Bachelorarbeit

im Studiengang Wirtschaftsmathematik
an der
UNIVERSITÄT MANNHEIM

Arbeit zur Erlangung des akademischen Grades Bachelor of Science zum Thema:

Nichtparametrische Regression durch tiefe neuronale Netzwerke mit ReLU Aktivierungsfunktion

vorgelegt von
Minh Duc Bui
Martikelnummer: 1557406

Abgabedatum: 22.07.2019

Erstgutachter: Prof. Dr. Claudia Strauch
Zweitgutachter: Prof. Dr. Andreas Neuenkirch

Inhaltsverzeichnis

1 Einleitung und Motivation

Diese Arbeit basiert auf der wissenschaftlichen Arbeit „Nonparametric regression using deep neural networks with ReLU activation function“ von Johannes Schmidt-Hieber, vgl. [16], und behandelt das Thema Deep Learning, eines der am meisten diskutierten Themen der Statistik von heute. Das steigende Interesse liegt in der Entwicklung der Technik und der Digitalisierung begründet. Diese beiden Faktoren haben es erst ermöglicht, tiefe neuronale Netzwerke, die Werkzeuge des Deep Learnings, effektiv zu nutzen. Neuronale Netzwerke brauchen nämlich nicht nur leistungsstarke Computer, sondern auch eine sehr große Menge an Daten. Erst die Digitalisierung hat es ermöglicht persönliche Daten, digitale Fotos, Sensordaten und mehr zu speichern und produzieren, durch beispielsweise leistungsstarke Speicherkarten, Social Media Plattformen, Smartphones und präzise Sensoren. Im Gegensatz dazu nutzen traditionelle Machine Learning Algorithmen diese Menge an Daten nicht effektiv aus.

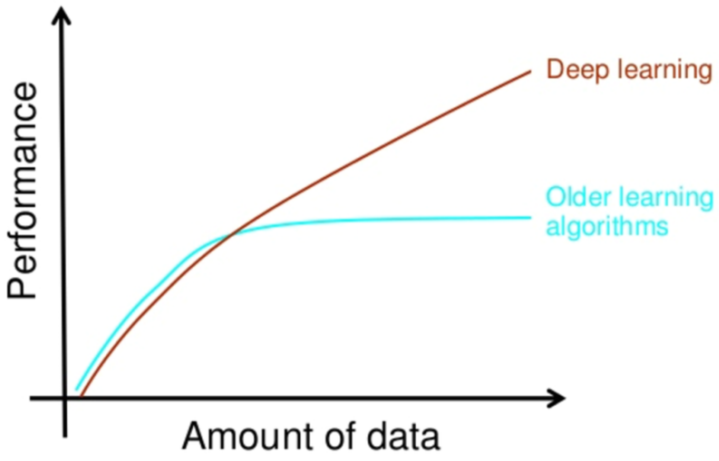

Abbildung 1: Bei großen Datenmengen performen Deep Learning Methoden viel besser als traditionelle Machine Learning Algorithmen.[1]

In den letzten Jahren gab es durch die Methoden des Deep Learnings beeindruckende praktische Resultate, wie zum Beispiel die Objekterkennung in Bildern. Dabei verarbeiten sogenannte faltende neuronale Netzwerke Bilder als Input, um bestimmte Objekte im Bild zu identifizieren. Der Einsatz in der Bildverarbeitung ist jedoch nur einer von vielen Bereichen, in denen neuronale Netzwerke bereits die Standardmethode bilden. Jedoch hat sich die Technik schneller entwickelt als die Theorie, weshalb sich die Frage stellt, wie man die

[1] Quelle: Analytics Vidhya: Web Url <https://www.analyticsvidhya.com/blog/2017/04/comparison-between-deep-learning-machine-learning/>, 15.06.2019.

beobachteten Phänomene der Praxis in einer statistischen Theorie begründet, um damit gegebenenfalls weitere Erkenntnisse für die Praxis zu gewinnen.

Im 1. Kapitel „Einleitung und Motivation“ werden wichtige Begriffe eingeführt, die für das Verständnis der Arbeit wichtig sind. Im 2. Kapitel „Beschreibung des Modells“ werden wir unser neuronales Netzwerk definieren und die Rahmenbedingungen des Modells festlegen, wie beispielsweise die Nutzung der ReLU Aktivierungsfunktion. In dem Abschnitt definieren wir auch die betrachtete Netzwerkklasse und Funktionsklasse der Regressionsfunktion. Die Haupttheoreme und Beweise findet man im 3. Kapitel „Hauptresultate“. Unter dem 4. Kapitel „Beispiele“ werden die additiven und verallgemeinerten additiven Modelle als Beispiele diskutiert. Im letzten Kapitel werden wir die wichtigsten Ergebnisse dieser Arbeit erwähnen und dabei Verbesserungen und Erweiterungen des Modells besprechen.

1.1 Ziel der Arbeit

Wir betrachten in dieser Arbeit ein multivariates nichtparametrisches Regressionsmodel und nehmen dabei an, dass die Regressionsfunktion aus einer Komposition von Funktionen besteht. Ziel dieser Arbeit ist es, für eine bestimmte gewählte Netzwerkarchitektur, eine obere Schranke für den L_2-Fehler zu beweisen, unter der Annahme, dass das tiefe neuronale Netzwerk *sparsly connected* ist mit einer ReLU Aktivierungsfunktion. Aus den Resultaten können wir folgern, dass unter bestimmen Voraussetzungen *tiefe feedforward neuronale Netzwerke* den Fluch der Dimensionalität umgehen. Wir werden anschließend eine untere Schranke für den L_2-Fehler angeben, für dessen Beweis wir auf [16] verweisen. Unter denselben Annahmen können wir damit die Minimax-Konvergenzrate für Schätzer aus solchen Netzwerken (bis auf einen in der Stichprobengröße n logarithmischen Faktor) folgern. Die Resultate in dieser Arbeit werden unter anderem auch zeigen, dass die Tiefe eines neuronalen Netzwerkes mit der Anzahl an Trainingsdaten steigen sollte, um die besten Resultate zu erhalten. Vorab müssen wir jedoch einige Fachtermini einführen, die für das Verständnis dieser Arbeit essenziell sind.

1.2 Nichtparametrische Regression

Wir werden zunächst das statistische Problem definieren, das wir in dieser Arbeit behandeln. In der Regressionsanalyse betrachten wir einen Zufallsvektor $(\mathbf{X}, Y)$ mit Werten in $\mathbb{R}^d \times \mathbb{R}$, wobei $\mathbb{E}Y^2 < \infty$ gilt. Das Ziel der Analyse ist es, den Wert für Y aus dem Beobachtungsvektor $\mathbf{X}$ vorherzusagen. Mathematisch kann man diese Beziehung darstellen als

$$Y = f(\mathbf{X}) + \underbrace{\epsilon}_{\textit{Störgröße}} . \tag{1}$$

Die Störgröße ist dabei eine standardnormalverteilte Zufallsvariable, die unabhängig von $\mathbf{X}$ ist. Das Ziel ist es nun, eine (messbare) Funktion $f' : \mathbb{R}^d \to \mathbb{R}$ zu finden, sodass $f'(\mathbf{X})$ unser

Y so gut wie möglich „approximiert“. Eine sogenannte *Verlustfunktion* ist eine messbare Funktion $L : \mathbb{R} \times \mathbb{R} \to \mathbb{R}_{\geq 0}$, die uns ein Maß gibt, wie gut wir Y mit unserer Funktion approximiert haben. Da aber nun $\mathbf{X}, Y$ Zufallsvariablen sind, können wir nicht einfach $|L(Y, f'(\mathbf{X}))|$ minimieren, da diese selber eine Zufallsvariable ist, sondern wir minimieren $\mathbb{E}[L(Y, f'(\mathbf{X}))]$, auch das Risiko genannt [20, S. 49].
Eine Wahl für die Verlustfunktion, die wir im Abschnitt 3 motivieren werden, ist die *quadratische Verlustfunktion* $L(y, s) = (y - s)^2$. Es folgt für das Risiko $\mathbb{E}[L(Y, f'(\mathbf{X})] = \mathbb{E}(|Y - f'(\mathbf{X})|^2)$, die man auch mittlere quadratische Abweichung von f' oder auch L_2-Risiko nennt. Wir suchen somit eine Funktion $f_0 : \mathbb{R}^d \to \mathbb{R}$, die die mittlere quadratische Abweichung minimiert, d.h. f_0 soll

$$\mathbb{E}[L(Y, f_0(\mathbf{X})] = \mathbb{E}(|Y - f_0(\mathbf{X})|^2) = \min_{f':\mathbb{R}^d \to \mathbb{R}} \mathbb{E}(|Y - f'(\mathbf{X})|^2) \tag{2}$$

erfüllen. Um herauszufinden, wie genau nun f_0 aussieht, definieren wir $m : \mathbb{R}^d \to \mathbb{R},\ m = \mathbb{E}(Y|\mathbf{X} = \mathbf{x})$ und betrachten eine beliebige Funktion $f' : \mathbb{R}^d \to \mathbb{R}$ mit der Gleichung

$$\begin{aligned}\mathbb{E}(|f'(\mathbf{X}) - Y|^2) &= \mathbb{E}(|f'(\mathbf{X}) - m(\mathbf{X}) + m(\mathbf{X}) - Y|^2) \\ &= \mathbb{E}(|f'(\mathbf{X}) - m(\mathbf{X})|^2) + \mathbb{E}(|m(\mathbf{X}) - Y)|^2),\end{aligned}$$

wobei wir diese als gegeben nehmen und in dieser Arbeit nicht herleiten werden, vgl. [7, S. 2]. Man sieht leicht ein, dass m unsere gesuchte Funktion ist, denn $\mathbb{E}(|f'(\mathbf{X}) - Y|^2)$ wird minimal genau dann, wenn $f' = m$. Man nennt die Funktion $f_0 = m = \mathbb{E}(Y|\mathbf{X} = \mathbf{x})$ Regressionsfunktion. Den Ausdruck $\mathbb{E}(|f'(\mathbf{X}) - m(\mathbf{X})|^2)$ nennt man auch L_2-Fehler von f', vgl. [7, S. 2]. Wir können also das L_2 Risiko als eine Summe des L_2 Risikos der Regressionsfunktion, der auch als unvermeidbarer Fehler bezeichnet wird, und des L_2-Fehler von f' darstellen.

Es verbirgt sich jedoch ein Problem: Wir kennen die Verteilung von $(\mathbf{X}, Y)$ nicht, weshalb die Schätzung von Y durch die Regressionsfunktion $f_0 = \mathbb{E}(Y|\mathbf{X} = \mathbf{x})$ nicht möglich ist. Doch falls wir Daten gegeben haben, die derselben Verteilung wie $(\mathbf{X}, Y)$ unterliegen, haben wir die Möglichkeit f_0 durch die Daten zu schätzen. Das statistische Problem lautet nun, eine Funktion $f_n : \mathbb{R}^d \to \mathbb{R}$ zur Schätzung der Regressionsfunktion f_0 durch gegebene Beobachtungen $D_n = (\mathbf{X}_1, Y_1, ..., \mathbf{X}_n, Y_n)$ zu konstruieren, wobei $(\mathbf{X}, Y), (\mathbf{X}_1, Y_1), ..., (\mathbf{X}_n, Y_n)$ u.i.v. Zufallsvariablen sind. Es gilt dabei, dass die Funktion $f_n(\mathbf{x}) = f_n(\mathbf{x}, D_n)$ messbar bezüglich $\mathbf{x}$ und den Daten ist. Das Ziel ist es nun, gegeben den Daten D_n, eine Funktion $f_n(\cdot) = f_n(\cdot, D_n)$ so zu konstruieren, sodass deren L_2-Fehler $\mathbb{E}(|f_n(\mathbf{X}) - f_0(\mathbf{X})|^2)$ minimal sind.

Als nächstes wollen wir den Unterschied zwischen der parametrischen und nichtparametrischen Regression erläutern. Bei der parametrischen Regression wird angenommen, dass die Struktur der Regressionsfunktion bekannt ist und zusätzlich wird angenommen, dass

diese nur von endlich vielen Parameter abhängt, deren Werte aber unbekannt sind. Diese unbekannten Werte werden nun aus den Daten geschätzt werden, vgl. [10, S. 4]. Beispielsweise wird bei der linearen Regression angenommen, dass die Regressionsfunktion linear ist und wir schätzen, die endliche Anzahl an Faktoren der Linearkombination. Es gibt jedoch wesentliche Nachteile der parametrischen Regression, denn man muss den Daten zunächst eine Struktur entnehmen, welche oftmals nicht direkt ersichtlich ist.

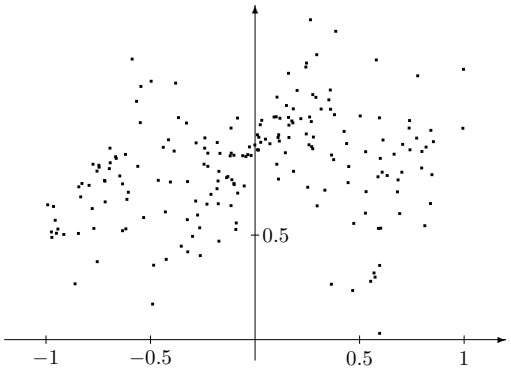

Abbildung 2: Datenpunkte.

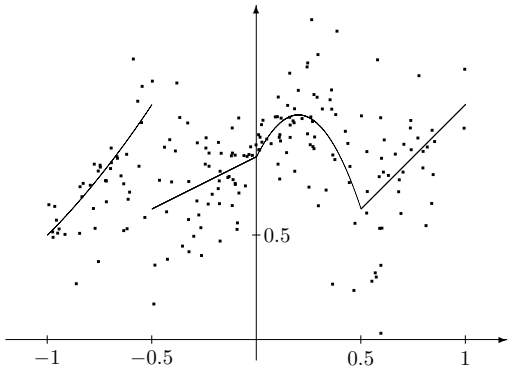

Abbildung 3: Datenpunkte mit der wahren Regressionsfunktion. [2]

Zum Beispiel kann man der Regressionsfunktion aus der Abbildung 2 nicht direkt mit bloßem Auge eine Strukturannahme unterstellen. Zusätzlich können bei Verletzung der angenommen Strukturannahme sehr schnell schlechte Schätzungen entstehen, vgl. [7, S. 11].

Im Gegensatz dazu wird bei der nichtparametrischen Regression angenommen, dass die Regressionsfunktion keine Funktionsstruktur aufweist, die durch endlich viele Parameter beschrieben werden kann. Es werden dabei nur bestimmte allgemeine Regularitätsbedingungen gestellt, wie zum Beispiel Stetigkeit, Differenzierbarkeit oder Hölder-Stetigkeit, vgl. [20, S. 3]. Somit können wir eine allgemeinere strukturelle Annahme an die Regressionsfunktion stellen. Dabei wird die Regressionsfunktion aus den Daten geschätzt, wobei es mehrere Verfahren gibt, wie zum Beispiel neuronale Netzwerke, Kernel-Regression und KNN Schätzer, vgl. [6, 7].

In dieser Arbeit werden wir uns auf die neuronalen Netzwerke und ihre möglichen Vorteile gegenüber anderen Schätzern fokussieren. Primär werden wir den durch die Konvergenzrate entstehenden Vorteil analysieren.

[2] [7, S. 11].

1.3 Konvergenzrate

Um den Fehler eines Schätzers f_n einer Regressionsfunktion f_0 zu messen, benutzen wir, wie oben motiviert, den L_2-Fehler und definieren ihn als

$$R(f_n, f_0) := \int |f_n(\mathbf{X}) - f_0(\mathbf{X})|^2 P_X(dx), \tag{3}$$

wobei P_X die Verteilung der Zufallsvariable $\mathbf{X}$ ist, vgl. [7, S.2]. Der Schätzer f_n ist dabei abhängig von den Daten D_n. Man will natürlich einen Schätzer finden, wofür der L_2 gegen 0 konvergiert (mit der Anzahl der Beobachtungen) und das so schnell wie möglich. Wir haben immer noch nicht die Verteilung $(\mathbf{X}, Y)$ in irgendeiner Weise eingeschränkt, außer dass $E(Y^2) < \infty$ gelten muss. Leider gibt es keinen Schätzer für den der L_2-Fehler für alle Verteilungen von $(\mathbf{X}, Y)$ mit einer festen Konvergenzrate gegen 0 konvergiert, d.h. für jeden Schätzer kann die Konvergenzrate beliebig langsam sein, vgl. [7], Abschnitt 3.1.

Wir müssen also unsere Verteilungsklasse, die in der die Verteilung von $(\mathbf{X}, Y)$ liegt, einschränken, um nicht triviale Konvergenzraten zu erhalten. Bevor wir jedoch die optimale Konvergenzrate eines Schätzverfahren gegeben einer Verteilungsklasse analysieren können, müssen wir vorher definieren, was optimal in diesem Kontext heißt. Wir sprechen von einer optimalen Konvergenzrate, falls die Rate der Minimax-Konvergenzrate entspricht. Wir müssen somit die Minimax-Konvergenz-rate einführen.

Wir versuchen zu einer gegebenen Klasse D von Verteilungen von $(\mathbf{X}, Y)$ den maximalen L_2-Fehler innerhalb der Klasse

$$\sup_{(\mathbf{X},Y)\in D} \int |f_n(\mathbf{X}) - f_0(\mathbf{X})|^2 P_X(dx) \tag{4}$$

durch einen Schätzer f_n zu minimieren, d.h. der Schätzer sollte „nah am“ Wert von

$$\inf_{\widetilde{f}_n} \sup_{(\mathbf{X},Y)\in D} \int |\widetilde{f}_n(\mathbf{X}) - f_0(\mathbf{X})|^2 P_X(dx) \tag{5}$$

liegen, wobei wir hier über alle Regressionsschätzer minimieren, vgl. [10]. Da wir an dem asymptotische Verhalten ($n \to \infty$) für (4) und (5) interessiert sind, bedeutet (4) „nah an“ (5), falls sie sich asymptotisch gleich verhalten in Abhängigkeit von n. Deshalb führen wir die folgenden Begriffe ein, die entnommen sind aus [10].

Definition 1. *Eine Folge von positiven Zahlen $(a_n)_{n\in\mathbb{N}}$ für die Klasse D von Verteilungen $(\mathbf{X}, Y)$ nennt man*
a) untere Minimax-Konvergenzrate für $D \Leftrightarrow$

$$\liminf_{n\to\infty} \inf_{f_n} \sup_{(\mathbf{X},Y)\in D} \frac{\int |f_n(\mathbf{X}) - f_0(\mathbf{X})|^2 P_X(dx)}{a_n} = C_1 > 0$$

b) obere Minimax-Konvergenzrate für D, falls für ein Schätzverfahren f_n gilt

$$\limsup_{n\to\infty} \sup_{(\mathbf{X},Y)\in D} \frac{\int |f_n(\mathbf{X}) - f_0(\mathbf{X})|^2 P_X(dx)}{a_n} = C_2 < \infty$$

c) optimale Minimax-Konvergenzrate für D, falls $(a_n)_{n\in\mathbb{N}}$ untere und obere Minimax-Konvergenzrate für D ist.

Wir hatten schon gesagt, dass wir die Verteilungsklasse von $(\mathbf{X}, Y)$ einschränken müssen, um nicht triviale Lösungen zu bekommen, weshalb wir eine typische Annahme der nichtparametrischen Statistik treffen. Wir nehmen an, dass die Regressionsfunktion β-glatt ist. Stone, siehe [19], hatte für β-glatte Regressionsfunktionen bestimmt, dass die optimale Konvergenzrate bei $n^{-\frac{2\beta}{2\beta+d}}$ liegt, wobei d die Dimension des Inputs ist. Jedoch beschäftigen sich neuronale Netzwerke häufig mit Problemen mit hochdimensionalen Input, weshalb die optimale Konvergenzrate groß ist, d.h. die Konvergenzrate für solche Probleme ist sehr langsam. Jedes statistisches Verfahren würde somit schlecht abschneiden, inklusive unser neuronales Netzwerk. Um zu verstehen, wie stark dieses Phänomen unser Modell einschränkt, machen wir ein kleines Zahlenbeispiel. Es sei ein Stichprobenumfang von $n = 100$, Inputdimension $d = 100$ und Glattheit $\beta = 2$ gegeben. Falls sich die Inputdimension auf $d = 1036$ erhöht, dann müsste man um die selbe Konvergenzrate zu erhalten, einen Stichprobenumfang von $n = 500^{10}$ haben! Sogar in der heutigen Zeit ist diese Menge an Daten praktisch unmöglich.
Es hat sich aber gezeigt, dass neuronale Netzwerke nicht von dem sogenannten „Fluch der Dimensionalität“ leiden. In dieser Arbeit werden wir unter anderem untersuchen, unter welchen Voraussetzungen und wie genau neuronale Netzwerke den Fluch umgehen, indem wir die Konvergenzrate des L_2-Fehlers analysieren.

1.4 Notationen

Vektoren werden mit fettgedruckten Buchstaben bezeichnet, zum Beispiel schreiben wir $\mathbf{x} := (x_1, ..., x_d)^\top$. Wir definieren die Vektornormen mit

$$|\mathbf{x}|_p := (\sum_{i=1}^{d} |x_i|^p)^{1/p}, \quad |\mathbf{x}|_\infty := max_i|x_i|, \quad |\mathbf{x}|_0 := \sum_{i=1}^{d} \mathbf{1}(x_i \neq 0).$$

Für Funktionen f schreiben wir die L^p-Norm auf D als $\|f\|_p := \|f\|_{L^p(D)}$, falls klar ist, dass D gemeint ist und es keine Mehrdeutigkeiten gibt.
Für zwei Folgen $(a_n)_{n\in\mathbb{N}}$ und $(b_n)_{n\in\mathbb{N}}$, schreiben wir

$$a_n \precsim b_n \iff \exists C \in \mathbb{R}_+ : \ a_n \leq C b_n \ \forall n \in \mathbb{N},$$
$$a_n \asymp b_n \iff a_n \precsim b_n \text{ und } b_n \precsim a_n.$$

Wir bezeichnen $\log_2$ als den Logarithmus zur Basis 2 und $\lceil x \rceil$ als die kleinste ganze Zahl mit $\geq x$.

2 Beschreibung des Modells

2.1 Motivation und Definition eines neuronalen Netzwerkes

Es sei $\mathbf{X}_i \in \mathbb{R}^d$ ein Beobachtungsvektor, wobei d die Dimension der Beobachtungen ist. In allen Anwendungsproblemen, die wir im Abschnitt 1 beschrieben haben, liegt ein Regressionsproblem mit einer hohen Inputdimension d vor. Dadurch werden wir, wie oben schon beschrieben, unter dem Fluch der Dimensionalität leiden, welches zur Folge hat, dass kein statistisches Verfahren das Problem in angemessener Rechenzeit bewältigt. Ein Lösungsansatz wäre die Reduktion des Informationsgehalt des ursprünglichen Beobachtungsvektors $\mathbf{X}_i$ auf eine neue Dimension $\widetilde{d} < d$ mittels „Expertenwissen“ und somit der Erhalt eines neuen Beobachtungsvektor $\widetilde{\mathbf{X}} \in \mathbb{R}^{\widetilde{d}}$. Algorithmen des maschinellen Lernens, angewandt auf die neuen Daten $(\widetilde{\mathbf{X}}_i, Y_i)_{i=1,...,n}$, liefern tatsächlich sehr gute Resultate, wobei wir jedoch annehmen, dass wir durch die Reduktion auf $\widetilde{X}_i$ alle wesentlichen Eigenschaften von $\mathbf{X}_i$ zusammengefasst haben. Bei einer sehr hohen Inputdimension d, wo wir dementsprechend auch eine starke Reduktion vornehmen müssen, ist es sehr unwahrscheinlich, dass wir kaum Informationsverlust erleiden und sich die Beziehung zwischen Y_i und $\widetilde{\mathbf{X}}_i$ nicht wesentlich verkomplizieren, vgl. [15, S. 194].

Neuronale Netzwerke reduzieren, genau wie oben, die Dimension d zu $\widetilde{d}$, jedoch nicht durch „Expertenwissen“, sondern auf Basis der Trainingsdaten. Wir brauchen mit neuronalen Netzwerken also kein vorheriges Expertenwissen und können direkt mit den ursprünglichen Trainingsdaten arbeiten. Neuronale Netzwerke reduzieren den Beobachtungsvektor $\mathbf{X}_i \in \mathbb{R}^d$ auf $\mathbf{X}_i \in \mathbb{R}^{\widetilde{d}}$ in $L \in \mathbb{N}$ Schritten gleicher Struktur, vgl. [15, S. 194]. Die Konstruktion eines Schrittes besteht aus einer einfachen Transformation des aktuell vorliegenden Vektors, wobei die Dimension der Daten dabei reduziert bzw. erhöht wird. Wir werden nach der Reduktion annehmen, dass die reduzierten Daten $(\widetilde{\mathbf{X}}_i, Y_i)_{i=1,...,n}$ dem Modell einer linearen Regression folgen, vgl. für Modelldefinition [15, S. 27].
Für die Reduktion der Dimension könnten wir in jedem Schritt beispielsweise eine einfache lineare Transformation ausprobieren, vgl. [15, S. 194].

Bemerkung. *[15, S. 194] (Linearer Ansatz zur Dimensionsreduktion)*
Seien $p_1, ..., p_L \in \mathbb{N}$. Für $i = 1, ..., n$:

(1) $\mathbf{X}_i^{(1)} := W_1 \mathbf{X}_i - \mathbf{v}_1 \ \in \mathbb{R}^{p_1}$ *mit geeignetem* $\mathbf{v}_1 \in \mathbb{R}^{p_1}, W_1 \in \mathbb{R}^{p_1 \times d}$.

(2) $\mathbf{X}_i^{(2)} := W_2 \mathbf{X}_i^{(1)} - \mathbf{v}_2 \ \in \mathbb{R}^{p_2}$ *mit geeignetem* $\mathbf{v}_2 \in \mathbb{R}^{p_2}, W_2 \in \mathbb{R}^{p_2 \times p_1}$.

$\vdots$

(L) $\mathbf{X}_i^{(L)} := W_L \mathbf{X}_i^{L-1} - \mathbf{v}_L \ \in \mathbb{R}^{p_L}$ *mit geeignetem* $\mathbf{v}_L \in \mathbb{R}^{p_L}, W_L \in \mathbb{R}^{p_L \times p_{L-1}}$.

Annahme: $(\mathbf{X}_i^{(L)}, Y_i)_{i=1,...,n}$ *folgt dem Modell einer linearen Regression.*

Wir haben damit unsere Trainingsdaten in $L \in \mathbb{N}$ Schritten gleicher Struktur von $\mathbf{X}_i \in \mathbb{R}^d$ auf $\mathbf{X}_i \in \mathbb{R}^{p_L}$ reduziert. Es gibt jedoch ein Problem: Kompositionen von linearen Abbildungen bilden wieder eine lineare Abbildung. Da wir aber nun auf unsere Daten lediglich lineare Abbildungen angewandt haben, können wir das Verfahren auch als

$$\mathbf{X}_i^{(L)} = \widetilde{\mathbf{v}} + \widetilde{W} \cdot \mathbf{X}_i \qquad \textit{mit geeignetem } \widetilde{\mathbf{v}} \in \mathbb{R}^{p_L},\ \widetilde{W} \in \mathbb{R}^{p_L \times d}$$

zusammenfassen. Unser Verfahren würde also einer linearen Regression entsprechen!

Um die Nichtlinearität in das Modell zu bekommen und somit auch komplexere Probleme lösen zu können, werden wir eine Art „Störung“ der Linearität einbauen. Um die Linearität zu stören, können wir eine möglichst einfache nichtlineare Funktion $\sigma : \mathbb{R} \to \mathbb{R}$ verwenden, eine sogenannte Aktivierungsfunktion.

Bemerkung. *[15, S. 194] [Nichtlinearer Ansatz zur Dimensionsreduktion]*
Seien $p_1, ..., p_L \in \mathbb{N}$. Für $i = 1, ..., n$:

(1) $\mathbf{Z}_i^{(1)} := W_1 \mathbf{X}_i - \mathbf{v}_1 \ \in \mathbb{R}^{p_1}$ *mit geeignetem* $\mathbf{v}_1 \in \mathbb{R}^{p_1}, W_1 \in \mathbb{R}^{p_1 \times d}$.
(1') $\mathbf{X}_i^{(1)} := \sigma(\mathbf{Z}_i^{(1)})$

(2) $\mathbf{Z}_i^{(2)} := W_2 \mathbf{X}_i^{(1)} - \mathbf{v}_2 \ \in \mathbb{R}^{p_2}$ *mit geeignetem* $\mathbf{v}_2 \in \mathbb{R}^{p_2}, W_2 \in \mathbb{R}^{p_2 \times p_1}$.
(2') $\mathbf{X}_i^{(2)} := \sigma(\mathbf{Z}_i^{(2)})$
$\vdots$
(L) $\mathbf{Z}_i^{(L)} := W_L \mathbf{X}_i^{(L-1)} - \mathbf{v}_L \ \in \mathbb{R}^{p_L}$ *mit geeignetem* $\mathbf{v}_L \in \mathbb{R}^{p_L}, W_L \in \mathbb{R}^{p_L \times p_{L-1}}$.

(L') $\mathbf{X}_i^{(L)} := \sigma(\mathbf{Z}_i^{(L)})$

Annahme: $(\mathbf{X}_i^{(L)}, Y_i)_{i=1,...,n}$ *folgt dem Modell einer linearen Regression.*

Nach Annahme über die reduzierten Daten $(\mathbf{X}_i^{(L)}, Y_i)_{i=1,...n}$ gilt mit geeignetem $W_{L+1} \in \mathbb{R}^{1 \times p_L}$ und u.i.v. ϵ_i mit $\mathbb{E}(\epsilon_i) = 0$:

$$Y_i = W_{L+1} \cdot \mathbf{X}_i^{(L)} + \epsilon_i, \quad i = 1, ...n$$

und nach dem linearen Regressionsmodel gilt für die Regressionsfunktion [15, S. 195]

$$f_0(x) = \mathbb{E}[Y_1 | \mathbf{X}_1 = \mathbf{x}] = W_{L+1} \cdot \widetilde{f}(\mathbf{x}),$$

wobei $\widetilde{f}$ sich ergibt durch zusammenfassen der Schritte $(1), (1'), ..., (L), (L')$ durch

$$\widetilde{f}(\mathbf{x}) = \sigma\left(W_L \cdot \sigma\left(...W_2 \cdot \sigma\left(W_1 \cdot \mathbf{x} - \mathbf{v}_1\right) - \mathbf{v}_2 ...\right) - \mathbf{v}_L\right).$$

ergibt. Somit gilt $\widetilde{f}(\mathbf{X}_i) = \mathbf{X}_i^{(L)},\ i = 1, ...n.$

Mithilfe der verschobenen Aktivierungsfunktion $\sigma_{\mathbf{v}} : \mathbb{R}^r \to \mathbb{R}^r$:

$$\sigma_{\mathbf{v}} \begin{pmatrix} y_1 \\ \vdots \\ y_r \end{pmatrix} = \begin{pmatrix} \sigma(y_1 - v_1) \\ \vdots \\ \sigma(y_r - v_r) \end{pmatrix},$$

wobei $\mathbf{v} = (v_1, ..., v_r) \in \mathbb{R}^r$, können wir die Funktion f_0 auch schreiben als

$$f_0 : \mathbb{R}^{p_0} \to \mathbb{R}^{p_{L+1}},\ \mathbf{x} \mapsto f_0(\mathbf{x}) = W_{L+1} \cdot \sigma_{\mathbf{v}_L}(W_L \cdot \sigma_{\mathbf{v}_{L-1}}(\cdots W_2 \cdot \sigma_{\mathbf{v}_1}(W_1 \cdot \mathbf{x}) \cdots)),$$

wobei W_i eine $p_i \times p_{i-1}$ Gewichtsmatrix und $\mathbf{v}_i \in \mathbb{R}^{p_i}$ Verschiebungsvektor ist. Für p_{L+1} gilt $p_{L+1} = 1$, da wir bei einer Regression einen einzigen Wert schätzen wollen gegeben dem Input mit der Dimension $p_0 = d$. Im Gegensatz dazu hätten wir für ein Klassifizierungsproblem mit K Klassen $p_{L+1} = K$, wobei im k-ten Output die Wahrscheinlichkeit (bzw. proportional zur Wahrscheinlichkeit), dass der Input der k-ten Klasse angehört, entspricht, vgl. [15, S. 196].

Definition 1. *[15, S. 196]*
Sei $L \in \mathbb{N}_0,\ \mathbf{p} = (p_0, ..., p_{L+1})^T \in \mathbb{N}^{L+2}$. *Ein neuronales Netzwerk mit Netzwerkarchitektur* $(L, \mathbf{p})$ *und verschobener Aktivierungsfunktion* $\sigma_{\boldsymbol{v}_i} : \mathbb{R}^{p_i} \to \mathbb{R}^{p_i}$ *ist eine Funktion* $f_0 : \mathbb{R}^{p_0} \to \mathbb{R}^{p_{L+1}}$ *mit*

$$f_0 : \mathbb{R}^{p_0} \to \mathbb{R}^{p_{L+1}},\ \mathbf{x} \mapsto f_0(\mathbf{x}) = W_{L+1} \cdot \sigma_{\boldsymbol{v}_L}(W_L \cdot \sigma_{\boldsymbol{v}_{L-1}}(\cdots W_2 \cdot \sigma_{\boldsymbol{v}_1}(W_1 \cdot \mathbf{x}) \cdots)), \quad (6)$$

wobei $W_l \in \mathbb{R}^{p_l \times p_{l-1}},\ l = 1, ..., L+1$ *Gewichtsmatrizen und* $\mathbf{v}_l \in \mathbb{R}^{p_l}$ *Verschiebungsvektoren heißen. L nennen wir die Anzahl der hidden Layer/Tiefe des neuronalen Netzwerkes und* $\mathbf{p}$ *heißt width vector.*

Die Netzwerkparameter sind die Einträge der Matrizen $(W_j)_{j=0,...,L}$ und der Vektoren $(\mathbf{v}_j)_{j=1,...,L}$, die durch die Daten geschätzt bzw. gelernt werden.
Eine weitere Möglichkeit, neuronale Netzwerke einzuführen, ist über eine graphische Betrachtung. Das Netzwerk wird durch einen gerichteten azyklischen Graphen dargestellt, der die Kompositionen von Funktionen aus (6) beschreiben soll. In dieser äquivalenten Definition werden die Knoten (units) des Graphen in Layers geschichtet. Der erste bzw. letzte Layer wird Input bzw. Output Layer genannt. Alle Layers zwischen der ersten und letzten nennt man *hidden Layers*, wobei die Anzahl der *hidden Layers* mit L bezeichnet wird. Jeder *Layer* ist typischerweise vektorwertig und die Dimension bzw. die Anzahl der Knoten der *Layer* bestimmen die jeweiligen Einträge des *width vector* p, vgl. [6, S. 165]. In jedem Knoten, der nicht im Input Layer ist, wird das Skalarprodukt der Aktivierungswerte ausgehend von den Knoten im vorherigem Layer mit einem Gewichtsvektor berechnet und auf die verschobene Aktivierungsfunktion angewendet. Somit erhält man für diesen Knoten auch einen Aktivierungswert.

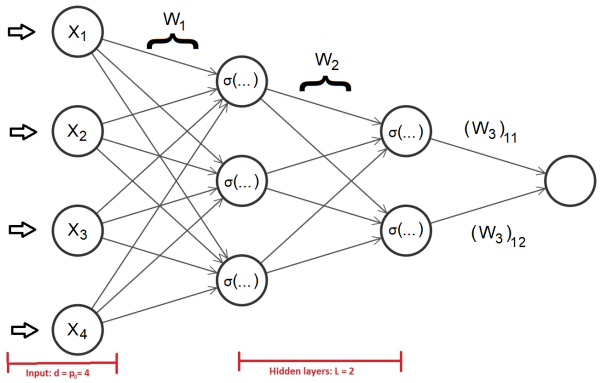

Abbildung 4: Graphische Darstellung eines neuronalen Netzwerkes mit *width vector* $\mathbf{p} = (4, 3, 2, 1)$

2.2 Rahmenbedingungen des Modells

Um eine statistische Theorie aufzubauen, müssen wir einige Eigenschaften eines neuronalen Netzwerkes fixieren, da ein allgemeines neuronales Netzwerk ohne Einschränkungen zu komplex ist, um es in einer statistischen Theorie zu fassen. Wir werden hierbei Eigenschaften wählen, die in der Praxis die besten Resultate liefern bzw. häufig vorkommen. Unser Modell soll damit alle Charakteristiken aufweisen, die ein typisches Netzwerk in der Praxis hat.
Es gab schon frühere statistische Resultate, die jedoch ein eingeschränkteres und nicht der Praxis entsprechendes Modell betrachten. Diese können in [14] nachgelesen werden.

2.2.1 Aktivierungsfunktion

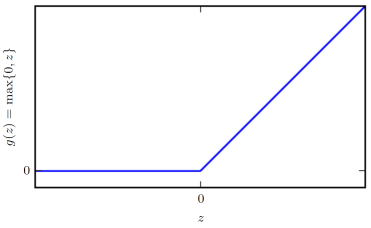

Abbildung 5: Die ReLU Funktion[3]

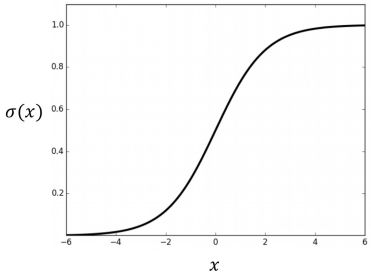

Abbildung 6: Die Sigmoid Funktion[4]

[3] [6, S. 170]

[4] Bernstein, Matthew: *Sigmoid functions*. 2016. Verfügbar über <https://mbernste.github.io/files/notes/SigmoidFunction.pdf>.

Wir werden in dieser Arbeit eine spezielle Aktivierungsfunktion betrachten, die ReLU (rectifier linear unit) Aktivierungsfunktion (Abb. 5) $\sigma : \mathbb{R} \to \mathbb{R}; \quad x \mapsto \sigma(x) = max(0, x) = (x)_+$.
Weitere Aktivierungsfunktionen wären zum Beispiel die Sigmoid-Aktivierungsfunktion (Abb. 6) oder die hyperbolic tangent-Aktivierungsfunktion. Die ReLU Funktion ist jedoch eine der erfolgreichsten und meist genutzten Aktivierungsfunktionen der heutigen Zeit, vgl. [4, S. 8]. Weshalb die ReLU Funktion anderen Aktivierungsfunktionen überlegen ist, hat vielerlei Gründe.
Zum einen produziert die ReLU Funktion viele inaktive *hidden units*, d.h. das Netzwerk ist *sparsely connected* (dünn verbunden), vgl. [13, S. 117]. Dünnbesetzte Netzwerke weisen besonders vorteilhafte Eigenschaften auf, vgl. [1], wobei einer der Vorteile im Abschnitt 2.2.3 erläutert wird.

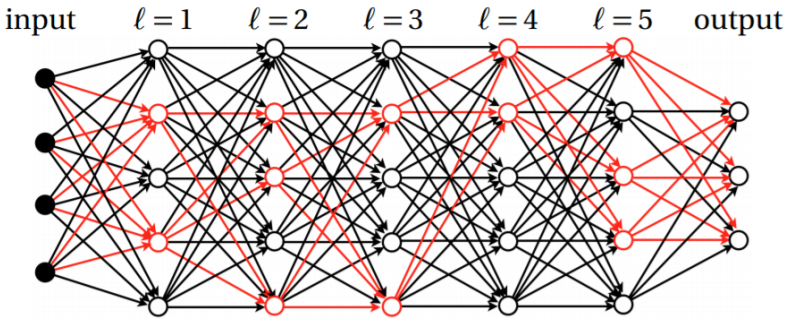

Abbildung 7: Ein *sparse* Netzwerk mit ReLU Aktivierungsfunktion. Die roten Linien illustrieren die Verbindungen zu den aktiven Knoten.[5]

Wie man in Abbildung 5 sehen kann, besteht die Funktion aus zwei stückweise linearen Teilen, wodurch die Funktion „annähernd“ linear ist. Aus diesen Grund bewahrt es viele Eigenschaften, die das Optimieren linearer Modelle auf Grundlage von Gradienten Methoden vereinfacht, vgl. [6, S. 170].
Eines der häufigen Probleme, das in anderen Aktivierungsfunktionen, beispielsweise in der Sigmoid Funktion, vorkommt, ist das Problem des Verschwinden des Gradienten (engl.: „The Vanishing Gradient Problem“). Das Problem stellt ein großes Hindernis im Lernalgorithmus eines tiefen neuronalen Netzwerkes dar und führt häufig zu Ungenauigkeiten des Modells. Mehr zu dem Problem kann man in [13], Abschnitt 7.2.1 nachlesen.
Zusammenfassend übertrifft die ReLU Aktivierungsfunktion in der Praxis tatsächlich andere Funktionen in Hinblick auf die statistische Performance und die Computerrechenzeit, vgl. [5]. Die Wahl gegenüber anderen Aktivierungsfunktionen kommt somit natürlich.

Auch in der Theorie benutzen wir zwei wichtige Eigenschaften der ReLU Funktion um

[5] [13, S. 118]

unser Haupttheorem 1 zu beweisen. Eine nützliche Besonderheit der Funktion ist, dass die ReLU Funktion eine Projektion ist, d.h. es gilt

$$\sigma \circ \sigma = \sigma. \tag{7}$$

Wir können somit ein Signal durch mehrere Schichten senden, ohne es zu verändern. Diese Eigenschaft wird später wichtig sein für den Beweis von Theorem 1, da der Beweis auf Grundlage der Approximationstheorie aufgebaut ist. Die Approximationstheorie wiederum basiert auf der Konstruktion von kleineren Netzwerken für einfache Aufgaben, die nicht unbedingt dieselbe Netzwerktiefe haben. Um nun Teilnetzwerke zu verbinden, müssen alle dieselbe Netzwerktiefe haben. Wir realisieren dies durch Hinzufügen von hidden Layers, die den Output nicht verändern. Eine einfache Methode wäre es die Gewichtsmatrizen in den hinzugefügten Layers als Einheitsmatrix zu initialisieren (angenommen wir haben gleiche Netzwerkbreite in aufeinanderfolgenden Layers) und somit wird jetzt die Eigenschaft (7) benutzt, um die Aktivierungswerte, die durch die hinzugefügten Layer gesendet werden, nicht zu verändern. Im Abschnitt 3.3.1 werden wir näher auf das Verbinden von Netzwerken eingehen. Tatsächlich wird diese Methode auch in der Praxis angewendet, beispielsweise machen Residuale Netzwerke, auch ResNet genannt, davon Gebrauch. Einer der Effekte ist, dass dadurch das Lernen beschleunigt wird, wodurch Residuale Netzwerke anderen Netzwerken überlegen sind, vgl. [8].

Außerdem können wir mit der ReLU Aktivierungsfunktion, die Netzwerkparameter mit eins im Betrag beschränken. Falls alle Netzwerkparameter im Intervall $[-1, 1]$ initialisiert werden, müssen die Netzwerkparameter im Training nur mit höchstens zwei variiert werden. Im nächsten Abschnitt werden wir motivieren, wieso es sinnvoll ist, die Netzwerkparameter mit eins im Betrag zu beschränken.

2.2.2 Netzwerkparameter

Mit der Zeit wurden die neuronalen Netzwerke immer tiefer, vgl. [18], womit unsere Anzahl an hidden Layers L dementsprechend hoch ist. Ein Grund für das Wachstum ist, dass man dadurch häufig bessere prognostische Treffsicherheit erlangt, vgl. [3]. Wir haben somit eine große Menge an potentiellen Netzwerkparametern, weshalb wir die Anzahl an Netzwerkparametern nicht einschränken werden.

Zusätzlich nehmen wir an, dass wir mehr Netzwerkparameter haben als Trainingsdaten. Diese Annahme scheint auf den ersten Blick eine starke Einschränkung zu sein, jedoch beobachten wir in der Praxis genau solche Netzwerke. Ein Beispiel wäre das bekannte AlexNet, vgl. [11], ein neuronales Netzwerk für die Objekterkennung in digitalen Bildern. Das Netzwerk besitzt 60 Millionen Netzwerkparameter, aber wurde nur auf 1.2 Millionen Trainingsdaten trainiert. Das Netzwerk erreicht trotz der Differenz an Netzwerkparameter und Trainingsdaten erstaunliche Resultate.

Eine weiter wichtige Eigenschaft der Netzwerkparameter, die wir berücksichtigen müssen, ist die Größe des Wertes der Parameter. Frühere statistische Resultate haben öfters die Größe der Werte der Netzwerkparameter nicht eingeschränkt, d.h. in den Resultaten steigen diese mit den Stichprobenumfang an und können somit bis unendlich wachsen, um eine bestimmte Genauigkeit zu erzielen. Falls wir nämlich erlauben, dass die Netzwerkparameter beliebig groß werden können, dann können wir die Indikatorfunktion mit

$$x \mapsto \sigma(ax) - (ax - 1) \tag{8}$$

beliebig genau approximieren, indem wir a groß wählen.

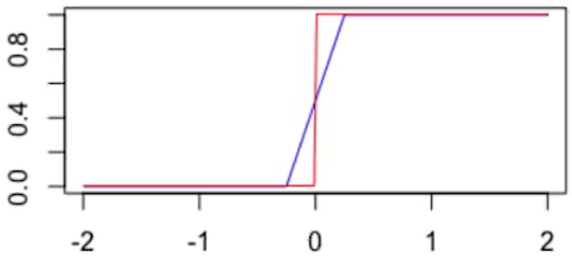

Abbildung 8: Die rote Kurve ist die Indikatorfunktion. Die blaue Kurve entsteht durch (8) und kann für große a die Indikatorfunktion beliebig gut approximieren.[6]

Durch diesen „Trick“ kann die Approximationstheorie viele verschiedene Aussagen treffen, vgl. [17].
Natürlich stimmen diese Annahmen nicht mit der Praxis überein, denn dort sind die Werte typischerweise nicht sehr groß. Der Grund ist dabei, dass falls zwei aufeinanderfolgende Layers die selbe Dimension haben, werden Gewichtsmatrizen W_j häufig als orthogonale Matrizen initialisiert, vgl. [6], Abschnitt 8.4. Nach Definition einer orthogonalen Matrix folgt, dass alle Einträge der Matrix im Betrag kleiner sind als eins und da nun die trainierten Parameter meistens nicht weit von den initialisierten Werten liegen, sind die trainierten Gewichtsmatrizen folglich im Wert nicht groß. Wir werden somit unsere Netzwerkparameter im Betrag kleiner 1 halten, indem wir im Lernalgorithmus die Netzwerkparameter in jeder Iteration auf $[-1, 1]$ projizieren. Wir erhalten die Klasse der Netzwerkfunktionen

$$F(L, \mathbf{p}) := \{f \text{ in der Form (6): } \max_{j=0,\dots,L} \|W_j\|_\infty \vee |\mathbf{v}_j|_\infty \leq 1\}$$

mit $\mathbf{v}_0$ als Nullvektor definiert und $\|W_j\|_\infty$ bezeichnet die Maximumsnorm von W_j.

[6] [17, Folie 7]

2.2.3 Dünnbesetzte Parameter

Ein Problem, mit dem jedes statistische Verfahren zu kämpfen hat, sind verrauschte Daten und das damit verbundene Problem der Überanpassung an die zufällige Streuung der Daten. Bei der Überanpassung modellieren wir das „Rauschen“ der Messdaten mit und würden dann für neue Testdaten, die unser Modell noch nicht gesehen hat, keine realistische Prognose abgeben. Wir können zwar mit größeren Netzwerken komplexere Aufgaben lösen, haben jedoch dadurch auch eine Neigung zum Overfitting, vgl. [23, S. 21]. Da wir hier tiefe neuronale Netzwerke betrachten, haben wir zwangsläufig ein großes Netzwerk. Wir laufen Gefahr der Überanpassung.

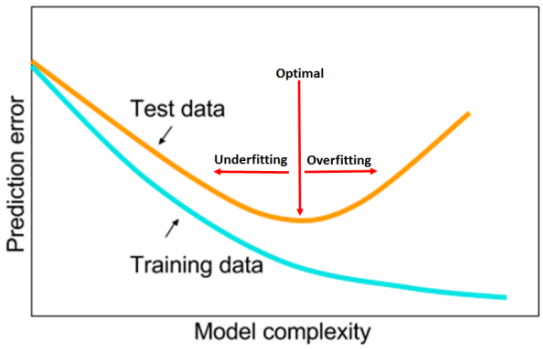

Abbildung 9: Problem des Overfittings durch tiefe neuronale Netzwerke ohne Regulierungen.[7]

Ein häufiger Lösungsansatz ist, dünnbesetzte Netzwerkparameter einzuführen, d.h. viele der Parametereinträge werden auf 0 (oder nah bei 0) gesetzt. Es soll dazu führen, dass die Komplexität des Modells vereinfacht wird. Eine Möglichkeit dies Umzusetzen wäre beispielsweise die Nutzung der ReLU Aktivierungsfunktion, wie wir schon im Abschnitt 2.2.1 gesehen haben, oder einer Regulierung, wie zum Beispiel die L^1-Regulierung, vgl. [6, S. 230].

In dieser Arbeit werden wir entsprechend der Praxis annehmen, dass wir dünnbesetzte Parameter haben. Falls $\|W_j\|_0$ die Anzahl der Einträge ungleich 0 von W_j bezeichnet, dann definieren wir ein *s-sparse* Netzwerk als

$$\begin{aligned} F(L, \mathbf{p}, s) &:= F(L, \mathbf{p}, s, F) \\ &:= \{f \in F(L, p) : \sum_{j=0}^{L} \|W_j\|_0 + |\mathbf{v}_j|_0 \leq s, \||f|_\infty\|_\infty \leq F\}, \end{aligned}$$

[7] Smith, Leslie N.: *A disciplined approach to neural network hyper-parameters: Part 1 - Learning rate, batch size, momentum and weight decay.* arXiv:1803.09820. April 2018.

wobei die obere Grenze der Supremumsnorm von f meistens überflüssig ist, weshalb man diese in der Notation weglässt. Wir betrachten also alle Netzwerkfunktionen $f \in F(L, \mathbf{p})$, deren Anzahl an nicht verschwindende Einträge der Netzwerkparameter höchstens s beträgt. Wir werden uns auf Netzwerke „beschränken“, bei denen s klein ist in Relation zu der Gesamtanzahl an Parametern im Netzwerk. Wie wir aber schon oben motiviert haben, werden in der Praxis durch Regulierungen und ReLU Funktionen *s-sparse* Netzwerke auftreten mit relativ zu Gesamtanzahl an Parametern kleinen s.

2.2.4 Hierarchische Komposition der Regressionsfunktion

Wir nehmen eine β-glatte Regressionsfunktion $f_0 : [0,1]^d \to \mathbb{R}$ an. Durch Reskalierung des Intervalls $[0,1]^d$ können alle dargestellten Ergebnisse auf jede kompakte Menge $\mathbf{X} \subset \mathbb{R}^d$ erweitert werden. Es gilt jedoch für β-glatte Funktion, wie wir schon im Abschnitt 1.3 gesehen haben, eine Minimax-Konvergenzrate von $n^{-2\beta/2\beta+d}$ für den L_2-Fehler, wobei d die Dimension des Inputs ist. Im Abschnitt 1.3 haben wir schon motiviert, dass wir unter diesen Annahmen keine schnellen Konvergenzraten erreichen können.

In der Praxis beobachten wir jedoch was anderes: Neuronale Netzwerke scheinen den Fluch der Dimensionalität zu umgehen. Die vorherige, sehr allgemeine Modellannahme scheint damit nicht plausibel zu sein. Die Funktionsklassen, die wir schätzen wollen, müssen also viel kleiner sein als angenommen, vgl. [12, S. 4].

Es müssen zusätzliche strukturelle Annahmen an die Regressionsfunktion getroffen werden, die uns erlauben schnellere Konvergenzraten zu erreichen, um somit die Ergebnisse der Praxis zu erklären. Die Annahmen sollten dabei bei Problemen, wo neuronale Netzwerke tatsächlich bessere Ergebnisse zeigen, zutreffen. Eine heuristische Idee ist, dass neuronale Netzwerke bei Problemen, wo es möglich ist, komplexe Objekte durch einfache Objekte in einer iterativen Weise aufzubauen, gut performen. So eine Struktur nennt man auch „hierarchische Struktur“. Diese Idee kann aus der Physik, Mathematik und aus der Neurowissenschaft heuristisch begründet werden, vgl [21, 22].

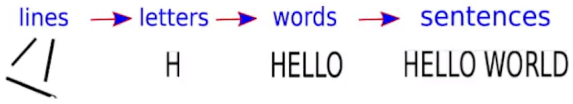

Abbildung 10: Beispiel einer hierarchischen Struktur: Aus Linien werden Buchstaben gebaut, aus Buchstaben werden Wörter geformt und zum Schluss werden die Wörter zu Sätzen zusammengesetzt.[8]

[8] [17, Folie 13]

Mathematisch können wir diese hierarchische Struktur formulieren als eine Regressionsfunktion f_0, die aus einer Komposition von Funktionen besteht, das heißt

$$f_0 = g_q \circ g_{q-1} \circ ... \circ g_1 \circ g_0 \tag{9}$$

mit $g_i : [a_i, b_i]^{d_i} \to [a_{i+1}, b_{i+1}]^{d_{i+1}}$, wobei $[a_i, b_i]$ ein Intervall ist. Wir bezeichnen die einzelnen Komponenten von g_i mit $g_i = (g_{ij})_{j=1,...,d_{i+1}}^T$ und t_i als die maximale Anzahl an Variablen, an denen die einzelnen g_{ij} abhängen. Die g_{ij} sind somit t_i-variate Funkionen. Ein einfaches Beispiel wäre eine Bivariate Funktion $f(x, y)$, die nur von 2 Variablen abhängt. Natürlich muss immer $t_i \leq d_i$ gelten. Die fehlende Eindeutigkeit der Darstellung (9) schadet uns nicht, da wir nicht interessiert sind, wie genau $g_0, ..., g_q$ und die Paare (β_i, t_i) aussehen, sondern wir werden lediglich die Eigenschaft ausnutzen, dass f_0 aus Kompositionen von Funktionen besteht, um damit letztendlich f_0 zu schätzen. Um zu verstehen, wie die einzelnen Dimensionen interagieren, schauen wir uns das folgende Beispiel an und ermitteln dabei d_i und t_i:

$$f_0(\underbrace{x_1, x_2, x_3}_{d_o=3}) = \underbrace{g_{11}(g_{01}(\underbrace{x_3}_{t_{01}=1}), g_{02}(\underbrace{x_2}_{t_{02}=1}))}_{t_1=2} = \underbrace{g_1(\underbrace{x_2, x_3}_{d_1=2})}_{d_2=1}.$$

Die maximale Anzahl an Variablen, an denen g_{01} und g_{02} abhängen, sind $t_{01} = 1$ und $t_{02} = 1$. Daraus folgt, dass $t_0 = 1$ gilt.
In unserem d-variaten Regressionsmodel (1) $f_0 : [0, 1]^d \to \mathbb{R}$ gilt offensichtlich $d_0 = d,\ a_0 = 0,\ b_0 = 1,$ und $d_{q+1} = 1$. Offensichtlich gibt es mehrere Möglichkeiten die Regressionsfunktion in einer Komposition zu schreiben, jedoch sollte man unter allen Möglichkeiten der Darstellung von f_0 immer eine auswählen, die die schnellste Konvergenzrate in Theorem 1 ermöglicht.

Einer der klassischen Ansätze der nichtparametrischen Statistik ist es anzunehmen, dass die Funktionen g_{ij} in der Hölderklasse sind mit Glattheitsindex β_i liegt. Die Definition der Hölderklasse mit Glattheitsindex β lautet:

Definition 1. *[7, S. 37]*
Sei $\beta = q + s$ für ein $q \in \mathbb{N}_0$ und $0 < s \leq 1$. Eine Funktion $f : \mathbb{R}^d \to \mathbb{R}$ liegt in der Hölderklasse mit Glattheitsindex β, falls für jedes $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_d) \in \mathbb{N}_0^d$ mit $\sum_{j=1}^{d} \alpha_j = q$ die partielle Ableitung $\left| \frac{\partial^q m}{\partial x_1^{\alpha_1} ... \partial x_d^{\alpha_d}}(\mathbf{x}) - \frac{\partial^q m}{\partial x_1^{\alpha_1} ... \partial x_d^{\alpha_d}}(\mathbf{z}) \right| \leq C \cdot \|\mathbf{x} - \mathbf{z}\|^s$ für alle $\mathbf{x}, \mathbf{z} \in \mathbb{R}^d$, wobei $\|\cdot\|$ die euklidische Norm ist.

Eine Funktion liegt also in der Hölderklasse mit Glattheitsindex β, falls alle partiellen Ableitungen bis zum Grad $\lfloor \beta \rfloor$ existieren, beschränkt sind und die partiellen Ableitungen

mit Grad $\lfloor\beta\rfloor$ sind $\beta - \lfloor\beta\rfloor$ Hölder-stetig, wobei $\lfloor\beta\rfloor$ die größte ganzzahlige Zahl ist, die strikt kleiner als β ist. Der Ball der β-Hölder Funktionen mit Radius K ist definiert als

$$C_r^\beta(D,K) = \Bigg\{ f : D \subset \mathbb{R}^r \to \mathbb{R} : \sum_{\boldsymbol{\alpha}:|\boldsymbol{\alpha}|<\beta} \|\partial^{\boldsymbol{\alpha}} f\|_\infty + \sum_{\boldsymbol{\alpha}:|\boldsymbol{\alpha}|=\lfloor\beta\rfloor} \sup_{\substack{\mathbf{x},\mathbf{y}\in D \\ \mathbf{x}\neq\mathbf{y}}} \frac{|\partial^{\boldsymbol{\alpha}} f(\mathbf{x}) - \partial^{\boldsymbol{\alpha}} f(\mathbf{y})|}{|\mathbf{x}-\mathbf{y}|_\infty^{\beta-\lfloor\beta\rfloor}} \leq K \Bigg\},$$

wobei $\partial^{\boldsymbol{\alpha}} = \partial^{\alpha_1} ... \partial^{\alpha_r}$ ein Multi-Index mit $\boldsymbol{\alpha} = (\alpha_1, ..., \alpha_r) \in \mathbb{N}^r$ und $|\boldsymbol{\alpha}| := |\boldsymbol{\alpha}|_1$.

Da nun die g_{ij} Hölder Glatt mit Index β_i und t_i-variate Funktionen sind, gilt $g_{ij} \in C_{t_i}^{\beta_i}([a_i,b_i]^{t_i}, K_i)$. Wir nehmen somit an, dass unsere Regressionsfunktion f_0 aus einer Komposition von Funktionen in der Klasse

$$\begin{aligned} G(q,\mathbf{d},\mathbf{t},\boldsymbol{\beta},K) := \{ f_0 = g_q \circ ... \circ g_0 : g_i = (g_{ij})_j : [a_i,b_i]^{d_i} \to [a_{i+1},b_{i+1}]^{d_{i+1}}, \\ g_{ij} \in C_{t_i}^{\beta_i}([a_i,b_i]^{t_i}, K), \text{ für beliebige } |a_i|, |b_i| \leq K \} \end{aligned} \tag{10}$$

mit $\mathbf{d} := (d_0, ..., d_{q+1}),\ \mathbf{t} := (t_0, ..., t_q),\ \boldsymbol{\beta} := (\beta_0, ..., \beta_q)$ besteht.

2.3 Glattheit einer kompositionalen Funktion

Für die Konvergenzrate in der nichtparametrischen Regression spielt die Glattheit der Funktion f_0 eine wichtige Rolle. Wir müssen demzufolge die Glattheit von f_0 berechnen, die wiederum durch die Glattheit der Funktionen g_i induziert wird, da f_0 aus einer Komposition der Funktionen g_i besteht. Aus [9] können wir entnehmen, wie die Glattheit einer kompositionalen Funktion induziert wird. Die Konvergenz eines Netzwerkschätzers hängt dementsprechend von dem sogenannten effektiven Glattheitsindex

$$\beta_i^* := \beta_i \prod_{l=i+1}^{q} (\beta_l \wedge 1), \qquad (\beta_l \wedge 1) := \min(\beta_l, 1) \tag{11}$$

ab und wird beschrieben durch die Konvergenzrate

$$\phi_n := \max_{i=0,...q} n^{-\frac{2\beta_i^*}{2\beta_i^* + t_i}}. \tag{12}$$

Wir betrachten dazu ein kleines Beispiel mit $q = 1,\ \beta_0, \beta_1 \leq 1,\ d_0 = d_1 = t_0 = t_1 = 1$, dann gilt $f = g_1 \circ g_0$. Die Funktion f hat somit nach (11) einen Glattheitsindex von $\beta_0\beta_1$. Die Konvergenzrate liegt dann bei mindestens $n^{-2\beta_0\beta_1/(2\beta_0\beta_1+1)}$. Falls $\beta_1 > 1$ gilt, dann folgt $\beta_0^* = \beta_0(\beta_1 \wedge 1) = \beta_0$. Die Konvergenzrate liegt nun bei mindestens $n^{-2\beta_0/(2\beta_0+1)}$.

2.4 Empirisches Risiko

Im Deep Learning verwendet man Varianten von stochastic gradient descent (SDG) in Verbindung mit anderen Methoden, um die Verlustfunktion, die durch die log-likelihood induziert wird, zu minimieren, vgl. [6], Abschnitt 6.2.1.1. Für nichtparametrische Regressionsfunktion mit normalverteilten Fehler ϵ entspricht dies der Methode der kleinsten Quadrate, vgl. [6, S.129]. Das Ziel ist es also, gegeben den Daten $D_n = (\mathbf{X}_1, Y_1, ..., \mathbf{X}_n, Y_n)$, wobei $(\mathbf{X}, Y), (\mathbf{X}_1, Y_1), ..., (\mathbf{X}_n, Y_n)$ u.i.v. Zufallsvariablen sind, eine Netzwerkfunktion f zu konstruieren, sodass das empirische Risiko $\frac{1}{n}\sum_{i=1}^{n}(Y_i - f(\mathbf{X}_i))^2$ minimal ist. Wir haben schon im Abschnitt 1 gesehen, dass für den L_2-Risiko

$$\mathbb{E}(|f_0(\mathbf{X} - Y|^2) = \inf_{f'} \mathbb{E}(|f' - Y|^2)$$

gilt, wobei f_0 die Regressionsfunktion und das Infinimum über alle messbaren Funktionen $f' : \mathbb{R}^d \to \mathbb{R}$ genommen wird. Da wir aber nicht die Regressionsfunktion berechnen können, minimieren wir das empirische Risiko, welche den L_2-Risiko schätzt, um die Regressionsfunktion zu berechnen. Die Verlustfunktion entspricht somit einer quadratischen Verlustfunktion und wir erhalten den L_2-Fehler

$$R(\widehat{f}_n, f_0) = \mathbb{E}(|\widehat{f}_n(\mathbf{X}) - f_0(\mathbf{X})|^2)$$

als Maß der statistischen Performance, wie wir schon im Abschnitt 1 motiviert haben, eines Schätzers $\widehat{f}_n$, der in der Netzwerkklasse $F(L, \mathbf{p}, s, F)$ liegt.
Für einen beliebigen Schätzer $\widehat{f}_n \in F(L, \mathbf{p}, s, F)$ definieren wir

$$\Delta_n(\widehat{f}_n, f_0) := \mathbb{E}_{f_0}\left[\frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{f}_n(\mathbf{X}_i))^2 - \inf_{f \in F(L,\mathbf{p},s,F)} \frac{1}{n}\sum_{i=1}^{n}(Y_i - f(\mathbf{X}_i))^2\right]. \tag{13}$$

Die Folge misst die Differenz zwischen dem erwarteten empirischen Risiko von $\widehat{f}_n$ und dem globalen Minimum über alle Netzwerkfunktionen in der Klasse. Wir müssen den Term Δ_n in unserem Theorem berücksichtigen, da stochastic gradient descent Methoden nur eine kleine Chance haben das globale Minimum zu erreichen. Es ist viel wahrscheinlicher, dass man in einem lokalen Minimum bzw. Sattelpunkt stecken bleibt.

3 Hauptresultat

Als Erinnerung: Wir untersuchen ein d-variates nichtparametrisches Regressionsmodel (1), dessen Regressionsfunktion f_0 in der Klasse $G(q, \mathbf{d}, \mathbf{t}, \boldsymbol{\beta}, K)$, definiert in (10), liegt. Wir betrachten hierbei einen Schätzer $\widehat{f}_n$, der aus der Netzwerkklasse $F(L, \mathbf{p}, s, F)$ stammt. Unser Ziel ist es die Minimax-Konvergenzrate für den L_2-Fehler $R(\widehat{f}_n, f_0) = \mathbb{E}_{f_0}[(\widehat{f}_n(\mathbf{X}) - f_0(\mathbf{X}))^2]$ aus (3) mit $\mathbf{X} \overset{D}{=} \mathbf{X_1}$ unabhängig von einer gegebenen Stichprobe $(\mathbf{X}_i, Y_i)_i$ zu folgern. Der

Index f_0 in $\mathbb{E}_{f_0}$ bezeichnet hierbei die Betrachtung des Erwartungswert, bezogen auf ein generierte Stichprobe aus dem Modell der nichtparametrischen Regression mit der Regressionsfunktion f_0.

Wir werden zunächst eine obere Schranke im Abschnitt 3.1 für den L_2-Fehler für Schätzer aus der Netzwerkklasse $F(L, \mathbf{p}, s, F)$ angeben und im Abschnitt 3.3 beweisen. Im Abschnitt 3.1.1 werden wir Folgerungen der oberen Schranke aufführen, wie zum Beispiel das Umgehen des Fluches der Dimensionalität. Im Abschnitt 3.2 werden wir eine untere Schranke für den L_2-Fehler, bewiesen in [16], angeben. Falls diese beiden Schranken identisch sind, können wir daraus die Minimax-Konvergenzrate folgern. Wir werden jedoch später sehen, dass unsere Schranken um den Faktor $L\log^2(n)$ abweichen, es fehlt uns damit noch an Präzision.

3.1 Obere Schranke des L_2-Fehler

Theorem 1. *[15, S. 219]*
Betrachte ein d-variates nichtparametrisches Regressionsmodell (1), wobei die Regressionsfunktion eine Komposition aus Funktionen besteht, wie in (9) definiert, und dabei in der Klasse $G(q, \mathbf{d}, \mathbf{t}, \boldsymbol{\beta}, K)$ liegt. Sei nun $\widehat{f}_n$ ein Schätzer, der Funktionen in der Netzwerkklasse $F(L, (p_i)_{i=0,\dots,L+1}, s, F)$ schätzt, wobei die Netzwerkklasse die folgenden Bedingungen erfüllt:

(i) Die zur Schätzung verwendeten neuronalen Netzwerke erlauben Funktionswerte, die mindestens so groß sind wie die maximalen Funktionswerte der Regressionsfunktion f_0: $F \geq \max(K, 1)$

(ii) Für die Anzahl der Layer soll gelten: $\sum_{i=0}^{q} \log_2(4t_i \vee 4\beta_i)\log_2 n \leq L \lesssim n\phi_n$

(iii) Die Größe der Layer muss mindestens mit Rate $n\phi_n$ in n gegen unendlich gehen: $n\phi_n \lesssim \min_{i=1,\dots,L} p_i$

(iv) Anzahl der nicht verschwindende Einträge der Gewichtsmatrizen und Verschiebungsvektoren muss mit Rate $n\phi_n \log(n)$ in n gegen unendlich gehen: $s \asymp n\phi_n \log n$

Dann existieren Konstanten C und C', die nur abhängen von q, $\boldsymbol{d}$, $\boldsymbol{t}$, $\boldsymbol{\beta}, F$, sodass, wenn $\Delta_n(\widehat{f}_n, f_0) \leq C\phi_n L\log^2(n)$ gilt, dann

$$R(\widehat{f}_n, f_0) \leq C'\phi_n L\log^2(n) \tag{14}$$

und falls $\Delta_n(\widehat{f_n}, f_0) \geq C\phi_n L\log^2(n)$, dann

$$\frac{1}{C'}\Delta_n(\widehat{f}_n, f_0) \leq R(\widehat{f}_n, f_0) \leq C'\Delta_n(\widehat{f}_n, f_0). \tag{15}$$

Die beste Wahl für L um die Rate $\phi_n L \log^2(n)$ zu minimieren, ist es L proportional zu $\log_2(n)$ zu wählen. Die Anzahl der Layer sollte also mit Rate $\log_2(n)$ in n gegen unendlich gehen, d.h. $L \asymp \log_2(n)$. Es folgt $\Delta_n(\widehat{f}_n, f_0) \leq C\phi_n \log^3(n)$ und damit auch

$$R(\widehat{f}_n, f_0) \leq C'\phi_n \log^3(n).$$

Es findet in der Konvergenzrate ϕ_n ein Tradeoff statt: Falls t_i groß ist, kann dies durch entsprechend großes β_i^* kompensiert werden [15, S.219].

3.1.1 Folgerungen aus Theorem 1

Es ergibt sich aus Theorem 1, dass die Konvergenzrate des L_2-Fehlers, unter den selben Annahmen, von ϕ_n und $\Delta_n(\widehat{f}_n, f_0)$ abhängen. Aus dem Abschnitt 3.2 können wir entnehmen, dass ϕ_n die untere Grenze des Minimax des L_2-Fehlers über der selben Funktionsklasse, wie aus Theorem 1, ist. Aus $\phi_n := \max_{i=0,\dots q} n^{-\frac{2\beta_i^*}{2\beta_i^* + t_i}}$ können wir sehen, dass die Rate nicht mehr von der ursprünglichen Inputdimension d abhängt, sondern von den t_i, die gegebenenfalls viel kleiner als d sein können. Die t_i können somit auch als die effektive Dimension angesehen werden, die es ermöglichen schnellere Konvergenzraten zu erreichen. Aufgrund dessen umgeht das Netzwerk den Fluch der Dimensionalität für bestimmte Strukturen von f_0.
Zusätzlich kommt $\Delta_n(\widehat{f}_n, f_0)$ in der unteren Grenze in (15) vor und ist somit in der Konvergenzrate unvermeidlich. Der Ausdruck $\Delta_n(\widehat{f}_n, f_0)$ nimmt einen großen Wert an, falls $\widehat{f}_n$ ein großes empirisches Risiko im Vergleich zum Minimierer des empirischen Risikos aufweist. Falls jedoch $\widehat{f}_n$ ein Minimierer des empirischen Risikos ist, gilt $\Delta_n = 0$ nach Definition. Die Bedingung $\Delta_n(\widehat{f}_n, f_0) \leq C\phi_n L \log^2(n)$ aus Theorem 1 ist in diesem Fall nun trivialerweise erfüllt, da $\Delta_n(\widehat{f}_n, f_0) = 0 \leq C\phi_n L \log^2(n)$. Es folgt das Korollar 1.

Korollar 1. *Sei* $\widetilde{f}_n \in \arg\min_{f \in F(L,\mathbf{p},s,F)} \sum_{i=1}^n (Y_i - f(\mathbf{X}_i))^2$ *der Minimierer des empirischen Risikos. Unter den selben Bedingungen, wie im Theorem 1, existiert eine Konstante* C'*, die nur von* $q, \mathbf{d}, \mathbf{t}, \boldsymbol{\beta}, F$ *abhängt, sodass*

$$R(\widetilde{f}_n, f_0) \leq C'\phi_n L \log^2(n).$$

Falls wir also ein Minimierer des empirischen Risikos finden, dann hätten wir eine obere Schranke für den L_2-Fehler von $\phi_n L \log^2(n)$. Leider findet man mit den Methoden, die wir heutzutage nutzen, wie beispielsweise stochastic gradient Methoden, nicht zuverlässig den Minimierer. Es wäre somit von großem Interesse eine Methode kennen zu lernen mit der man konstant ein Minimierer des empirischen Risikos finden könnte um so den Term Δ_n in der Konvergenzrate zu umgehen. Als nächstes wollen wir uns die Bedingungen des Theorem 1 näher anschauen.

Die Kondition (i) aus Theorem 1 ist eine sehr milde Bedingung an die Netzwerkfunktion, da sie nur aussagt, dass die Funktion mindestens die selbe Supremumsnorm haben soll wie die Regressionsfunktion.
Aus der Bedingung (*ii*) sehen wir, dass es für die Wahl der Netzwerktiefe L ausreicht, eine obere Grenze für $t_i \leq d_i$ und den Glattheitsindex β_i zu finden. Wir haben schon gesehen, dass die beste Wahl für die Anzahl der hidden Layers $L \asymp \log_2(n)$ ist. Die Tiefe des neuronalen Netzwerkes sollte somit mit den Stichprobenumfang wachsen.
Die Netzwerkbreite $\mathbf{p}$, die in der Bedingung (*iii*) beschränkt wird, kann man unabhängig von den Glattheitsindizes wählen, indem man anstatt $n\phi_n \lesssim \min_{i=1,\dots,L} p_i$ die Bedingung $n \lesssim \min_{i=1,\dots,L} p_i$ erfüllt. Es gilt nämlich $\phi_n \leq 1$ und somit folgt $n\phi_n \lesssim n \lesssim \min_{i=1,\dots,L} p_i$, falls $n \lesssim \min_{i=1,\dots,L} p_i$ gilt. Man könnte zum Beispiel den Stichprobenumfang n als Breite für jede hidden Layer wählen.
Aus der Bedingung *(iv)* können wir folgern, dass wir ein *sparse* Netzwerk vorliegen haben müssen. Der Grund liegt dabei, dass in einem *fully connected* Netzwerk die Anzahl an Gewichtsparameter $\sum_{i=0}^{L} p_i p_{i+1}$ und für die Verschiebungsparameter $\sum_{l=1}^{L} p_L$ beträgt. Die Größenordnung der Netzwerkparameter liegt somit bei $\sum_{i=0}^{L} p_i p_{i+1}$. Es gilt aber

$$\sum_{i=0}^{L} p_i p_{i+1} \geq (L-1) \min_{i=1,\dots,L} p_i^2$$

und mit den Bedingungen *(iii), (ii)* gilt

$$\sum_{i=0}^{L} p_i p_{i+1} \gtrsim \log(n)(n\phi_n)^2 - (n\phi_n)^2$$

und damit kann die Bedingung *(iv)* nicht erfüllt sein. Das zeigt, dass Theorem 1 *sparse* Netzwerke voraussetzt. Es gilt sogar, dass das Netzwerk mindestens $\sum_{i=1}^{L} p_i - s$ inaktive Knoten haben muss, dass heißt alle eingehenden dieser Knoten sind Null. Die Aussage kann man mit folgender Überlegung begründen: Mit einer festen Anzahl an aktiven Netzwerkparameter s erreichen wir die höchste Anzahl an aktiven Knoten, indem wir keine aktiven Verschiebungsparameter, sondern nur aktive Gewichtsparameter ins Netzwerk aufnehmen. Wir haben somit s aktive Gewichtsparameter. Um nun möglichst viele aktive Knoten zu generieren, verbinden wir nur zu inaktiven Knoten bis wir keine Gewichtsparameter oder keine inaktiven Knoten mehr übrig haben. Dadurch erreicht man offensichtlich höchstens s aktive Knoten in den *hidden Layers*. Es folgt, dass es mindestens $\sum_{i=1}^{L} p_i - s$ inaktive Knoten gibt.
Die Wahl $s \asymp n\phi_n \log(n)$ in Bedingung (*iv*) balanciert den quadrierten Bias und die Varianz. Aus dem Beweis von Theorem 1 kann man entnehmen, dass die Konvergenzrate auch für andere Größenordnungen von s hergeleitet werden kann.

Aus den Bedingungen in Theorem 1 können wir ableiten, dass wir eine sehr flexible Möglichkeit haben eine gute Netzwerkarchitektur zu wählen, solange die Anzahl der aktiven Parameter s die Bedingung (iv) erfüllt. Die Größe des Netzwerkes spielt somit nicht die wichtigste Rolle für die statistische Performance, sondern die richtige Regulation des Netzwerkes in Hinblick auf die Anzahl der aktiven Parameter aus Bedingung (iv)!

Der Einfachheit halber haben wir Theorem 1 ohne explizite Konstanten formuliert. Im Beweis hat man jedoch nicht versucht die Konstanten minimal zu halten, obwohl der Beweis nicht-asymptotisch durchgeführt wird. Es ist aber bekannt, dass Deep Learning nur andere Methoden leistungsmäßig übertreffen, falls man eine hohe Stichprobengröße hat. Somit ist die Nicht-Minimierung der Konstante bei hoher Stichprobengröße nicht mehr ausschlaggebend. Dies deutet darauf hin, dass die Methode zwar fähig ist sich der zugrunde liegende Struktur im Signal anzupassen und deswegen eine hohe Konvergenzrate erreicht, jedoch mit großen Konstanten oder übrige Terme, die die Resultate bei kleinen Stichproben beeinträchtigen.

Um das Theorem 1 zu beweisen, brauchen wir die folgende Ungleichung vom Orakel-Typus.

Theorem 2. *Betrachte ein d-variates nichtparametrische Regressionsmodel (1) mit unbekannter Regressionsfunktion f_0, die die Bedingung $\|f_0\| \leq F$ für ein beliebiges $F \geq 1$ erfüllt. Sei $\widehat{f}_n$ ein beliebiger Schätzer, der Funktionen in der Netzwerkklasse $F(L, \mathbf{p}, s, F)$ schätzt, und sei $\Delta_n(\widehat{f}_n, f_0)$, wie in (13) definiert. Für jedes beliebige $\epsilon \in (0, 1]$ existiert eine Konstante C_ϵ, die nur von ϵ abhängt, sodass*

$$\tau_{\epsilon,n} := C_\epsilon F^2 \frac{(s+1)\log(n(s+1)^L p_0 p_{L+1})}{n},$$

$$\begin{aligned}
&(1-\epsilon)^2 \Delta_n(\widehat{f}_n, f_0) - \tau_{\epsilon,n} \leq R(\widehat{f}_n, f_0) \\
&\leq (1+\epsilon)^2 \left(\inf_{f \in F(L,\mathbf{p},s,F)} \|f - f_0\|_\infty^2 + \Delta_n(\widehat{f}_n, f_0) \right) + \tau_{\epsilon,n}.
\end{aligned}$$

Den Beweis zum Theorem 2 findet man im Appendix von [16]. Eine Konsequenz von Theorem 2 ist, dass die obere Grenze des Risikos schlechter wird, falls die Anzahl der hidden Layer L steigt. In der Praxis beobachtet man auch solche Ergebnisse, wo zu viele hidden Layer die Performance verschlechtern. Residuale Netzwerke (ResNet) hingegen überbrücken dieses Problem, sind jedoch nicht in der Form von (6) und müssen separat analysiert werden.

Als nächstes werden wir die untere Schranke für den L_2-Fehler angeben.

3.2 Untere Schranke des L_2-Fehler

Theorem 3. *Betrachte ein d-variates nichtparametrisches Regressionsmodell (1) mit Beobachtungen* $\mathbf{X}_i$ *aus einer Wahrscheinlichkeitsverteilung mit einer Lebesque Dichte auf* $[0,1]^d$*, welche mit einer oberen und unteren positiven Konstante beschränkt ist. Für eine beliebige nicht-negative ganze Zahl q, beliebige Dimensionsvektoren* $\boldsymbol{d}$ *und* $\boldsymbol{t}$*, die* $t_i \leq \min(d_0, ..., d_{i-1})$ *für alle i erfüllen, ein beliebiger Glattheitsvektor* $\boldsymbol{\beta}$ *und alle hinreichend großen Konstanten* $K > 0$*, existiert eine positive Konstante c, sodass*

$$\inf_{\widehat{f}_n} \sup_{f_0 \in G(q,\mathbf{d},\mathbf{t},\boldsymbol{\beta},K)} R(\widehat{f}_n, f_0) \geq c\phi_n,$$

wobei das inf *über alle Schätzer* $\widehat{f}_n$ *genommen wird.*

Der Beweis zu Theorem 3 befindet sich im Appendix von [16] und wird hier nicht weiter diskutiert.
Wir sehen durch das Theorem 3, dass ϕ_n eine untere Schranke für den Minimax L_2-Fehler über der Funktionsklasse $G(q, \mathbf{d}, \mathbf{t}, \boldsymbol{\beta}, K)$ ist. Wir erinnern uns, dass das Theorem 1 uns eine obere Schranke für den L_2-Fehler für einen Schätzer aus der Netzwerkklasse $F(L, \mathbf{p}, s, F)$ über der Klasse $G(q, \mathbf{d}, \mathbf{t}, \boldsymbol{\beta}, K)$ angibt. Falls nun beide Schranken übereinstimmen würden, könnten wir aus den beiden (identischen) Schranken herleiten, dass Schätzer aus dünnbesetzten tiefen neuronalen Netzwerken die Minimax-Konvergenzrate erreichen. Da wir für den L_2-Fehler aber einerseits eine untere Schranke mit ϕ_n und andererseits eine obere Schranke für Schätzer aus $F(L, \mathbf{p}, s, F)$ mit $\phi_n L \log^2(n)$ haben, womit unsere Schranken nicht übereinstimmen, besitzen wir hier noch eine Lücke in unserer Theorie. Es könnte somit sein, dass unsere untere Schranke aus Theorem 3 oder unsere obere Schranke aus Theorem 1 zu ungenau sind. Johannes Schmidt-Hieber vermutet, dass der Faktor $L \log^2(n)$ aus der oberen Schranke vom Theorem 1 ein Artefakt vom Beweis ist.
Wir können aus Theorem 1 und 3 eine Minimax-Konvergenzrate angeben, die bis zu $L \log^2(n)$ abweicht.

3.3 Beweise

3.3.1 Einbettungseigenschaften einer Netzwerkfunktionsklasse

In diesem Abschnitt seien $\mathbf{p} = (p_0, ..., p_{L+1})$ und $\mathbf{p'} = (p'_0, ..., p'_{L+1})$.
Um eine Funktion durch ein Netzwerk zu approximieren, müssen wir im ersten Schritt Netzwerke konstruieren, die jeweils einfache Aufgaben bewältigen. Um nun zwei Netzwerke zu verbinden, benutzen wir die folgende Regeln:

Größenvergleich: Es gilt $F(L, \mathbf{p}, s) \subseteq F(L, \mathbf{q}, s')$, falls $\mathbf{p} \leq \mathbf{q}$ komponentenweise gilt und $s \leq s'$.

Komposition: Es seien $f \in F(L, \mathbf{p})$ und $g \in F(L', \mathbf{p}')$ zwei Netzwerke mit derselben Anzahl an $p_{L+1} = p'_0$. Für einen Vektor $\mathbf{v} \in \mathbb{R}^{p_{L+1}}$ definieren wir die Komposition der Netzwerke als $g \circ \sigma_{\mathbf{v}}(f)$, die in der Netzwerkklasse $F(L + L' + 1, (\mathbf{p}, p'_1, ..., p'_{L'+1}))$ liegt. In den meisten Fällen werden wir annehmen, dass der Output des ersten Netzwerkes nicht-negativ ist und dass der Verschiebungsvektor $\mathbf{v}$ Null ist.

Layers hinzufügen/Netzwerktiefe angleichen: Um die Anzahl an hidden Layers für zwei Netzwerke anzugleichen, können wir zusätzliche Layers im Input hinzufügen, die das Signal nicht verändern. Wir erreichen dies, indem wir die Gewichtsmatrizen der hinzugefügten Layer als Einheitsmatrix initialisieren, sodass

$$F(L, \mathbf{p}, s) \subset F(L + q, (\underbrace{p_0, ..., p_0}_{q\text{-mal}}, \mathbf{p}), s + qp_0).$$

Wie wir schon im Abschnitt 2.2.1 motiviert haben, nutzen wir hier die Eigenschaft der Projektion der ReLU Aktivierungsfunktion.

Parallelisierung: Es seien f, g zwei Netzwerke mit der gleichen Anzahl an hidden Layers und gleichen Inputdimension, d.h. $f \in F(L, \mathbf{p})$ und $g \in F(L, \mathbf{p}')$ mit $p_0 = p'_0$. Das *parallele* Netzwerk (f, g) berechnet f und g simultan in einem gemeinsamen Netzwerk, welches in der Klasse $F\left(L, (p_0, p_1 + p'_1, ..., p_{L+1} + p'_{L+1})\right)$ liegt.

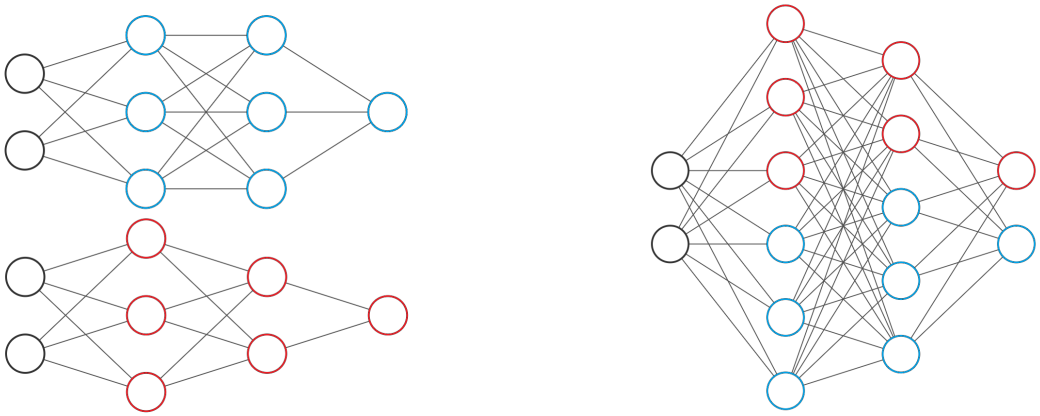

Abbildung 11: Beispiel für ein gemeinsames Netzwerk für $f \in F(2, (2, 3, 3, 1))$, $g \in F(2, (2, 3, 2, 1))$ und somit $(f, g) \in F(2, (2, 6, 5, 2))$.[9]

Beseitigung der inaktiven Knoten: Es gilt

$$F(L, \mathbf{p}, s) = F\left(L, (p_0, p_1 \wedge s, p_2 \wedge s, ..., p_L \wedge s, p_{L+1}), s\right).$$

[9]Erstellt in http://alexlenail.me/NN-SVG/LeNet.html, letzter Zugriff: 26.06.2019

Wir zeigen die Aussage, indem wir zunächst ein Netzwerk $f(x) = W_L \sigma_{\mathbf{v}_L} W_{L-1} ... \sigma_{\mathbf{v}_1} W_0 \mathbf{x} \in F(L, \mathbf{p}, s)$ betrachten. Sind alle Einträge der j-ten Spalte von W_i Null, dann können wir die Spalte zusammen mit der j-ten Zeile von W_{i-1} und dem j-ten Eintrag von $\mathbf{v}_i$ entfernen ohne die Funktion zu verändern. Das zeigt, dass $f \in F(L, (p_0, ..., p_{i-1}, p_i - 1, p_{i+1}, ..., p_{L+1}), s)$. In der Abbildung 12 haben wir eine graphische Betrachtung dieses Verfahrens.

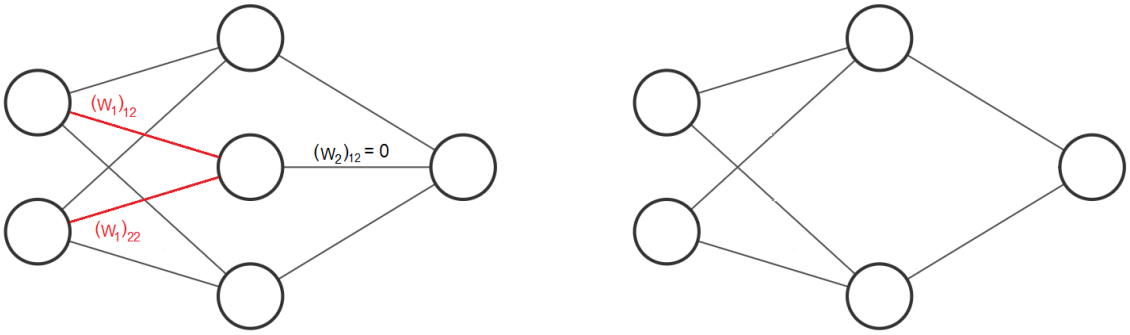

Abbildung 12: *Graphische Darstellung:* Alle ausgehenden Signale des 2. Knotens im 1. hidden Layer sind 0, d.h. die 2. Spalte von W_2 ist 0. Man erkennt, dass alle eingehenden Signale zu diesem Knoten „irrelevant“ sind, da kein Signal weitergeleitet wird. Somit können wir die 2. Zeile von W_1 und den entsprechenden Verschiebungsvektor weglassen.

Da nun höchstens s aktive Parameter existieren, haben wir mindestens $(p_i - s)$ inaktive Knoten im Layer i. Daraus lässt sich schließen, dass wir die Prozedur mindestens $(p_i - s)$-mal für jedes beliebiges $i = 1, ..., L$ iterieren können. Das zeigt $f \in F(L, (p_0, p_1 \wedge s, p_2 \wedge s, ..., p_L \wedge s, p_{L+1}), s)$.

Im Beweis werden wir Gebrauch von der Tatsache machen, dass für ein *fully connected* Netzwerk in $F(L, \mathbf{p})$ es $\sum_{l=0}^{L} p_l p_{l+1}$ Gewichtsparameter und $\sum_{l=1}^{L} p_l$ Verschiebungsparameter gibt. Insgesamt haben wir also

$$\sum_{l=0}^{L} p_l p_{l+1} - \sum_{l=1}^{L} p_l = \sum_{l=0}^{L} p_l p_{l+1} - \left(\sum_{l=0}^{L} p_{l+1} - p_{L+1} \right) = \sum_{l=0}^{L} (p_l - 1) p_{l+1} + p_{L+1}$$

Parameter.

3.3.2 Approximationsqualität neuronaler Netzwerke

Am Ende dieses Abschnittes werden die Resultate zeigen, dass wir jede Funktion, die in der Hölderklasse mit Glattheitsindex β liegen, durch ein neuronales Netzwerk geeigneter Größe annähern können. Wir werden das Theorem nicht beweisen, wollen aber eine Beweisskizze führen für mindestens einmal differenzierbare Funktionen $h : [0, 1]^d \to \mathbb{R}$, die sehr

analog läuft wie für Funktionen, die in der Hölderklasse mit Glattheitsindex β liegen. Wir werden dabei die Frage beantworten, wie viele Layer und welche Dimensionen der Layer hinreichend sind, um solche Funktionen zu approximieren.

Die Idee ist es, die Funktion h in Terme zu zerlegen, die nur aus Produkten bestehen, die wir dann durch Netzwerke annähern können. Da h mindestens einmal differenzierbar ist, besitzt h eine Taylor-Entwicklung. Die Taylor-Entwicklung ermöglicht es, die Funktion in Produkten $x_{i_1}^{\alpha_1} \cdot \ldots \cdot x_{i_1}^{\alpha_1}$ mit $\alpha_1, \ldots, \alpha_d \in \mathbb{N}_0$ zu beschreiben. Produkte zwischen r Variablen sind nur mehrmalige Anwendungen von Produkten zweier Variablen. Im ersten Schritt müssen wir also ein Netzwerk konstruieren, welches das Produkt zwischen zwei Variablen annähert, d.h. die Funktion

$$\text{mult: } [0,1]^2 \to [0,1], \quad \text{mult}(x,y) = x \cdot y$$

approximiert, gegeben den Input x und y. Falls wir nämlich dieses Problem gelöst haben, dann können wir auch Produkte zwischen r Variablen durch ein Netzwerk berechnen und damit auch die Taylor-Entwicklung der gesuchten Funktion h. Die zwei folgenden Lemmata lösen das Problem des Multiplizieren von zwei Variablen. Die Beweise der folgenden Lemmata können im Appendix von [16] nachgeschlagen werden.

Lemma 1. *[15, S.214]*
Sei $g : \mathbb{R} \to \mathbb{R},\ g(x) := x(1-x)$ *und*

$$T^k : [0, 2^{2-2k}] \to [0, 2^{-k}], \quad T^k(x) := \sigma(\frac{x}{2}) - \sigma(x - 2^{1-2k}),$$

sowie $R^k : [0,1] \to [0, 2^{-k}],\ R^k(x) := T^k \circ T^{k-1} \circ \ldots \circ T^1$. *Dann gilt*

$$\forall x \in [0,1] : \quad |\sum_{k=1}^{m} R^k(x) - g(x)| \leq 2^{-m}.$$

Das Lemma zeigt, dass wir mit der Funktion $\sum_{k=1}^{m} R^k$ die Funktion $x(1-x)$ exponentiell schnell in m approximieren können. Es gilt insbesondere $x(1-x) = \sum_{k=1}^{\infty} R^k(x)$ in $L^\infty[0,1]$. Unter Nutzung der Polarisationsgleichung

$$g\left(\frac{x-y+1}{2}\right) - g\left(\frac{x+y}{2}\right) + \frac{x+y}{2} - \frac{1}{4} = x \cdot y$$

können wir zusammen mit dem vorherigen Lemma die Funktion $\text{mult}(x,y)$ approximieren:

Lemma 2. *[16, S.3 im Appendix]*
Für jede positive Zahl m, *existiert ein Netzwerk* $\text{mult}_m \in F(m+4, (2,6,6,\ldots,6,1))$, *sodass* $\text{mult}_m(x,y) \in [0,1]$ *mit*

$$|\text{mult}_m(x,y) - \text{mult}(x,y)| \leq 2^{-m}, \quad \textit{für alle } x, y \in [0,1]$$

und $\text{mult}_m(0,y) = \text{mult}_m(x,0) = 0$.

Eine interessante Folgerung des Lemmas ist, dass wir mit tieferen Netzwerken, die Multiplikation von zwei Zahlen x und y besser approximieren können. Mittels Parallelisierung, siehe Abschnitt 3.3.1, werden wir Netzwerke aus mult_m zu einem Netzwerk zusammenfassen, welche die Multiplikation für r Variablen berechnet. Mithilfe des Lemma 2 können wir die Resultate auf die Multiplikation mit r Variablen erweitern.

Lemma 3. *[16, S.4 im Appendix]*
Für jede positive ganze Zahl m, existiert ein Netzwerk

$$\text{mult}_m^r \in F((m+5)\lceil \log_2 r \rceil, (r, 6r, 6r, \ldots, 6r, 1)),$$

sodass $\text{mult}_m^r \in [0,1]$ *und*

$$\left|\text{mult}_m^r(\mathbf{x}) - \prod_{i=1}^{r} x_i\right| \leq r^2 2^{-m}, \quad \textit{für alle } \mathbf{x} = (x_1, \ldots, x_r) \in [0,1]^r.$$

Außerdem gilt $\text{mult}_m^r(\mathbf{x}) = 0$*, falls einer der Komponenten von* $\mathbf{x}$ *Null ist.*

Auch hier sehen wir, dass tiefere Netzwerke die Multiplikation zwischen mehreren Variablen besser approximieren. Mit diesem Lemma können wir nun Produkte mit mehreren Variablen durch ein neuronales Netzwerk approximieren, d.h. wir können auch die Taylor-Entwicklung der Funktion h approximieren. Ist $k \in \mathbb{N}$ und h k-mal stetig partiell differenzierbar, so lautet das Taylor-Polynom der Ordnung k an einer Stelle $\mathbf{a} \in [0,1]^d$

$$P_{\mathbf{a}}^{k-1} h(\mathbf{x}) = \sum_{\boldsymbol{\alpha} \in \mathbb{N}_0^d, 0 \leq \|\boldsymbol{\alpha}\|_1 \leq k-1} (\partial^{\boldsymbol{\alpha}} h)(\mathbf{a}) \frac{(\mathbf{x}-\mathbf{a})^{\boldsymbol{\alpha}}}{\boldsymbol{\alpha}!}, \tag{16}$$

wobei $\partial^{\boldsymbol{\alpha}} h := \partial_{x_1^{\alpha_1}} \ldots \partial_{x_d^{\alpha_d}}$, $\|\boldsymbol{\alpha}\|_1 = \sum_{j=1}^d \alpha_j$, $\boldsymbol{\alpha}! := \alpha_1! \cdot \ldots \cdot \alpha_d!$ und $v_1^{\alpha_1} \cdot \ldots \cdot v_d^{\alpha_d}$ für $\mathbf{v} \in \mathbb{R}^d$, vgl. [15, S.214].
Durch die Restgliedabschätzung aus dem Satz von Taylor können wir folgern, dass falls $\sup_{\boldsymbol{\alpha} \in \mathbb{N}_0^d, \|\boldsymbol{\alpha}\|_1 = k} \sup_{\mathbf{x} \in [0,1]^d} |\partial^{\boldsymbol{\alpha}} h| \leq K$ mit einer Konstanten $K \in \mathbb{R}$, dann gilt $\forall \mathbf{x} \in [0,1]^d$:

$$|h(\mathbf{x}) - P_{\mathbf{a}}^{k-1} h(\mathbf{x})| \leq \frac{K}{k!} |\mathbf{x} - \mathbf{a}|_\infty^k, \tag{17}$$

vgl. [15, S. 215]. Aus der Ungleichung (17) sehen wir, dass wir durch einen Punkt $\boldsymbol{a} \in [0,1]^d$ eine Approximation von h mit Taylor keine besseren Resultate erlangen können als eine obere Schranke von $\frac{K}{k!}|\mathbf{x}-\mathbf{a}|_\infty^k$. Da wir im Intervall $[0,1]^d$ beschränkt sind, entspricht die Schranke in vielen Fällen der Form $\frac{K}{k!} \cdot \frac{1}{2^k}$. Wir können also mit einem neuronalen Netzwerk mittels dem Lemma 3 so beliebig gut $P_{\mathbf{a}}^{k-1} h(\mathbf{x})$ approximieren wir wir wollen, wir würden trotzdem eine zu grobe Abschätzung mit $\frac{K}{k!} \cdot \frac{1}{2^k}$ erhalten.

Um die Ungleichung (17) besser auszunutzen, wird der Inputraum in kleine Hyperwürfel aufgeteilt, worauf wir auf jeden dieser Hyperwürfel eine lokale Taylorreihe durch Netzwerke approximieren werden. Die Zerteilung in Hyperwürfel wird wie folgt vorgenommen:

$$D(M) := \{\mathbf{a}^{(r)} := (\frac{r_j}{M})_{j=1,\dots,d} : r = (r_1, \dots, r_d) \in \{0, 1, \dots, M\}^d\} \subset [0,1]^d$$

Wir nennen $D(M)$ auch Gitter. Als nächstes wird h je nach Argument von x mit dem Taylorpolynom, welches zu dem Gitterpunkt gehört, der am nächsten an x liegt, approximiert, vgl. [15, S. 215].

$$P^{k-1}h(\mathbf{x}) := \sum_{\mathbf{a}^{(r)} \in D(M)} P^{k-1}_{\mathbf{a}^{(r)}} h(\mathbf{x}) \cdot \prod_{j=1}^{d} \Big(1 - M \cdot |x_j - a_j^{(r)}|\Big)_+$$

Wir können das Polynom $P^{k-1}_{\mathbf{a}} h$ aus (16) durch Ausmultiplizieren als eine Linearkombination von Monomen

$$P^{k-1}_{\mathbf{a}} h(\mathbf{x}) = \sum_{\boldsymbol{\gamma} \in \mathbb{N}_0^d : 0 \le \|\boldsymbol{\gamma}\|_1 \le k-1} c_{\boldsymbol{\gamma}} \cdot \mathbf{x}^{\boldsymbol{\gamma}}, \tag{18}$$

für ein geeigneten Koeffizienten $c_{\boldsymbol{\gamma}}$ schreiben, wobei $\boldsymbol{\gamma} \in \mathbb{N}_0^d$ ein Multi-Index ist. Aus (18) sehen wir, dass $\partial^{\boldsymbol{\gamma}} P^{k-1}_{\mathbf{a}} h(\mathbf{x})|_{\mathbf{x}=0} = \boldsymbol{\gamma}! c_{\boldsymbol{\gamma}}$ gilt. Falls wir nun den Term $\partial^{\boldsymbol{\gamma}} P^{k-1}_{\mathbf{a}} h(\mathbf{x})|_{\mathbf{x}=0}$ mit der Gleichung (16) bilden, dann gilt

$$\begin{aligned}
\partial^{\boldsymbol{\gamma}} P^{k-1}_{\mathbf{a}} h(\mathbf{x})|_{\mathbf{x}=0} &= \sum_{\boldsymbol{\alpha} \in \mathbb{N}_0^d : 0 \le \|\boldsymbol{\alpha}\|_1 \le k-1, \boldsymbol{\gamma} \le \boldsymbol{\alpha}} (\partial^{\boldsymbol{\alpha}} h)(\mathbf{a}) \cdot \frac{\boldsymbol{\alpha}! (-\mathbf{a})^{\boldsymbol{\alpha}-\boldsymbol{\gamma}}}{(\boldsymbol{\alpha}-\boldsymbol{\gamma})! \boldsymbol{\alpha}!} \\
&= \sum_{\boldsymbol{\alpha} \in \mathbb{N}_0^d : 0 \le \|\boldsymbol{\alpha}\|_1 \le k-1, \boldsymbol{\gamma} \le \boldsymbol{\alpha}} (\partial^{\boldsymbol{\alpha}} h)(\mathbf{a}) \cdot \frac{(-\mathbf{a})^{\boldsymbol{\alpha}-\boldsymbol{\gamma}}}{(\boldsymbol{\alpha}-\boldsymbol{\gamma})!} \\
&= \boldsymbol{\gamma}! \sum_{\boldsymbol{\alpha} \in \mathbb{N}_0^d : 0 \le \|\boldsymbol{\alpha}\|_1 \le k-1, \boldsymbol{\gamma} \le \boldsymbol{\alpha}} (\partial^{\boldsymbol{\alpha}} h)(\mathbf{a}) \cdot \frac{(-\mathbf{a})^{\boldsymbol{\alpha}-\boldsymbol{\gamma}}}{\boldsymbol{\gamma}!(\boldsymbol{\alpha}-\boldsymbol{\gamma})!}.
\end{aligned}$$

Es folgt somit

$$c_{\boldsymbol{\gamma}} := \sum_{\boldsymbol{\alpha} \in \mathbb{N}_0^d : 0 \le \|\boldsymbol{\alpha}\|_1 \le k-1, \boldsymbol{\gamma} \le \boldsymbol{\alpha}} (\partial^{\boldsymbol{\alpha}} h)(\mathbf{a}) \cdot \frac{(-\mathbf{a})^{\boldsymbol{\alpha}-\boldsymbol{\gamma}}}{\boldsymbol{\gamma}!(\boldsymbol{\alpha}-\boldsymbol{\gamma})!}.$$

Man schreibt $\boldsymbol{\gamma} \le \boldsymbol{\alpha}$, falls für alle $j \in \{1, \dots, d\} :\ \gamma_j \le \alpha_j$ gilt.

Wir können nun die Terme $x^{\boldsymbol{\gamma}}$ in

$$P^{k-1}_{\mathbf{a}} h(\mathbf{x}) = \sum_{\boldsymbol{\gamma} \in \mathbb{N}_0^d : 0 \le \|\boldsymbol{\gamma}\|_1 \le k-1} c_{\boldsymbol{\gamma}} \cdot x^{\boldsymbol{\gamma}}, \qquad c_{\boldsymbol{\gamma}} := \sum_{\boldsymbol{\alpha} \in \mathbb{N}_0^d : 0 \le \|\boldsymbol{\alpha}\|_1 \le k-1, \boldsymbol{\gamma} \le \boldsymbol{\alpha}} (\partial^{\boldsymbol{\alpha}} h)(\mathbf{a}) \cdot \frac{(-\mathbf{a})^{\boldsymbol{\alpha}-\boldsymbol{\gamma}}}{\boldsymbol{\gamma}!(\boldsymbol{\alpha}-\boldsymbol{\gamma})!},$$

mittels dem Lemma 3 diskutieren und erhalten damit das folgende Theorem:

Theorem 4. *[15, S.215]*
Ist $h : [0,1]^d \to \mathbb{R}$ k-mal partiell differenzierbar und gilt $\sup_{\boldsymbol{\alpha} \in \mathbb{N}_0^d, \|\boldsymbol{\alpha}\|_1 = k} \sup_{\mathbf{x} \in [0,1]^d} |\partial^{\boldsymbol{\alpha}} h| \leq K$, so gibt es für alle $m, N \in \mathbb{N}$ mit $N \geq \max\{(k+1)^d, K+1\}$ ein

$$\widetilde{f} \in F(L, (d, 12dN, \ldots, 12dN, 1)), \quad L = 8 + (m+5)(1 + \lceil \log_2 d \rceil),$$

sodass

$$\forall \mathbf{x} \in [0,1]^d : |h(\mathbf{x}) - \widetilde{f}(\mathbf{x})| \leq (2K+1)3^{d+1} N 2^{-m} + K 2^k N^{-k/d}$$

Das Theorem 4 zeigt, dass wir mindestens einmal differenzierbare Funktionen durch ein neuronales Netzwerk geeigneter Größe approximieren können. In [16, Appendix] wird sogar gezeigt, dass wir tatsächlich die allgemeineren Funktionen, die in der Hölderklasse mit Glattheitsindex β liegen, durch ein neuronales Netzwerk annähern können.

Theorem 5. *Für jede beliebige Funktion $h \in C_r^\beta([0,1]^r, K)$ und jede beliebige ganze Zahl $m \geq 1$ und $N \geq (\beta+1)^r \vee (K+1)e^r$, existiert ein Netzwerk*

$$\widetilde{f} \in F(L, (r, 6(r + \lceil \beta \rceil)N, ..., 6(r + \lceil \beta \rceil)N, 1), s, \infty)$$

mit der Tiefe

$$L = 8 + (m+5)(1 + \lceil \log_2(r \vee \beta) \rceil)$$

und die Anzahl an Parameter

$$s \leq 141(r + \beta + 1)^{3+r} N(m+6),$$

sodass

$$\|\widetilde{f} - h\|_{L^\infty([0,1]^r)} \leq (2K+1)(1 + r^2 + \beta^2)6^r N 2^{-m} + K 3^\beta N^{-\frac{\beta}{r}}.$$

Der Beweis verläuft analog zu unserer Beweisskizze für mindestens einmal differenzierbare Funktionen und kann im Appendix von [16] nachgeschlagen werden. Der Unterschied fängt bei der Betrachtung der Taylor-Reihe an, da wir hier nun beachten müssen, dass wir Funktionen betrachten, die in der Hölderklasse liegen und nicht mindestens einmal differenzierbar sind. Die Güte der Näherung von Theorem 4 und Theorem 5 wird durch N und m bestimmt. N beschreibt dabei die Anzahl der Gitterpunkte von $D(M)$, da wir im Beweis M maximal wählen, sodass $(M+1)^d = \#D(M) \leq N$ gilt. Falls wir m, N groß wählen und deren Balancierung untereinander beachten, dann kann die Abschätzung in den zwei Theoremen beliebig klein werden.

Mit Theorem 5 können wir nun ein Netzwerk konstruieren, das die gesuchte Funktion $f_0 = g_q \circ ... \circ g_0$ approximiert, indem wir die einzelnen Komponenten mit dem Theorem annähern. Wie im Theorem 1 nehmen wir an, dass $g_{ij} \in C_{t_i}^{\beta_i}([a_i, b_i]^{t_i}, K_i)$ und zusätzlich o.B.d.A., dass $K_i \geq 1$. Theorem 5 setzt jedoch voraus, dass die zu approximierende Funktion im Intervall $[0,1]^r$ liegt. Deshalb zeigen wir im ersten Schritt, dass wir die Funktion f_0

immer als Komposition von Funktionen, die definiert sind auf einem Hyperwürfel $[0,1]^{t_i}$, schreiben können. Für $i = 1, ..., q-1$ definiere

$$h_0 := \frac{g_0}{2K_0} + \frac{1}{2}, \qquad h_i := \frac{g_i(2K_{i-1} \cdot -K_{i-1})}{2K_i} + \frac{1}{2}, \qquad h_q = g_q(2K_{q-1} \cdot -K_{q-1}).$$

Für $2K_{i-1}\mathbf{x} - K_{i-1}$ berechnen wir die Einträge durch $2K_{i-1}x_j - K_{i-1}$ für alle j. Offensichtlich gilt

$$f_0 = g_q \circ ... \circ g_0 = h_q \circ ... \circ h_0. \tag{19}$$

Zum Beispiel für $q = 2$ gilt:

$$h_1 \circ h_0 = h_1(h_0) = \frac{g_1(2K_0h_0 - K_0)}{2K_1} + \frac{1}{2} = \frac{g_1\left(2K_0\left(\frac{g_0}{2K_0} + \frac{1}{2}\right) - K_0\right)}{2K_1} + \frac{1}{2} = \frac{g_1(g_0)}{2K_1} + \frac{1}{2}$$

$$h_2 \circ h_1 \circ h_0 = h_2(h_1 \circ h_0) = g_2\left(2K_1\left(\frac{g_1(g_0)}{2K_1} + \frac{1}{2}\right) - K_1\right) = g_2\left(g_1\left(g_0\right)\right) = g_2 \circ g_1 \circ g_0.$$

Wir wollen nun die h_{ij} weiter analysieren. Da $g_{0j} \in C^{\beta_0}_{t_0}([0,1]^{t_0}, K_0)$ folgt, dass $h_{0j} = \frac{g_{0j}}{2K_0} + \frac{1}{2} \in C^{\beta_0}_{t_0}([0,1]^{t_0}, 1)$. Weiterhin gilt somit $h_{ij} \in C^{\beta_i}_{t_i}([0,1]^{t_i}, (2K_{i-1})^{\beta_i})$ für $i = 1, ..., q-1$ und $h_{qj} \in C^{\beta_q}_{t_q}([0,1]^{t_q}, K_q(2K_{q-1})^{\beta_q})$. Ohne Beschränkung der Allgemeinheit nehmen wir an, dass die Radien der Hölder Bälle mindestens eins ist, also $K_i \geq 1$.

Lemma 4. *Sei h_{ij} wie oben definiert mit $K_i \geq 1$. Dann gilt für jede beliebige Funktion $\widetilde{h}_i = (\widetilde{h}_{ij})_j^\top$ mit $\widetilde{h}_{ij} : [0,1]^{t_i} \to [0,1]$,*

$$\begin{aligned}
&\|h_q \circ ... \circ h_0 - \widetilde{h}_q \circ ... \circ \widetilde{h}_0\|_{L^\infty[0,1]^d} \\
&\leq K_q \prod_{l=0}^{q-1} (2K_l)^{\beta_{l+1}} \sum_{i=0}^{q} \| |h_i - \widetilde{h}_i|_\infty \|^{\prod_{l=i+1}^{q} \beta_l \wedge 1}_{L^\infty[0,1]^{d_i}}.
\end{aligned}$$

Beweis.. Definiere $H_i = h_i \circ ... \circ h_0$ und $\widetilde{H}_i = \widetilde{h}_i \circ ... \circ \widetilde{h}_0$. Falls Q_i ist eine obere Schranke für die Hölder Konstante von $h_{ij}, j = 1, ..., d_{i+1}$, dann gilt mit der Dreiecksungleichung

$$\begin{aligned}
&\left|H_i(\mathbf{x}) - \widetilde{H}_i(\mathbf{x})\right|_\infty = \left|h_i \circ H_{i-1}(\mathbf{x}) - \widetilde{h}_i \circ \widetilde{H}_{i-1}(\mathbf{x})\right|_\infty \\
&= \left|h_i \circ H_{i-1}(\mathbf{x}) - h_i \circ \widetilde{H}_{i-1}(\mathbf{x}) + h_i \circ \widetilde{H}_{i-1}(\mathbf{x}) - \widetilde{h}_i \circ \widetilde{H}_{i-1}(\mathbf{x})\right|_\infty \\
&\leq \left|h_i \circ H_{i-1}(\mathbf{x}) - h_i \circ \widetilde{H}_{i-1}(\mathbf{x})\right|_\infty + \left|h_i \circ \widetilde{H}_{i-1}(\mathbf{x}) - \widetilde{h}_i \circ \widetilde{H}_{i-1}(\mathbf{x})\right|_\infty \\
&\leq Q_i \left|H_{i-1}(\mathbf{x}) - \widetilde{H}_{i-1}(\mathbf{x})\right|^{\beta_i \wedge 1}_\infty + \| |h_i - \widetilde{h}_i|_\infty \|_{L^\infty[0,1]^{d_i}}
\end{aligned}$$

Mit dieser Abschätzung können wir nun unser Lemma beweisen: Sei h_{ij} mit $K_i \geq 1$ und $\widetilde{h}_i$ wie im Lemma definiert, dann gilt

$$\|h_q \circ ... \circ h_0 - \widetilde{h}_q \circ ... \circ \widetilde{h}_0\|_{L^\infty[0,1]^d}$$

$$
\begin{aligned}
&\leq K_q(2K_{q-1})^{\beta_q}\|h_{q-1}\circ ...\circ h_0 - \widetilde{h}_{q-1}\circ ...\circ \widetilde{h}_0\|_{L^\infty[0,1]^d}^{\beta_q\wedge 1} + \||h_q - \widetilde{h}_q|_\infty\|_{L^\infty[0,1]^{d_i}}\\
&\leq K_q(2K_{q-1})^{\beta_q}\Bigg((2K_{q-2})^{\beta_{q-1}}\|h_{q-2}\circ ...\circ h_0 - \widetilde{h}_{q-2}\circ ...\circ \widetilde{h}_0\|_{L^\infty[0,1]^d}^{\beta_{q-1}\wedge 1}\\
&\quad + \||h_{q-1} - \widetilde{h}_{q-1}|_\infty\|_{L^\infty[0,1]^{d_i}}\Bigg)^{\beta_q\wedge 1} + \||h_q - \widetilde{h}_q|_\infty\|_{L^\infty[0,1]^{d_i}}
\end{aligned}
$$

Aus $K_i \geq 1$ und $\beta_i \geq 0$ folgt $K_i^{\beta_i\wedge 1} \leq K_i$. Zusammen mit der Ungleichheit $(y+z)^\alpha \leq y^\alpha + z^\alpha$, dass für alle $y, z \geq 0$ und alle $\alpha \in [0,1]$ gilt, können wir nun weiter abschätzen:

$$
\begin{aligned}
&\leq K_q\Bigg((2K_{q-1})^{\beta_q}(2K_{q-2})^{\beta_{q-1}}\|h_{q-2}\circ ...\circ h_0 - \widetilde{h}_{q-2}\circ ...\circ \widetilde{h}_0\|_{L^\infty[0,1]^d}^{(\beta_{q-1}\wedge 1)\cdot(\beta_q\wedge 1)}\\
&\quad + (2K_{q-1})^{\beta_q}\||h_{q-1} - \widetilde{h}_{q-1}|_\infty\|_{L^\infty[0,1]^{d_i}}^{\beta_q\wedge 1}\Bigg) + \||h_q - \widetilde{h}_q|_\infty\|_{L^\infty[0,1]^{d_i}}\\
&\qquad\qquad\vdots\\
&\leq K_q\prod_{l=0}^{q-1}(2K_l)^{\beta_{l+1}}\sum_{i=0}^{q}\||h_i - \widetilde{h}_i|_\infty\|_{L^\infty[0,1]^{d_i}}^{\prod_{l=i+1}^q \beta_l\wedge 1},
\end{aligned}
$$

wobei wir hier sukzessiv $\|h_{q-2}\circ ...\circ h_0 - \widetilde{h}_{q-2}\circ ...\circ \widetilde{h}_0\|_{L^\infty[0,1]^d}^{(\beta_{q-1}\wedge 1)\cdot(\beta_q\wedge 1)}$ weiter abgeschätzt und ausgenutzt haben, dass

$$
\forall k \in \{0,...,q\}: \qquad \||h_k - \widetilde{h}_k|_\infty\|_{L^\infty[0,1]^{d_k}}^{\prod_{l=k+1}^q \beta_l\wedge 1} \leq \sum_{i=0}^{q}\||h_i - \widetilde{h}_i|_\infty\|_{L^\infty[0,1]^{d_i}}^{\prod_{l=i+1}^q \beta_l\wedge 1}
$$

und

$$
\||h_i - \widetilde{h}_i|_\infty\|_{L^\infty[0,1]^{d_i}} \leq K_q\prod_{l=0}^{q-1}(2K_l)^{\beta_{l+1}}\||h_i - \widetilde{h}_i|_\infty\|_{L^\infty[0,1]^{d_i}}.
$$

□

3.3.3 Beweis zum Theorem 1

Beweis zum Theorem 1. Es reicht aus das Theorem 1 für hinreichend große n zu beweisen. Als erstes werden wir unter der Annahme, dass $C\phi_n L\log^2(n) \leq \Delta_n(\widehat{f}_n, f_0)$ gilt, die untere Schranke aus (15) vom Theorem 1 zeigen. Wir nutzen dafür die Ungleichung vom Orakel-Typus aus dem Theorem 2 mit den angenommenen Schranken der Tiefe L (aus Bedingung (ii)) und der Anzahl an aktiven Parametern s (aus Bedingung (iv)). Es folgt für den Stichprobenumfang $n \geq 3$:

$$
\begin{aligned}
&\frac{1}{4}\Delta_n(\widehat{f}_n, f_0) - C'\phi_n L\log^2(n) \leq R(\widehat{f}_n, f_0)\\
&\leq 4\inf_{f^*\in F(L,\mathbf{p},s,F)}\|f^* - f_0\|_\infty^2 + 4\Delta_n(\widehat{f}_n, f_0) + C'\phi_n L\log^2(n).
\end{aligned}
\tag{20}
$$

Wir zeigen (20), indem wir im Theorem 2 für die untere Grenze $\epsilon = \frac{1}{2}$ und für die obere Grenze $\epsilon = 1$ wählen, dann gilt nämlich

$$\frac{1}{4}\Delta_n(\widehat{f}_n, f_0) - \tau_{\frac{1}{2},n} \leq R(\widehat{f}_n, f_0) \leq 4 \inf_{f \in F(L,\mathbf{p},s,F)} \|f - f_0\|_\infty^2 + 4\Delta_n(\widehat{f}_n, f_0) + \tau_{1,n}.$$

Um nun beide Seiten von (20) zu zeigen, müssen wir nachweisen, dass $\tau_{\epsilon,n} \leq C'\phi_n L \log^2(n)$ gilt. Es gilt für eine Konstante C', die von $(q, \mathbf{d}, \mathbf{t}, \beta, F)$ abhängt und sich von Zeile zu Zeile ändern kann, dass

$$\begin{aligned}
\tau_{\epsilon,n} &= C_\epsilon F^2 \frac{(s+1)\log\left(n(s+1)^L p_0 p_{L+1}\right)}{n} \leq C' \frac{s \log\left(n(s+1)^L p_0 p_{L+1}\right)}{n} \\
&\underset{Bed(iv)}{\leq} C'\phi_n \log(n) \log\left(n(s+1)^L p_0 p_{L+1}\right) \\
&= C'\phi_n \log(n) \left(\log(n) + L\log(s+1) + \log(p_0 p_{L+1})\right) \\
&\underset{Bed(ii)}{\leq} C'\phi_n \log(n) \left(L\log(s+1)\right) \\
&\underset{Bed(iv)}{\leq} C'\phi_n \log(n) L \log(n) = C'\phi_n L \log(n)^2.
\end{aligned}$$

Somit haben wir (20) gezeigt.
Falls nun $C\phi_n L \log^2(n) \leq \Delta_n(\widehat{f}_n, f_0)$ gilt und wir $C = 8C'$ wählen, dann folgt für

$$C'\phi_n L \log^2(n) = \frac{1}{8} C \phi_n L \log^2(n) \leq \frac{1}{8}\Delta_n(\widehat{f}, f_0).$$

Mit (20) gilt insgesamt

$$R(\widehat{f}_n, f_0) \geq \frac{1}{4}\Delta_n(\widetilde{f}_n, f_0) - C'\phi_n L \log^2(n) \geq \frac{1}{4}\Delta_n(\widehat{f}_n, f_0) - \frac{1}{8}\Delta_n(\widehat{f}_n, f_0) = \frac{1}{8}\Delta_n(\widehat{f}_n, f_0).$$

Damit gilt $\frac{1}{8}\Delta_n(\widehat{f}_n, f_0) \leq R(\widehat{f}_n, f_0)$ und somit ist die untere Grenze in (15) bewiesen.

Um die obere Grenze von (14) und (15) herzuleiten, werden wir die obere Abschätzung aus (20) nutzen. Wir müssen somit eine Schranke für den Approximationsfehler

$$\inf_{f^* \in F(L,\mathbf{p},s,F)} \|f^* - f_0\|_\infty$$

finden. Wir werden dafür die Regressionsfunktion f_0, wie in (19), in der Form $f_0 = h_q \circ ... \circ h_0$ mit $h_i = (h_{ij})_j^\top$, h_{ij} definiert auf $[0,1]^{t_i}$ darstellen und für jedes beliebige $i < q$ soll h_{ij} auf $[0,1]$ abbilden.

Nach Theorem 5 können wir ein Netzwerk geeigneter Größe finden, die die h_{ij} annähern können. Wir wenden das Theorem auf jede Funktion h_{ij} einzeln an, wobei, wie oben

schon gezeigt, $h_{ij} \in C_{t_i}^{\beta_i}\left([0,1]^{t_i}, (2K_{i-1})^{\beta_i}\right)$ gilt. Um die Bedingungen aus Theorem 5 zu erfüllen, wählen wir $m = \lceil \log_2(n) \rceil \geq 1$ und $N \leq (\beta_i + 1)^{t_i} \vee (Q_i + 1)e^{t_i}$, wobei Q_i eine obere Schranke für die Hölder Konstante von h_{ij} ist. Nach dem Theorem 5 gilt nun, dass ein Netzwerk existiert mit $\widetilde{h}_{ij} \in F(L_i', (t_i, 6(t_i + \lceil \beta_i \rceil)N, ..., 6(t_i + \lceil \beta_i \rceil)N, 1), s_i)$ mit der Tiefe $L_i' := 8 + (\lceil \log_2 n \rceil + 5)(1 + \lceil \log_2(t_i \vee \beta_i) \rceil)$ und der aktiven Parameter $s_i \leq 141(t_i + \beta_i + 1)^{3+t_i} N(\lceil \log_2 n \rceil + 6)$, sodass

$$\|\widetilde{h}_{ij} - h_{ij}\|_{L^\infty([0,1]^{t_i})} \leq (2Q_i + 1)(1 + t_i^2 + \beta_i^2)6^{t_i} N n^{-1} + Q_i 3^{\beta_i} N^{-\frac{\beta_i}{t_i}}. \tag{21}$$

Um aber $\inf_{f^* \in F(L,\mathbf{p},s,F)} \|f^* - h_q \circ ... \circ h_0\|_\infty$ mit einer Komposition von f^* aus $\widetilde{h}_i$ und dem Lemma 4 abzuschätzen, muss $\widetilde{h}_{ij} : [0,1]^{t_i} \to [0,1]$ gelten. Wir müssen also zunächst die Netzwerke $\widetilde{h}_{ij}$ transformieren.
Falls nun $i < q$, dann füge zu dem Output x zwei zusätzliche Layer hinzu um den Output zu $1-(1-x)_+$ zu transformieren. Man braucht insgesamt 4 zusätzliche aktive Parameter. Wir nennen das resultierende Netzwerk $h_{ij}^* \in F(L_i'+2, (t_i, 6(t_i+\lceil \beta_i \rceil)N, ..., 6(t_i+\lceil \beta_i \rceil)N, 1), s_i+4)$ und beobachten, dass $\sigma(h_{ij}^*) = (h_{ij}^*(\mathbf{x}) \vee 0) \wedge 1$. Die Transformation kann man graphisch in Abbildung 13 nachvollziehen.

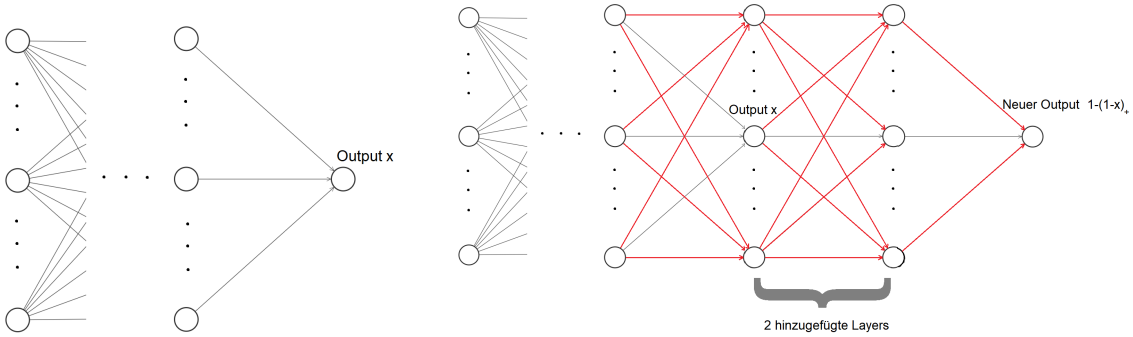

Abbildung 13: Rote Linien illustrieren inaktive Verbindungen. Man erkennt leicht, dass wir jeweils zwei zusätzliche Gewichts- und Verschiebungsparameter haben.

Da wir durch die Transformation $(1 - (1 - x)_+)_+$ von $\widetilde{h}_{ij}$ zu $\sigma(h_{ij}^*)$ das Supremum nicht erhöht haben und $h_{ij}(\mathbf{x}) \in [0,1]$ gilt, muss gelten, dass

$$\|\sigma(h_{ij}^*) - h_{ij}\|_{L^\infty([0,1]^r)} \leq \|\widetilde{h}_{ij} - h_{ij}\|_{L^\infty([0,1]^r)}. \tag{22}$$

Falls die Netzwerke h_{ij}^* parallel berechnet werden, vgl. Abschnitt 3.3.1, dann liegt $h_i^* = (h_{ij}^*)_{j=1,...,d_{i+1}}$ in der Klasse

$$F\left(L_i' + 2, (d_i, 6r_iN, ..., 6r_iN, d_{i+1}), d_{i+1}(s_i + 4)\right), \tag{23}$$

mit $r_i := \max_i d_{i+1}\,(t_i + \lceil \beta_i \rceil)$. Die Netzwerkklasse (23) ensteht dadurch, dass wir den Input durch $t_i \leq d_i$, die hidden Layer durch $\sum_{j=1}^{d_i+1} 6(t_i + \lceil \beta_i \rceil)N = d_{i+1}6(t_i + \lceil \beta_i \rceil)N \leq$

$6 \cdot \max_i d_{i+1}\,(t_i + \lceil \beta_i \rceil)\,N$ und den Output durch $\sum_{j=1}^{d_{i+1}}(s_i+4) = d_{i+1}(s_i+4)$ darstellen können.

Wir können nun das zusammengesetzte Netzwerk $f^* = \widetilde{h}_{q1} \circ \sigma(h^*_{q-1}) \circ ... \circ \sigma(h^*_0)$ konstruieren, indem wir die Kompositionsregel von Netzwerken, vgl. Abschnitt 3.3.1, anwenden und erhalten für die Anzahl der Layer

$$\sum_{i=0}^{q-1}(L'_i+2) + \sum_{i=0}^{q-2} 1 + L'_q + 1 = 3q + \sum_{i=0}^{q} L'_i =: E$$

und für die aktiven Parameter

$$\sum_{i=0}^{q-1} d_{i+1}(s_i+4) + s_q \leq \sum_{i=0}^{q} d_{i+1}(s_i+4).$$

Wir können das Netzwerk zusammenfassen zu

$$F\left(E, (d, 6r_iN, ..., 6r_iN, 1), \sum_{i=0}^{q} d_{i+1}(s_i+4)\right), \tag{24}$$

mit $E := 3q + \sum_{i=0}^{q} L'_i$.
Als nächstes wollen wir das Netzwerk so transformieren, sodass die Bedingungen aus dem Theorem für $L, \mathbf{p}, s$ erfüllt werden. Es existiert nun ein A_n, dass beschränkt ist in n, sodass $E = A_n + \lceil \log_2 n \rceil (\sum_{i=0}^{q} \lceil \log_2(t_i \vee \beta_i) \rceil + 1)$, da

$$\begin{aligned}
\sum_{i=0}^{q} L'_i &= \sum_{i=0}^{q} \Big(8 + (\lceil \log_2 n \rceil + 5) \cdot (1 + \lceil \log_2(t_i \vee \beta_i) \rceil)\Big) \\
&= \sum_{i=0}^{q} \Big(8 + (\lceil \log_2 n \rceil + 5 + \lceil \log_2(t_i \vee \beta_i) \rceil \cdot \lceil \log_2 n \rceil + 5 \lceil \log_2(t_i \vee \beta_i) \rceil\Big) \\
&= (q+1)8 + \sum_{i=0}^{q} \Big(\lceil \log_2 n \rceil + 5 + \lceil \log_2(t_i \vee \beta_i) \rceil \cdot \lceil \log_2 n \rceil + 5 \lceil \log_2(t_i \vee \beta_i) \rceil\Big) \\
&= (q+1)8 + \sum_{i=0}^{q} \Big(5 + 5 \lceil \log_2(t_i \vee \beta_i) \rceil\Big) + \sum_{i=0}^{q} \Big(\lceil \log_2 n \rceil \left(\lceil \log_2(t_i \vee \beta_i \rceil + 1\right)\Big) \\
&= (q+1)8 + \sum_{i=0}^{q} \Big(5 + 5 \lceil \log_2(t_i \vee \beta_i \rceil\Big) + \lceil \log_2 n \rceil \sum_{i=0}^{q} \Big(\lceil \log_2(t_i \vee \beta_i \rceil + 1\Big)
\end{aligned}$$

Insgesamt gilt also

$$\begin{aligned}
E &= 3q + \sum_{i=0}^{q} L_i' \\
&= 3q + (q+1)8 + \sum_{i=0}^{q} (5 + 5\lceil \log_2(t_i \vee \beta_i) \rceil) + \lceil \log_2 n \rceil \sum_{i=0}^{q} (\lceil \log_2(t_i \vee \beta_i) \rceil + 1) \\
&= A_n + \lceil \log_2 n \rceil (\sum_{i=0}^{q} \lceil \log_2(t_i \vee \beta_i) \rceil + 1),
\end{aligned}$$

wobei $A_n := 3q + (q+1)8 + \sum\limits_{i=0}^{q} (5 + 5\lceil \log_2(t_i \vee \beta_i) \rceil)$.
Aus der Tatsache, dass $\lceil x \rceil < x + 1$ gilt und A_n beschränkt in n ist, gilt für ausreichend große n, dass

$$\begin{aligned}
E &\leq \log_2(n) \left(\sum_{i=0}^{q} \log_2(t_i \vee \beta_i) + 1 + 1 \right) \\
&= \log_2(n) \left(\sum_{i=0}^{q} \log_2(t_i \vee \beta_i) + \log_2(4) \right) \underset{Bed(ii)}{\leq} L
\end{aligned}$$

Da nun $E \leq L$ für ausreichend große n gilt, kann durch Hinzufügen von Layers, vgl. Abschnitt 3.3.1, die Netzwerkklasse aus (24) zu $F(L, \mathbf{p}, s)$ für ausreichend große n erweitert werden. Wir erweitern die Netzwerkklasse unter Beachtung, dass wir die Bedingungen des Theorems durch $L, \mathbf{p}, s$ erfüllen und wählen dafür $N = \lceil c \max_{i=0,\dots,q} n^{\frac{t_i}{2\beta_i^* + t_i}} \rceil$ für ein ausreichend kleine Konstante $c > 0$, die nur von $q, \mathbf{d}, \mathbf{t}, \boldsymbol{\beta}$ abhängt. Wir können nun mit dem Lemma 4 und einer Konstanten C', die von $(q, \mathbf{d}, \mathbf{t}, \boldsymbol{\beta}, F)$ abhängt und sich von Zeile zu Zeile ändern kann, folgern, dass

$$\begin{aligned}
&\inf_{f^* \in F(L,\mathbf{p},s)} \|f^* - f_0\|_\infty^2 \\
&= \inf_{\widetilde{h}_{q1} \circ \sigma(h_{q-1}^*) \circ \dots \circ \sigma(h_0^*) \in F(L,\mathbf{p},s)} \left\| \widetilde{h}_{q1} \circ \sigma(h_{q-1}^*) \circ \dots \circ \sigma(h_0^*) - h_q \circ \dots \circ h_0 \right\|_\infty^2 \\
&\leq \inf_{\dots} \left(K_q \prod_{l=0}^{q-1} (2K_l)^{\beta_{l+1}} \right)^2 \left(\sum_{i=0}^{q-1} \| |\sigma(h_i^*) - h_i|_\infty \|_{L^\infty[0,1]^{d_i}}^{\prod_{l=i+1}^{q} \beta_l \wedge 1} + \| |\widetilde{h}_{q1} - h_q|_\infty \|_{L^\infty[0,1]^{d_q}}^1 \right)^2 \\
&\leq \inf_{\dots} C' \left(\sum_{i=0}^{q-1} \left(\sum_{j=1}^{d_{i+1}} \| |\sigma(h_{ij}^*) - h_{ij}|_\infty \|_{L^\infty[0,1]^{t_i}} \right)^{\prod_{l=i+1}^{q} \beta_l \wedge 1} + \| |\widetilde{h}_{q1} - h_q|_\infty \|_{L^\infty[0,1]} \right)^2
\end{aligned}$$

gilt. Mit der Gleichung (22) folgt weiter

$$\leq \inf_{\cdots} C' \left(\sum_{i=0}^{q-1} \left(\sum_{j=1}^{d_{i+1}} \| |h_{ij}^* - h_{ij}|_\infty \|_{L^\infty[0,1]^{t_i}} \right)^{\prod_{l=i+1}^{q} \beta_l \wedge 1} + \| |\widetilde{h}_{q1} - h_q|_\infty \|_{L^\infty[0,1]} \right)^2 .$$

Mit der Gleichung (21) können wir anschließend folgern

$$\leq C' \Bigg(\sum_{i=0}^{q-1} \left(d_{i+1} \left(2((2K_{i-1})^{\beta_i} + 1)(1 + t_i^2 + \beta_i^2) 6^{t_i} N n^{-1} + (2K_{i-1})^{\beta_i} 3^{\beta_i} N^{-\frac{\beta_i}{t_i}} \right) \right)^{\prod_{l=i+1}^{q} \beta_l \wedge 1}$$
$$+ 2(K_q(2K_{q-1})^{\beta_q} + 1)(1 + t_q^2 + \beta_q^2) 6^{t_q} N n^{-1} + K_q (2K_{q-1})^{\beta_q} 3^{\beta_q} N^{-\frac{\beta_q}{t_q}} \Bigg)^2$$

und für ausreichend kleines $c > 0$ in N gilt

$$\leq C' \Bigg(\sum_{i=0}^{q-1} \left(d_{i+1} \left(2((2K_{i-1})^{\beta_i} + 1)(1 + t_i^2 + \beta_i^2) 6^{t_i} N n^{-1} + (2K_{i-1})^{\beta_i} 3^{\beta_i} N^{-\frac{\beta_i}{t_i}} \right) \right)$$
$$+ 2(K_q(2K_{q-1})^{\beta_q} + 1)(1 + t_q^2 + \beta_q^2) 6^{t_q} N n^{-1} + K_q (2K_{q-1})^{\beta_q} 3^{\beta_q} N^{-\frac{\beta_q}{t_q}} \Bigg)^2$$

und damit ergibt sich insgesamt

$$\inf_{f^* \in F(L,\mathbf{p},s)} \|f^* - f_0\|_\infty^2 \leq C' \max_{i=0,\ldots,q} N^{-\frac{2\beta_i}{t_i}} \leq C' \max_{i=0,\ldots,q} c^{-\frac{2\beta_i^*}{t_i}} n^{-\frac{2\beta_i^*}{2\beta_i^* + t_i}}. \tag{25}$$

Für den Approximationsfehler in (20) brauchen wir jedoch eine Netzwerkfunktion, die in der Supremumsnorm von F beschränkt ist um den Term $\inf_{f^* \in F(L,\mathbf{p},s,F)} \|f^* - f_0\|_\infty^2$ abzuschätzen.
Aus (25) können wir entnehmen, dass es eine Folge $(\widetilde{f}_n)_n$ existiert, sodass für alle hinreichend große n, $\widetilde{f}_n \in F(L, \mathbf{p}, s)$ mit

$$\|\widetilde{f}_n - f_0\|_\infty^2 \leq 2C \max_{i=0,\ldots,q} c^{-\frac{2\beta_i^*}{t_i}} n^{-\frac{2\beta_i^*}{2\beta_i^* + t_i}} \tag{26}$$

gilt. Definiere $f_n^* = \widetilde{f}_n \cdot (\|f_0\|_\infty / \|\widetilde{f}_n\|_\infty \wedge 1)$. Das Netzwerk f_n^* liegt somit auch in $F(L, \mathbf{p}, s)$. Es gilt weiter, dass

$$\|f_n^*\|_\infty = \|\widetilde{f}_n \cdot \left(\frac{\|f_0\|_\infty}{\|\widetilde{f}_n\|_\infty} \wedge 1 \right) \|_\infty \leq \|\widetilde{f}_n \cdot \frac{\|f_0\|_\infty}{\|\widetilde{f}_n\|_\infty} \|_\infty \leq \|\widetilde{f}_n\|_\infty \cdot \frac{\|f_0\|_\infty}{\|\widetilde{f}_n\|_\infty} = \|f_0\|_\infty$$

und damit

$$\|f_n^*\|_\infty \leq \|f_0\|_\infty = \|g_q \circ \ldots \circ g_0\|_\infty = \|g_q\|_\infty \leq K \leq F,$$

wobei die letzte Ungleichung aus der Annahme (i) stammt. Wir haben somit gezeigt, dass $f_n^* \in F(L.\mathbf{p}, s, F)$ gilt. Aus $f_n^* - f_0 = (f_n^* - \widetilde{f}_n) + (\widetilde{f}_n - f_0)$ und

$$\begin{aligned}
\|f_n^* - \widetilde{f}_n\|_\infty &= \|\widetilde{f}_n \cdot \left(\frac{\|f_0\|_\infty}{\|\widetilde{f}_n\|_\infty} \wedge 1 \right) - \widetilde{f}_n\|_\infty = \| \left(\widetilde{f}_n \cdot \frac{\|f_0\|_\infty}{\|\widetilde{f}_n\|_\infty} - \widetilde{f}_n \right) \wedge \left(\widetilde{f}_n - \widetilde{f}_n \right) \|_\infty \\
&\leq \|\widetilde{f}_n \cdot \frac{\|f_0\|_\infty}{\|\widetilde{f}_n\|_\infty} - \widetilde{f}_n\|_\infty = \|\widetilde{f}_n \left(1 - \frac{\|f_0\|_\infty}{\|\widetilde{f}_n\|_\infty} \right) \|_\infty = \|\widetilde{f}_n \left(\frac{\|\widetilde{f}_n\|_\infty - \|f_0\|_\infty}{\|\widetilde{f}_n\|_\infty} \right) \|_\infty \\
&\leq \|\widetilde{f}_n\|_\infty \cdot \| \frac{\|\widetilde{f}_n\|_\infty - \|f_0\|_\infty}{\|\widetilde{f}_n\|_\infty} \|_\infty = \| \|\widetilde{f}_n\|_\infty - \|f_0\|_\infty \|_\infty \\
&\leq \|\widetilde{f}_n - f_0\|_\infty
\end{aligned}$$

erhalten wir

$$\begin{aligned}
\|f_n^* - f_0\|_\infty &= \|(f_n^* - \widetilde{f}_n) + (\widetilde{f}_n - f_0)\|_\infty \\
&\leq \|f_n^* - \widetilde{f}_n\|_\infty + \|\widetilde{f}_n - f_0\|_\infty \\
&\leq 2\|\widetilde{f}_n - f_0\|_\infty. \qquad (27)
\end{aligned}$$

Das zeigt, dass (25) auch gilt (mit Konstanten multipliziert mit 8), falls das Infimum über die beschränktere Netzwerkklasse $F(L, \mathbf{p}, s, F)$ genommen wird:

$$\begin{aligned}
\inf_{f^* \in F(L,\mathbf{p},s,F)} \|f^* - f_0\|_\infty^2 &\underset{(27)}{\leq} \inf_{\widetilde{f} \in F(L,\mathbf{p},s)} 4\|\widetilde{f} - f_0\|_\infty^2 \\
&\underset{(26)}{\leq} 8C \max_{i=0,\ldots,q} c^{-\frac{2\beta_i^*}{t_i}} n^{-\frac{2\beta_i^*}{2\beta_i^* + t_i}}.
\end{aligned}$$

Zusammen mit (20) folgt

$$\begin{aligned}
R(\widehat{f}_n, f_0) &\leq 4 \inf_{f^* \in F(L,\mathbf{p},s,F)} \|f^* - f_0\|_\infty^2 + 4\Delta_n(\widehat{f}_n, f_0) + C'\phi_n L \log^2(n) \\
&\leq 32C \max_{i=0,\ldots,q} c^{-\frac{2\beta_i^*}{t_i}} n^{-\frac{2\beta_i^*}{2\beta_i^* + t_i}} + 4\Delta_n(\widehat{f}_n, f_0) + C'\phi_n L \log^2(n).
\end{aligned}$$

Für jede beliebige Konstante $\widetilde{C}$ folgt mit der Bedingung $\Delta_n(\widehat{f}_n, f_0) \leq \widetilde{C}\phi_n L \log^2(n)$ offensichtlich die obere Schranke von (14) und mit der Bedingung $\Delta_n(\widehat{f}_n, f_0) \geq \widetilde{C}\phi_n L \log^2(n)$ die obere Schranke von (15). □

4 Beispiele

In diesem Abschnitt werden wir uns Modelle anschauen, die eine stärke strukturelle Zerlegung der Regressionsfunktion f_0 formulieren als wir es im Abschnitt 3 getan haben. Es sind somit Spezialfälle.

4.1 Additive Modelle

Bei additiven Modellen wird angenommen, dass die Regressionsfunktion $f_0(\mathbf{x})$ in eine Summe zerfällt, deren Summanden nur noch von einer Komponente von $\mathbf{x}$ abhängen.

Definition 2. *[15, S. 217]*
Es sei $K > 0$. Es sei $f_0(\mathbf{x}) = \sum_{j=1}^{d} g_{0j}(x_j)$ mit stetig differenzierbaren Funktionen $g_{0j} : \mathbb{R} \to [-K, K]$. Dann gilt

$$f_0 = g_1 \circ g_0,$$

wobei

$$g_0 : [0,1]^d \to \mathbb{R}^d, \quad g_0(\mathbf{x}) = g(g_{01}(x_1), ..., g_{0d}(x_d))^\top$$
$$g_1 : \mathbb{R}^d \to \mathbb{R}, \qquad g_1(\mathbf{y}) = \sum_{j=1}^{d} y_j$$

Die einzelnen Komponenten von g_0 sind zwar nur einmal differenzierbar, aber haben auch nur den Definitionsbereich $\mathbb{R}$. Dahingegen besitzt g_1 zwar den Definitionsbereich $\mathbb{R}^d$, ist aber als Summenfunktion unendlich oft differenzierbar.

Da die Funktionen $g_{0j}, j = 1, ..., d$ stetig differenzierbar sind mit dem Wertebereich $[-K, K]$, gilt für $g_0 : [0,1]^d \to [-K, K]^d$. Damit können wir für $g_1 : [-K, K]^d \to [-Kd, Kd]$ wählen. Die Dimensionen des Modells sind somit $d_0 = d$, $d_1 = d$, $d_2 = 1$. Die einzelnen Komponenten von g_0 sind stetig differenzierbar und hängen nur jeweils von einer Komponente ab, daher gilt für die effektive Dimension $t_0 = 1$ und für die Glattheit $\beta_0 = 1$. Die Funktion g_1 ist unendlich oft differenzierbar und besteht aus einer Summe von d Terme. Es gilt somit für die effektive Dimension $t_1 = d$ und $\beta_1 > 1$ beliebig groß. Zusammenfassend können wir sagen, dass die Regressionsfunktion in der Klasse $G(1, (d, d, 1), (1, d), (1, \beta_1), (K+1)d)$ liegt.
Der effektive Glattheitsindex liegt bei $\beta_0^* = 1 \cdot (\beta_1 \wedge 1) = 1$ und $\beta_1^* > 1$ kann beliebig groß gewählt werden. Als nächstes werden wir die Konvergenzrate

$$\phi_n = \max_{i=0,1} n^{-\frac{2\beta_i^*}{2\beta_i^* + t_i}}$$

berechnen. Es gilt mit den effektiven Glattheitsindizes

$$\min_{i=0,1} \frac{2\beta_i^*}{2\beta_i^* + t_i} = \min\left\{\frac{2}{3}, \frac{2\beta_1^*}{2\beta_1^* + d}\right\} = \frac{2}{3}$$

und damit folgt für die Konvergenzrate $\phi_n = n^{-2/3}$. Werden nun die Bedingungen

(i) $F \geq (K+1)d$

(ii) L *proportional zu* $\log_2(n)$*:* $\log_2(n)\,(\log_2(4)+\log_2(4d\vee 4\beta_1))\le L\lesssim \log_2(n)$

(iii) $n^{1/3}\lesssim \min_i p_i$

(iv) $s\asymp n^{1/3}\log(n)$

durch ein Netzwerkarchitektur $F(L,\mathbf{p},s,F)$ erfüllt, dann können wir die 2 verschiedenen Fälle im Theorem 1 für das Netzwerk zu

$$R(\widehat{f}_n,f_0)\lesssim n^{-2/3}\log(n)^3+\Delta_n(\widehat{f}_n,f_0)$$

zusammenfassen.

4.2 Verallgemeinerte additive Modelle

Wir nehmen an, dass die Regressionsfunktion in der Form

$$f_0(x_1,...,x_d)=h(\sum_{j=1}^{d} g_{0j}(x_j))$$

geschrieben werden kann, wobei $h:\mathbb{R}\to\mathbb{R}$ eine unbekannte Funktion ist. Wir können die Regressionsfunktion als Komposition $f_0=g_2\circ g_1\circ g_0$ mit g_0, g_1, wie im Abschnitt 4.1 definiert, und $g_2=h$ schreiben. Wir werden diesmal annehmen, dass wir eine allgemeinere Form für die $g_{0j}\in C_1^{\beta}([0,1],K),\ j=1,...,d$ haben. Sei nun $h\in C_1^{\gamma}(\mathbb{R},K)$, dann gilt $f_0:[0,1]^d\overset{g_0}{\to}[-K,K]^d\overset{g_1}{\to}[-Kd,Kd]\overset{g_2}{\to}[-K,K]$. Für jedes beliebige $\beta_1>1$ gilt $g_1\in C_d^{\beta_1}([-K,K]^d,(K+1)d)$. Wie bei den additiven Modellen begründet, gilt

$$f_0\in G\,(2,(d,d,1,1),(1,d,1),(\beta,(\beta\vee 2)d,\gamma),(K+1)d)\,.$$

Für die effektiven Glattheitsindizes gilt:

$$\begin{aligned}
\beta_0^* &= \beta\left(\underbrace{((\beta\vee 2)d)}_{\ge 2}\wedge 1\right)(\gamma\wedge 1)=\beta(\gamma\wedge 1)\\
\beta_1^* &= (\beta\vee 2)d(\gamma\wedge 1)\\
\beta_2^* &= \gamma
\end{aligned}$$

Wir können nun für die Konvergenzrate folgern, dass

$$\begin{aligned}
&\min\left\{\frac{2\beta(\gamma\wedge 1)}{2\beta(\gamma\wedge 1)+1},\frac{2(\beta\wedge 2)d(\gamma\wedge 1)}{2(\beta\wedge 2)d(\gamma\wedge 1)+d},\frac{2\gamma}{2\gamma+1}\right\}\\
&=\min\left\{\frac{2\beta(\gamma\wedge 1)}{2\beta(\gamma\wedge 1)+1},\frac{2(\beta\wedge 2)(\gamma\wedge 1)}{2(\beta\wedge 2)(\gamma\wedge 1)+1},\frac{2\gamma}{2\gamma+1}\right\}\\
&=\min\left\{\frac{2\beta(\gamma\wedge 1)}{2\beta(\gamma\wedge 1)+1},\frac{2\gamma}{2\gamma+1}\right\}
\end{aligned}$$

und damit

$$\phi_n = \max_{i=0,1,2} n^{-\frac{2\beta_i^*}{2\beta_i^*+t_i}} \leq n^{-\frac{2\beta(\gamma\wedge 1)}{2\beta(\gamma\wedge 1)+1}} + n^{-\frac{2\gamma}{2\gamma+1}}.$$

Für Netzwerkarchitekturen, die die Bedingungen aus Theorem 1 erfüllen, gilt

$$R(\widehat{f}_n, f_0) \lesssim \left(n^{-\frac{2\beta(\gamma\wedge 1)}{2\beta(\gamma\wedge 1)+1}} + n^{-\frac{2\gamma}{2\gamma+1}}\right) \log^3(n) + \Delta(\widehat{f}_n, f_0).$$

5 Ausblick

In diesem Abschnitt wollen wir diskutieren, welchen Wert die Arbeit an sich und die Ergebnisse, die hier präsentiert wurden, im Kontext der statistischen Ausarbeitung von Methoden des Deep Learnings darstellt. Zum Schluss werden wir uns mit möglichen Verbesserungen und Erweiterungen des Modells auseinandersetzen, um die Lücke zwischen der Theorie und der praktischen Implementation weiter schließen zu können.

In der Arbeit haben wir eine ausführliche Ausarbeitung und Erklärung des wissenschaftlichen Artikels „Nonparametric regression using deep neural networks with ReLU activation function“ von Johannes Schmidt-Hieber mit vielen Ergänzungen aus verschiedenen Quellen vorgenommen. Zudem haben wir grundlegende Begriffe definiert und erklärt, die essentiell sind für das Verständnis des Artikels. Diese Arbeit dient gewissermaßen als ein erleichterter Einstieg in die statistische Modellierung neuronaler Netzwerke und deren Approximation. Unter einer hierarchischen Struktur der Regressionsfunktion kann sich das tiefe neuronale Netzwerke der zugrunde liegenden Struktur im Signal anpassen und kann dadurch den Fluch der Dimensionalität umgehen. Die Resultate zeigen zum ersten Mal, dass man eine (fast) optimale Konvergenzrate mit dünnbesetzten multilayer neuronalen Netzwerken mit ReLU Aktivierungsfunktion erreicht. Für den Beweis benutzt man dabei neue Approximationen für multilayer feedforward neuronale Netzwerke mit beschränkten Gewichten und beschränkter Breite der Layer. Es gab zwar schon frühere statistische Arbeiten, jedoch wird nun ein viel allgemeineres und der Praxis entsprechendes Setting angenommen als es in früheren Arbeiten vorgenommen wurde. Wichtige Folgerungen, unter den selben Rahmenbedingungen wie im Haupttheorem 1, sind einmal, dass die hidden Layers mit den Stichprobenumfang mit $L \asymp \log_2(n)$ wachsen sollte und dass das Netzwerk mehr Parameter haben kann als die Anzahl an Stichproben. Eine weitere wichtige Erkenntnis ist, dass für die statistische Performance nicht die Größe des Netzwerkes die wichtigste Rolle spielt, sondern die Regulation des Netzwerkes, d.h. die Anzahl an aktiven Parameter. Die Ergebnisse geben uns auch einen Erklärungsansatz, warum solche Netzwerke in der Praxis gut funktionieren.

Trotz den allgemeinen Annahmen haben wir in dem Modell, wie jede Modellierung, Einschränkungen vorgenommen, die für verschiedene Fragestellungen potenziell ungeeignet

sind. Es könnte beispielsweise ein Klassifizierungs- anstatt Regressionsproblem vorliegen. Eine Erweiterung des Modells auf Klassifizierungsprobleme könnte man somit in Betracht ziehen. Es wäre dabei interessant zu untersuchen, wie sich die Resultate ändern, falls im letzten Layer eine Softmax-Aktivierungsfunktion angewandt wird.
Eine weitere wesentliche Einschränkung ist, dass wir ein tiefes *feedforward* neuronales Netzwerk betrachten. Viele der neusten Deep Learning Anwendungen basieren aber auf anderen spezifische Netzwerke, wie zum Beispiel ein faltendes neuronales Netzwerk oder rekurrentes neuronales Netzwerk. Es ist also natürlich sich die Frage zu stellen, wie man solche Arten von Netzwerken statistisch untersucht und dort ebenfalls die Konvergenzrate analysiert.
Eine offensichtliche Verbesserung an dem Modell wäre die Ungenauigkeit, die wir schon im Abschnitt 3 erwähnt haben, in unserer unteren bzw. oberen Schranke zu verbessern. Heuristische Resultate zeigen, dass wahrscheinlich der Faktor $L\log(n)^2$ in der oberen Schranke ein Artefakt des Beweises ist.

Die Theorie hinter tiefen Netzwerken hat viele verschiedene Bereiche, die eng miteinander zusammenhängen. Die Entwicklung dieser Bereiche könnte auch Auswirkungen auf die in dieser Arbeit betrachteten Fragestellungen, weshalb wir die möglichen Einflüsse von den verschiedenen Themenbereichen diskutieren wollen.
Die Approximationstheorie von Netzwerken hat zwar schon viele Resultate, die bis zu den frühen 1990er zurückreichen, sind dennoch immer noch aktueller Forschungsgegenstand. Im Beweis hat man dabei auch Gebrauch gemacht von neuen Approximationstheorien. Es wäre also möglich, dass wir auf Grundlage von zukünftigen Resultaten in der Approximationstheorie die Ungenauigkeit in unseren Schranken verbessern könnten.
Ein weiterer Themenbereich, der für die Resultate dieser Arbeit interessant sein könnte, ist die Forschung des *stochastic gradient descent*. Hierbei beschäftigt man sich heutzutage mit beispielsweise dem Entkommen von Sattelpunkten oder lokale Minima. Es könnten also Aussagen getroffen werden, wie stark der Term $\Delta_n(\widehat{f}_n, f_0)$ in n sinkt unter Anwendung des *stochastic gradient descent* oder sogar anderen Methoden.
Die Analyse würde aber stark von der angewandten Methode abhängen. Eine unabhängige Analyse könnte durch die Analyse des *loss landscape* tiefer Netzwerke erfolgen, d.h. wie Sattelpunkte, lokale und globale Minima der Verlustfunktion angeordnet sind und wie wir sie charakterisieren können. Wir könnten gegebenenfalls eine obere Schranke für den Term $\Delta_n(\widehat{f}_n, f_0)$ bekommen und diese im Theorem 1 berücksichtigen. Tatsächlich gibt es erste heuritische Ergebnisse für *fully connected* Netzwerke, die eine Verbindung zu *spherical spin glasses* sehen, vgl. [2], welche schon ausführlich untersucht wurden. Dort liegen alle lokale Minima mit einer hohen Wahrscheinlichkeit in einem Band, das offensichtlich nach unten beschränkt ist durch das globale Minima. Die Breite des Bandes hängt von der Breite des Netzwerkes ab und gibt uns auch eine obere Schranke des Terms $\Delta_n(\widehat{f}_n, f_0)$ für alle Methoden an, die zu einen lokalen Minima konvergieren. Falls wir dieses heuristische Ergebnis mathematisch festigen können, dann könnten wir die Abhängigkeit des Terms $\Delta_n(\widehat{f}_n, f_0)$ vom *width vector* $\mathbf{p}$ in unserer Konvergenzrate berücksichtigen.

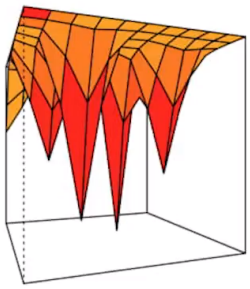

Abbildung 14: Beispielhaftes *loss landscape* eines neuronalen Netzwerkes.[10]

Wir sehen, dass viele Fragen, die aus den verschiedenen Bereichen des Deep Learning auftauchen, noch keine zufriedenstellende Antwort haben. Diese Arbeit soll gewissermaßen ein Start sein, um statistische Modelle aufzubauen, mit der man verwandte Probleme lösen kann, wie beispielsweise die Untersuchung der Konvergenzrate unter anderen Netzwerktypen.

[10] [17, Folie 10]

Literatur

[1] Ardakani, A., Condo, C. und Gross, W. J.: *Sparsely-Connected Neural Networks: Towards Efficient VLSI Implementation of Deep Neural Networks.* arXiv:1611.01427, 30. März 2017.

[2] Choromanska, A., Henaff, M., Mathieu, M., Arous, G. B. und LeCun Y.: *The Loss Surfaces of Multilayer Networks.* arXiv:1412.0233 , Januar 2015.

[3] Eldan, R. und Shamir, O.: *The Power of Depth for Feedforward Neural Networks* arXiv:1512.03965, Mai 2016.

[4] Nwankpa, C., Ijomah, W., Gachagan, A. und Marshall, S.: *Activation Functions: Comparison of Trends in Practice and Research for Deep Learning.* arXiv:1811.03378, November 2018.

[5] Glorot, X., Bordes, A., und Bengio, Y.: *Deep sparse rectifier neural networks.* In Aistats, 2011, vol. 15, S. 315-323.

[6] Goodfellow, I., Bengio, Y. und Courville, A.. *Deep Learning.* MIT Press, 2016.

[7] Györfi, L., Kohler, M., Krzyzak, A. und Walk, H.: *A Distribution-Free Theory of Nonparametric Regression.* Springer Series in Statistics. Springer-Verlag, New York, 2002.

[8] He, K., Zhang, X., Ren, S. und Sun, J.: *Deep Residual Learning for Image Recognition.* In Aistats, 2011, vol. 15, S. 315-323.

[9] Juditsky, A., B., Lepski, O. V. und Tsybakov, A., B.: *Nonparametric estimation of composite functions.* The Annals of Statistics, 2009, Vol. 37, No. 3, 1360?1404.

[10] Kohler, Michael: *Kurvenschätzung.* Unveröffentlichtes Skript, Universität Darmstadt. Erschienen im Sommersemester 2015.

[11] Krizhevsky, A., Sutskever, I., und Hinton, G. E: *ImageNet Classification with Deep Convolutional Neural Networks.* In Advances in Neural Information Processing Systems 25, F. Pereira, C. J. C. Burges, L. Bottou, und K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, S. 1097-1105.

[12] Lin, H. W., Tegmark, M. und Rolnick, D.: *Why does deep and cheap learning work so well?* arXiv:1608.08225, August 2017.

[13] Mehlig, Bernhard: *Artificial Neural Networks.* arXiv:1901.05639, Februar 2019.

[14] Pinkus, Allan: *Approximation theory of the MLP model in neural networks.* Acta Numerica, 1999, 143-195.

[15] Richter, Stefan: *Statistisches und maschinelles Lernen, Version 1.* Springer-Verlag Berlin Heidelberg, September 2018.

[16] Schmidt-Hieber, Johannes: *Nonparametric regression using deep neural networks with relu activation function.* arXiv:1708.06633, März 2019.

[17] Schmidt-Hieber, Johannes: *Statistical theory for deep neural networks with ReLU activation function.* MIFODS - Stochastics and Statistics joint seminar, Cambridge US, 23. März 2018.

[18] Schmidthuber, Jürgen: *Deep Learning in Neural Networks: An Overview* arXiv:1404.7828, Oktober 2014.

[19] Stone, C. J.: *Optimal global rates of convergence for nonparametric regression.* Annals of Statistics, 10, S.1040-1053. 1982.

[20] Strauch, Claudia: *Nonparametric Statistics.* Lecture notes, Universität Mannheim. Erschienen am 13. Mai 2019.

[21] Tegmark, M., Lin, H., W. und Rolnick D.: *Why does deep and cheap learning work so well?* arXiv:1608.08225v4, 3. August 2017.

[22] Tenenbaum, J., Tegmark, M. und Poggio, T.: *CBMM Research Meeting: On Compositionality (Debate/Discussion)* CBMM Special Seminars, 16. Dezember 2016.

[23] Woernle, Maike Inga: *Anwendbarkeit künstlicher neuronaler Netze zur Untergrundbewertung in der oberflächennahen Geothermie* unveröffentlichte Dissertation, Universität Fridericiana zu Karlsruhe, 2008.