“ $\mathbf{v}^*$ aus, also denjenigen unter den Vektoren $\mathbf{v}^*_{\mathbf{a},\mathbf{b}}$, dessen Wert bei der Zielfunktion $\Delta(\mathbf{v}, \mathbf{y}) + \sum_{i=1}^{r} v_i \langle \mathbf{w}, \mathbf{x_i} \rangle$ am größten ist.

Weitere Implementierungsdetails sind leicht einzufüllen. Es ist offensichtlich, dass dieser Algorithmus einen besten Vektor $\mathbf{v}^*$ des Optimierungsproblems (81) ausgibt.

13 Entscheidungsbäume

Ein Entscheidungsbaum T stellt eine Voraussagefunktion $h_T : \mathcal{X} \to \mathcal{Y}$ dar. Stellen wir uns der Einfachheit halber einen binären Entscheidungsbaum T vor. Dann gilt folgendes:

- Jeder innere Knoten von T repräsentiert eine Eigenschaft, die eine Instanz $x \in \mathcal{X}$ entweder hat oder nicht hat.
- Jedes Blatt in T ist mit einem Label aus $\mathcal{Y}$ markiert.

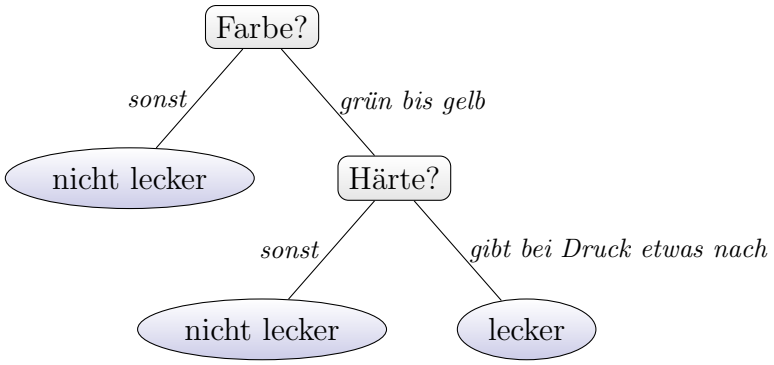


Abbildung 2: Entscheidungsbaum, der Papayas als „lecker“ bzw. „nicht lecker“ klassifiziert.

Um das Label $y = h_T(x)$ zu finden, wird x „top-down“ durch T geroutet (beginnend an der Wurzel). Befindet sich x an einem inneren Knoten v, welcher die Eigenschaft E_v repräsentiert, dann wird x zum rechten Kind von v weitergeleitet, falls x die Eigenschaft E_v besitzt, und zum linken Kind von v, falls x die Eigenschaft E_v nicht besitzt. Irgendwann landet x bei diesem Routing-Prozess in einem Blatt von T. Dort lesen wir das betreffende Label y ab und setzen $h_T(x) = y$. Abbildung 2, dem Lehrbuch von Shalev-Shwartz und Ben-David entnommen, illustriert diese Vorgehensweise mit einem Entscheidungsbaum, welcher Papayas entweder als „lecker“ oder „nicht lecker“ klassifiziert.

Wenn Abfragen mit mehr als zwei möglichen Ausgängen verwendet werden, führt das auf völlig analoge Weise zu Entscheidungsbäumen, die i.A. nicht binär sind: ein innerer Knoten, der eine Abfrage an x mit r möglichen Ausgängen repräsentiert, hat dann entsprechend r Kinder.

Abschnitt 13.1 ist den Booleschen Entscheidungsbäumen gewidmet. Die Informationskomplexität der Klasse der Booleschen Entscheidungsbäume lässt sich mit dem MDL-Ansatz[34] beherrschen (s. Abschnitt 13.1.1). Da das zugehörige Optimierungsproblem NP-hart ist, muss man sich beim Lernen von Booleschen Entscheidungsbäumen mit heuristischen Algorithmen (ohne mathematische Qualitätsgarantie) zufrieden geben. Wir besprechen hauptsächlich den ID3-Algorithmus von Quinlan (s. Abschnitt 13.1.2). In Abschnitt 13.2 besprechen wir einige Entscheidungsbaumvarianten, die über die einfachen Booleschen Entscheidungsbäume hinausgehen.

13.1 Boolesche Entscheidungsbäume

Definition 13.1 *Ein* Boolescher Entscheidungsbaum *über dem Grundbereich* $\{0,1\}^d$ *ist ein Binärbaum, bei welchem jeder innere Knoten mit einem Index aus* $[d]$ *und jedes Blatt mit einem Bit aus* $\{0,1\}$ *markiert ist.* $1-\mathrm{DT}_d$ *bezeichne die Menge aller Booleschen Entscheidungsbäume über* $\{0,1\}^d$.

Wir interpretieren das Label $i \in [d]$ eines inneren Knotens v als die Abfrage „$x_i = 1$?“. Auf

[34] MDL = Minimum Description Length

diese Weise repräsentiert jeder Baum $T \in 1{-}\mathrm{DT}_d$ eine Boolesche Funktion $h_T : \{0,1\}^d \to \{0,1\}$.

13.1.1 MDL-basierte Analyse der Informationskomplexität

Bemerkung 13.2 *Jede Boolesche Funktion $h : \{0,1\}^d \to \{0,1\}$ ist durch einen Baum aus $1{-}\mathrm{DT}_d$ repräsentierbar, so dass $\{0,1\}^d$ durch $1{-}\mathrm{DT}$ aufspaltbar ist und es gilt*

$$\mathrm{VCD}(1{-}\mathrm{DT}_d) = 2^d \ .$$

Beweis Der vollständige binäre Baum T_d der Tiefe d, bei welchem jeder innere Knoten einer Tiefe $i \in \{0,1,\ldots,d-1\}$ mit $i+1$ markiert ist, verteilt die 2^d Instanzen aus $\{0,1\}^d$ bijektiv auf seine 2^d Blätter. Da wir an den Blättern ein beliebiges Binärmuster aus $\{0,1\}^{2^d}$ anlegen können, kann jede Boolesche Funktion $h : \{0,1\}^d \to \{0,1\}$ mit Hilfe von T und einem entsprechenden Binärmuster an den 2^d Blättern dargestellt werden. •

Effiziente uniforme PAC-Lernbarkeit von $1{-}\mathrm{DT}$ ist damit ausgeschlossen. Wir weichen auf das Modell des nicht-uniformen Lernens aus. Die Klasse $1{-}\mathrm{DT} := \cup_{d\geq 1} 1{-}\mathrm{DT}_d$ ist zwar unendlich aber abzählbar. Aus dem Kapitel über Lernbarkeit im nichtuniformen Modell wissen wir folgendes. Wenn wir die Hypothesen h einer abzählbar unendlichen Klasse mit den Strings eines binären Präfixcodes darstellen und wenn $|h|$ die Länge des Codewortes für h bezeichnet, dann gilt (mit Wahrscheinlichkeit mindestens $1-\delta$ über $S \sim \mathcal{D}^m$) die folgende Fehlerschranke:

$$\forall h \in \mathcal{H} : L_{\mathcal{D}}(h) \leq L_S(h) + \sqrt{\frac{|h| + \ln(2/\delta)}{2m}} \ . \tag{82}$$

Folgende Rekursion liefert einen Präfixcode über dem Alphabet $\{0,1,(,)\}$ für die Binärbäume aus $1{-}\mathrm{DT}$:

1. Der nur aus einer mit $b \in \{0,1\}$ markierten Wurzel bestehende Baum wird kodiert mit dem String (b).
2. Wenn T einen Baum mit linkem Unterbaum T_0 und rechtem Unterbaum T_1 bezeichnet, dessen Wurzel mit i markiert ist, dann ergibt sich das Codewort von T aus der Rekursionsgleichung $K(T) = (\mathrm{bin}(i)K(T_0)K(T_1))$. Hierbei steht $\mathrm{bin}(\cdot)$ für die Binärdarstellung einer natürlichen Zahl.

Wir benötigen für jeden inneren Knoten $2 + \lceil \log(d) \rceil$ und für jedes Blatt 3 Zeichen. Durch Übergang zu einem binären Präfixcode verdoppelt sich die jeweilige Zeichenanzahl. Daher hat ein Baum aus $1{-}\mathrm{DT}_d$ mit n Knoten (und daher $(n-1)/2$ inneren Knoten und $(n+1)/2$ Blättern) die Kodierungslänge

$$\frac{n-1}{2}(4{+}2\lceil\log(d)\rceil){+}6\frac{n+1}{2} = 2(n{-}1){+}(n{-}1)\lceil\log(d)\rceil{+}3(n{+}1) = 5n{+}1{+}(n{-}1)\lceil\log(d)\rceil \ .$$

Wenn wir dies für $|h|$ in (82) einsetzen, erhalten wir für alle Wahlen von d, n und alle Bäume T aus $1-\mathrm{DT}_d$ mit n Knoten die Fehlerschranke

$$L_{\mathcal{D}}(T) \leq L_S(T) + \sqrt{\frac{5n + 1 + (n-1)\lceil \log(d) \rceil + \ln(2/\delta)}{2m}} \ . \tag{83}$$

Bei Anwendung der MDL-Lernregel zum Lernen von $1-$DT im nichtuniformen Modell würden wir zu gegebenem S einen Baum aus $1-$DT auswählen, der diese Fehlerschranke minimiert. Wie gewohnt dient der Kostenterm $L_S(T)$ der Vermeidung von „underfitting“; der Term

$$\sqrt{\frac{5n + 1 + (n-1)\lceil \log(d) \rceil + \ln(2/\delta)}{2m}}$$

bestraft unnötig komplexe Bäume und dient der Vermeidung von „overfittimg“.

13.1.2 Quinlan's ID3-Algorithmus

Das Problem, einen Baum aus $1-$DT zu bestimmen, der die obere Schranke für $L_{\mathcal{D}}(T)$ aus (83) minimiert, ist NP-hart. Es gibt aber eine Reihe von heuristischen Verfahren (ohne mathematische Qualitätsgarantie). Wir besprechen im Folgenden den ID3-Algorithmus von Quinlan.[35] Er benutzt eine Funktion $\mathrm{Gain}(S, i)$, die misst, welchen „Fortschritt“ wir machen, wenn wir die Trainingssequenz S gemäß der Abfrage „$x_i = 1$?“ zerlegen in $\{(\mathbf{x}, y) \in S |\ x_i = 0\}$ und $\{(\mathbf{x}, y) \in S |\ x_i = 1\}$. ID3 ist ein gieriger Algorithmus zur Konstruktion von T, der den aktuell betrachteten inneren Knoten immer so labelt, dass der betreffende Gain-Wert maximiert wird. Wenn am aktuell betrachteten Knoten v keine Aufteilung von S in zwei echte Untermengen möglich ist, weil alle in S befindlichen Instanzen das selbe Label besitzen, dann erklärt ID3 den Knoten v zu einem Blatt von T und markiert dieses mit dem betreffenden Label aus $\{0, 1\}$. Es folgt der Pseudocode für ID3 in Form einer rekursiven Prozedur:
Eingabe: Trainingssequenz S und $A \subseteq [d]$
Mehode:

Fall 1: Alle Instanzen in S besitzen das Label 0.
Dann gib ein mit 0 markiertes Blatt aus.

Fall 2: Alle Instanzen in S besitzen das Label 1.
Dann gib ein mit 1 markiertes Blatt aus.

Fall 3: S besitzt Instanzen mit unterschiedlichen Labels.

1. Setze $j := \mathrm{argmax}_{i \in A} \mathrm{Gain}(S, i)$.
2. Für $b = 0, 1$ setze $T_b := \mathrm{ID3}(\{(\mathbf{x}, y) \in S |\ x_j = b\}, A \setminus \{j\})$.
3. Gib den Baum aus, der aus der mit j markierten Wurzel, dem linken Unterbaum T_0 und dem rechten Unterbaum T_1 besteht.

[35] „ID3“ steht für „Iterative Dichotomizer 3“.

Im Hauptprogramm erfolgt der Aufruf ID3$(S, [d])$. Beachte, dass eine Menge S der Größe m höchstens $(m-1)$-mal echt aufgespalten werden kann. Die rekursive Prozedur ID3, die bei jedem Erreichen von Fall 3 einen inneren Knoten anlegt und eine Aufspaltung durchführt, wird also den Fall 3 höchstens $(m-1)$-mal erreichen. Daher wird die Prozedur Gain insgesamt höchstens $(m-1)d$-mal aufgerufen. Diese Aufrufe dominieren (asymptotisch gesehen) die gesamte Laufzeit.

Es sei $p(S)$ (bzw. $1-p(S)$) der Bruchteil, der mit 1 (bzw. mit 0) gelabelten Instanzen in S. Je näher $p(S)$ dem Wert $1/2$ kommt, desto „unreiner" ist S. Der Definition von Gain(S, i) liegt meist ein Kostenmaß $C(p(S))$ für die Unreinheit von S zugrunde. Die folgenden Funktionen in der Rolle von $C : (0,1) \to \mathbb{R}^+$ sind populär:

$$C_1(p) = \min\{p, 1-p\} \ , \ C_2(p) = -p\log(p) - (1-p)\log(1-p) \ \text{ und } C_3(p) = 2p(1-p) \ .$$

$C_2(p)$ wird auch „binäre Entropie" genannt. $C_2(p)$ und $C_3(p)$ sind konkave und differenzierbare obere Schranken von $C_1(p)$.

Die Funktion Gain(S, i) misst, wie stark die Unreinheit von S im Mittel reduziert wird, wenn wir die Aufteilung in $S_{i,0} := \{(\mathbf{x}, y) \in S | x_i = 0\}$ und $S_{i,1} := \{(\mathbf{x}, y) \in S | x_i = 1\}$ vornehmen:

$$\text{Gain}(S, i) := C(p(S)) - (p(S) \cdot C(p(S_{i,1})) + (1 - p(S)) \cdot C(p(S_{i,0}))) \ . \tag{84}$$

Die d Aufrufe Gain$(S, 1), \ldots,$ Gain(S, d) können in $O(m+d)$ Schritten implementiert werden: $O(m)$ Schritte zur Berechnung von $p(S)$ und $O(d)$ Schritte für die d Auswertungen gemäß (84). Da dieser Aufwand an jedem der maximal $m-1$ inneren Knoten von T betrieben werden muss, ergibt sich für die Laufzeit von ID3 die Schranke $O(m(m+d))$.

Pruning. Einen Baum einer mit $m = |S|$ linear wachsenden Größe für eine Trainingssequenz S anzulegen ist so etwas wie der Garantieschein für „overfitting". Es hat sich experimentell gezeigt, dass es besser ist, einen Baum T mit maximal $m-1$ inneren Knoten wieder auf ein kleineres Maß zurückzuschneiden (englisch: to prune) als ihn von vorneherein nur auf eine kleinere Größe anwachsen zu lassen. Die betreffende Pruning-Prozedur bearbeitet T „bottom-up" (in Richtung von den Blättern zur Wurzel). Bei jedem aktuell erreichten inneren Knoten v mit den Kindern v_0 und v_1 wird, der von v aufgespannte Unterbaum, notiert als $T(v)$, entweder unverändert gelassen oder ersetzt durch ein Blatt mit Label 0 oder 1, oder ersetzt durch einen der Bäume $T(v_0)$ und $T(v_1)$. Welche Option zum Zuge kommt hängt von einer Funktion $f(T, m)$ ab, die eine Schätzung bzw. eine obere Schranke für $L_{\mathcal{D}}(T)$ repräsentiert. Es folgt der Pseudocode für eine solche Pruning-Prozedur:

Eingabe: Baum T, Bewertungsfunktion $f(T, m)$

Methode: Durchlaufe T „bottom-up". Bei jedem inneren Knoten v von T mit Kindern v_0 und v_1 wähle unter den folgenden Modifikationen von T eine aus, die zur besten Bewertung $f(T, m)$ führt:

- Ersetzung von $T(v)$ durch ein mit 0 markiertes Blatt

- Ersetzung von $T(v)$ durch ein mit 1 markiertes Blatt
- Ersetzung von $T(v)$ durch $T(v_0)$
- Ersetzung von $T(v)$ durch $T(v_1)$
- Übernahme eines unveränderten Baumes T

Ausgabe: Nach Abschluss des „bottom-up“ Durchlaufes gib die aktuelle Version von T aus.

13.2 Weitere Entscheidungsbaumvarianten

13.2.1 Die Klasse k-DT

Wir bezeichnen mit $k{-}\mathrm{DT}_d$ die Klasse, die aus $1{-}\mathrm{DT}_d$ dadurch hervorgeht, dass wir an jedem inneren Knoten eine Abfrage der Form „$M(\mathbf{x}) = 1?$“ durchführen. Dabei ist M ein Boolescher Term bestehend aus maximal k Literalen, d.h., $M = \ell_1 \wedge \ldots \wedge \ell_{k'}$ mit $k' \in [k]$, und $\ell_1, \ldots, \ell_{k'} \in \{x_1, \bar{x}_1, \ldots, x_d, \bar{x}_d\}$.

Da eine Abfrage der Form „$\bar{x}_i = 1?$“ die selbe Information liefert wie die Abfrage „$x_i = 1?$“, ist die Definition von $k{-}\mathrm{DT}$ im Spezialfall $k = 1$ deckungsgleich zu der Definition 13.1 von $1{-}\mathrm{DT}$.

Eine polynomielle L-Reduktion von $k{-}\mathrm{DT}$ nach $1{-}\mathrm{DT}$ ist leicht zu konstruieren (mit neuen Variablen, die 1–zu–1 den Booleschen Termen der Maximallänge k entsprechen). Kombiniert man diese L-Reduktion mit ID3, erhält man einen (heuristischen) Lernalgorithmus für die Klasse $k{-}\mathrm{DT}$.

13.2.2 Binäre Abfragen zu reellwertigen Merkmalen

Wenn die Beispielinstanzen $\mathbf{x}$ aus $\mathbb{R}^d$ stammen, dann sind lineare Abfragen der Form „$\langle \mathbf{w}, \mathbf{x} \rangle \geq \theta?$“ gebräuchlich. Der Algorithmus von Quinlan kann an Anfragen dieser Art angepasst werden. In diesem Abschnitt diskutieren wir kurz spezielle lineare Abfragen von der Form „$x[i] \geq \theta?$“. Ein innerer Knoten kann dann einfach mit (i, θ) markiert werden. Es bezeichne $1{-}\mathrm{DT}_d^{\mathbb{R}}$ die Klasse aller solcher Entscheidungsbäume über dem Grundbereich $\mathbb{R}^d$. Wir wollen zeigen, dass sich $(1{-}\mathrm{DT}_d^{\mathbb{R}})_{d\geq 1}$ polynomiell auf $(1{-}\mathrm{DT}_d)_{d\geq 1}$ reduzieren lässt. Die Reduktion, die wir verwenden werden, ist im strengen Sinn keine L-Reduktion, weil wir die Instanzen- und die Hypothesentransformationen von der zugrunde liegenden Trainingssequenz abhängig machen. Die Reduktion erfüllt aber den gleichen Zweck wie eine „anständige“ L-Reduktion. Insbesondere kann sie mit dem ID3-Algorithmus zu einem Lernalgorithmus für $(1{-}\mathrm{DT}_d^{\mathbb{R}})_{d\geq 1}$ kombiniert werden.

Trainingssequenz: $S = [(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_m}, y_m)] \in (\mathbb{R}^d \times \{0,1\})^m$.

verwendete Schwellwerte: Für $i = 1, \ldots, m$ sei $\mathbf{x_1^{(i)}}, \ldots, \mathbf{x_m^{(i)}}$ eine Umordnung von $\mathbf{x_1}, \ldots, \mathbf{x_m}$, so dass $x_1^{(i)}[i] \leq \ldots \leq x_m^{(i)}[i]$. Wir unterstellen im Folgenden, dass die i-ten Koordinaten von $\mathbf{x_1}, \ldots, \mathbf{x_m}$ paarweise verschieden sind (aber unsere Ausführungen sind leicht auf den allgemeinen Fall erweiterbar). Für $i = 1, \ldots, m$ sei $\Theta_i := \{\theta_{i,j} |\ j = 0, \ldots, m\}$ mit

$$\theta_{i,0} < x_1^{(i)}[i] < \theta_{i,1} < \ldots < x_m^{(i)}[i] < \theta_{i,m} \ .$$

Kommentar: Zur Herstellung von ERM-Hypothesen auf S genügt es innere Knotenmarkierungen der Form (i, θ) mit $\theta \in \Theta_i$ zu verwenden.

Instanzentransformation: Ein Vektor $\mathbf{x} \in \mathbb{R}^d$ wird abgebildet auf einen Vektor $\mathbf{x}' \in \{0,1\}^{d'}$ für $d' = d(m+1)$. Für jedes Paar $(i,j) \in \{1,\ldots,d\} \times \{0,1,\ldots,m\}$ setze $x'[i,j] = \mathbb{1}_{[x[i] \geq \theta_{i,j}]}$.

1. Hypothesentransformation: Ersetze einen Baum T aus der Klasse $1{-}\mathrm{DT}_d^{\mathbb{R}}$ durch einen Baum der selben Form aber mit veränderten Markierungen: die Markierung $(i, \theta_{i,j})$, welche die Abfrage „$x[i] \geq \theta_{i,j}$?“ repräsentiert, wird ersetzt durch die Markierung (i,j), welche die Abfrage „$x'[i,j] = 1$?“ repräsentiert.

2. Hypothesentransformation: Wir benutzen einfach die umgekehrte Markierungssubstitution.

Beachte: wegen $x'[i,j] = \mathbb{1}_{[x[i] \geq \theta_{i,j}]}$ sind die Abfragen „$x[i] \geq \theta_{i,j}$?“ und „$x'[i,j] = 1$?“ äquivalent zueinander!

Da $\mathbf{x} \in \mathbb{R}^d$ auf $\mathbf{x}' \in \{0,1\}^{d(m+1)}$ abgebildet wird, muss bei Anwendung von ID3 auf eine transformierte Trainingssequenz S' die Abbildung Gain insgesamt $(m-1)(m+1)d$-mal ausgewertet werden. Eine cleverere Implementierung kommt jedoch mit $O(dm\log(m))$ Auswertungen von Gain aus.

13.2.3 Breiman's Konzept der zufälligen Wälder

Ein Entscheidungswald ist eine Kollektion $W = (T_1, \ldots, T_r)$ von Entscheidungsbäumen. W repräsentiert eine Voraussagefunktion h_W, welche ein Majoritätsvotum über $T_1, \ldots, T_r$ durchführt, d.h., $h_W(\mathbf{x}) = 1$ gilt genau dann, wenn mindestens $k/2$ der Werte $h_{T_1}(\mathbf{x}), \ldots, h_{T_r}(\mathbf{x})$ gleich 1 sind. Breiman's Grundidee besteht darin, „overfitting“ zu vermeiden, indem ein Baum T_i seine inneren Knoten nur mit Variablen einer Menge $A_i \subset [d]$ labeln darf. Dabei ist A_i eine zufällige Auswahl von $k << d$ Indizes aus der Grundmenge $[d]$ (und die Zufallsmengen $A_1, \ldots, A_r$ werden unabhängig voneinander ermittelt). Der Baum T_i könnte durch einen Aufruf $\mathrm{ID3}(S, A_i)$ bestimmt werden. Jeder einzelne Baum hat i.A. eine hohe Fehlerrate. Die Hoffnung ist, dass durch das Majoritätsvotum dennoch eine kleine Fehlerrate erzielt wird.

14 Die Nearest-Neighbor-Lernregel

In diesem Abschnitt setzen wir voraus, dass der Grundbereich $\mathcal{X}$ mit einer Metrik $\rho(\cdot,\cdot)$ versehen ist. Zum Beispiel könnte $\mathcal{X} = \mathbb{R}^d$ und $\rho(\mathbf{x},\mathbf{y}) = \|\mathbf{x}-\mathbf{y}\|_2$ gelten.

Es sei $S = [(x_1,y_1),\ldots,(x_m,y_m)] \in (\mathcal{X} \times \mathcal{Y})^m$ eine Trainingssequenz und $u \in \mathcal{X}$ eine Instanz (ohne vorgegebenes Label). Wir bezeichnen dann mit

$$x_1^u, \ldots, x_m^u$$

eine Umordnung der Instanzen $x_1, \ldots, x_m$, so dass

$$\rho(x_1^u, u) \leq \ldots \leq \rho(x_m^u, u) \ ,$$

wobei (zum Zwecke der Eindeutigkeit) bei gleicher Distanz zu u Vektoren mit kleinerem Index den Vorrang erhalten. $y_1^u, \ldots, y_m^u$ sei eine entsprechende Umordnung der Labels. Somit wäre $x_1^u \in S_\mathcal{X}$ der zu u nächstgelegene Nachbar, $x_2^u \in S_\mathcal{X}$ der zu u zweitnächstgelegene Nachbar usw.. Die k-Nearest-Neighbor-Lernregel macht den folgenden Vorschlag für das Label von u.

k-NN für binäre Klassifikationsprobleme: Weise u das Label 1 genau dann zu, wenn mindestens $k/2$ der Labels $y_1^u, \ldots, y_k^u$ identisch zu 1 sind (Majoritätsvotum).

k-NN für Regressionsprobleme: Weise u das Label $\frac{1}{k}\sum_{i=1}^k y_i^u$ zu.

Es bezeichne im Folgenden h_S die aus k-NN resultierende Hypothese. Offensichtlich gilt $h_S(u) = y_1^u$ im Falle $k = 1$, d.h., die Lernregel 1-NN weist einer Instanz u einfach das Label ihres „nearest neighbors“ in S zu.

Es sind natürlich auch Varianten von k-NN denkbar. Zum Beispiel könnte man im Falle von binären Klassifikationsproblemen ein gewichtetes Majoritätsvotum durchführen, bei dem y_i^u ein zu $\rho(x_i^u, u)$ umgekehrt proportionales Gewicht erhält. Eine entsprechende Bemerkung gilt für Regressionsprobleme.

14.1 Eine Fehlerschranke für 1-NN

Wir betrachten binäre Klassifikationsprobleme auf dem Grundbereich $\mathcal{X} = [0,1]^d$. Somit ist $\mathcal{Y} = \{0,1\}$ und ℓ ist die Null-Eins-Verlustfunktion. Es bezeichne $\|\cdot\| = \|\cdot\|_2$ die Euklidische Norm in $\mathbb{R}^d$. Wie üblich sei $\mathcal{D}$ eine Verteilung auf $\mathcal{X} \times \mathcal{Y}$. Weiter sei $\mathcal{D}_\mathcal{X}$ die von $\mathcal{D}$ induzierte Randverteilung auf $\mathcal{X}$ und $\eta(\mathbf{x}) = \Pr[y = 1|\mathbf{x}]$ bezeichne die bedingte Wahrscheinlichkeit für das 1-Label zu einer gegebenen Instanz $\mathbf{x}$. Eine Hypothese mit der kleinsten Fehlerrate, genannt *Bayes-optimale Hypothese*, ist dann gegeben durch

$$h^*(\mathbf{x}) = \mathbb{1}_{[\eta(\mathbf{x}) \geq 1/2]} \ .$$

Da $\mathcal{D}$ dem Lerner unbekannt ist, sind natürlich $\mathcal{D}_\mathcal{X}$, η und h^* dem Lerner ebenfalls unbekannt. Das Ziel ist dennoch, eine Hypothese zu lernen, deren Fehlerrate nicht dramatisch viel größer ist als die Fehlerrate von h^*. Wenn die räumliche Nähe zweier Instanzen $\mathbf{x}, \mathbf{x}' \in \mathbb{R}^d$ nichts über die Gleichartigkeit ihrer Labels aussagt, hat man mit der Lernregel 1-NN keine Chance. Wir setzen deshalb voraus, dass η für eine Konstante $c > 0$ c-Lipschitz ist, d.h.,

$$\forall \mathbf{x}, \mathbf{x}' \in \mathcal{X} : \ |\eta(\mathbf{x}) - \eta(\mathbf{x}')| \leq c \cdot \|\mathbf{x} - \mathbf{x}'\|$$

und beweisen das

Lemma 14.1 *Mit obigen Notationen und Voraussetzungen gilt für die aus*

$$S = [(\mathbf{x_1}, y_1), \ldots, (\mathbf{x_m}, y_m)] \sim \mathcal{D}^m$$

und Anwendung von 1-NN resultierende Hypothese h_S:

$$\mathbb{E}_{S\sim\mathcal{D}^m}[L_\mathcal{D}(h_S)] \leq 2 \cdot L_\mathcal{D}(h^*) + c \cdot \mathbb{E}_{S_\mathcal{X}\sim\mathcal{D}_\mathcal{X}^m, \mathbf{u}\sim\mathcal{D}_\mathcal{X}}[\|\mathbf{u} - \mathbf{x_1^u}\|] \ . \tag{85}$$

Beweis Das Zufallsexperiment

$$S \sim \mathcal{D}^m \text{ und } (\mathbf{u}, y) \sim \mathcal{D}$$

ist äquivalent zu

$$S_{\mathcal{X}} \sim \mathcal{D}_{\mathcal{X}}^m, \mathbf{u} \sim \mathcal{D}_{\mathcal{X}}, y_i \sim \eta(\mathbf{x_i}) \text{ für } i = 1, \ldots, m \text{ und } y \sim \eta(\mathbf{u}) \ ,$$

d.h., wir können zuerst zufällige unmarkierte Instanzen ermitteln und danach die zugehörigen Labels. Aus $S_{\mathcal{X}}$ und $\mathbf{u}$ lässt sich $\mathbf{x_1^u}$ ermitteln und freilich ist $y_1^{\mathbf{u}}$ gemäß $y_1^{\mathbf{u}} \sim \eta(\mathbf{x_1^u})$ verteilt. Daher gilt:

$$\begin{aligned}
\mathbb{E}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(h_S)] &= \mathbb{E}_{S\sim\mathcal{D}^m,(\mathbf{u},y)\sim\mathcal{D}}[\mathbb{1}_{[y_1^{\mathbf{u}}\neq y]}] \\
&= \mathbb{E}_{S_{\mathcal{X}}\sim\mathcal{D}_{\mathcal{X}}^m,\mathbf{u}\sim\mathcal{D}_{\mathcal{X}}}\mathbb{E}_{y_1^{\mathbf{u}}\sim\eta(\mathbf{x_1^u}),y\sim\eta(\mathbf{u})}[\mathbb{1}_{[y_1^{\mathbf{u}}\neq y]}] \\
&= \mathbb{E}_{S_{\mathcal{X}}\sim\mathcal{D}_{\mathcal{X}}^m,\mathbf{u}\sim\mathcal{D}_{\mathcal{X}}}\left[\Pr_{y_1^{\mathbf{u}}\sim\eta(\mathbf{x_1^u}),y\sim\eta(\mathbf{u})}[y_1^{\mathbf{u}} \neq y]\right]
\end{aligned}$$

Der Term $\Pr_{y_1^{\mathbf{u}}\sim\eta(\mathbf{x_1^u}),y\sim\eta(\mathbf{u})}[y_1^{\mathbf{u}} \neq y]$ lässt sich nach oben abschätzen wie folgt:

$$\begin{aligned}
\Pr_{y_1^{\mathbf{u}}\sim\eta(\mathbf{x_1^u}),y\sim\eta(\mathbf{u})}[y_1^{\mathbf{u}} \neq y] &= \eta(\mathbf{x_1^u})(1-\eta(\mathbf{u})) + (1-\eta(\mathbf{x_1^u}))\eta(\mathbf{u}) \\
&= \eta(\mathbf{x_1^u}) + \eta(\mathbf{u}) - 2\eta(\mathbf{x_1^u})\eta(\mathbf{u}) \\
&= 2\eta(\mathbf{u})(1-\eta(\mathbf{u})) + (2\eta(\mathbf{u})-1)(\eta(\mathbf{u}) - \eta(\mathbf{x_1^u})) \\
&\leq 2 \cdot \Pr_{y\sim\eta(\mathbf{u})}[h^*(\mathbf{u}) \neq y] + c \cdot \|\mathbf{u} - \mathbf{x_1^u}\|
\end{aligned}$$

Die letzte Ungleichung gilt, da η Werte in $[0,1]$ annimmt und c-Lipschitz ist, und weil daher folgendes gilt:

- $\eta(\mathbf{u})(1-\eta(\mathbf{u})) \leq \min\{\eta(\mathbf{u}), 1-\eta(\mathbf{u})\} = \Pr_{y\sim\eta(\mathbf{u})}[h^*(\mathbf{u}) \neq y]$.
- $|2\eta(\mathbf{u}) - 1| \leq 1$ und $|\eta(\mathbf{u}) - \eta(\mathbf{x_1^u})| \leq c \cdot \|\mathbf{u} - \mathbf{x_1^u}\|$.

Wenn wir beide Seiten der Ungleichung

$$\Pr_{y_1^{\mathbf{u}}\sim\eta(\mathbf{x_1^u}),y\sim\eta(\mathbf{u})}[y_1^{\mathbf{u}} \neq y] \leq 2 \cdot \Pr_{y\sim\eta(\mathbf{u})}[h^*(\mathbf{u}) \neq y] + c \cdot \|\mathbf{u} - \mathbf{x_1^u}\|$$

über $S_{\mathcal{X}} \sim \mathcal{D}_{\mathcal{X}}^m$ und $\mathbf{u} \sim \mathcal{D}_{\mathcal{X}}$ mitteln, erhalten wir (85). •

Wir wollen den Term $\mathbb{E}_{S_{\mathcal{X}}\sim\mathcal{D}_{\mathcal{X}}^m,\mathbf{u}\sim\mathcal{D}_{\mathcal{X}}}[\|\mathbf{u}-\mathbf{x_1^u}\|]$ nach oben abschätzen, benötigen dazu aber erst noch folgendes Hilfsresultat:

Lemma 14.2 *Es seien $C_1, \ldots, C_r$ (messbare) Teilmengen von $\mathcal{X}$ mit den Wahrscheinlichkeitsmassen $w_i := \mathcal{D}_{\mathcal{X}}(C_i)$ für $i = 1, \ldots, r$. Zu $S_{\mathcal{X}} \sim \mathcal{D}_{\mathcal{X}}^m$ assoziieren wir die Zufallsvariable*

$$W(S_{\mathcal{X}}) := \sum_{i:C_i\cap S_{\mathcal{X}}=\emptyset} w_i = \sum_{i=1}^{r} w_i \cdot \mathbb{1}_{[C_i\cap S_{\mathcal{X}}=\emptyset]} \ ,$$

die die Wahrscheinlichkeitsmassen der nicht von S getroffenen Teilmengen aufsummiert. Dann gilt

$$\mathbb{E}_{S_{\mathcal{X}}\sim\mathcal{D}_{\mathcal{X}}^m}[W(S_{\mathcal{X}})] \leq \frac{r}{em} \ .$$

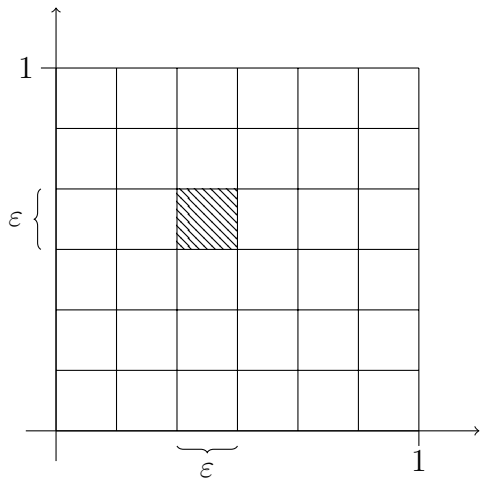


Abbildung 3: Zerlegung des Einheitsquadrats in Unterwürfel mit Kantenlänge ε.

Beweis Der Beweis ergibt sich aus folgender Rechnung:

$$\begin{aligned}
\mathbb{E}_{S_{\mathcal{X}} \sim \mathcal{D}_{\mathcal{X}}^m}[W(S_{\mathcal{X}})] &= \mathbb{E}\left[\sum_{i=1}^{r} w_i \cdot \mathbb{1}_{[C_i \cap S_{\mathcal{X}} = \emptyset]}\right] \\
&= \sum_{i=1}^{r} w_i \cdot \mathbb{E}[\mathbb{1}_{[C_i \cap S_{\mathcal{X}} = \emptyset]}] \\
&= \sum_{i=1}^{r} w_i (1 - w_i)^m \\
&\leq \sum_{i=1}^{r} w_i \cdot e^{-m w_i} \\
&\leq r \cdot \max_{a \in [0,1]} a \cdot e^{-ma} \leq \frac{r}{em}
\end{aligned}$$

Die letzte Ungleichung gilt, da $a = \frac{1}{m}$ eine Maximalstelle von $a \cdot e^{-ma}$ ist. •

Wir kommen jetzt auf unser eigentliches Ziel zurück: Herleitung einer oberen Schranke für den Term $\mathbb{E}_{S_{\mathcal{X}} \sim \mathcal{D}_{\mathcal{X}}^m, \mathbf{u} \sim \mathcal{D}_{\mathcal{X}}}[\|\mathbf{u} - \mathbf{x}_1^{\mathbf{u}}\|]$.

Lemma 14.3 *Es sei $\mathcal{X} = [0,1]^d$. Zu gegebenem $S_{\mathcal{X}} = (\mathbf{x_1}, \ldots, \mathbf{x_m}) \in \mathcal{X}^m$ und $\mathbf{u} \in \mathcal{X}$ bezeichne $\mathbf{x}_1^{\mathbf{u}}$ einen Punkt aus $S_{\mathcal{X}}$ mit der kleinsten Distanz zu $\mathbf{u}$. Mit diesen Notationen gilt:*

$$\mathbb{E}_{S_{\mathcal{X}} \sim \mathcal{D}_{\mathcal{X}}^m, \mathbf{u} \sim \mathcal{D}_{\mathcal{X}}}[\|\mathbf{u} - \mathbf{x}_1^{\mathbf{u}}\|] \leq 2\sqrt{d}(em)^{-\frac{1}{d+1}} \quad . \tag{86}$$

Beweis Es sei $0 < \varepsilon < 1$ ein Parameter, den wir an einer späteren Stelle des Beweises geschickt wählen. Wir zerlegen den Einheitswürfel $\mathcal{X} = [0,1]^d$ auf die offensichtliche Weise in $r = (1/\varepsilon)^d$ Unterwürfel der Kantenlänge ε. Für $d = 2$ ist dies in Abbildung 3 illustriert. Für $S_{\mathcal{X}} \sim \mathcal{D}_{\mathcal{X}}^m$ und $\mathbf{u} \sim \mathcal{D}_{\mathcal{X}}$ unterscheiden wir zwei Fälle:

Fall 1: $\mathbf{u}$ liegt in einem von $S_{\mathcal{X}}$ getroffenen Unterwürfel.

Fall 2: $\mathbf{u}$ liegt in einem von $S_{\mathcal{X}}$ nicht getroffenen Unterwürfel.

Es sei $W(S)$ die Zufallsvariable aus Lemma 14.2. Offensichtlich tritt Fall 2 (bzw. Fall 1) mit Wahrscheinlichkeit $W(S_{\mathcal{X}})$ (bzw. $1 - W(S_{\mathcal{X}})$) auf. Betrachte die Zufallsvariable $X := \|\mathbf{u} - \mathbf{x_1^u}\|$. Im Fall 1 gilt $X \leq \varepsilon\sqrt{d}$, da $\varepsilon\sqrt{d}$ der Durchmesser eines d-dimensionalen Würfels mit Kantenlänge ε ist. Im Fall 2 gilt die triviale Schranke $X \leq \sqrt{d}$, da $\sqrt{d}$ der Durchmesser von $[0,1]^d$ ist. Weiter geht es mit folgender Rechnung:

$$\begin{aligned}
\mathbb{E}_{S_{\mathcal{X}} \sim \mathcal{D}_{\mathcal{X}}^m, \mathbf{u} \sim \mathcal{D}_{\mathcal{X}}}[X] &= \mathbb{E}_{S_{\mathcal{X}} \sim \mathcal{D}_{\mathcal{X}}^m}\left[\mathbb{E}_{\mathbf{u} \sim \mathcal{D}_{\mathcal{X}}}[X|\text{Fall 1}] \cdot (1 - W(S)) + \mathbb{E}_{\mathbf{u} \sim \mathcal{D}_{\mathcal{X}}}[X|\text{Fall 2}] \cdot W(S)\right] \\
&\leq \varepsilon\sqrt{d} + \sqrt{d} \cdot \mathbb{E}_{S_{\mathcal{X}} \sim \mathcal{D}_{\mathcal{X}}^m} W(S_{\mathcal{X}}) \\
&\leq \varepsilon\sqrt{d} + \sqrt{d} \cdot \frac{\varepsilon^{-d}}{em} = \sqrt{d} \cdot \left(\varepsilon + \frac{\varepsilon^{-d}}{em}\right)
\end{aligned}$$

Bei der letzten Ungleichung wurde Lemma 14.2 verwendet. Wir können die Bedingung $\varepsilon = \frac{\varepsilon^{-d}}{em}$ erzwingen, indem wir $\varepsilon = (em)^{-\frac{1}{d+1}}$ setzen. Auf diese Weise erhalten wir schließlich (86). •

Wir merken kurz an, dass der Beweis eine kleine Schwierigkeit unter den Teppich kehrt: mit unserer Wahl von ε ist $1/\varepsilon$ keine ganze Zahl, d.h., die Unterteilung einer Kante der Länge 1 in Teilkanten der Länge ε geht nicht exakt auf. Einen exakten Beweis erhält man, wenn man $\varepsilon = 1/T$ für eine geeignete Wahl einer ganzen Zahl $T \geq 1$ setzt. Dann kann man allerdings die Bedingung $\varepsilon = \frac{\varepsilon^{-d}}{em}$ nur noch approximativ erreichen. Ein „sauberer“ Beweis würde das marginal schlechtere Resultat

$$\mathbb{E}_{S_{\mathcal{X}} \sim \mathcal{D}_{\mathcal{X}}^m, \mathbf{u} \sim \mathcal{D}_{\mathcal{X}}}[\|\mathbf{u} - \mathbf{x_1^u}\|] \leq 2\sqrt{d} \frac{1}{\left\lfloor (em)^{\frac{1}{d+1}} \right\rfloor}$$

liefern. Eine entsprechende Bemerkung gilt auch für das folgende Resultat, welches sich direkt aus den Lemmas 14.1 und 14.3 ergibt:

Theorem 14.4 *Mit den Notationen und Voraussetzungen wie in Lemma 14.1 gilt:*

$$\mathbb{E}_{S \sim \mathcal{D}^m}[L_{\mathcal{D}}(h_S)] \leq 2 \cdot L_{\mathcal{D}}(h^*) + 2c\sqrt{d}(em)^{-\frac{1}{d+1}} \quad . \tag{87}$$

14.2 Fluch der Hochdimensionalität (Curse of Dimensionality)

Um zu erreichen, dass der Term $2c\sqrt{d}(em)^{-\frac{1}{d+1}}$ in (87) einen Beitrag von maximal ε leistet, müssten wir

$$m \geq \frac{1}{e} \cdot \left(\frac{2c\sqrt{d}}{\varepsilon}\right)^{d+1}$$

setzen. Am meisten schmerzt es, dass die erforderliche Größe m der Trainingssequenz exponentiell mit d wächst, ein Umstand der auch als „Curse of Dimensionality“ bezeichnet wird. Man kann sich natürlich fragen, ob eine verbesserte Analyse oder die Verwendung einer anderen Lernregel nicht vielleicht eine freundlichere Abhängigkeit von d zu Tage fördern könnte. Dies ist i.A. leider nicht der Fall:

Theorem 14.5 *Zu jeder Wahl von $c, d \geq 1$ und jeder Lernregel $S \mapsto h_S$ existiert eine Verteilung $\mathcal{D}$ auf $[0,1]^d \times \{0,1\}$, so dass folgendes gilt: die von $\mathcal{D}$ induzierte Funktion $\eta : \mathcal{X} \to [0,1]$ ist c-Lipschitz und $L_{\mathcal{D}}(h^*) = 0$, aber für alle $m \leq (c+1)^d/2$ gilt $\mathbb{E}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(h_S)] \geq 1/4$.*

Beweis Es sei G_c^d das Gitter in $[0,1]^d$ der Maschenweite $1/c$, d.h.,

$$G_c^d = \left\{ \left(\frac{a_1}{c}, \ldots, \frac{a_d}{c} \right) \middle| \, a_1, \ldots, a_d \in \{0, 1, \ldots, c\} \right\} \ .$$

Das Gitter enthält $|G_c^d| = (c+1)^d$ Gitterpunkte. Da zwei verschiedene Punkte in G_c^d mindestens Abstand $1/c$ voneinander haben, ist jede Funktion von G_c^d nach $[0,1]$ automatisch c-Lipschitz. Zu jeder Funktion $f : G_c^d \to \{0,1\}$ assoziieren wir die Verteilung $\mathcal{D}^f$ mit:

1. $\mathcal{D}_{\mathcal{X}}^f$ ist uniform auf G_c^d.
2. Die von $\mathcal{D}^f$ induzierte Funktion η ist identisch zu f, d.h., jedem $\mathbf{x} \in G_c^d$ wird (mit Wahrscheinlichkeit 1) das Label $f(\mathbf{x})$ zugewiesen. In diesem Fall ist auch die Bayes-optimale Hypothese h^* identisch zu f und es gilt $L_{\mathcal{D}}(h^*) = 0$.

Wenn wir f uniform zufällig aus der Menge aller Funktionen von G_c^d nach $\{0,1\}$ wählen und $m \leq (c+1)^d/2$, dann sind die Labels auf den mindestens $(c+1)^d/2$ nicht in $S_{\mathcal{X}}$ enthaltenen Punkten aus G_c^d aus der Perspektive des Lerners perfekte Zufallsbits. Daher beträgt die über $\{\mathcal{D}^f | \ f : G_c^d \to \{0,1\}\}$ und S gemittelte Fehlerrate mindestens $1/4$. Nach dem Schubfachprinzip muss ein $f^* \in \{f | \ f : G_c^d \to \{0,1\}\}$ existieren mit:

$$\mathcal{D} = \mathcal{D}^{f^*} \Rightarrow \mathbb{E}_{S\sim\mathcal{D}^m}[L_{\mathcal{D}}(h_S)] \geq \frac{1}{4} \ ,$$

womit der Beweis des Theorems erbracht wäre. •

Effizienzbetrachtungen: Eine naive Methode zur Ermittlung des „Nearest Neighbor“ würde m Datenpunkte im $\mathbb{R}^d$ durchmustern und daher Laufzeit $O(md)$ benötigen. Durch Einsatz raffinierter Datenstrukturen zur Abspeicherung der m Datenpunkte kann man die Laufzeit auf $O(\text{poly}(d)\log(m))$ verkleinern. Allerdings belegen solche Datenstrukturen Speicherplatz in der Größenordnung $m^{O(d)}$. Es gibt daher Versuche, das „Nearest Neighbor Problem“ approximativ, dafür aber effizienter, zu lösen. Im Rahmen dieser Vorlesung können wir auf Details dazu nicht näher eingehen.

15 Neuronale Netzwerke

Künstliche neuronale Netzwerke sind Berechnungsmodelle, deren Design auf einer groben Modellierung natürlicher neuronaler Netzwerke (wie wir sie zum Beispiel im Gehirn vorfinden) basiert. Die Knoten dieser Netzwerke repräsentieren die Nervenzellen, welche auch Neurone genannt werden. Die Kanten repräsentieren die Vernetzungsstruktur. Mathematisch gesehen bilden die Neurone und ihre Verbindungen einen gerichteten Graphen. Ist dieser Graph azyklisch, so spricht man von einem *vorwärtsgerichteten neuronalen Netzwerk*, das auch „Feedforward Neural Network“ oder kurz „FFNN“ genannt wird.

Die Kanten e in einem neuronalen Netzwerk sind mit einem reellen Parameter $w(e)$ gewichtet. Der Absolutbetrag $|w(e)|$ steht für die Stärke der Reizübertragung; mit dem Vorzeichen unterscheiden wir exzitatorische (anregende) von inhibitorischer (hemmender) Reizübertragung. Die Gewichtsparameter sind programmierbar. Durch die Festlegung der Gewichte kann eine Anpassung des Netzwerkes an Trainingsdaten erreicht werden.

In diesem Kapitel werden wir uns auf die Diskussion vorwärtsgerichteter neuronaler Netzwerke konzentrieren. In Abschnitt 15.1 führen wir ein FFNN-Standardmodell ein, welches in Abschnitt 15.2 geringfügig erweitert wird. Abschnitt 15.3 beschäftigt sich mit der Rechenkraft von FFNN. Die Informationskomplexität einer FFNN-Architektur und der durch sie repräsentierten Hypothesenklasse wird in Abschnitt 15.4 analysiert. Im abschließenden Abschnitt 15.5 besprechen wir die Heuristik namens „Backpropagation“, die häufig zur Adjustierung der Gewichtsparameter verwendet wird.

15.1 Das FFNN-Standardmodell

Ein *vorwärtsgerichtetes neuronales Netzwerk $\mathcal{N}$ (auch „Feedforward Neural Network“ oder „FFNN“ genannt)* wird durch folgende Komponenten spezifiziert:

- ein gerichteter azyklischer Graph (V, E) mit Kantengewichten $w : E \to \mathbb{R}$, dessen Knoten als *Neurone* bezeichnet werden
- eine Funktion $\sigma : \mathbb{R} \to \mathbb{R}$, genannt *Aktivierungsfunktion*

Diese Spezifikation wird in der Gleichung $\mathcal{N} = (V, E, \sigma, w)$ zusammengefasst. Das Tripel (V, E, σ) heißt die *Architektur* des Netzwerkes. Die Gewichte $w(e)$ mit $e \in E$ sind programmierbare Parameter, deren Belegung mit reellen Zahlen durch einen Lernalgorithmus fixiert werden kann. Durch die konkrete Wahl der Gewichte passt sich das Netzwerk an vorliegende Trainingsdaten an.

Wir setzen im Folgenden voraus, dass das Netzwerk $\mathcal{N}$ *geschichtet (layered)* ist, d.h., die Knotenmenge V besitzt eine Zerlegung der Form $V = V_0 \cup V_1 \cup \ldots \cup V_T$, und die Kanten in E verlaufen stets von einer Schicht zu der darauf folgenden, d.h., $E \subseteq \cup_{t=1}^{T}(V_{t-1} \times V_t)$. Wir nennen V_0 die *Eingabeschicht (input layer)*, $V_1, \ldots, V_{T-1}$ die *verborgenen Schichten (hidden layers)* und V_T die *Ausgabeschicht (output layer)*. T heißt die *Tiefe* und $\max_{1 \leq t \leq T} |V_t|$ heißt die *Weite* von $\mathcal{N}$.

Es sei $n = |V_0| - 1$ und $d = |V_T|$. Dann repräsentiert $\mathcal{N} = (V, E, \sigma, w)$ eine Funktion $h_{\mathcal{N}} : \mathbb{R}^n \to \mathbb{R}^d$. Es folgt eine Beschreibung, wie die Berechnung auf einer Netzeingabe $\mathbf{x} \in \mathbb{R}^n$ die Schichten $0, 1, \ldots, T$ der Reihe nach durchläuft, wobei $o_v(\mathbf{x})$ den Wert bezeichnet, den ein Knoten $v \in V$ berechnet:

1. Die Eingabeschicht V_0 reicht im Wesentlichen $\mathbf{x}$ an die erste Schicht V_1 weiter. Genauer: für $i = 1, \ldots, n$ produziert der i-te Knoten in V_0 die Ausgabe x_i (= die i-te Komponente der Netzeingabe $\mathbf{x}$) und der $(n+1)$-te Knoten in V_0 produziert die Ausgabe 1.

2. Nehmen wir induktiv an, dass die Neurone der t-ten Schicht ihre Ausgabewerte schon berechnet haben und betrachten nun ein Neuron v in V_{t+1}. Es seien $u_1, \ldots, u_r$ alle Neurone in V_t, die durch eine Kante mit v verbunden sind. Weiter seien $w_1, \ldots, w_r$ die diesen Kanten zugeordneten Gewichte. Wir bezeichnen dann

$$a_v(\mathbf{x}) := \sum_{i=1}^{r} w_i o_{u_i}(\mathbf{x}) \tag{88}$$

   als die *Eingabe für den Knoten* v und dieser berechnet

$$o_v(\mathbf{x}) := \sigma(a_v(\mathbf{x})) = \sigma\left(\sum_{i=1}^{r} w_i o_{u_i}(\mathbf{x})\right) \ .$$

Seien $v_1, \ldots, v_d$ die d Knoten der Ausgabeschicht V_T. Dann setzen wir

$$h_{\mathcal{N}}(\mathbf{x}) := (o_{v_1}(\mathbf{x}), \ldots, o_{v_d}(\mathbf{x})) \in \mathbb{R}^d \ ,$$

d.h., die von $v_1, \ldots, v_d \in V_T$ berechneten Werte bilden die Ausgabe des Netzwerkes.

Da $\mathcal{N} = (V, E, \sigma, w)$ mit $n + 1 = |V_0|$ und $d = |V_T|$ eine Funktion $h_{\mathcal{N}} : \mathbb{R}^n \to \mathbb{R}^d$ repräsentiert, repräsentiert die Architektur (V, E, σ) die Hypothesenklasse

$$\mathcal{H}_{V,E,\sigma} := \{h_{V,E,\sigma,w} |\ w : E \to \mathbb{R}\} \subseteq \{f : \mathbb{R}^n \to \mathbb{R}^d\} \ .$$

Das Adjustieren der Gewichtsparameter w an vorliegende Trainingsdaten entspricht also der Wahl einer Hypothese aus der Klasse $\mathcal{H}_{V,E,\sigma}$.

15.2 Erweiterung des Standardmodells

Wir erweitern $\mathcal{N}$ um eine Komponente $\theta : V \to \mathbb{R}$, so dass $\mathcal{N} = (V, E, \sigma, w, \theta)$. Die Eingabe $a_v(\mathbf{x})$ für einen Knoten v bei Netzeingabe $\mathbf{x}$ — vergleiche mit (88) — hat dann die Form

$$a_v(\mathbf{x}) := \sum_{i=1}^{r} w_i o_{u_i}(\mathbf{x}) + \theta(v) \ ,$$

d.h., θ assoziiert zu jedem Knoten ein Bias. Ansonsten verläuft die Berechnung der Funktion $h_{\mathcal{N}}$ genau so wie wir es in Abschnitt 15.1 beschrieben haben. Wir betrachten die Biaswerte θ ebenso wie die Gewichte w als programmierbare Parameter. Somit repräsentiert eine Architektur (V, E, σ) die Hypothesenklasse

$$\mathcal{H}_{V,E,\sigma} := \{h_{V,E,\sigma,w,\theta} |\ w : E \to \mathbb{R}, \theta : V \to \mathbb{R}\} \subseteq \{f : \mathbb{R}^n \to \mathbb{R}^d\}\ .$$

Es ist nicht schwer zu zeigen, dass jedes neuronale Netzwerk in dem erweiterten Modell simulierbar ist durch ein (nicht wesentlich größeres) neuronales Netzwerk im Standardmodell. Hierzu muss lediglich die 1-Konstante (welche in der Eingabeschicht zur Verfügung gestellt wird) mit Hilfe eines Extra-Neurons pro versteckter Schicht weiterpropagiert werden, so dass ein Bias $\theta(v)$ mit Hilfe eines zusätzlichen Gewichtsparameters einer in v mündenden Kante dargestellt werden kann. Die Details sind leicht einzufüllen. Im Folgenden bedienen wir uns meistens des erweiterten Modells.

15.3 Die Rechenkraft von FFNN

Für eine Boolesche Variable v_i identifizieren wir v_i^1 mit v_i und v_i^0 mit ihrem Negat $\overline{v_i}$. Es sei $\mathbf{b} = (b_1, \ldots, b_n) \in \{0,1\}$ ein Boolescher Vektor. Dann berechnet der Term $T = v_1^{b_1} \wedge \ldots \wedge v_n^{b_n}$, angewendet auf einen weiteren Booleschen Vektor $\mathbf{a} = (a_1, \ldots, a_n)$ den Wert $T(\mathbf{a}) = 1$ genau dann, wenn $\mathbf{a} = \mathbf{b}$. Es sei $f : \{0,1\}^n \to \{0,1\}$ eine beliebige Boolesche Funktion in n Variablen. Weiter sei $B(f) = f^{-1}(1)$ die Menge der Booleschen Vektoren, die von f auf 1 abgebildet werden. Offensichtlich kann f durch die folgende DNF-Formel dargestellt werden:

$$f(\mathbf{v}) = \bigvee_{\mathbf{b} \in B(f)} (v_1^{b_1} \wedge \ldots \wedge v_n^{b_n})\ .$$

Es ergibt sich folgendes Resultat:

Lemma 15.1 *Jede Boolesche Funktion kann durch eine DNF-Formel dargestellt werden.*

Es bezeichne $\oplus$ die Addition modulo 2 und es sei $P(v_1, \ldots, v_n) = v_1 \oplus \ldots \oplus v_n$ die Funktion, die den Wert 1 genau dann liefert, wenn sich in $v_1, \ldots, v_n$ ungeradzahlig viele Einsen befinden (die sogenannte Paritätsfunktion). Man kann leicht zeigen, dass jede DNF-Formel, welche P darstellt, 2^{n-1} Konjunktionen verwenden muss. Hieraus folgt, dass wir i.A. DNF-Formeln exponentieller Größe benötigen, um alle Booleschen Funktionen in n Variablen darzustellen. Es seien $v, u_1, \ldots, u_r \in \{v_1, \ldots, v_n\}$ Boolesche Variable. Wir setzen

$$L_b(v) := \begin{cases} 1 - v & \text{falls } b = 0 \\ v & \text{falls } b = 1 \end{cases}\ .$$

Da die Boolesche Formel v^b den selben Wert (0 oder 1) liefert wie die arithmetische Formel $L_b(v)$, wird $L_b(v)$ auch als „Arithmetisierung“ des Literals v^b bezeichnet. Eine Konjunktion

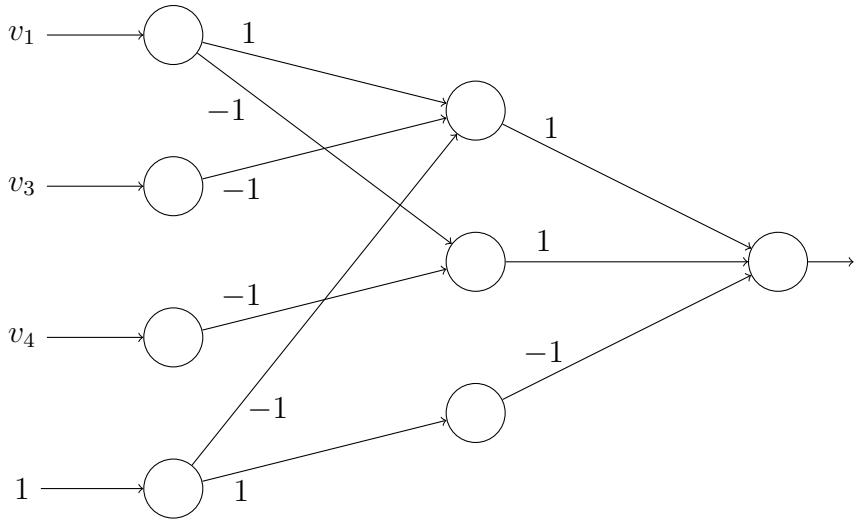


Abbildung 4: Netzwerk für die DNF-Formel $(v_1 \wedge \overline{v_3}) \vee (\overline{v_1} \wedge \overline{v_4})$.

oder Disjunktion von Literalen kann von einem Neuron mit sign$(\cdot)$ als Aktivierungsfunktion nach folgendem Schema simuliert werden:

$$\begin{aligned} u_1^{b_1} \wedge \ldots \wedge u_r^{b_r} = 1 &\Leftrightarrow L_{b_1}(u_1) + \ldots + L_{b_n}(u_r) \geq r \Leftrightarrow L_{b_1}(u_1) + \ldots + L_{b_n}(u_r) - r \geq 0 \\ u_1^{b_1} \vee \ldots \vee u_r^{b_r} = 1 &\Leftrightarrow L_{b_1}(u_1) + \ldots + L_{b_r}(u_r) \geq 1 \Leftrightarrow L_{b_1}(u_1) + \ldots + L_{b_r}(u_r) - 1 \geq 0 \end{aligned}$$

Insbesondere gilt

$$u_1 \vee \ldots \vee u_r = 1 \Leftrightarrow u_1 + \ldots + u_r \geq 1 \Leftrightarrow u_1 + \ldots + u_r - 1 \geq 0 \ .$$

Wir können daher aus Lemma 15.1 direkt folgendes Resultat ableiten:

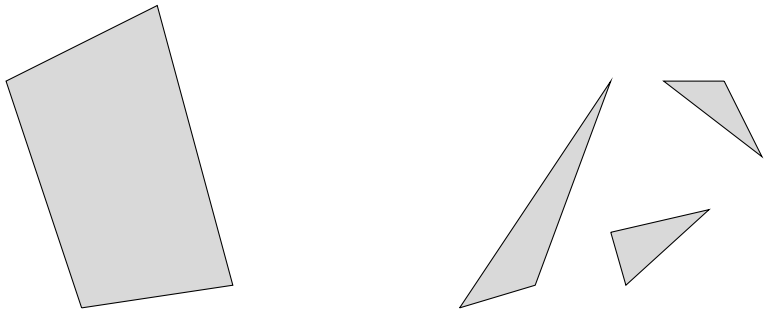
Lemma 15.2 *Jede DNF-Formel (und damit jede Boolesche Funktion) kann von einem neuronalen Netzwerk der Tiefe* 2 *berechnet werden.*

Beispiel 15.3 *Wir bauen ein Netzwerk für die DNF-Formel* $(v_1 \wedge \overline{v_3}) \vee (\overline{v_1} \wedge \overline{v_4})$*. Dazu nutzen wir aus:*

$$\begin{aligned} v_1 \wedge \overline{v_3} = 1 &\Leftrightarrow v_1 + (1 - v_3) \geq 2 \Leftrightarrow v_1 - v_3 - 1 \geq 0 \\ \overline{v_1} \wedge \overline{v_4} = 1 &\Leftrightarrow (1 - v_1) + (1 - v_4) \geq 2 \Leftrightarrow -v_1 - v_4 \geq 0 \end{aligned}$$

Dies führt zu dem in Abbildung 4 dargestellten Netzwerk.

Wir werden später zeigen, dass die VC-Dimension von $\mathcal{H}_{V,E,\text{sign}}$ durch $O(|E| \ln(|E|)) = O(|V|^3)$ nach oben beschränkt ist. Da 2^n die VC-Dimension von $\{f : \{0,1\}^n \to \{0,1\}\}$ ist, muss es Boolesche Funktionen geben, die nur von einer Architektur mit $\Omega(2^{n/3})$ Neuronen berechnet werden können. Man benötigt also i.A. Netzwerke einer exponentiellen Größe. Ähnliche Resultate gelten, wenn σ die Sigmoid-Funktion ist.



a) b)

Abbildung 5: a) Konvexer Polyeder als Durchschnitt von vier affinen Halbebenen. b) Vereinigung von drei konvexen Polyedern

Neuronale Netzwerkfamilien polynomiell beschränkter Größe. *P/poly* bezeichnet die Klasse aller Familien $(f_n)_{n\geq 1}$ Boolescher Funktionen, die von einer Schaltkreisfamilie $(C_n)_{n\geq 1}$ polynomiell beschränkter Größe über der Standardbasis $\{\neg, \wedge, \vee\}$ berechnet werden können. Da wir solche Schaltkreise (wie weiter oben bereits ausgeführt) mit Hilfe neuronaler Netzwerke effizient simulieren können, kann jedes $(f_n)_{n\geq 1} \in P/poly$ auch durch eine Familie $(\mathcal{N}_n)_{n\geq 1}$ neuronaler Netzwerke einer polynomiell beschränkten Größe berechnet werden. Die Klasse *P/poly* ist recht groß. Sie umfasst zum Beispiel die Klasse P der in Polynomialzeit Turing-berechenbaren Booleschen Funktionsfamilien.

FFNN und Vereinigung von konvexen Polyedern. Ein Neuron v in der ersten versteckten Schicht berechnet eine Funktion der Form

$$o_v(\mathbf{x}) = \text{sign}\left(\sum_{i=1}^{n} w_i x_i + \theta\right) \quad .$$

Das ist die Indikatorfunktion für einen affinen Halbraum in $\mathbb{R}^n$. Ein *konvexer Polyeder* ist ein Durchschnitt von affinen Halbräumen. Die Indikatorfunktion eines Polyeder ist daher die Konjunktion (Ver-Und-ung) von Indikatorfunktionen von affinen Halbräumen. Somit können neuronale Netzwerke mit einer versteckten Schicht die Indikatorfunktionen zu Polyedern berechnen. Mit zwei versteckten Schichten sind dann Indikatorfunktionen von Vereinigungen konvexer Polyeder berechenbar. Abbildung 5 veranschaulicht diese Beobachtungen in $\mathbb{R}^2$.

15.4 Die Informationskomplexität von FFNN

Es sei $\mathcal{H} \subseteq \{h : \mathcal{X} \to \mathcal{Y}\}$ für einen Grundbereich $\mathcal{X}$ und einen endlichen Zielbereich $\mathcal{Y}$. Wie gewohnt bezeichnet $\mathcal{H}_M$ mit $M \subseteq \mathcal{X}$ die Menge der auf M eingeschränkten Funktionen aus $\mathcal{H}$. Der Beweis des folgenden Resultates verwendet die einer binären Hypothesenklasse $\mathcal{H}$ zugeordnete Kapazitätsfunktion $\tau_{\mathcal{H}}(m)$, die wir im Kapitel „Lernen via uniforme Konvergenz“ kennengelernt hatten.

Theorem 15.4 *Es sei $(V, E, sign)$ eine FFNN-Architektur mit einem Ausgabeneuron und sign als Aktivierungsfunktion. Dann gilt:*

$$\mathrm{VCD}(\mathcal{H}_{V,E,sign}) = O(|E| \ln(|E|)) \ . \tag{89}$$

Beweis Der Kürze halber setzen wir $\mathcal{H} := \mathcal{H}_{V,E,\mathrm{sign}}$. Es sei $v_1, \ldots, v_N$ eine topologische Sortierung der Neurone in (V, E), so dass aus $(v_i, v_j) \in E$ stets $i < j$ folgt. Insbesondere ist v_N das Ausgabeneuron. Es sei n die Dimension der Netzeingabevektoren. Für $i = 1, \ldots, N$ sei r_i die Anzahl der in v_i eingehenden Kanten aus E. Beachte, dass dann $|E| = \sum_{i=1}^{N} r_i$. Zu gegebenem $w : E \to \mathbb{R}$ bezeichne $h_w^{(i)}(\mathbf{x})$ die vom i-ten Neuron v_i in Abhängigkeit von der Netzeingabe $\mathbf{x} \in \mathbb{R}^n$ berechnete Funktion. Betrachte die folgende Familie von Funktionen:

$$\mathcal{H}^{\leq i} = \{\mathbf{x} \mapsto (h_w^{(1)}(\mathbf{x}), \ldots, h_w^{(i)}(\mathbf{x}))|\ w : E \to \mathbb{R}\}$$

Fixiere ein $m \geq 1$ und eine beliebige Menge $M \subset \mathbb{R}^n$ mit $m = |M|$. Es ist leicht zu sehen[36], dass

$$|\mathcal{H}_M^{\leq i}| \leq |\mathcal{H}_M^{\leq i-1}| \cdot \left(\frac{em}{r_i}\right)^{r_i} \leq |\mathcal{H}_M^{\leq i-1}| \cdot (em)^{r_i} \ .$$

Expandieren dieser Rekursion liefert

$$|\mathcal{H}_M^{\leq N}| \leq \prod_{i=1}^{N} (em)^{r_i} = (em)^{|E|} \ .$$

Wegen $|\mathcal{H}_M| \leq |\mathcal{H}_M^{\leq N}|$ können wir nun schlussfolgern, dass $\tau_{\mathcal{H}}(m) \leq (em)^{|E|}$. Für $m = \mathrm{VCD}(\mathcal{H})$ müsste gelten $2^m \leq (em)^{|E|}$. Geschicktes Auflösen nach m liefert $m = \mathrm{VCD}(\mathcal{H}) \leq c|E| \ln(|E|)$ für eine geeignete Konstante $c > 0$. •

15.5 FFNN und Backpropagation

Das Konsistenzproblem für $\mathcal{H}_{V,e,\sigma}$ ist selbst für einfache Architekturen NP-hart und das Lernproblem für $\mathcal{H}_{V,e,\sigma}$ ist unter bestimmten kryptographischen Voraussetzungen selbst dann hart, wenn der Lernalgorithmus die Form seiner Hypothese frei wählen kann. Daher existieren für FFNN nur heuristische Lernverfahren. Das bekannteste unter ihnen ist „Backpropagation“. Es handelt sich dabei, wie wir im Folgenden herausarbeiten werden, im Wesentlichen um ein stochastisches Gradientenabstiegsverfahren, das eine lokal optimale Wahl der Gewichtsparameter findet.

[36] Das wäre eine gute Übungsaufgabe.

15.5.1 Grundlagen aus der Analysis

Es sei seien $f_1, \ldots, f_m$ differenzierbare Funktionen in n reellen Variablen und $f = (f_1, \ldots, f_m) : \mathbb{R}^n \to \mathbb{R}^m$. Es bezeichne $J_{\mathbf{w}}(f) \in \mathbb{R}^{m \times n}$ die Funktionalmatrix (englisch: Jacobian) von f am Punkt $\mathbf{x} = \mathbf{w}$, d.h., $J_{\mathbf{w}}(f)[i,j] = \frac{\partial f_i}{\partial x_j}(\mathbf{w})$. Mit anderen Worten: die i-te Zeile von $J_{\mathbf{w}}(f)$ ist der Gradient von f_i am Punkt $\mathbf{x} = \mathbf{w}$. Betrachte die Komposition $f \circ g : \mathbb{R}^k \to \mathbb{R}^m$ für eine weitere differenzierbare Funktion $g : \mathbb{R}^k \to \mathbb{R}^n$. Gemäß der Kettenregel gilt $J_{\mathbf{w}}(f \circ g) = J_{g(\mathbf{w})}(f) \cdot J_{\mathbf{w}}(g)$.

Beispiel 15.5 *Für $f(\mathbf{w}) = A\mathbf{w}$ mit $A \in \mathbb{R}^{n \times k}$ gilt $J_{\mathbf{w}}(f) = A$ für alle $\mathbf{w} \in \mathbb{R}^k$.*
Es sei $\sigma : \mathbb{R} \to \mathbb{R}$ die Sigmoid-Funktion, d.h., $\sigma(\theta) = \frac{1}{1+e^{-\theta}}$. Man rechnet leicht nach, dass $\sigma'(\theta) = \frac{1}{(1+e^{\theta})(1+e^{-\theta})}$. Weiter sei $\bar{\sigma} : \mathbb{R}^n \to \mathbb{R}^n$ die Funktion, welche σ komponentenweise anwendet, d.h., für $\bar{\theta} = (\theta_1, \ldots, \theta_n)$ gilt $\bar{\sigma}(\bar{\theta}) = (\sigma(\theta_1), \ldots, \sigma(\theta_n))$. Dann ist $J_{\bar{\theta}}(\bar{\sigma})$ eine $(n \times n)$-Diagonalmatrix mit den Werten $\sigma'(\theta_1), \ldots, \sigma'(\theta_n)$ auf der Hauptdiagonalen. Wir notieren diese Matrix im Folgenden als $\mathrm{diag}(\overline{\sigma'}(\bar{\theta}))$.
Sei nun $A \in \mathbb{R}^{n \times k}$ und $g(\mathbf{w}) = A\mathbf{w}$. Dann gilt

$$J_{\mathbf{w}}(\sigma \circ g) = \mathrm{diag}(\overline{\sigma'}(A\mathbf{w})) \cdot A \ . \tag{90}$$

Die Aktualisierung eines programmierbaren Parameters $\mathbf{w} \in \mathbb{R}^k$ im Rahmen eines Gradientenabstieges bezüglich einer Kostenfunktion $f : \mathbb{R}^k \to \mathbb{R}$ hat die allgemeine Form

$$\mathbf{w} := \mathbf{w} - \eta \nabla f(\mathbf{w}) \ ,$$

wobei $\nabla f(\mathbf{w}) = J_{\mathbf{w}}(f)$ den Gradienten von f an der Stelle $\mathbf{w}$ und $\eta > 0$ die sogenannte „Schrittweite“ oder „Lernrate“ bezeichnet. Der Vektor $-\nabla f(\mathbf{w})$ gibt, vom Punkt $\mathbf{w}$ aus gesehen, die Richtung an, in welcher die f-Werte am stärksten abfallen.

15.5.2 Gradientenabstieg und Backpropagation

Wir betrachten eine geschichtete Architektur (V, E, σ), wobei σ die Sigmoidfunktion bezeichnet, $|V_0| = n + 1$ und $|V_t| = k_t$ für $t = 1, \ldots, T$. Wir fassen im Folgenden $w : E \to \mathbb{R}$ als Vektor $\mathbf{w} \in \mathbb{R}^E$ auf (mit einer Gewichtskomponente für jede Kante). Ein Netzwerk $\mathcal{N} = (V, E, \sigma, \mathbf{w})$ berechnet eine Funktion $h_{\mathbf{w}} : \mathbb{R}^n \to \mathbb{R}^{k_T}$. Wir verwenden die regularisierte quadratische Verlustfunktion, die einer Hypothese $h_{\mathbf{w}}$ und einem Beispiel $(\mathbf{x}, \mathbf{y})$ die Kosten

$$\ell(h_{\mathbf{w}}, (\mathbf{x}, \mathbf{y})) = \frac{1}{2}\|h_{\mathbf{w}}(\mathbf{x}) - \mathbf{y}\|^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{91}$$

zuordnet. Dabei ist, wie üblich, $\lambda > 0$ ein Regularisierungsparameter. Die eigentliche Kostenfunktion ist $L_{\mathcal{D}}(h_{\mathbf{w}}) + \frac{\lambda}{2}\|\mathbf{w}\|^2$ mit

$$L_{\mathcal{D}}(h_{\mathbf{w}}) = \frac{1}{2} \cdot \mathbb{E}_{(\mathbf{x},\mathbf{y}) \sim \mathcal{D}}[\|h_{\mathbf{w}}(x) - \mathbf{y}\|^2] \ . \tag{92}$$

Den Gradienten dazu können wir aber ohne Kenntnis von $\mathcal{D}$ nicht bestimmen. Wir gehen daher rundenweise vor, wählen in jeder Runde ein neues zufälliges Beispiel $(\mathbf{x}, \mathbf{y}) \sim \mathcal{D}$ und

realisieren den nächsten Schritt des Gradientenabstiegs mit der in (91) spezifizierten Kostenfunktion. Wegen (92) entspricht ℓ im statistischen Mittel der idealen (aber uns unbekannten) Kostenfunktion. Verfahren dieser Bauart heißen „stochastische Gradientenabstiegsverfahren“.

Es erweist sich als vorteilhaft, die Schrittweite η des Gradientenabstiegs im Laufe der Iterationen zu reduzieren. Die in der i-ten Iteration verwendete Schrittweite bezeichnen wir mit η_i. Weiterhin ist es gut, den Gewichtsvektor nicht mit $\mathbf{0}$ zu initialisieren sondern mit einem zufällig gewählten Vektor (gemäß einer Verteilung, deren Realisierung i.A. Vektoren in der lokalen Umgebung von $\mathbf{0}$ produziert). Ein Startvektor $\mathbf{0}$ hätte nämlich zwei Nachteile:

- Falls zwei aufeinander folgende Schichten vollständig vernetzt sind, dann würden Gewichtsparameter, welche der selben Schicht angehören, auch nach Aktualisierung gleich bleiben (was die Freiheitsgrade zu sehr einschränkt).
- Mit identischen Startvektoren in verschiedenen Iterationen des Verfahrens wird, im Vergleich zu wechselnden Startvektoren, ein kleinerer Teil des Parameterraumes exploriert.

Es folgt der Pseudocode für das stochastische Gradientenabstiegsverfahren. Es verwendet die Hilfsprozedur „backpropagation“, die wir im Anschluss erläutern.

Parameter: Anzahl $r \geq 1$ an Iterationen, Schrittweiten $\eta_1, \ldots, \eta_r > 0$ und Regularisierungsparameter $\lambda > 0$

Eingabe: geschichtete FFNN-Architektur (V, E, σ)

Initialisierung: Wähle $\mathbf{w}^{(\mathbf{1})}$ zufällig.

Hauptschleife: Für $i = 1, \ldots, r$ mache folgendes:

1. Nimm ein zufälliges Beispiel $(\mathbf{x}, \mathbf{y}) \sim \mathcal{D}$.
2. Berechne den Gradienten $\mathbf{G}^{(\mathbf{i})} :=$ backpropagation$(V, E, \sigma, \mathbf{x}, \mathbf{y}, \mathbf{w}^{(\mathbf{i})})$.
3. Setze $\mathbf{w}^{(\mathbf{i+1})} = \mathbf{w}^{(\mathbf{i})} - \eta_i(\mathbf{G}^{(\mathbf{i})} + \lambda \mathbf{w}^{(\mathbf{i})})$.

Ausgabe Gib einen Gewichtsvektor $\mathbf{w} \in \{\mathbf{w}^{(\mathbf{1})}, \ldots, \mathbf{w}^{(\mathbf{r})}\}$ aus, der auf einer Validierungsmenge am besten abschneidet.

Wir werden nachweisen, dass die Aktualisierung von $\mathbf{w}$ in Schritt 3 der Hauptschleife dem allgemeinen Schema des Gradientenabstiegs folgt, und zwar bezüglich der in (91) spezifizierten Kostenfunktion. Dabei ist der Term $\lambda \mathbf{w}^{(\mathbf{i})}$ gerade der Gradient von $\frac{\lambda}{2}\|\mathbf{w}\|^2$ an der Stelle $\mathbf{w}^{(\mathbf{i})}$. Es wird also noch zu zeigen sein, dass der durch den Aufruf von Backpropagation berechnete Term $\mathbf{G}^{(\mathbf{i})}$ den Gradienten von $\|\mathbf{h}_{\mathbf{w}}(\mathbf{x}) - \mathbf{y}\|^2 = \|\mathbf{o}_{\mathbf{T}}(\mathbf{x}) - \mathbf{y}\|^2$ an der Stelle $\mathbf{w}^{(\mathbf{i})}$ bildet.[37]

[37] Dieser Gradient lässt sich nicht analytisch, sondern nur algorithmisch, ermitteln.

Wir setzen im Folgenden der Einfachheit halber voraus, dass zwei aufeinander folgende Schichten vollständig vernetzt sind. Die allgemeine Prozedur ergibt sich aus der hier geschilderten, indem die Gewichtsparameter zu fehlenden Kanten stets auf den Wert 0 gesetzt werden.
Backpropagation ist die folgende Prozedur.

Eingabe: geschichtete FFNN-Architektur (V, E, σ), gelabeltes Beispiel $(\mathbf{x}, \mathbf{y})$ und ein Gewichtsvektor $\mathbf{w} \in \mathbb{R}^E$.

Initialisierung:
1. Kreiere die Schichten $V_0, V_1, \ldots, V_T$ mit $V_t = \{v_{t,1}, \ldots, v_{t,k_t}\}$.
2. Für $t = 0, \ldots, T-1$, $i = 1, \ldots, k_{t+1}$ und $j = 1, \ldots, k_t$ setze $W_{t,i,j} := w((v_{t,j}, v_{t+1,i}))$. W_t sei dann die resultierende $(k_{t+1} \times k_t)$-Matrix.

Vorwärtsphase: (Berechnung der Ausgabe des Netzwerks)

1. Es bezeichne $\mathbf{o_0} = (\mathbf{x}, 1) \in \mathbb{R}^{n+1}$ den Vektor der Ausgabewerte der Eingabeschicht.
2. Für $t = 1, \ldots, T$ und $i = 1, \ldots, k_t$ setze

$$a_{t,i} := \sum_{j=1}^{k_{t-1}} W_{t-1,i,j} o_{t-1,j} \quad \text{und} \quad o_{t,i} := \sigma(a_{t,i}) \ .$$

   $\mathbf{a_t}, \mathbf{o_t} \in \mathbb{R}^{k_t}$ seien dann die resultierenden Vektoren.

Rückwärtsphase: (Berechnung eines Gradienten)

1. Setze $\mathbf{d_T} := \mathbf{o_T} - \mathbf{y}$.
2. Für $t = T-1, \ldots, 1$ und $i = 1, \ldots, k_t$ setze

$$d_{t,i} := \sum_{j=1}^{k_{t+1}} W_{t,j,i} d_{t+1,j} \sigma'(a_{t+1,j}) \ .$$

   $\mathbf{d_t} \in \mathbb{R}^{k_t}$ sei dann der resultierende Vektor.

Ausgabe: Gib $\mathbf{G} \in \mathbb{R}^E$ aus mit

$$G((v_{t-1,j}, v_{t,i})) := d_{t,i} \sigma'(a_{t,i}) o_{t-1,j} \ .$$

In der nun folgenden Analyse werden wir den Zusammenhang zum Gradientenabstieg herstellen.

Es sei $\ell_t(\mathbf{u})$ der (quadratische) Verlustwert, der bezüglich des Beispiels $(\mathbf{x}, \mathbf{y})$ entsteht, wenn $\mathbf{u}$ die Ausgabewerte der t-ten Schicht repräsentiert. Diese Werte ergeben sich rekursiv wie folgt:

1. $\ell_T(\mathbf{u}) = \frac{1}{2}\|\mathbf{u}-\mathbf{y}\|^2$.

2. $\ell_t(\mathbf{u}) = \ell_{t+1}(\bar{\sigma}(W_t\mathbf{u}))$ für $t = T-1,\ldots,1$.

Aus dieser Rekursion ergeben sich die Funktionalmatrizen (bzw. Gradienten, da die Verlustfunktion Werte in $\mathbb{R} = \mathbb{R}^1$ annimmt) $J_{\mathbf{u}}(\ell_T) = \mathbf{u}-\mathbf{y}$ und, für $t = T-1,\ldots,1$ unter Anwendung der Kettenregel, $J_{\mathbf{u}}(\ell_t) = J_{\bar{\sigma}(W_t\mathbf{u})}(\ell_{t+1})\mathrm{diag}(\overline{\sigma'}(W_t\mathbf{u}))W_t$. Setzen wir $\mathbf{u} := \mathbf{o_t}$ und $\mathbf{d_t^\top} := J_{\mathbf{o_t}}(\ell_t)$, dann ergibt sich

$$\mathbf{d_T} = \mathbf{o_T} - \mathbf{y}$$

und für $t = T-1,\ldots,1$ gilt

$$\begin{aligned}\mathbf{d_t^\top} = J_{\mathbf{o_t}}(\ell_t) &= J_{\bar{\sigma}(W_t\mathbf{o_t})}(\ell_{t+1})\mathrm{diag}(\overline{\sigma'}(W_t\mathbf{o_t}))W_t \\ &= J_{\mathbf{o_{t+1}}}(\ell_{t+1})\mathrm{diag}(\overline{\sigma'}(\mathbf{a_{t+1}}))W_t = \mathbf{d_{t+1}^\top}\mathrm{diag}(\overline{\sigma'}(\mathbf{a_{t+1}}))W_t \ .\end{aligned}$$

Diese rekursive Rechenvorschrift zur Berechnung der $\mathbf{d_t}$-Werte stimmt überein mit der Berechnung der $d_{t,i}$-Werte während der Rückwärtsphase von Backpropagation.

Da $\mathbf{w} \in \mathbb{R}^E$ die programmierbaren Parameter sind, reicht es nicht aus die Gradienten $\mathbf{d_t^\top} = J_{\mathbf{o_t}}(\ell_t)$ zu kennen. Vielmehr müssen wir die Funktion

$$g_t(W_{t-1}) := \ell_t(\mathbf{o_t}) = \ell_t(\bar{\sigma}(\mathbf{a_t})) = \ell_t(\bar{\sigma}(W_{t-1}\mathbf{o_{t-1}}))$$

betrachten, welche den auf Schicht t entstehenden Verlust als eine Funktion in W_{t-1} auffasst[38], um dann ihren Gradienten an der Stelle W_{t-1} zu bestimmen. Dies geschieht zunächst für $t = T$ und danach für $t = T-1,\ldots,1$, so dass nach und nach der Gradient $\mathbf{G}$ für alle Gewichtsparameter bestimmt wird.

Zu diesem Zweck ist es besser, $W_{t-1} \in \mathbb{R}^{k_t \times k_{t-1}}$ als Vektor $\mathbf{w_{t-1}} \in \mathbb{R}^{k_t k_{t-1}}$ aufzufassen. Sei also $\mathbf{w_{t-1}}$ der Vektor der entsteht, wenn die Zeilen von W_{t-1} aneinander gehängt werden. Weiter sei

$$O_{t-1} := \begin{pmatrix} \mathbf{o_{t-1}^\top} & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathbf{o_{t-1}^\top} & \ldots & \mathbf{0} \\ \vdots & \vdots & & \vdots \\ \mathbf{0} & \mathbf{0} & \ldots & \mathbf{o_{t-1}^\top} \end{pmatrix} \in \mathbb{R}^{k_t \times k_t k_{t-1}} \ ,$$

so dass $W_{t-1}\mathbf{o_{t-1}} = O_{t-1}\mathbf{w_{t-1}}$. Wir erhalten dann die Gleichung

$$g_t(\mathbf{w_{t-1}}) = \ell_t(\bar{\sigma}(O_{t-1}\mathbf{w_{t-1}})) \ .$$

Anwendung der Kettenregel liefert

$$\begin{aligned} J_{\mathbf{w_{t-1}}}(g_t) &= J_{\bar{\sigma}(O_{t-1}\mathbf{w_{t-1}})}(\ell_t)\mathrm{diag}(\overline{\sigma'}(O_{t-1}\mathbf{w_{t-1}}))O_{t-1} \\ &= J_{\mathbf{o_t}}(\ell_t)\mathrm{diag}(\overline{\sigma'}(\mathbf{a_t}))O_{t-1} \\ &= \mathbf{d_t^\top}\mathrm{diag}(\overline{\sigma'}(\mathbf{a_t}))O_{t-1} \\ &= (d_{t,1}\sigma'(a_{t,1})\mathbf{o_{t-1}^\top},\ldots,d_{t,k_t}\sigma'(a_{t,k_t})\mathbf{o_{t-1}^\top}) \ . \end{aligned}$$

[38] wobei alle anderen Gewichtsparameter als Konstanten betrachtet werden, wie das bei partiellen Ableitungen stets der Fall ist

Der Gradient $J_{\mathbf{w_{t-1}}}(g_t)$ ist ein Vektor mit $k_t k_{t-1}$ Komponenten. Die Komponente, die für die Aktualisierung des Gewichtsparameters zur Kante $(v_{t-1,j}, v_{t,i})$ zuständig ist, finden wir in Position $(i-1)k_{t-1}+j$. Für den Gradientenvektor $\mathbf{G} \in \mathbb{R}^E$ muss daher gelten: $G((v_{t-1,j}, v_{t,i}))$ stimmt überein mit der Komponente von $J_{\mathbf{w_{t-1}}}(g_t)$ in Position $(i-1)k_{t-1}+j$. Inspektion von $(d_{t,1}\sigma'(a_{t,1})\mathbf{o_{t-1}^\top}, \ldots, d_{t,k_t}\sigma'(a_{t,k_t})\mathbf{o_{t-1}^\top})$ liefert

$$G((v_{t-1,j}, v_{t,i})) = d_{t,i}\sigma'(a_{t,i})o_{t-1,j} \ .$$

Dies deckt sich aber mit der Art wie Backpropagation $\mathbf{G}$ in der Ausgabephase berechnet.

Résumée. Das in diesem Abschnitt angegebene Verfahren „Gradientenabstieg mit Verwendung von Backpropagation“ realisiert einen stochastischen Gradientenabstieg.

Literatur

[1] David Haussler. Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, University of California, Santa Cruz, 1999.

[2] Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, 1994.

[3] Shai Shalev-Shwartz and Shai Ben-David. *Understandig Machine Learning: From Theory to Algorithms*. Cambridge University Press, 2014.

[4] John Shawe-Taylor and Nello Cristianini. *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.