TECHNISCHE
UNIVERSITÄT
DARMSTADT

Fachbereich Mathematik

Masterarbeit

Analyse der Konvergenzgeschwindigkeit eines einfach berechenbaren Neuronale-Netze-Regressionsschätzers

Adrian Gabel

19. April 2020

Betreuer: Prof. Dr. Michael Kohler

Erklärung zur Abschlussarbeit gemäß § 22 Abs. 7 und § 23 Abs. 7 APB TU Darmstadt

Hiermit versichere ich, Adrian Gabel, die vorliegende Master-Thesis gemäß § 22 Abs. 7 APB der TU Darmstadt ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.
Mir ist bekannt, dass im Falle eines Plagiats (§ 38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.
Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß § 23 Abs. 7 APB überein.

Hamburg, 19. April 2020

Adrian Gabel

Inhaltsverzeichnis

Einleitung

Als programmierbare Computer zum ersten Mal konzipiert wurden, fragten sich die Menschen, ob solche Maschinen intelligent werden könnten, mehr als hundert Jahre bevor man sie baute. Heute ist die künstliche Intelligenz (*KI*) ein blühendes Feld mit vielen praktischen Anwendungen und aktiven Forschungsthemen. Künstliche Intelligenz ist längst in unserem Alltag präsent und dringt in immer mehr Bereiche vor. Sprachassistenten etwa sind bereits als Helfer auf dem Smartphone, im Auto oder zu Hause Normalität geworden. Fortschritte im Bereich der KI beruhen vor allem auf der Verwendung *neuronaler Netze*. Vergleichbar mit der Funktionsweise des menschlichen Gehirns verknüpfen sie mathematisch definierte Einheiten miteinander [GBC16].

In vielen Anwendungen der KI geht es darum, aus einer Menge von Daten eine allgemeine Regel abzuleiten (*maschinelles Lernen*). Maschinelles Lernen kann *als Lernen einer Funktion* f zusammengefasst werden, die Eingangsvariablen X auf Ausgangsvariablen Y abbildet, sodass die Abbildungsvorschrift $f(X) = Y$ entsteht.

Ein Algorithmus lernt diese Zielabbildungsfunktion aus Trainingsdaten. Die Form der Funktion ist unbekannt, sodass es unsere Aufgabe als Praktiker des maschinellen Lernens ist, verschiedene Algorithmen des maschinellen Lernens zu evaluieren und zu sehen, welcher die zugrunde liegende Funktion besser annähert. Unterschiedliche Algorithmen machen unterschiedliche Annahmen über die Form der Funktion und die Art und Weise, wie sie gelernt werden kann.

Algorithmen, die keine Annahmen über die Form der Abbildungsfunktion treffen, werden als nichtparametrische Algorithmen des maschinellen Lernens bezeichnet. Indem sie keine Annahmen treffen, können sie jede beliebige Funktionsform aus den Trainingsdaten lernen. Nichtparametrische Methoden sind gut, wenn viele Daten verfügbar sind und man sich nicht allzu sehr um die Auswahl der richtigen Funktionen kümmern will [RN09, Seite 757]. Mathematisch führt dies zu einem Approximationsproblem.

Im Kontext der KI wurden hierzu unter anderem neuronale Netze vorgeschlagen, die als universale Funktionsapproximatoren (*Schätzer*) eingesetzt werden können, jedoch insbesondere bei vielen verdeckten Schichten schwer zu analysieren sind. Dies führt

unter anderem dazu, dass eine große Lücke zwischen den Schätzungen, für die schöne Konvergenzergebnisse in der Theorie nachgewiesen wurden, und den Schätzungen entsteht, die in der Praxis verwendet werden können.
Ziel dieser Arbeit ist es, die folgende Frage genauer zu betrachten: Wenn wir eine Regressionsschätzung des neuronalen Netzes theoretisch genau so definieren, wie sie in der Praxis umgesetzt wird, welches Konvergenzergebnis können wir dann für diese Schätzung vorweisen?
Als Erstes werden wir in Kapitel 1 Grundlagen zu neuronalen Netzen für den weiteren Verlauf der Arbeit sammeln. Anschließend definieren wir in Kapitel 2 eine neue, leicht zu implementierende Neuronale-Netze-Regressionsschätzung. Die Besonderheit des dem Schätzer zugrunde liegenden neuronalen Netzes besteht darin, dass die meisten Gewichte unabhängig von den Daten gewählt werden. In Kapitel 3 zeigen wir, dass wir für diesen Schätzer Konvergenzraten ableiten können, falls die Regressionsfunktion bestimmte Glattheitsvoraussetzungen erfüllt. Abschließend untersuchen wir in Kapitel 4 die Leistung unseres Neuronale-Netze-Regressionsschätzers aus Kapitel 2 anhand von Anwendungsbeispielen auf simulierte Daten. Diese Arbeit orientiert sich an [BKK19].

Kapitel 1

Grundlagen für neuronale Netze

Das Ziel dieses Kapitels ist es, auf die in dieser Arbeit verwendeten neuronalen Netze einzugehen, die als Bausteine für unseren einfach berechenbaren Neuronale-Netze-Regressionsschätzer verwendet werden. Weiterhin werden wir Approximationsresultate darstellen und beweisen, welche wir für das Resultat der Konvergenzgeschwindigkeit des einfach berechenbaren Neuronale-Netze-Regressionsschätzers benötigen werden. Wenn wir in dieser Arbeit von *unserem Neuronale-Netze-Regressionsschätzer* sprechen, beziehen wir uns immer auf den *einfach berechenbaren Neuronale-Netze-Regressionsschätzer* aus [BKK19]. In dieser Arbeit bezeichnen wir mit $d \geq 1$ immer eine natürliche Zahl.
Da wir uns in dieser Arbeit mit neuronalen Netzen beschäftigen, ist es hilfreich zu wissen, was man darunter versteht. Ein neuronales Netz ist nichts anderes als eine Ansammlung von Neuronen, welche als Informationsverarbeitungseinheiten dienen, die schichtweise in einer Architektur angeordnet sind. Beginnend mit der Eingabeschicht (*Input Layer*) fließen Informationen über eine oder mehrere verborgene Schichten (*Hidden Layer*) bis hin zur Ausgabeschicht (*Output Layer)*. Die Informationsweitergabe der Neuronen verläuft wie folgt: Für jedes Neuron j werden die Eingaben $x_1, \ldots, x_n$ mit $w_{1_j}, \ldots, w_{n_j}$ gewichtet an eine Aktivierungsfunktion σ übergeben, welche die Neuronenaktivierung berechnet. Die Eingaben können aus dem beobachteten Prozess resultieren oder andererseits aus den Ausgaben anderer Neuronen stammen. Der Endpunkt des Informationsflusses in einem neuronalen Netz ist die Ausgabeschicht, die hinter den verborgenen Schichten liegt. Sie bildet damit die letzte Schicht in einem neuronalen Netz. Die Ausgabeschicht enthält somit das Ergebnis der Informationsverarbeitung durch das Netz. Zwei wichtige Charakteristika, die neuronale Netze aufweisen können, sind:

- Wenn in einem neuronalen Netz die Information von der Eingabeschicht über die verborgenen Schichten bis hin zur Ausgabeschicht in eine Richtung („vorwärts“) weitergereicht wird, spricht man von einem *feedforward* neuronalen Netz.

- Ein neuronales Netz wird als ein *Fully-connected* („vollständig verbundenes“) neuronales Netz bezeichnet, wenn sämtliche Neuronen einer Schicht mit allen der darauffolgenden verbunden sind. Da man bei Fully-connected neuronalen Netzen die Gewichte der Verbindungen zwischen zwei Neuronen auf Null setzen kann, unterscheiden wir hier nicht mehr zwischen neuronalen Netzen die Fully-connected sind oder nicht.

Als Nächstes kommen wir zur mathematischen Definition eines neuronalen Netzes, welche wir in dieser Form im weiteren Verlauf dieser Arbeit benötigen werden.

Definition 1.1. Sei $L \in \mathbb{N}$, $\mathbf{k} \in \mathbb{N}^L$ und $\sigma\colon \mathbb{R} \to \mathbb{R}$. Ein *mehrschichtiges feedforward neuronales Netz mit Architektur* $(L,\mathbf{k})$ *und Aktivierungsfunktion* σ, ist eine reellwertige Funktion $f\colon \mathbb{R}^d \to \mathbb{R}$ definiert durch:

$$f(x) = \sum_{i=1}^{k_L} c_i^{(L)} \cdot f_i^{(L)}(x) + c_0^{(L)}$$

mit Gewichten $c_0^{(L)}, \ldots, c_{k_L}^{(L)} \in \mathbb{R}$ und für $f_i^{(L)}$ mit $i = 1, \ldots, k_L$ rekursiv definiert durch:

$$f_i^{(r)}(x) = \sigma_r\left(\sum_{j=1}^{k_r - 1} c_{i,j}^{(r-1)} \cdot f_j^{(r-1)}(x) + c_{i,0}^{(r-1)} \right)$$

mit Gewichten $c_{i,0}^{(r-1)}, \ldots, c_{i,k_{r-1}}^{(r-1)} \in \mathbb{R}$ mit $r = 2, \ldots, L$, wobei $\sigma_r \in \{\sigma, \mathrm{id}\}$ und:

$$f_i^{(1)}(x) = \sigma_1\left(\sum_{j=1}^{d} c_{i,j}^{(0)} \cdot x^{(j)} + c_{i,0}^{(0)} \right)$$

für die Gewichte $c_{i,0}^{(0)}, \ldots, c_{i,d}^{(0)} \in \mathbb{R}$, wobei $\sigma_1 \in \{\sigma, \mathrm{id}\}$. Weiterhin sei $\mathfrak{N}(L, \mathbf{k}, \sigma)$ die Klasse aller mehrschichtigen feedforward neuronalen Netze mit Architektur $(L,\mathbf{k})$ und Aktivierungsfunktion σ.

Wir werden ab jetzt der Einfachheit halber nur noch von *neuronalen Netzen mit Architektur* $(L,\mathbf{k})$ reden, wenn die Aktivierungsfunktion σ aus dem Kontext bekannt ist oder keine Rolle spielt und beziehen uns damit immer auf Definition 1.1. Wenn aus dem Kontext die *Architektur* $(L,\mathbf{k})$ und die *Aktivierungsfunktion* σ bereits bekannt ist, sprechen wir nur noch von einem *neuronalen Netz*, beziehen uns damit aber dennoch auf Definition 1.1.

Bemerkung 1.2. In obiger Definition versteht man unter:

- $L \in \mathbb{N}$ die Anzahl an verborgenen Schichten von f.
- $\mathbf{k} = (k_1, \ldots, k_L) \in \mathbb{N}^L$ einen Vektor, der die Anzahl an Neuronen in jeder verborgenen Schicht angibt.

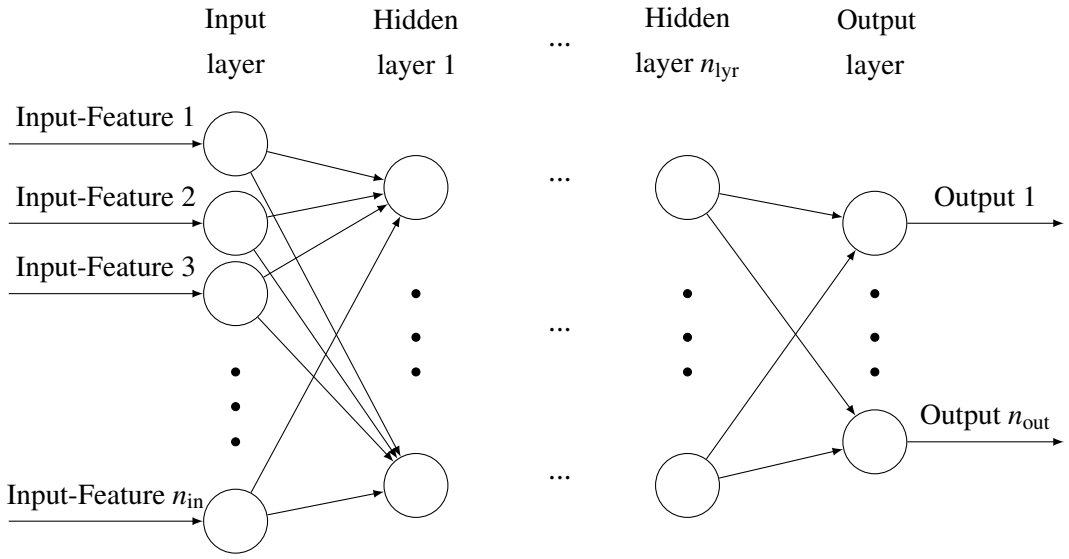

Abbildung 1.1: Mehrschichtiges feedforward neuronales Netz mit einer Eingabeschicht mit n_{in} Neuronen, n_{lyr} vielen verborgenen Schichten, deren Anzahl an Neuronen variieren kann und einer Ausgabeschicht bestehend aus n_{out} Neuronen.

Ist $f = \sum_{i=1}^{n} a_i f_i$ eine Linearkombination von neuronalen Netzen $f_1, \ldots, f_n \in \mathfrak{N}(L, \mathbf{k}, \sigma)$ mit Linearfaktoren $a_1, \ldots, a_n \in \mathbb{R}$, so können wir f als ein neuronales Netz mit Architektur $(L+1, (k_1, \ldots, k_L, n))$ betrachten, wobei in diesem Fall $\sigma_L = \text{id}$ gilt.

Da wir bei Neuronale-Netze-Regressionsschätzern Funktionswerte schätzen möchten, haben wir in der Ausgabeschicht nur ein Neuron. Damit erreichen wir einen eindimensionalen Output. Wie wir an der Konstruktion des neuronalen Netzes in Definition 1.1 erkennen können, nehmen wir in der Ausgabeschicht die Identität als Aktivierungsfunktion.
Abbildung 1.1 zeigt schematisch ein mehrschichtiges feedforward neuronales Netz, welches aus einer Eingabeschicht (*Input-Feature 1* - *Input-Feature* n_{in}), n_{lyr} verborgenen Schichten und einer Ausgabeschicht (*Output 1* - *Output* n_{out}) besteht.

Wie zuvor erwähnt ist einer der Ausgangspunkte für die Definition eines neuronalen Netzes die Wahl einer Aktivierungsfunktion $\sigma\colon \mathbb{R} \to \mathbb{R}$. Wir kommen nun zu einer häufig vorausgesetzten Eigenschaft an Aktivierungsfunktionen.

Definition 1.3. Sei $N \in \mathbb{N}_0$. Eine Funktion $\sigma\colon \mathbb{R} \to [0,1]$ wird *N-zulässig* genannt, wenn sie monoton wachsend und lipschitzstetig ist und wenn zusätzlich die folgenden drei Bedingungen erfüllt sind:

(i) Die Funktion σ ist $(N+1)$-mal stetig differenzierbar mit beschränkten Ableitungen.

(ii) Es existiert ein Punkt $t_\sigma \in \mathbb{R}$ mit

$$\sigma^{(i)}(t_\sigma) \neq 0 \text{ für } i = 0, \ldots, N.$$

(iii) Wenn $y > 0$ ist, gilt $|\sigma(y) - 1| \leq \frac{1}{y}$. Wenn $y < 0$ ist, gilt $|\sigma(y)| \leq \frac{1}{|y|}$.

Ein in dieser Arbeit wichtiges Beispiel für eine Aktivierungsfunktion bildet der sogenannte *sigmoidal* bzw. *logistische Squasher*:

$$\sigma(x) = \frac{1}{1+\exp(-x)} \quad (x \in \mathbb{R}). \tag{1.1}$$

Lemma 1.4. *Sei $n \in \mathbb{N}$. Dann existiert zur n-ten Ableitung $\sigma^{(n)}$ des logistischen Squashers σ aus Gleichung* (1.1) *ein Polynom P_n vom Grad $(n+1)$, sodass $P_n(\sigma) = \sigma^{(n)}$ gilt. Insbesondere ist P_n ungleich dem Nullpolynom für alle $n \in \mathbb{N}$.*

Beweis. Wir zeigen die Aussage per Induktion über n.
Induktionsanfang (IA): Die Funktion σ erfüllt die gewöhnliche Differentialgleichung $\sigma^{(1)} = (1-\sigma)\cdot\sigma$, da:

$$\begin{aligned}\sigma^{(1)}(x) &= -\frac{1}{(1+\exp(-x))^2}\cdot(-\exp(-x)) = \frac{\exp(-x)}{1+\exp(-x)}\cdot\frac{1}{1+\exp(-x)} \\ &= \left(1-\frac{1}{1+\exp(-x)}\right)\cdot\frac{1}{1+\exp(-x)} = (1-\sigma(x))\cdot\sigma(x).\end{aligned}$$

Damit gilt $\sigma^{(1)} = P_1(\sigma)$ mit $P_1(x) = x - x^2$. Es gilt $\deg(P_1) = 2$, da der führende Koeffizient von P_1 ungleich Null ist.
Induktionshypothese (IH): Die Aussage gelte für ein beliebiges aber festes $n \in \mathbb{N}$.
Induktionsschritt (IS): Nach der Induktionshypothese existiert ein Polynom P_{n-1} mit $P_{n-1}(x) = \sum_{k=0}^{n} a_k \cdot x^k$ und $\sigma^{(n-1)} = P_{n-1}(\sigma)$. Mit der Kettenregel aus der Differentialrechnung erhalten wir:

$$\begin{aligned}\sigma^{(n)} = (\sigma^{(n-1)})' &\overset{\text{(IH)}}{=} \Big(\sum_{k=0}^{n} a_k\cdot\sigma^k\Big)' = \sum_{k=1}^{n} a_k\cdot k\cdot\sigma^{k-1}\cdot\sigma' \overset{\text{(IA)}}{=} \sum_{k=1}^{n} a_k\cdot k\cdot\sigma^{k-1}\cdot\sigma\cdot(1-\sigma) \\ &= \sum_{k=1}^{n} a_k\cdot k\cdot\sigma^k - \sum_{k=1}^{n} a_k\cdot k\cdot\sigma^{k+1}. \end{aligned} \tag{1.2}$$

Gleichung (1.2) definiert ein Polynom P_n vom Grad $(n+1)$ mit $\sigma^{(n)} = P_n(\sigma)$. Damit haben wir die Aussage für alle $n \in \mathbb{N}$ bewiesen. □

Lemma 1.5. *Sei $N \in \mathbb{N}_0$ beliebig, dann ist der logistische Squasher σ aus Gleichung* (1.1) *N-zulässig.*

Beweis. Sei $N \in \mathbb{N}_0$ beliebig. Dadurch, dass $0 \leq \frac{1}{1+\exp(-x)} \leq 1$ für alle $x \in \mathbb{R}$ gilt, erhalten wir schließlich auch $\sigma : \mathbb{R} \to [0,1]$. Wir wissen, dass σ monoton wachsend ist, da für beliebige $s,t \in \mathbb{R}$ mit $s \leq t$ gilt:

$$\sigma(s) = \frac{1}{1+\exp(-s)} \leq \frac{1}{1+\exp(-t)} = \sigma(t),$$

wobei wir verwendet haben, dass aufgrund der Monotonie der Exponentialfunktion für $s \leq t$ auch $\exp(-s) \geq \exp(-t)$ gilt. Zudem ist σ als Komposition glatter Funktionen insbesondere $(N+1)$-mal stetig differenzierbar.
Mit Lemma 1.4 wissen wir, dass alle Ableitungen von σ, Polynome in σ sind. Dadurch folgt Bedingung (i) aus Definition 1.3, da σ nach Voraussetzung beschränkt ist und die Ableitungen von σ als endliche Summe von Produkten beschränkter Faktoren ebenfalls beschränkt sind. Da hiermit auch insbesondere die erste Ableitung von σ beschränkt ist, wissen wir, dass σ lipschitzstetig ist.
Nun kommen wir zum Beweis von Bedingung (ii). Sei $\mathscr{N}$ die Menge aller Nullstellen der Polynome $P_1, \ldots, P_n$ aus Lemma 1.4. Da keines dieser Polynome das Nullpolynom ist, wissen wir, dass $|\mathscr{N}| < \infty$ gilt. Insbesondere ist $(0,1) \setminus \mathscr{N}$ nicht leer. Sei $y \in (0,1) \setminus \mathscr{N}$. Da σ surjektiv nach $(0,1)$ ist, existiert ein $t_\sigma \in \mathbb{R}$ mit $\sigma(t_\sigma) = y$. Per Konstruktion ist $\sigma^{(n)}(t_\sigma) \neq 0$. Damit ist Bedingung (ii) ebenfalls erfüllt.
Nun zu Bedingung (iii). Wir wissen, dass für $x \in \mathbb{R}$ und damit insbesondere für ein beliebiges $x > 0$ die Ungleichung

$$x \leq \exp(x) + 1$$

gilt. Daraus folgt die Ungleichung $x \cdot \exp(-x) \leq 1 + \exp(-x)$ und schließlich $\frac{\exp(-x)}{1+\exp(-x)} \leq \frac{1}{x}$ für alle $x > 0$. Damit erhalten wir nun

$$|\sigma(x) - 1| = 1 - \frac{1}{1+\exp(-x)} = \frac{\exp(-x)}{1+\exp(-x)} \leq \frac{1}{x} \tag{1.3}$$

für $x > 0$. Dies zeigt den ersten Teil von Definition 1.3 (iii).
Der zweite Teil folgt auf die gleiche Art und Weise. Sei dazu $x < 0$. Aus

$$\frac{1}{1+\exp(x)} - \frac{1}{2} = \sigma(0-x) - \frac{1}{2} = -\sigma(0+x) + \frac{1}{2} = -\frac{1}{1+\exp(-x)} + \frac{1}{2} \tag{1.4}$$

wissen wir, dass σ punktsymmetrisch zum Punkt $(0, \frac{1}{2})$ ist. Gleichung (1.4) folgt aus $\frac{1}{1+\exp(x)} + \frac{1}{1+\exp(-x)} = \frac{2+\exp(-x)+\exp(x)}{2+\exp(-x)+\exp(x)} = 1$. Aus der Punktsymmetrie aus Gleichung (1.4) folgt zunächst

$$\sigma(-x) - 1 = \frac{1}{1+\exp(x)} - 1 = -\frac{1}{1+\exp(-x)} = -\sigma(x)$$

für $x < 0$. Da $-x > 0$ ist, folgt mit Gleichung (1.3):

$$|\sigma(x)| = |-\sigma(x)| = |\sigma(-x) - 1| \leq \frac{1}{-x} = \frac{1}{|x|}.$$

Damit haben wir alle drei Bedingungen aus Definition 1.3 gezeigt und unsere Aussage bewiesen. □

Als Nächstes stellen wir ein Resultat zur Taylorformel mit Rest vor, welches wir im weiteren Verlauf der Arbeit benötigen werden.

Lemma 1.6 (Lagrangesche Form des Restglieds, [For16, §22, Satz 2])**.**
Sei $I \subseteq \mathbb{R}$ ein Intervall und $f\colon I \to \mathbb{R}$ eine $(N+1)$-mal stetig differenzierbare Funktion und $u, x \in I$. Dann existiert ein ξ zwischen u und x, so dass

$$f(x) = \sum_{k=0}^{N} \frac{f^{(k)}(u)}{k!}(x-u)^k + \frac{f^{(N+1)}(\xi)}{(N+1)!}(x-u)^{N+1},$$

wobei hier $f^{(k)}$ für $k = 0, \ldots, N+1$ die k-te Ableitung von f bezeichnet.

Es folgen nun drei Approximationsresultate, welche wir in Kapitel 2 für die Konstruktion unseres Neuronale-Netze-Regressionsschätzers benötigen werden.
Wir betrachten als Erstes eine Approximation der Identität durch das neuronale Netz $f_{\mathrm{id}}\colon \mathbb{R} \to \mathbb{R}$ mit Architektur $(1,(1))$, welches in Abbildung 1.2 veranschaulicht wird und anschließend die Approximation eines Quadrats auf einem Intervall durch das neuronale Netz $f_{\mathrm{sq}}\colon \mathbb{R} \to \mathbb{R}$ mit Architektur $(1,(2))$.

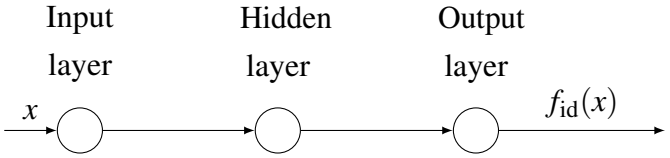

Abbildung 1.2: Neuronales Netz $f_{\mathrm{id}}(x)$ mit Architektur $(1,(1))$.

Lemma 1.7. *Sei $\sigma\colon \mathbb{R} \to \mathbb{R}$ eine beschränkte Funktion, R und $a > 0$.*

a) Angenommen σ ist zweimal stetig differenzierbar und $t_{\sigma,\mathrm{id}} \in \mathbb{R}$ so, dass $\sigma'(t_{\sigma,\mathrm{id}}) \neq 0$ ist. Dann gilt für das neuronale Netz

$$f_{\mathrm{id}}(x) = \frac{R}{\sigma'(t_{\sigma,\mathrm{id}})} \cdot \left(\sigma\left(\frac{x}{R} + t_{\sigma,\mathrm{id}}\right) - \sigma(t_{\sigma,\mathrm{id}})\right)$$

für beliebiges $x \in [-a,a]$:

$$|f_{\mathrm{id}}(x) - x| \leq \frac{\|\sigma''\|_\infty \cdot a^2}{2 \cdot |\sigma'(t_{\sigma,\mathrm{id}})|} \cdot \frac{1}{R}.$$

b) Angenommen σ ist dreimal stetig differenzierbar und $t_{\sigma,\mathrm{sq}} \in \mathbb{R}$ so, dass $\sigma''(t_{\sigma,\mathrm{sq}}) \neq 0$ ist. Dann gilt für das neuronale Netz

$$f_{\mathrm{sq}}(x) = \frac{R^2}{\sigma''(t_{\sigma,\mathrm{sq}})} \cdot \left(\sigma\left(\frac{2 \cdot x}{R} + t_{\sigma,\mathrm{sq}} \right) - 2 \cdot \sigma\left(\frac{x}{R} + t_{\sigma,\mathrm{sq}} \right) + \sigma(t_{\sigma,\mathrm{sq}}) \right)$$

für beliebiges $x \in [-a,a]$:

$$|f_{\mathrm{sq}}(x) - x^2| \leq \frac{5 \cdot \|\sigma'''\|_\infty \cdot a^3}{3 \cdot |\sigma''(t_{\sigma,\mathrm{sq}})|} \cdot \frac{1}{R}.$$

Beweis. a) Wir wissen, dass f_{id} zweimal differenzierbar ist, da σ nach Voraussetzung zweimal stetig differenzierbar ist. Damit folgt mit Lemma 1.6 für $f = f_{\mathrm{id}}$, $N = 1$, $u = 0$ und $I = [-a,a]$, dass ein ξ zwischen 0 und x existiert, mit

$$f_{\mathrm{id}}(x) = \sum_{k=0}^{1} \frac{f_{\mathrm{id}}^{(k)}(0)}{k!} x^k + \frac{f_{\mathrm{id}}^{(2)}(\xi)}{2!} x^2.$$

Es gilt:

$$f_{\mathrm{id}}'(x) = \frac{1}{\sigma'(t_{\sigma,\mathrm{id}})} \cdot \sigma'\left(\frac{x}{R} + t_{\sigma,\mathrm{id}} \right).$$

Mit $f_{\mathrm{id}}(0) = 0$ und $f_{\mathrm{id}}'(0) = 1$ erhalten wir:

$$\begin{aligned} &|f_{\mathrm{id}}(x) - x| \\ &\quad = \left| 0 + 1 \cdot x + \frac{1}{2} \cdot \frac{1}{R \cdot \sigma'(t_{\sigma,\mathrm{id}})} \sigma''\left(\frac{\xi}{R} + t_{\sigma,\mathrm{id}} \right) \cdot x^2 - x \right| \\ &\quad = \left| \frac{\sigma''\left(\frac{\xi}{R} + t_{\sigma,\mathrm{id}} \right) \cdot x^2}{2 \cdot R \cdot \sigma'(t_{\sigma,\mathrm{id}})} + x - x \right| \\ &\quad \leq \frac{\|\sigma''\|_\infty \cdot a^2}{2 \cdot |\sigma'(t_{\sigma,\mathrm{id}})|} \cdot \frac{1}{R}, \end{aligned}$$

wobei sich die letzte Ungleichung aus den Eigenschaften der Supremumsnorm und $x^2 \leq a^2$ ergibt, was aus $x \in [-a,a]$ folgt. Daraus erhalten wir die Behauptung.

b) Die Funktion f_{sq} ist dreimal differenzierbar, da σ nach Voraussetzung dreimal stetig differenzierbar ist. Wie in a) folgt durch Lemma 1.6 mit $f = f_{\mathrm{sq}}$, $N = 2$, $u = 0$ und $I = [-a,a]$, dass ein ξ zwischen 0 und x existiert, so dass:

$$f_{\mathrm{sq}}(x) = \sum_{k=0}^{2} \frac{f_{\mathrm{sq}}^{(k)}(0)}{k!} x^k + \frac{f_{\mathrm{sq}}^{(3)}(\xi)}{3!} x^3.$$

Es gilt:

$$f_{\mathrm{sq}}'(x) = \frac{R^2}{\sigma''(t_{\sigma,\mathrm{sq}})} \cdot \left(\frac{2}{R} \cdot \sigma'\left(\frac{2x}{R} + t_{\sigma,\mathrm{sq}} \right) - \frac{2}{R} \cdot \sigma'\left(\frac{x}{R} + t_{\sigma,\mathrm{sq}} \right) \right)$$

und

$$f''_{\mathrm{sq}}(x) = \frac{R^2}{\sigma''(t_{\sigma,\mathrm{sq}})} \cdot \left(\frac{4}{R^2} \cdot \sigma''\left(\frac{2x}{R} + t_{\sigma,\mathrm{sq}} \right) - \frac{2}{R^2} \cdot \sigma''\left(\frac{x}{R} + t_{\sigma,\mathrm{sq}} \right) \right).$$

Mit $f_{\mathrm{sq}}(0) = 0$, $f'_{\mathrm{sq}}(0) = 0$ und $f''_{\mathrm{sq}}(0) = 2$ erhalten wir:

$$\begin{aligned}
&|f_{\mathrm{sq}}(x) - x^2| \\
&= \left| x^2 + \frac{1}{6} \cdot \frac{R^2}{\sigma''(t_{\sigma,\mathrm{sq}})} \left(\frac{8}{R^3} \sigma'''\left(\frac{2\xi}{R} + t_{\sigma,\mathrm{sq}} \right) - \frac{2}{R^3} \sigma'''\left(\frac{\xi}{R} + t_{\sigma,\mathrm{sq}} \right) \right) \cdot x^3 - x^2 \right| \\
&\leq \frac{a^3}{6 \cdot |\sigma''(t_{\sigma,\mathrm{sq}})|} \cdot \frac{1}{R} \cdot \left(8 \cdot \left| \sigma'''\left(\frac{2\xi}{R} + t_{\sigma,\mathrm{sq}} \right) \right| + 2 \cdot \left| \sigma'''\left(\frac{\xi}{R} + t_{\sigma,\mathrm{sq}} \right) \right| \right) \\
&\leq \frac{10 \cdot a^3}{6 \cdot |\sigma''(t_{\sigma,\mathrm{sq}})|} \cdot \frac{1}{R} \cdot \|\sigma'''\|_\infty \\
&= \frac{5 \cdot \|\sigma'''\|_\infty \cdot a^3}{3 \cdot |\sigma''(t_{\sigma,\mathrm{sq}})|} \cdot \frac{1}{R}.
\end{aligned}$$

Daraus folgt die Behauptung. □

Wir betrachten als Nächstes die Approximation einer Multiplikation zweier Werte auf einem Intervall durch das neuronale Netz $f_{\mathrm{mult}} \colon \mathbb{R}^2 \to \mathbb{R}$ mit Architektur $(1,(4))$, welches in Abbildung 1.3 veranschaulicht wird.

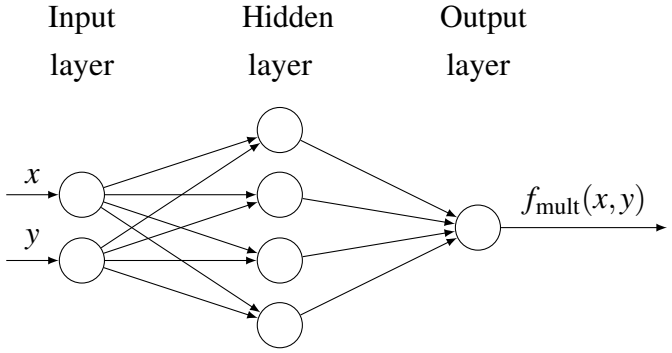

Abbildung 1.3: Neuronales Netz $f_{\mathrm{mult}}(x,y)$ mit Architektur $(1,(4))$.

Lemma 1.8. *Sei $\sigma \colon \mathbb{R} \to [0,1]$ 2-zulässig. Zudem sei $R > 0$ und $a > 0$ beliebig. Dann gilt für das neuronale Netz*

$$\begin{aligned}
f_{\mathrm{mult}}(x,y) = \frac{R^2}{4 \cdot \sigma''(t_\sigma)} \cdot \Big(&\sigma\Big(\frac{2 \cdot (x+y)}{R} + t_\sigma \Big) - 2 \cdot \sigma\Big(\frac{x+y}{R} + t_\sigma \Big) \\
&- \sigma\Big(\frac{2 \cdot (x-y)}{R} + t_\sigma \Big) + 2 \cdot \sigma\Big(\frac{x-y}{R} + t_\sigma \Big) \Big)
\end{aligned}$$

für beliebige $x, y \in [-a,a]$ folgende Ungleichung:

$$|f_{\text{mult}}(x,y) - x \cdot y| \leq \frac{20 \cdot \|\sigma'''\|_\infty \cdot a^3}{3 \cdot |\sigma''(t_\sigma)|} \cdot \frac{1}{R}.$$

Beweis. Durch Einsetzen von f_{sq} in f_{mult} erhalten wir

$$f_{\text{mult}}(x,y) = \frac{1}{4}(f_{\text{sq}}(x+y) - f_{\text{sq}}(x-y))$$

und durch Umformungen

$$x \cdot y = \frac{1}{4}\big((x+y)^2 - (x-y)^2\big).$$

Aus diesen beiden Gleichungen folgt durch Ausklammern von $\frac{1}{4}$, der Homogenität des Betrags und der Dreiecksungleichung:

$$\begin{aligned}
|f_{\text{mult}}(x,y) - x \cdot y| &= \frac{1}{4} \cdot \big|f_{\text{sq}}(x+y) - f_{\text{sq}}(x-y) - (x+y)^2 + (x-y)^2\big| \\
&\leq \frac{1}{4} \cdot \big|f_{\text{sq}}(x+y) - (x+y)^2\big| + \frac{1}{4} \cdot \big|f_{\text{sq}}(x-y) - (x-y)^2\big| \\
&\leq 2 \cdot \frac{1}{4} \cdot \frac{40 \cdot \|\sigma'''\|_\infty \cdot a^3}{3 \cdot |\sigma''(t_\sigma)|} \cdot \frac{1}{R} \\
&= \frac{20 \cdot \|\sigma'''\|_\infty \cdot a^3}{3 \cdot |\sigma''(t_\sigma)|} \cdot \frac{1}{R},
\end{aligned}$$

wobei wir bei der letzten Ungleichung Lemma 1.7 b) mit $x+y, x-y \in [-2a, 2a]$ verwendet haben. Daraus folgt die Behauptung. □

Wir betrachten als Nächstes die Approximation der Maximumsfunktion $\max\{x,0\}$ für x aus einem beschränkten Intervall durch das neuronale Netz $f_{\text{ReLU}} \colon \mathbb{R} \to \mathbb{R}$ mit Architektur $(2,(2,4))$, welches in Abbildung 1.4 veranschaulicht wird.

Lemma 1.9. *Sei $\sigma \colon \mathbb{R} \to [0,1]$ 2-zulässig und $a \geq 1$. Sei f_{id} das neuronale Netz aus Lemma 1.7 und f_{mult} das neuronale Netz aus Lemma 1.8. Angenommen es gelte die Ungleichung*

$$R \geq \frac{\|\sigma''\|_\infty \cdot a}{2 \cdot |\sigma'(t_\sigma)|}. \tag{1.5}$$

Dann erfüllt das neuronale Netz

$$f_{\text{ReLU}}(x) = f_{\text{mult}}(f_{\text{id}}(x), \sigma(R \cdot x)) \tag{1.6}$$

für alle $x \in [-a,a]$ folgende Ungleichung:

$$|f_{\text{ReLU}}(x) - \max\{x,0\}| \leq 56 \cdot \frac{\max\{\|\sigma''\|_\infty, \|\sigma'''\|_\infty, 1\}}{\min\{2 \cdot |\sigma'(t_\sigma)|, |\sigma''(t_\sigma)|, 1\}} \cdot a^3 \cdot \frac{1}{R}.$$

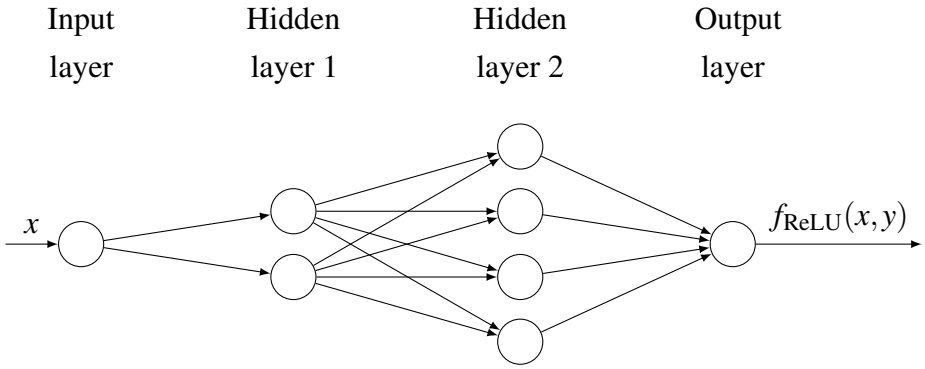

Abbildung 1.4: Neuronales Netz $f_{\mathrm{ReLU}}(x)$ mit Architektur $(2,(2,4))$.

Beweis. Da σ nach Voraussetzung 2-zulässig ist, gilt für $R \geq 0$ und $x \in \mathbb{R} \setminus \{0\}$:

$$|\sigma(R \cdot x) - 1| \leq \frac{1}{R \cdot x} \text{ für } x > 0$$

und

$$|\sigma(R \cdot x)| \leq \frac{1}{|R \cdot x|} \text{ für } x < 0.$$

Damit folgt aus der Homogenität des Betrags für alle $x \neq 0$:

$$|\sigma(R \cdot x) - \mathbb{1}_{[0,\infty)}(x)| \leq \frac{1}{|R \cdot x|} = \frac{1}{R \cdot |x|}. \tag{1.7}$$

Nach Lemma 1.7 gilt:

$$|f_{\mathrm{id}}(x) - x| \leq \frac{\|\sigma''\|_\infty \cdot a^2}{2 \cdot |\sigma'(t_\sigma)|} \cdot \frac{1}{R} \text{ für } x \in [-a,a] \tag{1.8}$$

und nach Lemma 1.8:

$$|f_{\mathrm{mult}}(x,y) - x \cdot y| \leq \frac{160 \cdot \|\sigma'''\|_\infty \cdot a^3}{3 \cdot |\sigma''(t_\sigma)|} \cdot \frac{1}{R} \text{ für } x,y \in [-2a,2a]. \tag{1.9}$$

Da nach Voraussetzung $a \geq 1$ ist, gilt insbesondere $[0,1] \subseteq [-2a,2a]$ und daher gilt insbesondere $\sigma(Rx) \in [0,1] \subseteq [-2a,2a]$. Zudem erhalten wir durch eine Nulladdition und die Dreiecksungleichung:

$$|f_{\mathrm{id}}(x)| = |f_{\mathrm{id}}(x) - x + x| \leq |f_{\mathrm{id}}(x) - x| + |x|. \tag{1.10}$$

Aus Gleichung (1.8) und Voraussetzung (1.5) folgt

$$|f_{\mathrm{id}}(x) - x| \leq \frac{\|\sigma''\|_\infty \cdot a^2}{2 \cdot |\sigma'(t_\sigma)|} \cdot \frac{1}{R} \leq \frac{\|\sigma''\|_\infty \cdot a^2}{2 \cdot |\sigma'(t_\sigma)|} \cdot \frac{2 \cdot |\sigma'(t_\sigma)|}{\|\sigma''\|_\infty \cdot a} = a. \tag{1.11}$$

Aus $x \in [-a,a]$ folgt schließlich mit den Ungleichungen (1.10) und (1.11):

$$|f_{\mathrm{id}}(x)| \leq |f_{\mathrm{id}}(x) - x| + |x| \leq 2 \cdot a.$$

Daraus folgt insbesondere $f_{\mathrm{id}}(x) \in [-2a, 2a]$. Mithilfe von $\max\{x,0\} = x \cdot \mathbb{1}_{[0,\infty)}(x)$, der Definition des Netzes in Gleichung (1.6), zweier Nulladditionen und der Dreiecksungleichung erhalten wir:

$$\begin{aligned} &|f_{\mathrm{ReLU}}(x) - \max\{x,0\}| \\ &\qquad = \left|f_{\mathrm{mult}}(f_{\mathrm{id}}(x), \sigma(R\cdot x)) - x \cdot \mathbb{1}_{[0,\infty)}(x)\right| \\ &\qquad \leq \left|f_{\mathrm{mult}}(f_{\mathrm{id}}(x), \sigma(R\cdot x)) - f_{\mathrm{id}}(x)\cdot\sigma(R\cdot x)\right| \\ &\qquad\qquad + \left|f_{\mathrm{id}}(x)\cdot\sigma(R\cdot x) - x\cdot\sigma(R\cdot x)\right| + \left|x\cdot\sigma(R\cdot x) - x\cdot\mathbb{1}_{[0,\infty)}(x)\right|. \end{aligned} \tag{1.12}$$

Wenden wir nun auf den ersten Summanden in Gleichung (1.12) die Ungleichung (1.9) an, erhalten wir:

$$\left|f_{\mathrm{mult}}(f_{\mathrm{id}}(x), \sigma(R\cdot x)) - f_{\mathrm{id}}(x)\cdot\sigma(R\cdot x)\right| \leq \frac{160\cdot\|\sigma'''\|_\infty \cdot a^3}{3\cdot|\sigma''(t_\sigma)|}\cdot\frac{1}{R}. \tag{1.13}$$

Klammern wir im nächsten Schritt im zweiten Summanden in Gleichung (1.12) den Faktor $\sigma(R\cdot x) \in [0,1]$ aus und wenden Ungleichung (1.8) an, erhalten wir:

$$\left|f_{\mathrm{id}}(x)\cdot\sigma(R\cdot x) - x\cdot\sigma(R\cdot x)\right| \leq \frac{\|\sigma''\|_\infty\cdot a^2}{2\cdot|\sigma'(t_\sigma)|}\cdot\frac{1}{R}\cdot 1 \leq \frac{\|\sigma''\|_\infty\cdot a^3}{2\cdot|\sigma'(t_\sigma)|}\cdot\frac{1}{R}, \tag{1.14}$$

wobei wir verwendet haben, dass nach der Voraussetzung $a \geq 1$ und damit $a^2 \leq a^3$ gilt. Wenn wir schließlich im dritten Summanden in Gleichung (1.12) den Faktor x durch die Homogenität des Betrags ausklammern und Ungleichung (1.7) anwenden, erhalten wir:

$$\left|x\cdot\sigma(R\cdot x) - x\cdot\mathbb{1}_{[0,\infty)}(x)\right| \leq \frac{1}{R}. \tag{1.15}$$

Setzen wir nun Ungleichungen (1.13)- (1.15) in Ungleichung (1.12) ein, ergibt sich durch Ausklammern von $\frac{1}{R}$ und weiteren Abschätzungen:

$$\begin{aligned} |f_{\mathrm{ReLU}}(x) - \max\{x,0\}| &\leq \left(\frac{160}{3}\cdot\frac{\|\sigma'''\|_\infty\cdot a^3}{|\sigma''(t_\sigma)|} + \frac{\|\sigma''\|_\infty\cdot a^3}{2\cdot|\sigma'(t_\sigma)|} + \frac{a^3}{a^3}\right)\cdot\frac{1}{R} \\ &\leq \left(\frac{160\cdot\|\sigma'''\|_\infty\cdot a^3 + 3\cdot\|\sigma''\|_\infty\cdot a^3 + 3\cdot a^3}{3\cdot\min\{2\cdot|\sigma'(t_\sigma)|, |\sigma''(t_\sigma)|, 1\}}\right)\cdot\frac{1}{R} \\ &\leq \frac{166}{3}\cdot\left(\frac{\max\{\|\sigma'''\|_\infty, \|\sigma''\|_\infty, 1\}}{\min\{2\cdot|\sigma'(t_\sigma)|, |\sigma''(t_\sigma)|, 1\}}\right)\cdot a^3\cdot\frac{1}{R}. \end{aligned}$$

Da $\frac{166}{3} \leq 56$ gilt, folgt schließlich die Behauptung. □

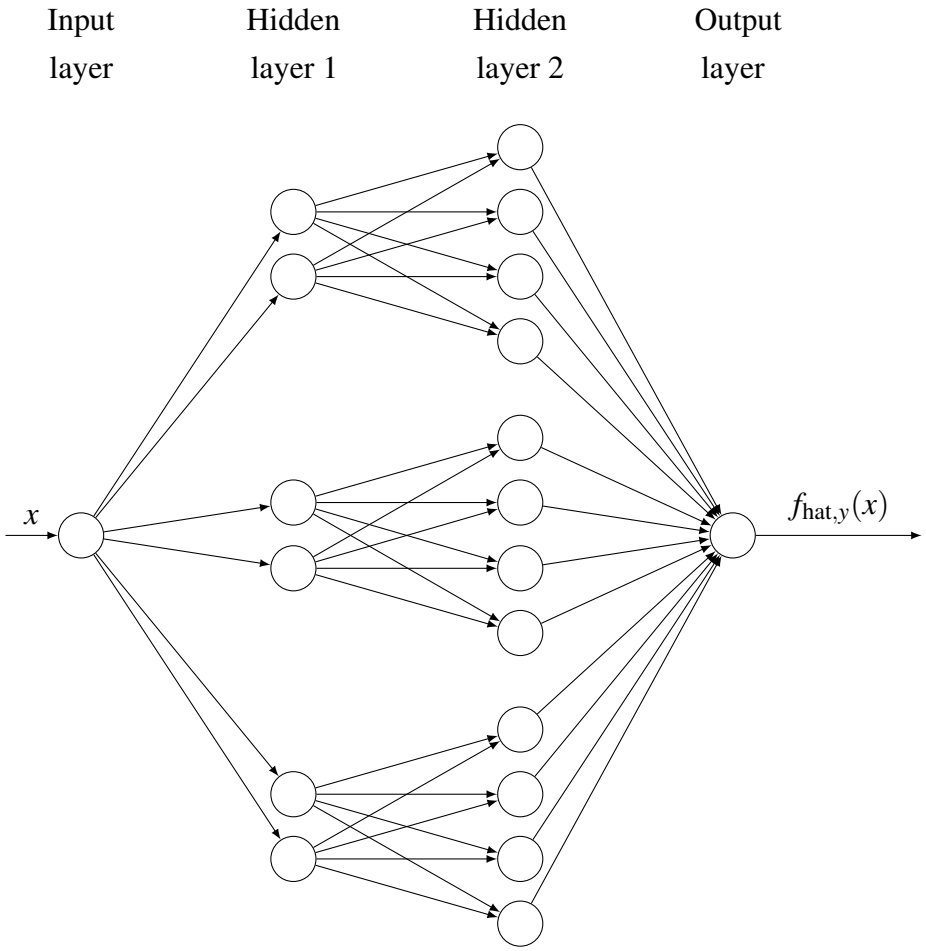

Abbildung 1.5: Neuronales Netz $f_{\text{hat},y}(x)$ mit Architektur $(2,(6,12))$.

Wir betrachten als Nächstes die Approximation des Positivteils einer Funktion auf einem Intervall durch das neuronale Netz $f_{\text{hat},y}\colon \mathbb{R} \to \mathbb{R}$ mit Architektur $(2,(6,12))$, welches in Abbildung 1.5 veranschaulicht wird.

Lemma 1.10. *Sei $M \in \mathbb{N}$ und sei $\sigma\colon \mathbb{R} \to [0,1]$ 2-zulässig. Sei $a > 0$ und*

$$R \geq \frac{\|\sigma''\|_\infty \cdot (M+1)}{2 \cdot |\sigma'(t_\sigma)|},$$

sei $y \in [-a,a]$ und f_{ReLU} das neuronale Netz aus Lemma 1.9. Dann erfüllt das neuronale Netz

$$f_{\text{hat},y}(x) = f_{\text{ReLU}}\left(\frac{M}{2a} \cdot (x-y) + 1\right) - 2 \cdot f_{\text{ReLU}}\left(\frac{M}{2a} \cdot (x-y)\right) + f_{\text{ReLU}}\left(\frac{M}{2a} \cdot (x-y) - 1\right)$$

für alle $x \in [-a,a]$ mit $z_+ = \max\{0,z\}$ $(z \in \mathbb{R})$:

$$\left| f_{\text{hat},y}(x) - \left(1 - \frac{M}{2a} \cdot |x-y|\right)_+ \right| \leq 1792 \cdot \frac{\max\{\|\sigma''\|_\infty, \|\sigma'''\|_\infty, 1\}}{\min\{2 \cdot |\sigma'(t_\sigma)|, |\sigma''(t_\sigma)|, 1\}} \cdot M^3 \cdot \frac{1}{R}.$$

Beweis. Für $x \in \mathbb{R}$ gilt die Gleichung:

$$\left(1 - \frac{M}{2a} \cdot |x|\right)_+ = \max\left\{\frac{M}{2a} \cdot x + 1, 0\right\} - 2 \cdot \max\left\{\frac{M}{2a} \cdot x, 0\right\} + \max\left\{\frac{M}{2a} \cdot x - 1, 0\right\}, \tag{1.16}$$

die wir im zweiten Teil dieses Beweises zeigen werden. Damit beweisen wir das Resultat mit Hilfe von Lemma 1.9, denn mit der Definition von $f_{\text{hat},y}(x)$ und der Dreiecksungleichung folgt:

$$\begin{aligned}
&\left| f_{\text{hat},y}(x) - \left(1 - \frac{M}{2a} \cdot |x-y|\right)_+ \right| \\
&\quad \leq \left| f_{\text{ReLU}}\left(\frac{M}{2a} \cdot (x-y) + 1\right) - \max\left\{\frac{M}{2a} \cdot (x-y) + 1, 0\right\} \right| \\
&\qquad + 2 \cdot \left| f_{\text{ReLU}}\left(\frac{M}{2a} \cdot (x-y)\right) - \max\left\{\frac{M}{2a} \cdot (x-y), 0\right\} \right| \\
&\qquad + \left| f_{\text{ReLU}}\left(\frac{M}{2a} \cdot (x-y) - 1\right) - \max\left\{\frac{M}{2a} \cdot (x-y) - 1, 0\right\} \right| \\
&\quad \leq 1792 \cdot \frac{\max\{\|\sigma''\|_\infty, \|\sigma'''\|_\infty, 1\}}{\min\{2 \cdot |\sigma'(t_\sigma)|, |\sigma''(t_\sigma)|, 1\}} \cdot M^3 \cdot \frac{1}{R},
\end{aligned}$$

wobei die letzte Ungleichung daraus folgt, dass wir auf jeden Summanden Lemma 1.9 mit

$$1 \leq a = M + 1$$

angewendet haben, da aus $x, y \in [-a, a]$ folgt, dass $\frac{M}{2a} \cdot (x-y) \in [-M, M]$ gilt. Schließlich haben wir

$$(M+1)^3 \leq (2M)^3 = 8M^3$$

verwendet, da $M \geq 1$ ist. $(\square)$

Um Gleichung (1.16) zu zeigen unterscheiden wir vier Fälle.

Fall 1 $(x < 0)$ In diesem Fall hat die linke Seite von Gleichung (1.16) nach der Definition des Betrags die Gestalt

$$\max\left\{1 + \frac{M}{2a} \cdot x, 0\right\}$$

und die rechte Seite von Gleichung (1.16) die Form

$$\max\left\{\frac{M}{2a} \cdot x + 1, 0\right\} - 2 \cdot 0 + 0,$$

da $x < 0$ und damit die letzten zwei Summanden Null sind. Es erfordert hier eine weitere Fallunterscheidung.

Fall 1.1 $(0 > x \geq -\frac{2a}{M})$ In diesem Fall gilt für die linke und rechte Seite von Gleichung (1.16):

$$\max\left\{1 + \frac{M}{2a} \cdot x, 0\right\} = 1 + \frac{M}{2a} \cdot x.$$

Fall 1.2 $(x < -\frac{2a}{M})$ In diesem Fall sind beide Seiten gleich Null, da $1 + \frac{M}{2a} \cdot x \leq 0$ ist. $(\square)$

Fall 2 $(x \geq 0)$ In diesem Fall hat die linke Seite von Gleichung (1.16) nach der Definition des Betrags die Gestalt

$$\max\left\{1 - \frac{M}{2a} \cdot x, 0\right\}$$

und die rechte Seite von Gleichung (1.16) die Form

$$\max\left\{\frac{M}{2a} \cdot x + 1, 0\right\} - 2 \cdot \max\left\{\frac{M}{2a} \cdot x, 0\right\} + \max\left\{\frac{M}{2a} \cdot x - 1, 0\right\}$$

und erfordert daher eine weitere Fallunterscheidung.

Fall 2.1 $(0 \leq x < \frac{2a}{M})$ In diesem Fall hat die linke Seite von Gleichung (1.16) die Gestalt

$$1 - \frac{M}{2a} \cdot x$$

und die rechte Seite von Gleichung (1.16) die Form

$$\frac{M}{2a} \cdot x + 1 - 2 \cdot \frac{M}{2a} \cdot x + 0 = 1 - \frac{M}{2a} \cdot x$$

und stimmt daher mit der linken Seite überein.

Fall 2.2 ($x \geq \frac{2a}{M}$) In diesem Fall ist die linke Seite von Gleichung (1.16) gleich 0, da wir wissen, dass $1 - \frac{M}{2a} \cdot x < 0$ ist und die rechte Seite die Form

$$\frac{M}{2a} \cdot x + 1 - 2 \cdot \frac{M}{2a} \cdot x + \frac{M}{2a} \cdot x - 1 = 0$$

besitzt. (□)

Durch diese Fallunterscheidung wurde die Gleichung (1.16) bewiesen und damit ist der Beweis vollständig. □

Im nächsten Kapitel werden wir die hier eingeführten neuronalen Netze als Bausteine verwenden, um daraus unseren Neuronale-Netze-Regressionsschätzer zu konstruieren.

Kapitel 2

Konstruktion eines Neuronale-Netze-Schätzers

In dieser Arbeit behandeln wir Neuronale-Netze-Schätzer im Kontext der *nichtparametrischen Regression*. In der Regressionsanalyse betrachtet man einen $\mathbb{R}^d \times \mathbb{R}$-wertigen Zufallsvektor (X,Y). Man ist daran interessiert, wie der Wert der *Reaktionsvariable Y* vom Wert des *Beobachtungsvektors X* abhängt. Dies bedeutet, dass man eine (messbare) Funktion $f\colon \mathbb{R}^d \to \mathbb{R}$ sucht, sodass $f(X)$ eine „gute Approximation von Y" ist. Das wiederum bedeutet, dass $f(X)$ nah an Y sein sollte, was in gewisser Weise gleichwertig damit ist den Ausdruck $|f(X)-Y|$ zu „minimieren". Da aber X und Y Zufallsvektoren sind und damit $|f(X)-Y|$ auch zufällig ist, ist nicht klar was unter „$|f(X)-Y|$ minimal" zu verstehen ist. Wir können dieses Problem lösen, indem wir das sogenannte *L_2-Risiko*

$$\mathbb{E}\,[|f(X)-Y|^2],$$

einführen und verlangen, dass dieses so klein wie möglich ist. Bei der nichtparametrischen Regression ist die Bauart der schätzenden Funktion f komplett unbekannt. Wir sind daher an einer Funktion interessiert, die das L_2-Risiko von f minimiert. Dies führt auf die *Regressionsfunktion* $m(x) = \mathbb{E}\,[Y \mid X = x]$, da für das das L_2-Risiko einer beliebigen messbaren Funktion $f\colon \mathbb{R}^d \to \mathbb{R}$ gilt:

$$\mathbb{E}\,[|f(X)-Y|^2] = \mathbb{E}\,[|m(X)-Y|^2] + \int |f(x)-m(x)|^2 \mathbb{P}_X(dx),$$

d.h. der mittlere quadratische Vorhersagefehler einer Funktion f ist darstellbar als Summe des L_2-Risikos der Regressionsfunktion (unvermeidbarer Fehler) und des L_2-Fehlers, der aufgrund der Verwendung von f an Stelle von m bei der Vorhersage bzw. Approximation des Wertes von Y entsteht. Im Hinblick auf die Minimierung des L_2-Risikos sollte dabei der L_2-Fehler der Schätzfunktion möglichst klein sein. Dieser L_2-Fehler ist immer

nichtnegativ und für $f(x) = m(x)$ sogar Null. Daher ist m die beste Wahl für f. In Anwendungsfällen ist aber üblicherweise die Verteilung von (X,Y) unbekannt, daher kann die Regressionsfunktion m nicht berechnet werden. Oft ist es aber möglich, Werte von (X,Y) zu beobachten und damit die Regressionsfunktion m zu schätzen. Formal führt das auf folgende Problemstellung:

Problem 2.1 ([GKKW02, Kapitel 1.1 und Kapitel 1.2])**.** *Seien* $(X,Y),(X_1,Y_1),(X_2,Y_2),\ldots$ *u.i.v.* $\mathbb{R}^d \times \mathbb{R}$*-wertige Zufallsvariablen mit* $\mathbb{E}[Y^2] < \infty$ *und* $m\colon \mathbb{R}^d \to \mathbb{R}$ *definiert durch* $m(x) = \mathbb{E}[Y \mid X = x]$ *sei die zugehörige Regressionsfunktion. Gegeben sei die Datenmenge*

$$\mathscr{D}_n = \{(X_1,Y_1),\ldots,(X_n,Y_n)\}. \tag{2.1}$$

Gesucht ist eine Schätzung

$$m_n(\cdot) = m_n(\cdot,\mathscr{D}_n)\colon \mathbb{R}^d \to \mathbb{R}$$

von m, für die der L_2*-Fehler*

$$\int |m_n(x) - m(x)|^2 \mathbb{P}_X(dx)$$

möglichst „klein“ ist.

In diesem Kapitel werden wir mithilfe von neuronalen Netzen einen Regressionsschätzer $\tilde{m}_n$ konstruieren. Dieser Schätzer besitzt ebenfalls die Gestalt eines neuronalen Netzes nach Definition 1.1 und daher werden wir ihn als *Neuronale-Netze-Regressionsschätzer* bezeichnen.
Für die Konstruktion unseres Neuronale-Netze-Regressionsschätzers wählen wir den logistischen Squasher aus Gleichung (1.1) als Aktivierungsfunktion σ, verwenden die gegebene Datenmenge $\mathscr{D}_n$ und wählen die Gewichte des neuronalen Netzes so, dass die resultierende Funktion aus Definition 1.1 eine gute Schätzung für die Regressionsfunktion m ist. Dafür wählen wir die Gewichte bis auf die in der Ausgabeschicht fest und bestimmen die Gewichte in der Ausgabeschicht, indem wir mit unserer Datenmenge $\mathscr{D}_n$ ein Kleinste-Quadrate-Problem lösen.
Es ist bekannt (vgl. [DGL96, Theorem 7.2 und Problem 7.2] und [DW80, Section 3]), dass man Glattheitsvoraussetzungen an die Regressionsfunktion stellen muss, um nichttriviale Konvergenzresultate für nichtparametrische Regressionsschätzer herzuleiten. Dafür verwenden wir die folgende Definition.

Definition 2.2 ((p,C)-Glattheit)**.** Sei $p = q+s$ mit $q \in \mathbb{N}_0$ und $s \in (0,1]$ und sei $C > 0$. Eine Funktion $f\colon \mathbb{R}^d \to \mathbb{R}$ heißt *(p,C)-glatt*, falls für alle Multiindizes $\alpha = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}_0^d$ mit $\sum_{j=1}^d \alpha_j = q$ die partielle Ableitung

$$\frac{\partial^q f}{\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d}}$$

existiert und falls für alle $x, z \in \mathbb{R}^d$ die Abschätzung

$$\left| \frac{\partial^q f}{\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d}}(x) - \frac{\partial^q f}{\partial x_1^{\alpha_1} \ldots \partial x_d^{\alpha_d}}(z) \right| \leq C \cdot \|x - z\|^s,$$

gilt, wobei $\|\cdot\|$ die euklidische Norm in $\mathbb{R}^d$ bezeichnet.

2.1 Definition der Netzwerkarchitektur

In diesem Abschnitt stellen wir die *Netzwerkarchitektur* unseres Neuronale-Netze-Regressionsschätzers vor. Dafür legen wir die Architektur $(L, \mathbf{k})$ fest und gehen auf die konkrete Konstruktion unseres Schätzers ein.
Zunächst fixieren wir die Multiindexnotation, die wir der Übersichtlichkeit halber im weiteren Verlauf dieser Arbeit verwenden werden. Sei $a > 0$ fest und $M \in \mathbb{N}$. Wir definieren $[M]^d := \{0, 1, \ldots, M\}^d$. Für $(\mathbf{i}^{(1)}, \ldots, \mathbf{i}^{(d)}) = \mathbf{i} \in [M]^d$ und $x \in \mathbb{R}^d$ definieren wir

$$|\mathbf{i}|_1 := \sum_{k=1}^d \mathbf{i}^{(k)}, \quad \mathbf{i}! := \mathbf{i}^{(1)}! \cdots \mathbf{i}^{(d)}! \quad \text{und} \quad x^{\mathbf{i}} := x_1^{\mathbf{i}^{(1)}} \cdots x_d^{\mathbf{i}^{(d)}}.$$

Für $f\colon \mathbb{R}^d \to \mathbb{R}$ ausreichend oft differenzierbar definieren wir

$$\partial^{\mathbf{i}} f(x) := \frac{\partial^{|\mathbf{i}|_1} f}{\partial^{\mathbf{i}^{(1)}} x_1 \cdots \partial^{\mathbf{i}^{(d)}} x_d}(x).$$

Das nächste Lemma ist ein Resultat aus der Kombinatorik, welches wir in Kapitel 2.2 benötigen werden.

Lemma 2.3. *Sei $d, N \in \mathbb{N}$ und $k \in \mathbb{N}_0$ mit $k \leq N$. Dann gilt:*

$$\left| \left\{ \mathbf{j} \in [N]^d : |\mathbf{j}|_1 = k \right\} \right| = \binom{d+k-1}{k}.$$

Beweis. Diese Aussage folgt aus einer Analogie zu einem Urnenexperiment. Wir betrachten eine Urne mit d Kugeln, die wir mit $1, \ldots, d$ beschriften. Wir ziehen k-mal aus dieser Urne mit Zurücklegen und ohne Beachtung der Reihenfolge und konstruieren so einen Vektor $\mathbf{j} = (j_1, \ldots, j_d)$ mit $|\mathbf{j}|_1 = k$. Der Koeffizient j_i mit $i = 1, \ldots, d$ gibt an wie oft die Kugel mit der Nummer i gezogen wurde. Damit stimmt die Kardinalität der Menge auf der linken Seite mit der Anzahl aller Möglichkeiten überein, die man erhält, wenn man k-mal aus dieser Urne mit Zurücklegen und ohne Beachtung der Reihenfolge zieht. □

Wir betrachten im Folgenden ein d-dimensionales äquidistantes Gitter im Würfel $[-a,a]^d$ mit Schrittweite $\frac{2a}{M}$. Dann ordnen wir jedem Multiindex $\mathbf{i} \in [M]^d$ einen Gitterpunkt

$$x_{\mathbf{i}} = \left(-a + \mathbf{i}^{(1)} \cdot \frac{2a}{M}, \ldots, -a + \mathbf{i}^{(d)} \cdot \frac{2a}{M} \right) = -\mathbf{a} + \frac{2a}{M} \cdot \mathbf{i} \tag{2.2}$$

mit $\mathbf{a} = (a, a, \ldots, a) \in \mathbb{R}^d$ zu.
Hiermit lässt sich das zu m gehörige Taylorpolynom der Ordnung $q \in \mathbb{N}_0$ mit Entwicklungspunkt $x_{\mathbf{i}}$ schreiben als

$$p_{\mathbf{i}}^m(x) = \sum_{\substack{\mathbf{j} \in [q]^d \\ |\mathbf{j}|_1 \leq q}} \partial^{\mathbf{j}} m(x_{\mathbf{i}}) \cdot \frac{(x - x_{\mathbf{i}})^{\mathbf{j}}}{\mathbf{j}!}.$$

$$P_m(x) = \sum_{\mathbf{i} \in [M]^d} p_{\mathbf{i}}^m(x) \prod_{j=1}^{d} \left(1 - \frac{M}{2a} \cdot |x^{(j)} - x_{\mathbf{i}}^{(j)}| \right)_+, \tag{2.3}$$

mit der wir die Regeressionsfunktion m approximieren wollen.
Als Nächstes stellen wir ein Resultat zur Taylorformel mit Rest vor, welches der Restgliedformel aus Kapitel 1 in höherer Raumdimension entspricht.

Lemma 2.4 (Lagrangesche Form des Restglieds, [Kö04, Kapitel 2.4, Seite 67])**.**
Sei $U \subseteq \mathbb{R}^d$ und $f\colon U \to \mathbb{R}$ eine $(N+1)$-mal stetig differenzierbare Funktion und $u, x \in U$ Punkte, deren Verbindungsstrecke in U liegt, so gilt:

$$f(x) = T_N f(x;u) + R_{N+1}(x;u),$$

wobei

$$T_N f(x;u) := \sum_{\substack{\mathbf{j} \in [N]^d \\ |\mathbf{j}|_1 \leq N}} \partial^{\mathbf{j}} f(u) \cdot \frac{(x-u)^{\mathbf{j}}}{\mathbf{j}!}$$

das Taylorpolynom der Ordnung N von f in u *ist und das* Restglied *mit einem geeigneten Punkt ξ auf der Verbindungsstrecke zwischen u und x in der Form*

$$R_{N+1}(x;u) = \sum_{\substack{\mathbf{j} \in [N+1]^d \\ |\mathbf{j}|_1 = N+1}} \partial^{\mathbf{j}} f(\xi) \cdot \frac{(x-u)^{\mathbf{j}}}{\mathbf{j}!}$$

dargestellt werden kann.

Wir zeigen mithilfe des folgenden Lemmas, dass wir $P_m(x)$ als eine lokale Spline-Interpolation von Taylorpolynomen von m auffassen können.

Lemma 2.5. *Sei $a > 0$ und $M \in \mathbb{N}$. Dann sind für $\mathbf{i} \in [M]^d$ die Funktionen*

$$B_{\mathbf{i}}(x) = \prod_{j=1}^{d} \left(1 - \frac{M}{2a} \cdot |x^{(j)} - x_{\mathbf{i}}^{(j)}|\right)_+ \quad \text{für } x \in [-a,a]^d, \tag{2.4}$$

mit $x_{\mathbf{i}}$ als Gitterpunkte aus Gleichung (2.2), B-Splines *auf $[-a,a]^d$, für die die folgenden drei Bedingungen gelten:*

(i) Zerlegung der Eins*:* $\sum_{\mathbf{i}\in[M]^d} B_{\mathbf{i}}(x) = 1$ *für $x \in [-a,a]^d$.*

(ii) Nichtnegativität*:* $B_{\mathbf{i}}(x) \geq 0$ *für alle $\mathbf{i} \in [M]^d$.*

(iii) Lokaler Träger*: Für Multiindizes $\mathbf{i} \in [M]^d$ ist $B_{\mathbf{i}}(x) > 0$ falls $|x^{(j)} - x_{\mathbf{i}}^{(j)}| < \frac{2a}{M}$ für alle $j \in \{1,\dots,d\}$ gilt und andernfalls $B_{\mathbf{i}}(x) = 0$.*

Beweis. Als Erstes möchten wir für $d = 2$ und $M = 3$ eine Skizze angeben, um die Idee des Beweises zu veranschaulichen. Es ist ein Gitter mit $(M+1)^d$ Gitterpunkten, die

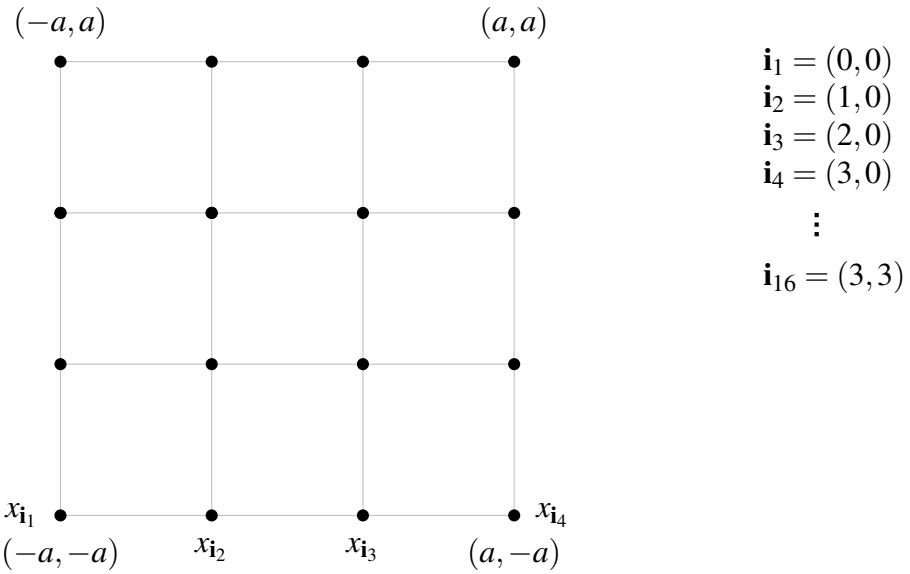

Abbildung 2.1: Beispielhafte Darstellung der $x_{\mathbf{i}_k}$ für $d = 2$ und $M = 3$.

den Punkten $x_{\mathbf{i}_k}$ entsprechen. Der Abstand zwischen zwei Gitterpunkten beträgt $\frac{2a}{M}$. Man betrachtet in Gleichung (2.4) immer den Abstand zu den nächsten 2^d Gitterpunkten, da $(1 - \frac{M}{2a} \cdot |x^{(j)} - x_{\mathbf{i}}^{(j)}|)_+ = 0$ immer dann gilt, wenn der Abstand zwischen $x^{(j)}$ und $x_{\mathbf{i}}^{(j)}$ größer als $\frac{2a}{M}$ ist.
(i) Im Folgenden wollen wir

$$\sum_{\mathbf{i}\in[M]^d} B_{\mathbf{i}}(x) = 1 \text{ für } x \in [-a,a]^d \tag{2.5}$$

per Induktion über d zeigen.

Induktionsanfang (IA): Für $d = 1$ kann x nur zwischen zwei Gitterpunkten $x_{\mathbf{i}_1} \neq x_{\mathbf{i}_2}$ liegen. Sei ohne Beschränkung der Allgemeinheit $x_{\mathbf{i}_1} \leq x \leq x_{\mathbf{i}_2}$, dann gilt mit der gleichen Begründung wie im einleitenden Beispiel:

$$\begin{aligned}\sum_{\mathbf{i}\in[M]^d} \left(1 - \frac{M}{2a} \cdot |x - x_{\mathbf{i}}|\right)_+ &= \left(1 - \frac{M}{2a} \cdot |x - x_{\mathbf{i}_1}|\right)_+ + \left(1 - \frac{M}{2a} \cdot |x - x_{\mathbf{i}_2}|\right)_+ \\ &= 1 + 1 - \frac{M}{2a} \cdot (x - x_{\mathbf{i}_1} + x_{\mathbf{i}_2} - x) \\ &= 1 + 1 - \frac{M}{2a} \cdot \frac{2a}{M} \\ &= 1,\end{aligned}$$

wobei wir unter anderem verwendet haben, dass beide Summanden unabhängig von dem Positivteil nichtnegativ sind, da der Abstand von x zu den beiden Gitterpunkten $x_{\mathbf{i}_1}$ und $x_{\mathbf{i}_2}$ kleiner gleich $\frac{2a}{M}$ ist. Zudem haben wir verwendet, dass $x_{\mathbf{i}_2} - x_{\mathbf{i}_1} = \frac{2a}{M}$ gilt, da beides Gitterpunkte sind.

Induktionshypothese (IH): Aussage (2.5) gelte für ein beliebiges aber festes $d \in \mathbb{N}$.

Induktionsschritt (IS): Wir nehmen ohne Beschränkung der Allgemeinheit an, dass $x_{(0,\ldots,0)} \leq x \leq x_{(1,\ldots,1)}$ komponentenweise gilt. Das heißt also, dass $x \in [-a, -a + \frac{2a}{M}]^{d+1}$ gilt. Im Folgenden zeigen wir

$$\sum_{\mathbf{i}\in[M]^{(d+1)}} B_{\mathbf{i}}(x) = \sum_{\mathbf{i}\in[M]^{(d+1)}} \prod_{j=1}^{d+1} \left(1 - \frac{M}{2a} \cdot \left|x^{(j)} - x_{\mathbf{i}}^{(j)}\right|\right)_+ = 1.$$

Ein Summand der obigen Summe ist Null, wenn ein $j \in \{1, \ldots, d+1\}$ existiert mit $\left|x^{(j)} - x_{\mathbf{i}}^{(j)}\right| \geq \frac{2a}{M}$. Zudem haben wir ohne Beschränkung der Allgemeinheit angenommen, dass $x \in [-a, -a + \frac{2a}{M}]^{d+1}$ gilt. Damit haben wir also nur noch 2^{d+1} Summanden, was der Anzahl der Gitterpunkte, die am nächsten bei x liegen, entspricht. Zudem wissen wir, dass alle Gitterpunkte, die in der $(d+1)$-ten Komponente den selben Wert haben, in dieser Dimension gleich weit von $x^{(d+1)}$ entfernt sind. Daraus ergibt sich, dass der Faktor $\left(1 - \frac{M}{2a} \cdot \left|x^{(d+1)} - x_{(0,\ldots,0)}^{(d+1)}\right|\right)$ bzw. $\left(1 - \frac{M}{2a} \cdot \left|x^{(d+1)} - x_{(1,\ldots,1)}^{(d+1)}\right|\right)$ in jedem Summanden vorkommt, da

$$\left(1 - \frac{M}{2a} \cdot \left|x^{(d+1)} - x_{\mathbf{i}}^{(d+1)}\right|\right) = \begin{cases} \left(1 - \frac{M}{2a} \cdot \left|x^{(d+1)} - x_{(0,\ldots,0)}^{(d+1)}\right|\right) & \mathbf{i} \in \{0,1\}^d \times \{0\} \\ \left(1 - \frac{M}{2a} \cdot \left|x^{(d+1)} - x_{(1,\ldots,1)}^{(d+1)}\right|\right) & \mathbf{i} \in \{0,1\}^d \times \{1\} \end{cases}$$

gilt. Daraus ergibt sich:

$$
\begin{aligned}
&\sum_{\mathbf{i}\in[M]^{(d+1)}} \prod_{j=1}^{d+1} \left(1 - \frac{M}{2a} \cdot \left|x^{(j)} - x_{\mathbf{i}}^{(j)}\right|\right)_+ \\
&\quad = \sum_{\mathbf{i}\in\{0,1\}^{d+1}} \prod_{j=1}^{d+1} \left(1 - \frac{M}{2a} \cdot \left|x^{(j)} - x_{\mathbf{i}}^{(j)}\right|\right) \\
&\quad = \left(\sum_{\mathbf{i}\in\{0,1\}^{d}\times\{0\}} \prod_{j=1}^{d} \left(1 - \frac{M}{2a} \cdot \left|x^{(j)} - x_{\mathbf{i}}^{(j)}\right|\right)\right) \cdot \left(1 - \frac{M}{2a} \cdot \left|x^{(d+1)} - x_{(0,\dots,0)}^{(d+1)}\right|\right) \\
&\qquad + \left(\sum_{\mathbf{i}\in\{0,1\}^{d}\times\{1\}} \prod_{j=1}^{d} \left(1 - \frac{M}{2a} \cdot \left|x^{(j)} - x_{\mathbf{i}}^{(j)}\right|\right)\right) \cdot \left(1 - \frac{M}{2a} \cdot \left|x^{(d+1)} - x_{(1,\dots,1)}^{(d+1)}\right|\right) \\
&\quad \overset{\text{(IH)}}{=} 1 \cdot \left(1 - \frac{M}{2a} \cdot \left|x^{(d+1)} - x_{(0,\dots,0)}^{(d+1)}\right|\right) + 1 \cdot \left(1 - \frac{M}{2a} \cdot \left|x^{(d+1)} - x_{(1,\dots,1)}^{(d+1)}\right|\right) \\
&\quad = 1 + 1 - \frac{M}{2a} \cdot \left(x^{(d+1)} - x_{(0,\dots,0)}^{(d+1)} + x_{(1,\dots,1)}^{(d+1)} - x^{(d+1)}\right) \\
&\quad = 1 + 1 - 1 \\
&\quad = 1,
\end{aligned}
$$

wobei wir bei der vorletzten Gleichung angewendet haben, dass $x_{(1,\dots,1)}^{(d+1)} - x_{(0,\dots,0)}^{(d+1)} = \frac{2a}{M}$ ist, da beides Gitterpunkte sind. (□)

(ii) Es folgt $\prod_{j=1}^{d}(1 - \frac{M}{2a} \cdot |x^{(j)} - x_{\mathbf{i}}^{(j)}|)_+ \geq 0$ für alle $\mathbf{i} \in [M]^d$, da

$$z_+ = \max\{z, 0\} \geq 0 \text{ für } z \in \mathbb{R}$$

gilt.

(iii) Es handelt sich hierbei um einen lokalen Träger, da nach der Konstruktion von $B_{\mathbf{i}}(x)$ der Funktionswert genau dann Null ist, wenn ein $j \in \{1, \dots, d\}$ existiert, sodass $|x^{(j)} - x_{\mathbf{i}}^{(j)}| \geq \frac{2a}{M}$ gilt. Andernfalls erhalten wir mit Bedingung (ii), dass $B_{\mathbf{i}}(x) > 0$ ist. □

Mit der Definition der B-Splines aus Lemma 2.5 erhalten wir nun:

$$P_m(x) = \sum_{\mathbf{i}\in[M]^d} p_{\mathbf{i}}^m(x) \cdot B_{\mathbf{i}}(x). \tag{2.6}$$

Daher können wir $P_m(x)$ als eine Spline-Interpolation von Taylorpolynomen von m auffassen. Die Wahl der Architektur $(L, \mathbf{k})$ unseres Neuronale-Netze-Regressionsschätzers und der Werte aller Gewichte bis auf die der Ausgabeschicht ist durch folgendes Approximationsresultat motiviert.

Lemma 2.6. *Sei $M \in \mathbb{N}$, $a > 0$ und f eine (p,C)-glatte Funktion, wobei $p = q + s$ mit $q \in \mathbb{N}_0$, $s \in (0,1]$ und $C > 0$ sind. Sei zudem $P_f(x)$ analog zu Gleichung (2.6) eine lokale Spline-Interpolation von Taylorpolynomen von f auf dem Würfel $[-a,a]^d$. Dann gilt:*

$$\sup_{x \in [-a,a]^d} |f(x) - P_f(x)| \leq c \cdot \left(\frac{a}{M}\right)^p,$$

mit einer Konstante c, die von p, d, s und C abhängt.

Beweis. Nach Lemma 2.4 über die Lagrange Form des Restglieds existiert ein ξ auf der Verbindungsstrecke zwischen $x_{\mathbf{i}}$ und x so, dass

$$\begin{aligned} f(x) &= T_{q-1} f(x; x_{\mathbf{i}}) + R_q(x; x_{\mathbf{i}}) \\ &= \sum_{\substack{\mathbf{j} \in [q-1]^d \\ |\mathbf{j}|_1 \leq q-1}} \partial^{\mathbf{j}} f(x_{\mathbf{i}}) \cdot \frac{(x - x_{\mathbf{i}})^{\mathbf{j}}}{\mathbf{j}!} + \sum_{\substack{\mathbf{j} \in [q]^d \\ |\mathbf{j}|_1 = q}} \partial^{\mathbf{j}} f(\xi) \cdot \frac{(x - x_{\mathbf{i}})^{\mathbf{j}}}{\mathbf{j}!} \end{aligned} \tag{2.7}$$

gilt. Nach der B-Spline Eigenschaft (i) aus Gleichung (2.5) erhalten wir

$$f(x) = \sum_{\mathbf{i} \in [M]^d} f(x) \prod_{j=1}^{d} \left(1 - \frac{M}{2a} \cdot |x^{(j)} - x_{\mathbf{i}}^{(j)}|\right)_+.$$

Mithilfe der Dreiecksungleichung und der Konstruktion von $P_f(x)$ erhalten wir:

$$|f(x) - P_f(x)| \leq \sum_{\mathbf{i} \in [M]^d} \left| f(x) - \sum_{\substack{\mathbf{j} \in [q]^d \\ |\mathbf{j}|_1 \leq q}} \partial^{\mathbf{j}} f(x_{\mathbf{i}}) \cdot \frac{(x - x_{\mathbf{i}})^{\mathbf{j}}}{\mathbf{j}!} \right| \prod_{j=1}^{d} \left(1 - \frac{M}{2a} \cdot |x^{(j)} - x_{\mathbf{i}}^{(j)}|\right)_+. \tag{2.8}$$

Nach Gleichung (2.7) erhalten wir:

$$\begin{aligned} &\left| f(x) - \sum_{\substack{\mathbf{j} \in [q]^d \\ |\mathbf{j}|_1 \leq q}} \partial^{\mathbf{j}} f(x_{\mathbf{i}}) \cdot \frac{(x - x_{\mathbf{i}})^{\mathbf{j}}}{\mathbf{j}!} \right| \\ &= \left| \sum_{\substack{\mathbf{j} \in [q-1]^d \\ |\mathbf{j}|_1 \leq q-1}} \partial^{\mathbf{j}} f(x_{\mathbf{i}}) \cdot \frac{(x - x_{\mathbf{i}})^{\mathbf{j}}}{\mathbf{j}!} + \sum_{\substack{\mathbf{j} \in [q]^d \\ |\mathbf{j}|_1 = q}} \partial^{\mathbf{j}} f(\xi) \frac{(x - x_{\mathbf{i}})^{\mathbf{j}}}{\mathbf{j}!} - \sum_{\substack{\mathbf{j} \in [q]^d \\ |\mathbf{j}|_1 \leq q}} \partial^{\mathbf{j}} f(x_{\mathbf{i}}) \cdot \frac{(x - x_{\mathbf{i}})^{\mathbf{j}}}{\mathbf{j}!} \right| \\ &= \left| \sum_{\substack{\mathbf{j} \in [q]^d \\ |\mathbf{j}|_1 = q}} \partial^{\mathbf{j}} f(\xi) \frac{(x - x_{\mathbf{i}})^{\mathbf{j}}}{\mathbf{j}!} - \sum_{\substack{\mathbf{j} \in [q]^d \\ |\mathbf{j}|_1 = q}} \partial^{\mathbf{j}} f(x_{\mathbf{i}}) \frac{(x - x_{\mathbf{i}})^{\mathbf{j}}}{\mathbf{j}!} \right|. \end{aligned} \tag{2.9}$$

Aus der (p,C)-Glattheit von f mit $\xi, x_{\mathbf{i}} \in \mathbb{R}^d$ folgt durch Gleichung (2.9) und Lemma 2.3:

$$\left| \sum_{\substack{\mathbf{j} \in [q]^d \\ |\mathbf{j}|_1 = q}} \partial^{\mathbf{j}} f(\xi) \frac{(x - x_{\mathbf{i}})^{\mathbf{j}}}{\mathbf{j}!} - \sum_{\substack{\mathbf{j} \in [q]^d \\ |\mathbf{j}|_1 = q}} \partial^{\mathbf{j}} f(x_{\mathbf{i}}) \frac{(x - x_{\mathbf{i}})^{\mathbf{j}}}{\mathbf{j}!} \right| \leq \binom{d+q-1}{q} \cdot \|\xi - x_{\mathbf{i}}\|^s \cdot C \cdot \|x - x_{\mathbf{i}}\|_\infty^q, \tag{2.10}$$

wobei $\|x-x_{\mathbf{i}}\|_\infty = \max_{1\le k\le d}\left|x^{(k)}-x_{\mathbf{i}}^{(k)}\right|$. Fassen wir die Gleichungen (2.8), (2.9) und (2.10) zusammen, erhalten wir:

$$\begin{aligned}|f(x)-P_f(x)| &\le \sum_{\mathbf{i}\in[M]^d}\left|f(x)-\sum_{\substack{\mathbf{j}\in[q]^d\\ |\mathbf{j}|_1\le q}}\partial^{\mathbf{j}}f(x_{\mathbf{i}})\cdot\frac{(x-x_{\mathbf{i}})^{\mathbf{j}}}{\mathbf{j}!}\right|\prod_{j=1}^{d}\left(1-\frac{M}{2a}\cdot|x^{(j)}-x_{\mathbf{i}}^{(j)}|\right)_+\\ &\le \binom{d+q-1}{q}\cdot C\cdot\sum_{\mathbf{i}\in[M]^d}\|\xi-x_{\mathbf{i}}\|^s\cdot\|x-x_{\mathbf{i}}\|_\infty^q\prod_{j=1}^{d}\left(1-\frac{M}{2a}\cdot|x^{(j)}-x_{\mathbf{i}}^{(j)}|\right)_+.\end{aligned} \tag{2.11}$$

Wir können in Gleichung (2.4) immer den Abstand zu den nächsten 2^d Gitterpunkten betrachten, da wir wissen, dass $(1-\frac{M}{2a}\cdot|x^{(j)}-x_{\mathbf{i}}^{(j)}|)_+=0$ immer dann gilt, wenn der Abstand zwischen $x^{(j)}$ und $x_{\mathbf{i}}^{(j)}$ größer als $\frac{2a}{M}$ ist. Daraus folgt aus der Eigenschaft der Zerlegung der Eins aus Gleichung (2.5)

$$\sum_{\mathbf{i}\in[M]^d}\|\xi-x_{\mathbf{i}}\|^s\cdot\|x-x_{\mathbf{i}}\|_\infty^q\prod_{j=1}^{d}\left(1-\frac{M}{2a}\cdot|x^{(j)}-x_{\mathbf{i}}^{(j)}|\right)_+\le d^{s/2}\cdot\left(\frac{2a}{M}\right)^p \tag{2.12}$$

mit $p=q+s$. Setzen wir nun Ungleichung (2.12) in (2.11) ein, erhalten wir:

$$\begin{aligned}|f(x)-P_f(x)| &\le \binom{d+q-1}{q}\cdot C\cdot\left(\frac{2a}{M}\right)^p\cdot d^{s/2}\\ &= c\cdot\left(\frac{a}{M}\right)^p.\end{aligned} \tag{2.13}$$

Die Konstante in Gleichung (2.13) lautet

$$c=\binom{d+q-1}{q}\cdot C\cdot 2^p\cdot d^{s/2}.$$

Bilden wir schließlich in Gleichung (2.13) noch das Supremum über $x\in[-a,a]^d$, erhalten wir die Behauptung. □

Durch geeignet gewählte $a_{\mathbf{i},\mathbf{j}}\in\mathbb{R}$ lässt sich $P_m(x)$ in die Form

$$\sum_{\mathbf{i}\in[M]^d}\sum_{\substack{\mathbf{j}\in[q]^d\\ |\mathbf{j}|_1\le q}}a_{\mathbf{i},\mathbf{j}}\cdot(x-x_{\mathbf{i}})^{\mathbf{j}}\prod_{j=1}^{d}\left(1-\frac{M}{2a}\cdot|x^{(j)}-x_{\mathbf{i}}^{(j)}|\right)_+$$

bringen, da sich jedes $p_{\mathbf{i}}^m(x)$ als Polynom umordnen lässt und wir daher auch $P_m(x)$ umschreiben können.

Als Nächstes wollen wir geeignete neuronale Netze $f_{\text{net},\mathbf{j},\mathbf{i}}$ mit Architektur $(L,\mathbf{k})$ definieren, die die Funktionen

$$x\mapsto(x-x_{\mathbf{i}})^{\mathbf{j}}\prod_{j=1}^{d}\left(1-\frac{M}{2a}\cdot|x^{(j)}-x_{\mathbf{i}}^{(j)}|\right)_+$$

approximieren, um anschließend Linearkombinationen

$$\sum_{\mathbf{i}\in[M]^d} \sum_{\substack{\mathbf{j}\in[q]^d \\ |\mathbf{j}|_1\leq q}} a_{\mathbf{i},\mathbf{j}} \cdot f_{\text{net},\mathbf{j},\mathbf{i}}(x) \qquad (a_{\mathbf{i},\mathbf{j}} \in \mathbb{R})$$

zu betrachten. Um dies zu erreichen, wählen wir als Aktivierungsfunktion den logistischen Squasher

$$\sigma(x) = \frac{1}{(1+\exp(-x))} \quad (x \in \mathbb{R})$$

aus Gleichung (1.1). Zudem wählen wir $R \geq 1$ und erhalten für die neuronalen Netze aus Kapitel 1:

- $f_{\text{id}}(x) = 4R \cdot \sigma\left(\frac{x}{R}\right) - 2R.$
- $f_{\text{mult}}(x,y) = \frac{R^2}{4} \cdot \frac{(1+\exp(-1))^3}{\exp(-2)-\exp(-1)} \cdot \left(\sigma\left(\frac{2(x+y)}{R}+1\right) - 2 \cdot \sigma\left(\frac{x+y}{R}+1\right) - \sigma\left(\frac{2(x-y)}{R}+1\right) + 2 \cdot \sigma\left(\frac{x-y}{R}+1\right)\right).$
- $f_{\text{ReLU}}(x) = f_{\text{mult}}(f_{\text{id}}(x), \sigma(R \cdot x)).$
- $f_{\text{hat},y}(x) = f_{\text{ReLU}}\left(\frac{M}{2a} \cdot (x-y)+1\right) - 2 \cdot f_{\text{ReLU}}\left(\frac{M}{2a} \cdot (x-y)\right) + f_{\text{ReLU}}\left(\frac{M}{2a} \cdot (x-y)-1\right).$

Mit diesen neuronalen Netzen können wir nun $f_{\text{net},\mathbf{j},\mathbf{i}}$ rekursiv definieren.

Definition 2.7. Sei $N \in \mathbb{N}$ und $q \in \mathbb{N}_0$ mit $N \geq q$. Sei zudem $s = \lceil \log_2(N+d) \rceil$, $\mathbf{i} \in [M]^d$ und $\mathbf{j} \in [N]^d$. Dann ist das neuronale Netz $f_{\text{net},\mathbf{j},\mathbf{i}}$ definiert durch:

$$f_{\text{net},\mathbf{j},\mathbf{i}}(x) = f_1^{(0)}(x),$$

wobei

$$f_k^{(l)}(x) = f_{\text{mult}}\left(f_{2k-1}^{(l+1)}(x), f_{2k}^{(l+1)}(x)\right)$$

für $k \in \{1,2,\ldots,2^l\}$ und $l \in \{0,\ldots,s-1\}$. Zudem ist

$$f_k^{(s)}(x) = f_{\text{id}}(f_{\text{id}}(x^{(l)} - x_{\mathbf{i}_k}^{(l)}))$$

für $j_1 + j_2 + \cdots + j_{l-1} + 1 \leq k \leq j_1 + j_2 + \cdots + j_l$ und $1 \leq l \leq d$ und

$$f_{|\mathbf{j}|_1+k}^{(s)}(x) = f_{\text{hat},x_{\mathbf{i}_k}^{(k)}}(x^{(k)})$$

für $1 \leq k \leq d$ und

$$f_k^{(s)}(x) = 1$$

für $|\mathbf{j}|_1 + d + 1 \leq k \leq 2^s$.

Das folgende Lemma ist ein Spezialfall von [BKK19, Lemma 5] und liefert das eingangs erwähnte Approximationsresultat für neuronale Netze $f_{\text{net},\mathbf{j},\mathbf{i}}$ aus Definition 2.7. Dieses Lemma wird hier nur der Vollständigkeit halber und ohne Beweis aufgeführt

Lemma 2.8. *Sei $M \in \mathbb{N}$ und $\sigma\colon \mathbb{R} \to [0,1]$ 2-zulässig. Sei $a \geq 1$ und $R \in \mathbb{R}$ mit*

$$R \geq \max\Bigg\{ \frac{\|\sigma''\|_\infty \cdot (M+1)}{2 \cdot |\sigma'(t_\sigma)|}, \frac{9 \cdot \|\sigma''\|_\infty \cdot a}{|\sigma'(t_\sigma)|}, \frac{20 \cdot \|\sigma'''\|_\infty}{3 \cdot |\sigma''(t_\sigma)|} \cdot 3^{3 \cdot 3^s} \cdot a^{3 \cdot 2^s},$$
$$1792 \cdot \frac{\max\{\|\sigma''\|_\infty, \|\sigma'''\|_\infty, 1\}}{\min\{2 \cdot |\sigma'(t_\sigma)|, |\sigma''(t_\sigma)|, 1\}} \cdot M^3 \Bigg\},$$

wobei $s = \lceil \log_2(N+d) \rceil$ mit $N \in \mathbb{N}$. Sei $x_{\mathbf{i}_k} \in [-a,a]^d$, $\mathbf{j} \in [N]^d$, $\mathbf{i} \in [M]^d$ und $f_{\text{net},\mathbf{j},\mathbf{i}}$ das neuronale Netz aus Definition 2.7. Dann erhalten wir für $x \in [-a,a]^d$:

$$\left| f_{\text{net},\mathbf{j},\mathbf{i}}(x) - (x - x_{\mathbf{i}_k})^{\mathbf{j}} \prod_{j=1}^{d} \left(1 - \frac{M}{2a} \cdot \left| x^{(j)} - x_{\mathbf{i}_k}^{(j)} \right| \right)_+ \right| \leq c \cdot 3^{3 \cdot 3^s} \cdot a^{3 \cdot 2^s} \cdot M^3 \cdot \frac{1}{R}$$

für eine von n unabhängige Konstante $c > 0$.

Da das neuronale Netz $f_{\text{net},\mathbf{j},\mathbf{i}}$ aus mehreren neuronalen Netzen zusammengebaut wurde, lässt sich dadurch auch die Anzahl an Schichten und Neuronen pro Schicht durch diese Struktur erklären. Aus der rekursiven Definition 2.7 entnimmt man, dass $f_{\text{net},\mathbf{j},\mathbf{i}}$ insgesamt $s+2$ verborgene Schichten, durch s-maliges Anwenden von f_{mult} und einer Anwendung von f_{hat} bzw. $f_{\text{id}}(f_{\text{id}})$ hat. Da f_{hat} zwei verborgene Schichten besitzt, ergibt sich daraus die Anzahl an verborgenen Schichten von $f_{\text{net},\mathbf{j},\mathbf{i}}$. Die Anzahl der Neuronen pro verborgener Schicht von $f_{\text{net},\mathbf{j},\mathbf{i}}$ ergibt sich wie folgt:

- Die erste verborgene Schicht enthält maximal $3 \cdot 2 \cdot 2^s = 6 \cdot 2^s$ Neuronen, da dies die erste verborgene Schicht von f_{hat} ist und maximal 2^s-mal aufgerufen wird.
- Die zweite verborgene Schicht enthält maximal $3 \cdot 4 \cdot 2^s = 12 \cdot 2^s$ Neuronen, da dies die zweite verborgene Schicht von f_{hat} ist und maximal 2^s-mal aufgerufen wird.
- Die verborgenen Schichten $3, 4, \ldots, s+1, s+2$ enthalten maximal $2^{s+1}, 2^s, \ldots, 2^3, 2^2$ Neuronen, da wir s-mal f_{mult} verschachtelt aufrufen.

Wie in Kapitel 1 bereits erwähnt, erhält man ein nicht Fully-connected neuronales Netz, indem man die Gewichte der Verbindungen zwischen zwei Neuronen in einem Fully-connected neuronalen Netz auf Null setzt. Daher liegt auch $f_{\text{net},\mathbf{j},\mathbf{i}}$ nach Definition 1.1 in $\mathfrak{N}(s+2, \{24 \cdot (N+d)\}^{s+2}, \sigma)$, da die größte Anzahl an Neuronen in einer Schicht

$$12 \cdot 2^s = 12 \cdot 2^{\lceil \log_2(N+d) \rceil} \leq 12 \cdot 2^{\log_2(N+d)+1} = 24 \cdot (N+d)$$

ist. Weiterhin erkennt man durch die Zusammensetzung der neuronalen Netze, dass alle Gewichte im Betrag durch $c \cdot \max\{\frac{M}{2a}, R^2\}$ beschränkt sind, wobei $c > 0$ ist.

2.2 Bestimmung der Gewichte der Ausgabeschicht

Wir definieren unseren Neuronale-Netze-Regressionsschätzer $\tilde{m}_n(x)$ durch:

$$\tilde{m}_n(x) := \sum_{\mathbf{i}\in[M]^d} \sum_{\substack{\mathbf{j}\in[N]^d \\ |\mathbf{j}|_1\leq N}} a_{\mathbf{i},\mathbf{j}} \cdot f_{\text{net},\mathbf{j},\mathbf{i}}(x), \tag{2.14}$$

wobei n die Größe unserer gegebenen Datenmenge $\mathscr{D}_n$ ist und wir die Koeffizienten $a_{\mathbf{i},\mathbf{j}}$ durch die Lösung eines Kleinste-Quadrate-Problems erhalten. Dazu betrachten wir die Tikhonov Regularisierung (vgl. [Kre98, Kapitel 16.1])

$$\frac{1}{n}\sum_{i=1}^{n} |Y_i - \tilde{m}_n(X_i)|^2 + \frac{c}{n} \cdot \|a_{\mathbf{i},\mathbf{j}}\|_2^2 \tag{2.15}$$

mit Regularitätsparameter $\frac{c}{n}$ für eine von n unabhängige Konstante $c > 0$ und wollen im Folgenden die Gleichung (2.15) minimieren. Zunächst stellen wir Gleichung (2.15) als Gleichungssystem dar. Dafür definieren wir uns die Menge

$$\mathscr{U} := \{U_s : s = 1,\dots,S\} := \left\{ f_{\text{net},\mathbf{j},\mathbf{i}}(x) : \mathbf{i} \in [M]^d \text{ und } |\mathbf{j}|_1 \leq N \text{ mit } \mathbf{j} \in [N]^d \right\}$$

wobei

$$S := \left|[M]^d\right| \cdot \binom{N+d}{d} = (M+1)^d \cdot \binom{N+d}{d}$$

die Kardinalität von $\mathscr{U}$ ist. Dies sieht man wie folgt über ein Kombinatorik Argument. Wir wissen, dass es insgesamt $(M+1)^d$ Möglichkeiten gibt d-viele Zahlen aus einer Menge der Größe $(M+1)$ mit Zurücklegen und mit Beachtung der Reihenfolge zu ziehen. Mit Zurücklegen, da man mehrmals eine Zahl ziehen kann und mit Beachtung der Reihenfolge, da wir Vektoren betrachten und die Komponenten nicht vertauschbar sind. Für jede dieser $(M+1)^d$ Möglichkeiten ist noch zu beachten, dass wir zusätzlich d-mal aus einer Menge mit $(N+1)$-vielen Zahlen ziehen und gleichzeitig die Bedingung beachten müssen, dass die Summe der gezogenen d Elemente zwischen Null und N liegt. Gesucht ist also

$$\left| \left\{ \mathbf{j} \in [N]^d : |\mathbf{j}|_1 \leq N \right\} \right| =: H.$$

Wir stellen fest, dass

$$\begin{aligned}&\left\{ \mathbf{j} \in [N]^d : |\mathbf{j}|_1 \leq N \right\} \\ &= \left\{ \mathbf{j} \in [N]^d : |\mathbf{j}|_1 = 0 \right\} \cup \left\{ \mathbf{j} \in [N]^d : |\mathbf{j}|_1 = 1 \right\} \cup \dots \cup \left\{ \mathbf{j} \in [N]^d : |\mathbf{j}|_1 = N \right\}\end{aligned}$$

gilt. Mit Lemma 2.3 wissen wir, dass für $d, N \in \mathbb{N}$ und $k \in \mathbb{N}_0$ mit $k \leq N$ die Identität

$$\left| \left\{ \mathbf{j} \in [N]^d : |\mathbf{j}|_1 = k \right\} \right| = \binom{d+k-1}{k} = \binom{d+k-1}{d-1}$$

gilt. Damit erhalten wir

$$H = \sum_{k=0}^{N} \left| \left\{ \mathbf{j} \in [N]^d : |\mathbf{j}|_1 = k \right\} \right| = \sum_{k=0}^{N} \binom{d+k-1}{d-1} = \binom{N+d}{d},$$

mit der *Hockey-Stick Identität* (vgl. [TEC19, Theorem 10.14])

$$\sum_{i=0}^{n-r} \binom{i+r}{r} = \sum_{i=r}^{n} \binom{i}{r} = \binom{n+1}{r+1} \quad (n, r \in \mathbb{N} \text{ mit } n \geq r).$$

Wir setzen nun

$$\mathbf{U} = (U_s(X_i))_{1 \leq i \leq n, 1 \leq s \leq S} \quad \text{und} \quad \mathbf{Y} = (Y_i)_{1 \leq i \leq n}.$$

Der Schätzer aus Gleichung (2.14) lässt sich nun umschreiben zu

$$\tilde{m}_n(x) = \sum_{s=1}^{S} a_s \cdot U_s(x) \tag{2.16}$$

mit $(a_s)_{s=1,\dots,S} = \mathbf{a} \in \mathbb{R}^S$. Für die Tikhonov Regularisierung aus Gleichung (2.15) erhalten wir:

$$\begin{aligned} &\frac{1}{n} \sum_{i=1}^{n} |Y_i - \tilde{m}_n(X_i)|^2 + \frac{c}{n} \cdot \sum_{\mathbf{i} \in [M]^d} \sum_{\substack{\mathbf{j} \in [N]^d \\ |\mathbf{j}|_1 \leq N}} a_{\mathbf{i},\mathbf{j}}^2 \\ &= \frac{1}{n} (\mathbf{Y} - \mathbf{U}\mathbf{a})^T (\mathbf{Y} - \mathbf{U}\mathbf{a}) + \frac{c}{n} \cdot \mathbf{a}^T \mathbf{a}. \end{aligned} \tag{2.17}$$

Im folgenden Lemma bestimmen wir den Koeffizientenvektor $\mathbf{a}$ so, dass Gleichung (2.17) minimal wird.

Lemma 2.9. *Das Funktional*

$$\varphi(\mathbf{a}) := \frac{1}{n} (\mathbf{Y} - \mathbf{U}\mathbf{a})^T (\mathbf{Y} - \mathbf{U}\mathbf{a}) + \frac{c}{n} \cdot \mathbf{a}^T \mathbf{a},$$

besitzt einen eindeutigen Minimierer $\mathbf{a} \in \mathbb{R}^S$.

Beweis. Es gilt $\mathbf{a}^T \mathbf{U}^T \mathbf{Y} = \mathbf{Y}^T \mathbf{U}\mathbf{a}$, da dieser Ausdruck eine reelle Zahl und damit insbesondere symmetrisch ist. Damit erhalten wir für $\varphi(\mathbf{a})$:

$$\begin{aligned} \varphi(\mathbf{a}) &= \frac{1}{n} (\mathbf{Y}^T \mathbf{Y} - \mathbf{Y}^T \mathbf{U}\mathbf{a} - \mathbf{a}^T \mathbf{U}^T \mathbf{Y} + \mathbf{a}^T \mathbf{U}^T \mathbf{U}\mathbf{a}) + \frac{c}{n} \cdot \mathbf{a}^T \mathbf{a} \\ &= \frac{1}{n} (\mathbf{Y}^T \mathbf{Y} - 2\mathbf{Y}^T \mathbf{U}\mathbf{a}) + \mathbf{a}^T \left(\frac{1}{n} \mathbf{U}^T \mathbf{U} + \frac{c}{n} \cdot \mathbf{1} \right) \mathbf{a} \\ &= \frac{1}{n} (\mathbf{Y}^T \mathbf{Y} - 2\mathbf{Y}^T \mathbf{U}\mathbf{a}) + \mathbf{a}^T \mathbf{A}\mathbf{a}, \end{aligned} \tag{2.18}$$

mit

$$\mathbf{A} := \frac{1}{n} \mathbf{U}^T \mathbf{U} + \frac{c}{n} \cdot \mathbf{1}.$$

Die Matrix $\mathbf{U}^T\mathbf{U} \in \mathbb{R}^{S\times S}$ ist positiv definit, denn aufgrund der Rechenregeln der Transponierten und des Standardskalarprodukts sowie der positiven Definitheit des Standardskalarprodukts gilt für alle $x \in \mathbb{R}^s \setminus \{0\}$:

$$\langle x, \mathbf{U}^\mathbf{T}\mathbf{U}x\rangle = \langle \mathbf{U}x, \mathbf{U}x\rangle > 0.$$

Zudem wissen wir dass $\frac{c}{n}\mathbf{1}$ durch die Wahl von c nur positive Eigenwerte besitzt und damit positiv definit ist. Daher wissen wir, dass die Matrix $\mathbf{A}$ positiv definit und insbesondere invertierbar ist, da die Eigenwerte positiv sind. Das folgt daraus, dass die ohnehin schon positiven Eigenwerte von $\frac{1}{n}\mathbf{U}^T\mathbf{U}$ um $\frac{c}{n}$ verschoben werden und damit positiv bleiben. Zudem ist die Matrix $\mathbf{A}$ als Summe von zwei symmetrischen Matrizen ebenfalls symmetrisch. Setzen wir

$$\mathbf{b} := \frac{1}{n}\cdot\mathbf{A}^{-1}\mathbf{U}^T\mathbf{Y} \in \mathbb{R}^S$$

folgt mit der Symmetrie der Matrix $\mathbf{A}$ die Identität

$$\mathbf{b}^T\mathbf{A}\mathbf{a} = \mathbf{a}^T\mathbf{A}\mathbf{b} = \frac{1}{n}\cdot\mathbf{a}^T\mathbf{U}^T\mathbf{Y} = \frac{1}{n}\cdot\mathbf{Y}^T\mathbf{U}\mathbf{a} \in \mathbb{R}.$$

Daraus ergibt sich mit

$$\mathbf{Y}^T\mathbf{U}\mathbf{a} = \frac{n}{2}\cdot\Big(\mathbf{b}^T\mathbf{A}\mathbf{a} + \mathbf{a}^T\mathbf{A}\mathbf{b}\Big)$$

und

$$0 = \mathbf{b}^T\mathbf{U}^T\mathbf{Y} - \mathbf{Y}^T\mathbf{U}\mathbf{b}$$

die Gleichung

$$\begin{aligned}\mathbf{Y}^T\mathbf{U}\mathbf{a} &= \frac{n}{2}\Big(\mathbf{b}^T\mathbf{A}\mathbf{a} + \mathbf{a}^T\mathbf{A}\mathbf{b}\Big) = \frac{n}{2}\Big(\mathbf{b}^T\mathbf{A}\mathbf{a} + \mathbf{a}^T\mathbf{A}\mathbf{b} - \frac{1}{n}\mathbf{b}^T\mathbf{U}^T\mathbf{Y} + \frac{1}{n}\mathbf{Y}^T\mathbf{U}\mathbf{b}\Big)\\ &= \frac{n}{2}\Big(\mathbf{b}^T\mathbf{A}\mathbf{a} + \mathbf{a}^T\mathbf{A}\mathbf{b} - \frac{1}{n}\mathbf{b}^T\mathbf{A}\Big(\mathbf{A}^{-1}\mathbf{U}^T\mathbf{Y}\Big) + \frac{1}{n}\mathbf{Y}^T\mathbf{U}\mathbf{b}\Big) \qquad (2.19)\\ &= \frac{n}{2}\Big(\mathbf{b}^T\mathbf{A}\mathbf{a} + \mathbf{a}^T\mathbf{A}\mathbf{b} - \mathbf{b}^T\mathbf{A}\mathbf{b} + \frac{1}{n^2}\mathbf{Y}^T\mathbf{U}\mathbf{A}^{-1}\mathbf{U}^T\mathbf{Y}\Big).\end{aligned}$$

Damit erhalten wir in Gleichung (2.18):

$$\begin{aligned}\varphi(\mathbf{a}) &= \frac{1}{n}(\mathbf{Y}^T\mathbf{Y} - 2\mathbf{Y}^T\mathbf{U}\mathbf{a}) + \mathbf{a}^T\mathbf{A}\mathbf{a}\\ &= \frac{1}{n}\mathbf{Y}^T\mathbf{Y} - \mathbf{b}^T\mathbf{A}\mathbf{a} - \mathbf{a}^T\mathbf{A}\mathbf{b} + \mathbf{b}^T\mathbf{A}\mathbf{b} - \frac{1}{n^2}\mathbf{Y}^T\mathbf{U}\mathbf{A}^{-1}\mathbf{U}^T\mathbf{Y} + \mathbf{a}^T\mathbf{A}\mathbf{a}\\ &= (\mathbf{a}-\mathbf{b})^T\mathbf{A}(\mathbf{a}-\mathbf{b}) + \frac{1}{n}\mathbf{Y}^T\mathbf{Y} - \frac{1}{n^2}\mathbf{Y}^T\mathbf{U}\mathbf{A}^{-1}\mathbf{U}^T\mathbf{Y}.\end{aligned}$$

Hierbei erkennen wir, dass für $\mathbf{a} = \mathbf{b}$ das Funktional φ minimal wird, da wir wissen, dass $\mathbf{A}$ positiv definit ist und damit $x^T\mathbf{A}x > 0$ für alle $x \in \mathbb{R}^S$ mit $x \neq 0$ gilt. Aus der

positiven Definitheit von $\mathbf{A}$ folgt zudem $(\mathbf{a}-\mathbf{b})^T\mathbf{A}(\mathbf{a}-\mathbf{b}) = 0$ genau dann, wenn $\mathbf{a} = \mathbf{b}$ gilt. Daraus folgt, dass das Funktional einen eindeutigen Minimierer $\mathbf{a} = \mathbf{b} = \frac{1}{n} \cdot \mathbf{A}^{-1}\mathbf{U}^T\mathbf{Y} \in \mathbb{R}^S$ besitzt. □

Für den Koeffizientenvektor unseres Schätzers $\tilde{m}_n$ wählen wir die Lösung aus Lemma 2.9.

Bemerkung 2.10. Da der Koeffizientenvektor $\mathbf{a}$ die Gleichung (2.17) minimiert, erhalten wir, wenn wir den Koeffizientenvektor gleich Null setzen:

$$\frac{c}{n} \cdot \mathbf{a}^T\mathbf{a} \leq \frac{1}{n}(\mathbf{Y}-\mathbf{U}\mathbf{a})^T(\mathbf{Y}-\mathbf{U}\mathbf{a}) + \frac{c}{n} \cdot \mathbf{a}^T\mathbf{a} \leq \frac{1}{n}\sum_{i=1}^{n} Y_i^2,$$

was uns erlaubt eine obere Schranke für den absoluten Wert unserer Koeffizienten abzuleiten. Daraus können wir folgern, dass unser Neuronale-Netze-Regressionsschätzer $\tilde{m}_n$ beschränkt ist, da die neuronalen Netze $f_{\text{net},\mathbf{j},\mathbf{i}}$ nach Konstruktion ebenfalls beschränkt sind.

Kapitel 3

Resultat zur Konvergenzgeschwindigkeit

In diesem Kapitel stellen wir das Hauptresultat dieser Arbeit vor. Wir betrachten im Folgenden das Problem 2.1 und wollen nun eine Abschätzung des erwarteten L_2-Fehlers

$$\mathbb{E}\left[\int |m_n(x) - m(x)|^2 \mathbb{P}_X(dx)\right]$$

im Falle des Schätzers

$$m_n(x) := T_{\beta_n} \tilde{m}_n(x), \tag{3.1}$$

mit $\beta_n = c_1 \cdot \log(n)$ für eine hinreichend große und von n unabhängige Konstante $c_1 > 0$ und unter Annahme einer (p,C)-glatten Regressionsfunktion m herleiten. Hierbei bezeichnet $T_\beta z = \max\{\min\{z,\beta\},-\beta\}$ für $z \in \mathbb{R}$ und $\beta > 0$. Weiterhin bezeichnen wir mit $\tilde{m}_n$ unseren Neuronale-Netze-Regresssionsschätzer aus Kapitel 2.2, welcher aus mehreren neuronalen Netzen konstruiert wurde. Für diesen gilt: Die Aktivierungsfunktion σ ist der logistische Squasher, $N \geq q$, $M = M_n = \lceil c_2 \cdot n^{1/(2p+d)} \rceil$ mit $c_2 > 0$ und unabhängig von n, $R = R_n = n^{d+4}$ und $a = a_n = (\log n)^{1/(6(N+d))}$.
Nun kommen wir zu dem Hauptresultat dieser Arbeit.

Hauptsatz 3.1 ([BKK19, Theorem 1])**.** *Angenommen die Verteilung von Y erfüllt*

$$\mathbb{E}\left[e^{c_3 \cdot |Y|^2}\right] < \infty$$

für eine Konstante $c_3 > 0$ und die Verteilung von X besitzt einen beschränkten Träger $\operatorname{supp}(\mathbb{P}_X)$. *Sei $m(x) = \mathbb{E}[Y \mid X = x]$ die zu dem Tupel (X,Y) gehörige Regressionsfunktion. Angenommen m ist (p,C)-glatt, mit $p = q + s$, wobei $q \in \mathbb{N}_0$, $s \in (0,1]$ und $C > 0$ ist. Des Weiteren sei m_n der in Gleichung (3.1) definierte Regressionsschätzer.*
Dann gilt für hinreichend großes n:

$$\mathbb{E}\left[\int |m_n(x) - m(x)|^2 \mathbb{P}_X(dx)\right] \leq c_{\text{fin}} \cdot (\log n)^3 \cdot n^{-\frac{2p}{2p+d}},$$

wobei $c_{\text{fin}} > 0$ *eine von n unabhängige Konstante ist.*

Bevor wir zum Beweis von Hauptresultat 3.1 kommen, stellen wir im folgenden Abschnitt die dafür notwendigen Hilfsresultate vor.

3.1 Approximationsresultate für Hauptsatz 3.1

Die nächsten Definitionen und Lemmata benötigen wir für den Beweis unseres Hauptresultats, einer Aussage über die Konvergenzgeschwindigkeit unseres Neuronale-Netze-Regressionsschätzers. Die Lemmata werden hier nur der Vollständigkeit halber und ohne Beweis aufgeführt. Als Nächstes geben wir eine Definition von Überdeckungszahlen an, da wir im Beweis für unser Hauptresultat eine Abschätzung einer L_p-ε-Überdeckungszahl anwenden.

Definition 3.2. Sei (X, δ) ein pseudometrischer Raum (vgl. [Bar15, Definition 2.1.1]). Für $x \in X$ und $\varepsilon > 0$ sei:

$$U_\varepsilon(x) = \{z \in X : \delta(x,z) < \varepsilon\}$$

die offene Kugel um x mit Radius ε.

a) $\{z_1, \dots, z_N\} \subseteq X$ heißt *ε-Überdeckung* einer Menge $A \subseteq X$, falls gilt:

$$A \subseteq \bigcup_{k=1}^{N} U_\varepsilon(z_k).$$

b) Ist $A \subseteq X$ und $\varepsilon > 0$, so ist die sogenannte *ε-Überdeckungszahl* von A in (X, δ) definiert als:

$$\mathcal{N}_{(X,\delta)}(\varepsilon, A) := \inf\big\{|U| : U \subseteq X \text{ ist } \varepsilon\text{-Überdeckung von } A\big\}.$$

Da wir in unserem Hauptresultat eine Überdeckungszahl von Mengen von Funktionen betrachten werden, benötigen wir folgende Definition.

Definition 3.3. Sei $\mathcal{F}$ eine Menge von Funktionen $f\colon \mathbb{R}^d \to \mathbb{R}$, sei $\varepsilon > 0$, $1 \leq p < \infty$ und seien $x_1, \dots, x_n \in \mathbb{R}^d$ und $x_1^n = (x_1, \dots, x_n)$. Dann ist die *$L_p$-$\varepsilon$-Überdeckungszahl von $\mathcal{F}$ auf x_1^n* definiert durch:

$$\mathcal{N}_p(\varepsilon, \mathcal{F}, x_1^n) := \mathcal{N}_{(X,\delta)}(\varepsilon, \mathcal{F}),$$

wobei der pseudometrische Raum (X, δ) gegeben ist durch X als Menge aller Funktionen $f\colon \mathbb{R}^d \to \mathbb{R}$ und durch die Pseudometrik $\delta(f,g) = \delta_p(f,g) = (\frac{1}{n}\sum_{i=1}^n |f(x_i) - g(x_i)|^p)^{1/p}$.

Das folgende Lemma beschreibt wie man den erwarteten L_2-Fehler eines Schätzers mithilfe einer L_p-ε-*Überdeckungszahl* abschätzen kann. Dieses Lemma ist ein Spezialfall von [BKK19, Lemma 8].

Lemma 3.4. *Sei $\beta_n = c_1 \cdot \log(n)$ für eine hinreichend große Konstante $c_1 > 0$. Angenommen die Verteilung von Y erfüllt*

$$\mathbb{E}\left[\mathrm{e}^{c_2 \cdot |Y|^2}\right] < \infty$$

für eine Konstante $c_2 > 0$. Zudem nehmen wir an, dass die Regressionsfunktion m beschränkt ist. Sei $\mathscr{F}_n$ eine Menge von Funktionen $f\colon \mathbb{R}^d \to \mathbb{R}$. Sei $\hat{m}_n$ ein Schätzer für m mit

$$\hat{m}_n = T_{\beta_n} \mathring{m}_n = \max\left\{ \min\{\mathring{m}_n, \beta_n\}, -\beta_n \right\}$$

für eine Funktion

$$\mathring{m}_n(\cdot) = \mathring{m}_n(\cdot, (X_1, Y_1), \ldots, (X_n, Y_n)) \in \mathscr{F}_n,$$

welche die Ungleichung

$$\frac{1}{n}\sum_{i=1}^{n} |Y_i - \mathring{m}_n(X_i)|^2 \leq \frac{1}{n}\sum_{i=1}^{n} |Y_i - g_n(X_i)|^2 + \mathrm{pen}_n(g_n) \tag{3.2}$$

mit einer deterministischen Funktion $g_n\colon \mathbb{R}^d \to \mathbb{R}$ und deterministischem Penalty Term $\mathrm{pen}_n(g_n) \geq 0$ erfüllt. Dann gilt für den erwarteten L_2-Fehler die Ungleichung

$$\begin{aligned}
&\mathbb{E}\left[\int |\hat{m}_n(x) - m(x)|^2 \mathbb{P}_X(dx)\right] \\
&\leq \frac{1}{n} \cdot c \cdot \log(n)^2 \cdot \left(\log\left(\sup_{x_1^n \in (\mathrm{supp}(X))^n} \mathscr{N}_1\Big(\frac{1}{n \cdot \beta_n}, \mathscr{F}_n, x_1^n\Big)\right) + 1 \right) \\
&\quad + 2 \cdot \mathbb{E}\left[\int |g_n(x) - m(x)|^2 \mathbb{P}_X(dx) + \mathrm{pen}_n(g_n)\right],
\end{aligned} \tag{3.3}$$

für $n > 1$ und eine von n unabhängige Konstante $c > 0$.

Das nächste Lemma benötigen wir, um in Ungleichung (3.3) die Überdeckungszahl $\mathscr{N}_1\big(\frac{1}{n\cdot\beta_n}, \mathscr{F}_n, x_1^n\big)$ weiter abzuschätzen.

Lemma 3.5 ([BKK19, Lemma 9])**.** *Seien $a > 0$ und $d, L, N, J_n \in \mathbb{N}$ so, dass $J_n \leq n^{c_1}$ für eine Konstante $c_1 > 0$ gilt und setze $\beta_n = c_2 \cdot \log(n)$ für eine hinreichend große Konstante $c_2 > 0$. Sei die Funktion $\sigma\colon \mathbb{R} \to [0,1]$ 2-zulässig. Sei $\mathscr{F}$ die Menge aller Funktionen die durch Definition 1.1 definiert sind mit $k_1 = k_2 = \cdots = k_L = 24 \cdot (N + d)$ und einer Beschränkung des Betrags der Gewichte durch $c_3 \cdot n^{c_4}$ für Konstanten $c_3, c_4 > 0$. Sei*

$$\mathscr{F}^{(J_n)} := \left\{ \sum_{j=1}^{J_n} a_j \cdot f_j : f_j \in \mathscr{F} \quad \textit{und} \quad \sum_{j=1}^{J_n} a_j^2 \leq c_5 \cdot n^{c_6} \right\}$$

für Konstanten $c_5, c_6 > 0$. Dann gilt für $n > 1$:

$$\log\left(\sup_{x_1^n \in [-a,a]^{d\cdot n}} \mathscr{N}_1\Big(\frac{1}{n\cdot\beta_n}, \mathscr{F}^{(J_n)}, x_1^n\Big)\right) \leq c\cdot\log(n)\cdot J_n,$$

für eine Konstante c die nur von L, N, a und d abhängt.

Mit den Approximationsresultaten aus diesem Abschnitt verfügen wir nun über alle Bausteine, um unser Hauptresultat zu beweisen.

3.2 Der Beweis von Hauptsatz 3.1

Wir betrachten das Ereignis

$$A_n := \left[\frac{1}{n}\sum_{i=1}^{n} Y_i^2 \leq 1 + \mathbb{E}[Y^2]\right]. \tag{3.4}$$

Damit können wir den erwarteten L_2-Fehler umschreiben zu:

$$\begin{aligned}
\mathbb{E}\left[\int |m_n(x) - m(x)|^2 \mathbb{P}_X(dx)\right] &= \mathbb{E}\left[\int |m_n(x) - m(x)|^2 \mathbb{P}_X(dx)\cdot(\mathbb{1}_{A_n^c} + \mathbb{1}_{A_n})\right] \\
&= \mathbb{E}\left[\int |m_n(x) - m(x)|^2 \mathbb{P}_X(dx)\mathbb{1}_{A_n^c}\right] \\
&\quad + \mathbb{E}\left[\int |m_n(x) - m(x)|^2 \mathbb{P}_X(dx)\mathbb{1}_{A_n}\right] \\
&=: T_{1,n} + T_{2,n}.
\end{aligned} \tag{3.5}$$

In den folgenden Abschnitten kümmern wir uns um die Abschätzung der Summanden $T_{1,n}$ und $T_{2,n}$.

3.2.1 Abschätzung von $T_{1,n}$

Für zwei beliebige reelle Zahlen $u, v \in \mathbb{R}$ folgt aus $0 \leq (u-v)^2 = u^2 + v^2 - 2uv$ die Ungleichung $u^2 + v^2 \geq 2uv$ und damit schließlich:

$$|(u-v)^2| = |u^2 - 2uv + v^2| \leq u^2 + 2uv + v^2 \leq 2u^2 + 2v^2. \tag{3.6}$$

Wir wissen, dass aufgrund der Unabhängigkeit und identischen Verteiltheit der $\mathbb{R}^d \times \mathbb{R}$-wertigen Zufallsvariablen $(X,Y), (X_1,Y_1), (X_2,Y_2), \ldots$ die Zufallsvariablen $Y_1, \ldots, Y_n$ ebenso wie die Zufallsvariablen $X_1, \ldots, X_n$ unabhängig und identisch verteilt sind. Mit Ungleichung (3.6) und durch die Unabhängigkeit von A_n von den Zufallsvariablen $X, X_1, \ldots, X_n$

erhalten wir für den ersten Summanden aus Gleichung (3.5):

$$\begin{aligned} T_{1,n} = \mathbb{E}\left[\int |m_n(x)-m(x)|^2 \mathbb{P}_X(dx) \mathbb{1}_{A_n^c}\right] &= \mathbb{E}\left[\int |m_n(x)-m(x)|^2 \mathbb{P}_X(dx)\right] \cdot \mathbb{P}(A_n^c) \\ &\leq \mathbb{E}\left[\int 2m_n(x)^2 + 2m(x)^2 \mathbb{P}_X(dx)\right] \cdot \mathbb{P}(A_n^c) \\ &\leq \mathbb{E}\left[\int 2\beta_n^2 + 2\beta_n^2 \mathbb{P}_X(dx)\right] \cdot \mathbb{P}(A_n^c). \end{aligned} \tag{3.7}$$

Da nach Voraussetzung $\operatorname{supp}(\mathbb{P}_X)$ beschränkt und definitionsgemäß immer abgeschlossen ist, wissen wir nach dem Satz von Heine-Borel (siehe z.B. [For16, Satz 5]), dass $\operatorname{supp}(\mathbb{P}_X)$ kompakt ist. Da m als (p,C)-glatte Funktion insbesondere stetig ist, wissen wir, dass sie auf einer kompakten Menge ein Maximum und Minimum annimmt. Dadurch können wir n so groß wählen, dass ohne Beschränkung der Allgemeinheit $\|m\|_\infty \leq \beta_n$ gilt. Da wir nach Bemerkung 2.10 wissen, dass $\tilde{m}_n$ beschränkt ist, ist m_n nach Konstruktion ebenfalls beschränkt. Wir haben daher bei Ungleichung (3.7) zudem verwendet, dass $\max\{\|m\|_\infty, \|m_n\|_\infty\} \leq \beta_n$ nach Definition von m_n gilt.
Im nächsten Schritt wollen wir die Wahrscheinlichkeit $\mathbb{P}(A_n^c)$ abschätzen. Da die Zufallsvariablen $Y_1, \ldots, Y_n$ unabhängig und identisch verteilt sind, folgern wir daraus mit der Linearität des Erwartungswerts $\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^n Y_i^2\right] = \mathbb{E}[Y^2]$. Mithilfe der Monotonie der Wahrscheinlichkeitsfunktion $\mathbb{P}$ und der Chebyshev-Ungleichung für $\varepsilon = 1$ (siehe z.B. [Kle13, Satz 5.11]) erhalten wir:

$$\mathbb{P}(A_n^c) = \mathbb{P}\Big(\frac{1}{n}\sum_{i=1}^n Y_i^2 - \mathbb{E}[Y^2] \geq 1\Big) \leq \mathbb{P}\Big(\Big|\frac{1}{n}\sum_{i=1}^n Y_i^2 - \mathbb{E}[Y^2]\Big| \geq 1\Big) \leq \mathbb{V}\Big[\frac{1}{n}\sum_{i=1}^n Y_i^2\Big].$$

Da die Zufallsvariablen $Y_1, \ldots, Y_n$ u.i.v. sind, folgt mit den Rechenregeln der Varianz:

$$\mathbb{P}(A_n^c) \leq \frac{n \cdot \mathbb{V}[Y^2]}{n^2} = \frac{\mathbb{V}[Y^2]}{n} = \frac{c_4}{n}, \tag{3.8}$$

wobei $c_4 := \mathbb{V}[Y^2]$ ist.
Mit Ungleichung (3.8) erhalten wir in Ungleichung (3.7):

$$\mathbb{E}\left[\int 2\beta_n^2 + 2\beta_n^2 \mathbb{P}_X(dx)\right] \cdot \mathbb{P}(A_n^c) \leq 4\beta_n^2 \cdot \mathbb{P}(A_n^c) \overset{(3.8)}{\leq} \frac{4 \cdot c_4 \cdot \beta_n^2}{n},$$

wobei wir bei der ersten Ungleichung verwendet haben, dass β_n deterministisch und die Wahrscheinlichkeit $\mathbb{P}(X \in \operatorname{supp}(\mathbb{P}_X)) = 1$ ist. Schließlich erhalten wir in Ungleichung (3.7), da für n hinreichend groß $\log(n)^3 > 1$ gilt:

$$T_{1,n} \leq \frac{4 \cdot c_4 \cdot \beta_n^2}{n} \leq c_5 \cdot \log(n)^3 \cdot n^{-1} \leq c_5 \cdot \log(n)^3 \cdot n^{-\frac{2p}{2p+d}}, \tag{3.9}$$

mit einer von n unabhängigen Konstante $c_5 := 4 \cdot c_4 \cdot c_1^2 > 0$.
Damit haben wir $T_{1,n}$ entsprechend der rechten Seite von Hauptsatz 3.1 abgeschätzt und wollen als Nächstes $T_{2,n}$ abschätzen.

3.2.2 Abschätzung von $T_{2,n}$

In diesem Abschnitt wollen wir Lemma 3.4 für die Abschätzung von $T_{2,n}$ anwenden. Sei dafür

$$\hat{m}_n := \mathbb{1}_{A_n} m_n + \mathbb{1}_{A_n^c} T_{\beta_n} g_n = T_{\beta_n}\big(\mathbb{1}_{A_n}\tilde{m}_n + \mathbb{1}_{A_n^c} g_n\big),$$

wobei g_n definiert ist über

$$g_n(x) := \sum_{\mathbf{i}\in[M_n]^d} \sum_{\substack{\mathbf{j}\in[q]^d \\ |\mathbf{j}|_1 \le q}} \frac{1}{\mathbf{j}!} \cdot \partial^{\mathbf{j}} m(x_{\mathbf{i}}) \cdot f_{\text{net},\mathbf{j},\mathbf{i}}(x)$$

mit $x_{\mathbf{i}} \in [-a_n, a_n]^d$.
Durch unsere Definition von $\hat{m}_n$ erhalten wir durch die Monotonie des Erwartungswerts und einer Abschätzung über den ganzen Raum:

$$\begin{aligned} T_{2,n} = \mathbb{E}\left[\int |m_n(x) - m(x)|^2 \mathbb{P}_X(dx) \mathbb{1}_{A_n}\right] &= \mathbb{E}\left[\int |\hat{m}_n(x) - m(x)|^2 \mathbb{P}_X(dx) \mathbb{1}_{A_n}\right] \\ &\le \mathbb{E}\left[\int |\hat{m}_n(x) - m(x)|^2 \mathbb{P}_X(dx)\right]. \end{aligned} \tag{3.10}$$

Da m nach Voraussetzung (p,C)-glatt ist, existiert für alle $\alpha = (\alpha_1, \ldots, \alpha_d) \in \mathbb{N}_0^d$ mit $\sum_{j=1}^d \alpha_j = q$ die partielle Ableitung $\partial^\alpha m$. Insbesondere existiert ein $z \in \mathbb{R}$ mit

$$z := \max_{\mathbf{i}\in[M_n]^d, \mathbf{j}\in[q]^d, |\mathbf{j}|_1 \le q} \left| \frac{1}{\mathbf{j}!} \cdot \partial^{\mathbf{j}} m(x_{\mathbf{i}}) \right| < \infty. \tag{3.11}$$

Sei $\mathscr{F}$ die Menge aller Funktionen aus Definition 1.1 mit Aktivierungsfunktion σ,

$$L = s + 2 = \lceil \log_2(N+d) \rceil + 2, \quad \text{mit } k_1 = k_2 = \cdots = k_L = 24 \cdot (N+d)$$

und einer Konstante $c_6 > 0$ so, dass der Betrag der Gewichte durch n^{c_6} beschränkt ist. Wir definieren

$$\mathscr{F}^{(J_n)} := \left\{ \sum_{j=1}^{J_n} a_j \cdot f_j : f_j \in \mathscr{F} \text{ und } \sum_{j=1}^{J_n} a_j^2 \le c_7 \cdot n \right\}$$

mit

$$J_n = (M_n + 1)^d \cdot \left| \left\{ \mathbf{j} \in [N]^d : |\mathbf{j}|_1 \le N \right\} \right|$$

und

$$c_7 := \max\left\{\frac{1+\mathbb{E}[Y^2]}{c_8}, c_2^d \cdot (N+1)^d \cdot z^2\right\}. \tag{3.12}$$

Hierbei bezeichnet c_8 die Konstante des Regularitätsterms aus Gleichung (2.15). Da wir uns in der nichtparametrischen Regressionsschätzung befinden, gilt unter anderem die Bedingung $\mathbb{E}[Y^2] < \infty$ und daher ist c_7 auch wohldefiniert. Weiterhin folgt wie in Kapitel 2.2 mit $S = J_n$,

$$J_n = (M_n+1)^d \cdot \binom{N+d}{d} \leq (M_n+1)^d \cdot (N+1)^d. \tag{3.13}$$

Um Lemma 3.4 anwenden zu können, zeigen wir im Folgenden zunächst $g_n, \tilde{m}_n \in \mathscr{F}^{(J_n)}$. Da nach Konstruktion $f_{\text{net},\mathbf{j},\mathbf{i}} \in \mathscr{F}$ ist, folgt mit Gleichung (3.12), dass für n hinreichend groß die Abschätzung

$$\begin{aligned}\sum_{\mathbf{i}\in[M_n]^d} \sum_{\substack{\mathbf{j}\in[q]^d \\ |\mathbf{j}|_1 \leq q}} \left|\frac{1}{\mathbf{j}!} \cdot \partial^{\mathbf{j}} m(x_{\mathbf{i}})\right|^2 &\leq (M_n+1)^d (N+1)^d \cdot z^2 \\ &\leq (2c_2 \cdot n^{1/2p+d})^d \cdot (N+1)^d \cdot z^2 \\ &\leq c_7 \cdot n\end{aligned}$$

gilt und damit g_n in $\mathscr{F}^{(J_n)}$ liegt.
Wir zeigen nun $\tilde{m}_n \in \mathscr{F}^{(J_n)}$. Nach Gleichung (2.16) können wir unseren Schätzer $\tilde{m}_n$ darstellen durch:

$$\tilde{m}_n(x) = \sum_{j=1}^{J_n} \hat{a}_j \cdot f_j$$

für geeignete $f_j \in \mathscr{F}$ und $\hat{a}_j$, welche die Ungleichung

$$\frac{c_8}{n}\sum_{j=1}^{J_n} \hat{a}_j^2 = \frac{c_8}{n} \sum_{\mathbf{i}\in[M_n]^d} \sum_{\substack{\mathbf{j}\in[q]^d \\ |\mathbf{j}|_1 \leq q}} a_{\mathbf{i},\mathbf{j}}^2 \leq \frac{1}{n}\sum_{i=1}^{n} |Y_i - \tilde{m}_n(X_i)|^2 + \frac{c_8}{n} \sum_{\mathbf{i}\in[M_n]^d} \sum_{\substack{\mathbf{j}\in[q]^d \\ |\mathbf{j}|_1 \leq q}} a_{\mathbf{i},\mathbf{j}}^2 \leq \sum_{i=1}^{n} Y_i^2$$

erfüllen, wobei wir bei der letzten Ungleichung wie in Kapitel 2 die minimierende Eigenschaft von $a_{\mathbf{i},\mathbf{j}}$ verwendet haben und zum Schluss die Koeffizienten Null gesetzt haben. Da $c_8 > 0$ ist, erhalten wir damit, dass die Koeffizienten $\hat{a}_j$ die Eigenschaft

$$\sum_{j=1}^{J_n} \hat{a}_j^2 \leq \frac{1}{n}\sum_{i=1}^{n} Y_i^2 \cdot \frac{n}{c_8} \tag{3.14}$$

erfüllen müssen. Da Ungleichung (3.14) insbesondere auf dem Ereignis A_n gilt, folgt aus der Definition von A_n in Gleichung (3.4) und der Definition der Konstante c_7 in

Gleichung (3.12):

$$\sum_{j=1}^{J_n} \hat{a}_j^2 \leq \frac{1+\mathbb{E}[Y^2]}{c_8} \cdot n \leq c_7 \cdot n,$$

woraus durch $f_j \in \mathscr{F}$ schließlich $\tilde{m}_n \in \mathscr{F}^{(J_n)}$ folgt.

Als Nächstes wollen wir Ungleichung (3.2) aus Lemma 3.4 für $\tilde{m}_n$ und g_n zeigen. Die Funktionen $\tilde{m}_n$ und g_n unterscheiden sich in den Vorfaktoren von f_j. Die Koeffizienten $a_{\mathbf{i},\mathbf{j}}$ von $\tilde{m}_n$ haben wir durch Minimierung des Funktionals φ aus Lemma 2.9 erhalten. Nach Voraussetzung ist $N \geq q$ und damit gilt dann insbesondere $\{0,\ldots,q\} \subseteq \{0,\ldots,N\}$, daher wurden bei der Minimierung des Funktionals unter anderem die Koeffizienten von g_n betrachtet. Daher erhalten wir:

$$\begin{aligned}
\frac{1}{n}\sum_{i=1}^{n} |Y_i - \tilde{m}_n(X_i)|^2 &\leq \frac{1}{n}\sum_{i=1}^{n} |Y_i - \tilde{m}_n(X_i)|^2 + \frac{c_8}{n} \cdot \sum_{\mathbf{i}\in[M_n]^d} \sum_{\substack{\mathbf{j}\in[N]^d \\ |\mathbf{j}|_1 \leq N}} a_{\mathbf{i},\mathbf{j}}^2 \\
&\leq \frac{1}{n}\sum_{i=1}^{n} |Y_i - g_n(X_i)|^2 + \frac{c_8}{n} \cdot \sum_{\mathbf{i}\in[M_n]^d} \sum_{\substack{\mathbf{j}\in[q]^d \\ |\mathbf{j}|_1 \leq q}} \left| \frac{1}{\mathbf{j}!} \cdot \partial^{\mathbf{j}} m(x_{\mathbf{i}}) \right|^2 \qquad (3.15) \\
&\leq \frac{1}{n}\sum_{i=1}^{n} |Y_i - g_n(X_i)|^2 + c_9 \cdot \frac{(M_n+1)^d}{n},
\end{aligned}$$

mit $c_9 = c_8 \cdot (q+1)^d \cdot z^2$ als Konstante, die unabhängig von n ist. Wir erhalten damit für $\overline{m}_n \in \{\tilde{m}_n, g_n\} \subseteq \mathscr{F}^{(J_n)}$:

$$\frac{1}{n}\sum_{i=1}^{n} |Y_i - \overline{m}_n(X_i)|^2 \leq \frac{1}{n}\sum_{i=1}^{n} |Y_i - g_n(X_i)|^2 + c_9 \cdot \frac{(M_n+1)^d}{n}, \qquad (3.16)$$

da für $\overline{m}_n = g_n$ die Ungleichung unmittelbar folgt. Da g_n nach Definition deterministisch, damit also unabhängig von $(X_1,Y_1),\ldots,(X_n,Y_n)$ ist, sind mit der Abschätzung (3.16) für $\hat{m}_n = T_{\beta_n}\overline{m}_n$ mit $\overline{m}_n \in \mathscr{F}^{(J_n)}$ und dem *Penalty Term* $\mathrm{pen}_n(g_n) = c_9 \cdot \frac{(M_n+1)^d}{n} > 0$ die

Voraussetzungen für Lemma 3.4 erfüllt. Wir erhalten durch dessen Anwendung:

$$\begin{aligned}
&\mathbb{E}\left[\int |\hat{m}_n(x) - m(x)|^2 \mathbb{P}_X(dx)\right] \\
&\leq \frac{1}{n} \cdot c_9 \cdot \log(n)^2 \cdot \left(\log\left(\sup_{x_1^n \in (\mathrm{supp}(X))^n} \mathscr{N}_1\left(\frac{1}{n \cdot \beta_n}, \mathscr{F}^{(J_n)}, x_1^n\right)\right) + 1 \right) \\
&\quad + 2 \cdot \mathbb{E}\left[\int |g_n(x) - m(x)|^2 \mathbb{P}_X(dx) + c_9 \cdot \frac{(M_n+1)^d}{n}\right] \\
&= \frac{1}{n} \cdot c_9 \cdot \log(n)^2 \cdot \left(\log\left(\sup_{x_1^n \in (\mathrm{supp}(X))^n} \mathscr{N}_1\left(\frac{1}{n \cdot \beta_n}, \mathscr{F}^{(J_n)}, x_1^n\right)\right) + 1 \right) \\
&\quad + 2 \int |g_n(x) - m(x)|^2 \mathbb{P}_X(dx) + 2 \cdot c_9 \cdot \frac{(M_n+1)^d}{n} \\
&=: T_{2,A,n} + 2 \cdot T_{2,B,n} + 2 \cdot c_9 \cdot \frac{(M_n+1)^d}{n},
\end{aligned} \tag{3.17}$$

wobei wir bei der letzten Gleichheit verwendet haben, dass der letzte Summand deterministisch ist. Zudem wissen wir, dass c_9 unabhängig von n ist und $n > 1$, da wir n hinreichend groß wählen. Des Weiteren erhalten wir, da für n hinreichend groß $\log(n)^3 > 1$ gilt:

$$c_9 \cdot \frac{(M_n+1)^d}{n} \leq c_{10} \cdot \frac{n^{\frac{d}{2p+d}}}{n} \leq c_{10} \cdot \log(n)^3 \cdot n^{-\frac{2p}{2p+d}}, \tag{3.18}$$

mit einer von n unabhängigen Konstanten $c_{10} \coloneqq c_9 \cdot (2c_2)^d > 0$. Mithilfe von Ungleichung (3.18) erhalten wir nun in Gleichung (3.17):

$$\mathbb{E}\left[\int |\hat{m}_n(x) - m(x)|^2 \mathbb{P}_X(dx)\right] \leq T_{2,A,n} + 2 \cdot T_{2,B,n} + 2 \cdot c_{10} \cdot \log(n)^3 \cdot n^{-\frac{2p}{2p+d}}.$$

Wir wollen im Folgenden die Summanden $T_{2,A,n}$ und $T_{2,B,n}$ weiter abschätzen.

3.2.2.1 Abschätzung von $T_{2,A,n}$

Als Erstes überprüfen wir die Voraussetzungen von Lemma 3.5, um damit dann den Summanden, welcher die Überdeckungszahl $\mathscr{N}_1$ enthält, in Ungleichung (3.17) weiter abzuschätzen. Nach Voraussetzung ist $\beta_n = c_1 \cdot \log(n)$ und $a_n = (\log n)^{1/(6(N+d))} > 0$ für hinreichend großes n. Nach Voraussetzung sind zudem $d, N, J_n \in \mathbb{N}$ und es gilt nach Gleichung (3.13):

$$J_n \leq (M_n+1)^d \cdot (N+1)^d \leq n^\gamma,$$

für hinreichend großes n und Konstante $\gamma > 0$. Wir betrachten hier den logistischen Squasher σ welcher nach Lemma 1.5 insbesondere 2-zulässig ist. Da die hier betrachtete Menge

von Funktionen $\mathscr{F}^{(J_n)}$ identisch mit der aus Lemma 3.5 ist, sind nun alle Voraussetzungen für Lemma 3.5 erfüllt. Nach Voraussetzung wissen wir, dass $\operatorname{supp}(\mathbb{P}_X)$ beschränkt ist und wir können n so groß wählen, dass wir ohne Beschränkung der Allgemeinheit annehmen können, dass $\operatorname{supp}(\mathbb{P}_X) = \{x \in \mathbb{R}^d \mid \forall \varepsilon > 0 : \mathbb{P}_X(S_\varepsilon(x)) > 0\} \subseteq [-a_n, a_n]^d$ ist, mit S_ε als ε-Umgebung um $x \in \mathbb{R}^d$. In der nächsten Ungleichungskette bezeichnen wir mit c die Konstante aus Lemma 3.5. Wir erhalten damit durch Lemma 3.5 für hinreichend großes n:

$$\begin{aligned} T_{2,A,n} &= \frac{1}{n} \cdot c_9 \cdot \log(n)^2 \cdot \left(\log \left(\sup_{x_1^n \in (\operatorname{supp}(X))^n} \mathscr{N}_1\Big(\frac{1}{n \cdot \beta_n}, \mathscr{F}^{(J_n)}, x_1^n\Big)\right) + 1 \right) \\ &\leq \frac{1}{n} \cdot c_9 \cdot \log(n)^2 \cdot \Big(\big(c \cdot \log(n) \cdot J_n\big) + 1 \Big) \\ &\leq \frac{1}{n} \cdot c_9 \cdot \log(n)^2 \cdot \big(2 \cdot c \cdot \log(n) \cdot J_n\big) \\ &\leq c_{11} \cdot \frac{1}{n} \cdot \log(n)^3 \cdot J_n, \end{aligned} \tag{3.19}$$

wobei $c_{11} \coloneqq 2c_9 \cdot c$ eine von n unabhängige Konstante ist. Durch Einsetzen der Definition von M_n erhalten wir für hinreichend großes n

$$(M_n + 1)^d \leq (c_2 \cdot n^{\frac{1}{2p+d}} + 2)^d \leq 2^d \cdot c_2^d \cdot n^{\frac{d}{2p+d}}$$

und damit schließlich für Ungleichung (3.19):

$$\begin{aligned} c_{11} \cdot \frac{1}{n} \cdot \log(n)^3 \cdot J_n &\leq c_{11} \cdot (N+1)^d \cdot 2^d \cdot c_2^d \cdot \log(n)^3 \cdot n^{-\frac{2p}{2p+d}} \\ &= c_{12} \cdot \log(n)^3 \cdot n^{-\frac{2p}{2p+d}}, \end{aligned} \tag{3.20}$$

mit einer von n unabhängigen Konstante $c_{12} \coloneqq c_{11} \cdot (N+1)^d \cdot 2^d \cdot c_2^d > 0$. Damit haben wir den Summanden aus Ungleichung (3.17), welcher die Überdeckungszahl $\mathscr{N}_1$ enthält, abschließend passend zum Endresultat abgeschätzt.

3.2.2.2 Abschätzung von $T_{2,B,n}$

Sei

$$P_{m,n}(x) \coloneqq \sum_{\mathbf{i} \in [M_n]^d} \sum_{\substack{\mathbf{j} \in [q]^d \\ |\mathbf{j}|_1 \leq q}} \frac{1}{\mathbf{j}!} \cdot \partial^{\mathbf{j}} m(x_{\mathbf{i}}) \cdot (x - x_{\mathbf{i}})^{\mathbf{j}} \prod_{j=1}^{d} \left(1 - \frac{M_n}{2a_n} \cdot \left| x^{(j)} - x_{\mathbf{i}}^{(j)} \right| \right)_+$$

wie in Gleichung (2.3) eine lokale Spline-Interpolation von Taylorpolynomen von m. Mit Ungleichung (3.6) zusammen mit einer Nulladdition und der Linearität des Integrals

erhalten wir:

$$\begin{aligned} T_{2,B,n} &= \int |g_n(x) - m(x)|^2 \mathbb{P}_X(dx) \\ &= \int |g_n(x) - P_{m,n}(x) + P_{m,n}(x) - m(x)|^2 \mathbb{P}_X(dx) \\ &\leq \int 2|g_n(x) - P_{m,n}(x)|^2 \mathbb{P}_X(dx) + \int 2|P_{m,n}(x) - m(x)|^2 \mathbb{P}_X(dx). \end{aligned}$$

Aus der Supremumseigenschaft folgt weiter

$$\begin{aligned} &\int \sup_{x\in[-a_n,a_n]^d} |g_n(x) - P_{m,n}(x)|^2 \mathbb{P}_X(dx) + \int \sup_{x\in[-a_n,a_n]^d} |P_{m,n}(x) - m(x)|^2 \mathbb{P}_X(dx) \\ &= \sup_{x\in[-a_n,a_n]^d} |g_n(x) - P_{m,n}(x)|^2 + \sup_{x\in[-a_n,a_n]^d} |P_{m,n}(x) - m(x)|^2 \qquad (3.21) \\ &=: \sup_{x\in[-a_n,a_n]^d} T^2_{2,B_1,n}(x) + \sup_{x\in[-a_n,a_n]^d} T^2_{2,B_2,n}(x), \end{aligned}$$

wobei wir im ersten Schritt $\text{supp}(\mathbb{P}_X) \subseteq [-a_n, a_n]^d$ und $\mathbb{P}(X \in \text{supp}(\mathbb{P}_X)) = 1$ verwendet haben. Damit folgt schließlich:

$$T_{2,B,n} \leq 2 \cdot \Big(\sup_{x\in[-a_n,a_n]^d} T^2_{2,B_1,n}(x) + \sup_{x\in[-a_n,a_n]^d} T^2_{2,B_2,n}(x) \Big).$$

Um die letzten beiden Summanden der Gleichung (3.21) weiter abzuschätzen, möchten wir Lemma 2.8 anwenden.

Abschätzung von $T_{2,B_1,n}$

Wir überprüfen, ob für Lemma 2.8 alle Voraussetzungen erfüllt sind. Wir betrachten wieder den logistischen Squasher σ aus Gleichung 1.1, welcher nach Lemma 1.5 insbesondere 2-zulässig ist. Zudem ist für hinreichend großes n die Bedingung

$$\begin{aligned} R_n \geq \max\Bigg\{ &\frac{\|\sigma''\|_\infty \cdot (M_n + 1)}{2 \cdot |\sigma'(t_\sigma)|}, \frac{9 \cdot \|\sigma''\|_\infty \cdot a_n}{|\sigma'(t_\sigma)|}, \frac{20 \cdot \|\sigma'''\|_\infty}{3 \cdot |\sigma''(t_\sigma)|} \cdot 3^{3 \cdot 3^s} \cdot a_n^{3 \cdot 2^s}, \\ &1792 \cdot \frac{\max\{\|\sigma''\|_\infty, \|\sigma'''\|_\infty, 1\}}{\min\{2 \cdot |\sigma'(t_\sigma)|, |\sigma''(t_\sigma)|, 1\}} \cdot M_n^3 \Bigg\} \end{aligned}$$

erfüllt. Daher gelten für unser neuronales Netz aus Definition 2.7 mit $x_{\mathbf{i}} \in [-a_n, a_n]^d$ alle Voraussetzungen für Lemma 2.8. Wir erhalten damit für $x \in [-a_n, a_n]^d$ und n hinreichend

groß:

$$\begin{aligned}\left| f_{\text{net},\mathbf{j},\mathbf{i}}(x) - (x-x_{\mathbf{i}})^{\mathbf{j}} \cdot \prod_{j=1}^{d} (1 - \frac{M_n}{2a_n} \cdot |x^{(j)} - x_{\mathbf{i}}^{(j)}|)_+ \right| &\leq c \cdot 3^{3\cdot 3^s} \cdot a_n^{3\cdot 2^s} \cdot M_n^3 \cdot \frac{1}{R_n} \\ &\leq c \cdot a_n^{3\cdot(N+d)\cdot 2} \cdot \frac{M_n^3}{R_n} \\ &= c \cdot \log(n) \cdot \frac{M_n^3}{R_n},\end{aligned} \tag{3.22}$$

wobei wir in c alle von n unabhängigen Konstanten zusammenfassen. Diese Konstante enthält unter anderem die Konstante aus Lemma 2.8. Des Weiteren haben wir verwendet, dass für hinreichend großes n die Ungleichung

$$a_n^{2^{\lceil \log_2(N+d) \rceil}} \leq a_n^{2^{\log_2(N+d)+1}} = a_n^{(N+d)\cdot 2}$$

gilt. Im letzten Schritt haben wir in Ungleichung (3.22) die Definition von a_n eingesetzt. Mit Ungleichung (3.22) und Gleichung (3.11) erhalten wir nun:

$$\begin{aligned}T_{2,B_1,n}(x) &= |g_n(x) - P_{m,n}(x)| \\ &= \left| \sum_{\mathbf{i}\in[M_n]^d} \sum_{\substack{\mathbf{j}\in[q]^d \\ |\mathbf{j}|_1 \leq q}} \frac{1}{\mathbf{j}!} \cdot \partial^{\mathbf{j}} m(x_{\mathbf{i}}) \right| \cdot \left| f_{\text{net},\mathbf{j},\mathbf{i}}(x) - (x-x_{\mathbf{i}})^{\mathbf{j}} \cdot \prod_{j=1}^{d} \left(1 - \frac{M_n}{2a_n} \cdot \left| x^{(j)} - x_{\mathbf{i}}^{(j)} \right| \right)_+ \right| \\ &\leq (M_n+1)^d \cdot (q+1)^d \cdot z \cdot \left| f_{\text{net},\mathbf{j},\mathbf{i}}(x) - (x-x_{\mathbf{i}})^{\mathbf{j}} \cdot \prod_{j=1}^{d} \left(1 - \frac{M_n}{2a_n} \cdot \left| x^{(j)} - x_{\mathbf{i}}^{(j)} \right| \right)_+ \right| \\ &\leq (M_n+1)^d \cdot c \cdot \log(n) \cdot \frac{M_n^3}{R_n},\end{aligned} \tag{3.23}$$

wobei wir in c alle von n unabhängigen Konstanten zusammenfassen, was insbesondere die Konstante c aus Lemma 2.8 einschließt. Durch Einsetzen der Definitionen von M_n und R_n erhalten wir für n hinreichend groß:

$$\frac{(M_n+1)^{8d}}{R_n^2} \leq c \cdot \frac{n^{\frac{8d}{2p+d}}}{n^{2d+8}} = c \cdot n^{\frac{8d}{2p+d} - 2d - 8} \leq c \cdot n^{\frac{8d}{2p+d} - 8\frac{2p+d}{2p+d}} = c \cdot n^{-\frac{16p}{2p+d}} \leq c \cdot n^{-\frac{2p}{2p+d}},$$

wobei in dieser Ungleichungskette $c = (2c_2)^{8d}$ gilt. Bei der letzten Ungleichung haben wir verwendet, dass für $p > 0$ auch $\frac{16p}{2p+d} > \frac{2p}{2p+d}$ gilt. Damit erhalten wir für alle $x \in [-a_n, a_n]^d$:

$$\begin{aligned}
|g_n(x) - P_{m,n}(x)|^2 &\leq \left((M_n+1)^d \cdot c \cdot \log(n) \cdot \frac{M_n^3}{R_n} \right)^2 \\
&\leq c^2 \cdot (M_n+1)^{2d} \cdot \log(n)^2 \cdot \frac{M_n^6}{R_n^2} \\
&\leq c^2 \cdot (M_n+1)^{2d} \cdot \log(n)^2 \cdot \frac{(M_n+1)^{6d}}{R_n^2} \qquad (3.24) \\
&\leq c^2 \cdot \log(n)^2 \cdot \frac{(M_n+1)^{8d}}{R_n^2} \\
&\leq c_{13} \cdot n^{-\frac{2p}{2p+d}} \cdot \log(n)^3,
\end{aligned}$$

mit $c_{13} = c^2 \cdot (2c_2)^{8d} > 0$ und unabhängig von n ist. Bei der letzten Ungleichung haben wir verwendet, dass $\log(n)^2 < \log(n)^3$ für n hinreichend groß gilt. Wenden wir schließlich noch das Supremum über $x \in [-a_n, a_n]^d$ auf Ungleichung(3.24) an, erhalten wir wie verlangt eine Abschätzung für $\sup_{x \in [-a_n, a_n]^d} T_{2,B_1,n}(x)^2$.

Abschätzung von $T_{2,B_2,n}$

Da nach Konstruktion $a_n > 0$, m (p,C)-glatt und $P_{m,n}(x)$ nach Lemma 2.5 eine Spline-Interpolation von Taylorpolynomen von m ist, erhalten wir mit Lemma 2.6 für $x \in [-a_n, a_n]^d$:

$$T_{2,B_2,n}(x) = |P_{m,n}(x) - m(x)| \leq c_{14} \cdot \frac{a_n^p}{M_n^p} \leq c_{14} \cdot \log(n) \cdot \frac{1}{M_n^p}. \qquad (3.25)$$

In dieser Ungleichung ist $c_{14} > 0$ eine von n unabhängige Konstante aus Lemma 2.6 und wir haben zudem verwendet, dass:

$$a_n^p = a_n^{q+s} \leq a_n^{N+d} \leq a_n^{6\cdot(N+d)} = \log(n),$$

für $p = q + s$ für hinreichend großes n gilt, da nach Voraussetzung $N \geq q$ und $d \geq s$ mit $s \in (0,1]$ ist. Durch Quadrieren bleibt die Ungleichung (3.25) auch erhalten und, da in der Ungleichung die rechte Seite unabhängig von x ist, gilt die Ungleichung ebenfalls für das Supremum über $x \in [-a_n, a_n]^d$. Für n hinreichend groß, erhalten wir durch Einsetzen der Definition von M_n:

$$\begin{aligned}
\sup_{x \in [-a_n, a_n]^d} |P_{m,n}(x) - m(x)|^2 &\leq \left(c_{14} \cdot \log(n) \cdot \frac{1}{M_n^p} \right)^2 \\
&\leq c_{14}^2 \cdot \log(n)^2 \cdot c_2^{-2p} \cdot n^{-\frac{2p}{2p+d}} \qquad (3.26) \\
&\leq \left(\frac{c_{14}}{c_2^p} \right)^2 \cdot \log(n)^3 \cdot n^{-\frac{2p}{2p+d}}.
\end{aligned}$$

Hiermit haben wir $\sup_{x\in[-a_n,a_n]^d} T_{2,B_2,n}(x)^2$ aus Ungleichung (3.21) abgeschätzt. Dadurch wurde schließlich auch $T_{2,B,n}$ und damit auch $T_{2,n}$ abgeschätzt.
Mit der Abschätzung von $T_{1,n}$ haben wir nun alle Summanden von Ungleichung (3.5) abgeschätzt und erhalten schließlich:

$$\mathbb{E}\left[\int |m_n(x)-m(x)|^2 \mathbb{P}_X(dx)\right] \leq c_{\text{fin}} \cdot \log(n)^3 \cdot n^{-\frac{2p}{2p+d}},$$

mit

$$c_{\text{fin}} = c_5 + 2\cdot c_{10} + c_{12} + 2\cdot\left(2\cdot\left(\left(\frac{c_{14}}{c_2^p}\right)^2 + c_{13}\right)\right),$$

wobei c_{fin} als Summe nichtnegativer und positiver Konstanten, die unabhängig von n sind, nichtnegativ und unabhängig von n ist. Damit haben wir unser Hauptresultat bewiesen. □

Kapitel 4

Anwendungsbeispiel auf simulierte Daten

In diesem Kapitel untersuchen wir die Leistung unseres Neuronale-Netze-Regressionsschätzers aus Kapitel 2 anhand von Anwendungsbeispielen auf simulierte Daten. Der Schätzer und die Beispiele wurden in *Python* [VRDJ95, Version 3.7.3] implementiert (vgl. Appendix). Im ersten Abschnitt führen wir eine Parameterstudie für unseren Neuronale-Netze-Regressionsschätzer durch. Im zweiten Abschnitt quantifizieren wir im Rahmen eines Simulationsbeispiels die Leistung unseres Neuronale-Netze-Regressionsschätzers, indem wir den empirischen L_2-Fehler dieses Schätzers und weiterer Standardschätzer berechnen und vergleichen.

4.1 Parameterstudie

In diesem Abschnitt führen wir für die Implementation unseres Neuronale-Netze-Regressionsschätzers *new_neural_network_estimate* ($m_{n,1}$) eine Parameterstudie durch. Die konkrete Implementierung des Schätzers gemäß Kapitel 2 erfordert die Festsetzung von Parametern, deren Einfluss auf die Schätzung wir in diesem Abschnitt genauer betrachten werden. Wir haben als Regressionsfunktion $m\colon [-3,3] \to \mathbb{R}$ mit

$$m(x) = \sin\left(\frac{\pi}{2} \cdot x^2\right)$$

gewählt. Diese Funktion stellt als Potenzreihe und durch ihr starkes Schwingen für Regressionsschätzer und insbesondere für unseren Neuronale-Netze-Regressionsschätzer, welcher Polynome approximiert, eine Herausforderung dar. Wir erzeugen als Realisierung von X ein Training-Sample der Größe 800 und ein Testing-Sample der Größe $n = 200$ von unabhängigen auf dem Intervall $[-3,3]$ gleichverteilten Zufallsvariablen. Zudem wählen

wir ε standardnormalverteilt und unabhängig von X und wir definieren Y durch:

$$Y = m(X) + \sigma \cdot \lambda \cdot \varepsilon. \tag{4.1}$$

Den Skalierungsfaktor $\lambda > 0$ wählen wir als Interquartilsabstand *(IQA)* von $m(X)$ auf dem Trainingsdatensatz. Für den Rauschfaktor σ gilt $\sigma = 0.05$. Dem Schätzer wird nun der Trainingsdatensatz von X und Y zum Lernen bzw. Festlegen der Gewichte nach Kapitel 2.1 gegeben.
In der Parameterstudie verändern wir bei unserem Neuronale-Netze-Regressionsschätzer den Parameter N, welcher den maximalen Grad der Polynome bestimmt, welche wir mit $f_{\text{net},\mathbf{j},\mathbf{i}}$ schätzen möchten und die ihrerseits die Regressionsfunktion m approximieren sollen (vgl. Lemma 2.8). Zudem verändern wir den Parameter M, welcher den Abstand zwischen zwei benachbarten Gitterpunkten steuert. Durch wachsendes M verfeinert sich das Gitter, welches wir über das Intervall $[-3,3]$ legen und wir vermuten, dass sich dadurch die Approximationsgüte verbessert (vgl. Lemma 2.6).
Um die Vergleichbarkeit der Ergebnisse sicherzustellen, erzeugen wir mit einem *Seed* eine reproduzierbare Realisierung der Zufallsvariablen. In der Parameterstudie betrachten wir den Neuronale-Netze-Regressionsschätzer mit Parametern $d = 1$, $q = 2$, $R = 10^6$, $a = 3$, $M \in \{2,4,8,9,16\}$ und $N \in \{2,4,8,9,16\}$. Für die Wahl der Parameter haben wir uns an [BKK19] orientiert. Es ist zu beachten, dass die Approximation durch den Schätzer besser wird je höher die Parameter sind. Hohe Parameter führen aber auch zu einem höheren Ressourcenbedarf wie z.B. der Rechenzeit. Es ist daher auch interessant zu wissen, ob der Schätzer auch bei niedriger Parameterwahl akzeptable Ergebnisse liefert.
In Abbildung 4.1 erkennen wir die Approximation der Regressionsfunktion m auf dem Testing-Sample durch unseren Neuronale-Netze-Regressionsschätzer, wobei wir $M = 2$ und $N \in \{2,4,8,16\}$ wählen. Mit diesem Test versuchen wir zu erkennen, ob die Approximation der Regressionsfunktion besser wird, wenn der maximale Grad der Polynome, die wir schätzen möchten, steigt. Anhand der Plots erkennen wir, dass sich die Approximation mit steigendem N verbessert. Der nächste Test besteht darin die Parameter $N = 2$ und $M \in \{2,4,8,16\}$ zu wählen. Durch das steigende M verfeinern wir das Gitter welches auf dem Intervall $[-3,3]$ liegt und wir können in Abbildung 4.2 beobachten wie sich mit steigendem M die Approximation der Regressionsfunktion verbessert. In Abbildung 4.3 haben wir $N = 16$ fest gewählt und betrachten variables $M \in \{2,4,8,16\}$. Der Test unterscheidet sich zu dem vorherigen nur darin, dass N hoch gewählt wurde und wir können auch wieder beobachten, dass sich die Approximation mit steigendem M verbessert. Im Vergleich zu dem vorherigen Test sind aber die Approximationen bereits mit geringem M deutlich besser. In Abbildung 4.4 haben wir $M = 16$ fest gewählt und betrachten variables $N \in \{2,4,8,16\}$. Durch alleiniges Betrachten der Plots erkennen wir, dass sich die

Approximation mit steigendem N verbessert.
Im letzten Test möchten wir, dass der Aufwand zur Lösung des Kleinste-Quadrate-Problems gleicht bleibt. Gemäß Kapitel 2.2 lösen wir für die Bestimmung der Gewichte der Ausgabeschicht ein Kleinste-Quadrate-Problem. Der Rechenaufwand für das Lösen der Normalengleichungen, welche wir in unserer Implementation verwendet haben, ist ähnlich zu dem der numerisch stabileren Lösung einer QR-Zerlegung. Bei der QR-Zerlegung mittels Housholdertransformationen beträgt die Anzahl an Operationen in unserem Fall ca. $n \cdot \big((M+1)\cdot(N+1)\big)^2$ [DR08, Kapitel 4, Seite 130]. Wir betrachten daher die Tupel $(M,N) \in \{2,4,9,16\}$ mit $(M+1)\cdot(N+1) \approx 51$. In Abbildung 4.5 erkennen wir, dass die Approximation bei $(2,16)$ und $(16,2)$ besser ist.

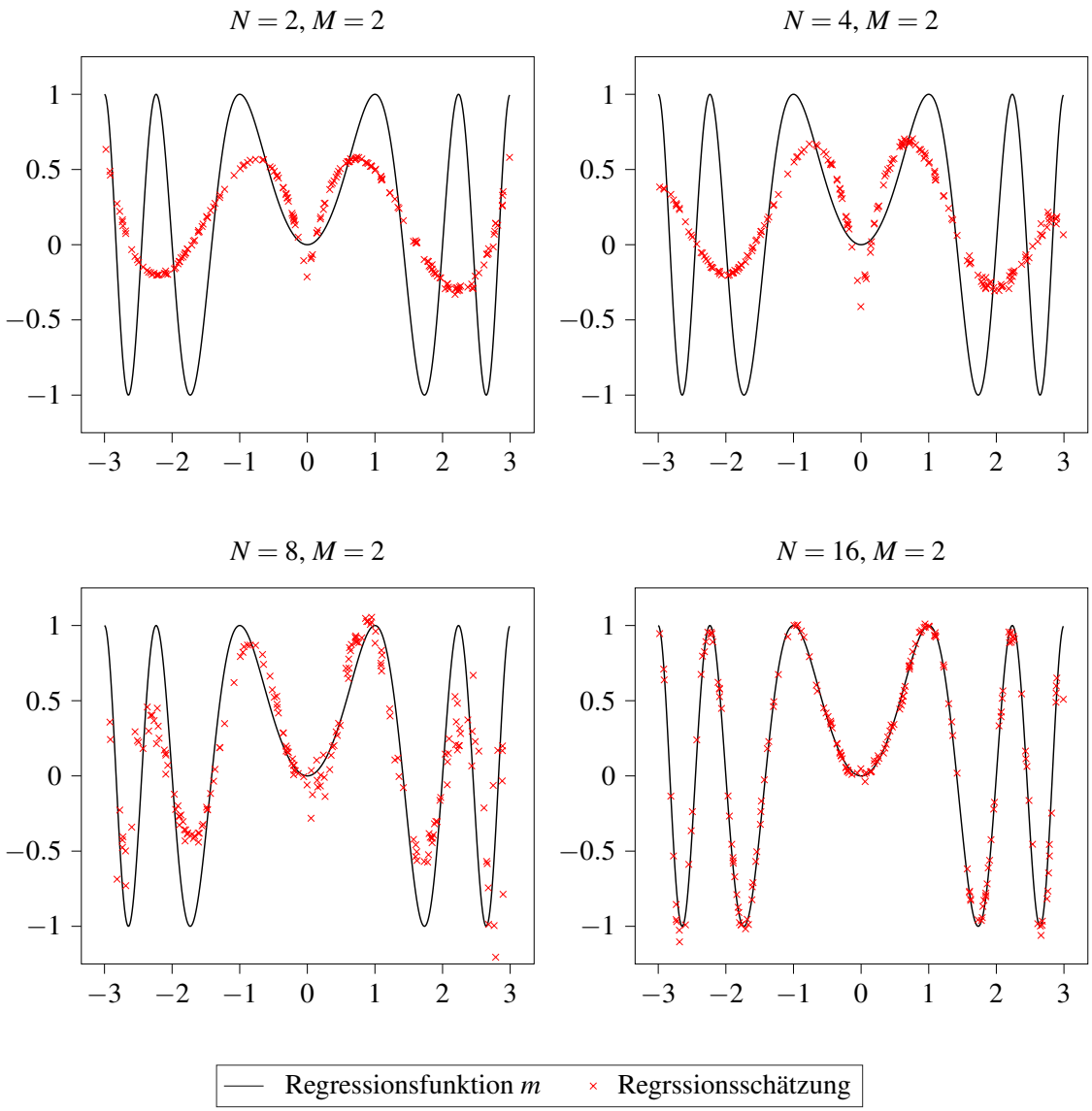

Abbildung 4.1: Approximation der Regressionsfunktion $m(x) = \sin\left(\frac{\pi}{2} \cdot x^2\right)$ durch unseren Neuronale-Netze-Regressionsschätzer mit Parametern $d = 1$, $q = 2$, $R = 10^6$, $a = 3$, $M = 2$ und $N \in \{2, 4, 8, 16\}$.

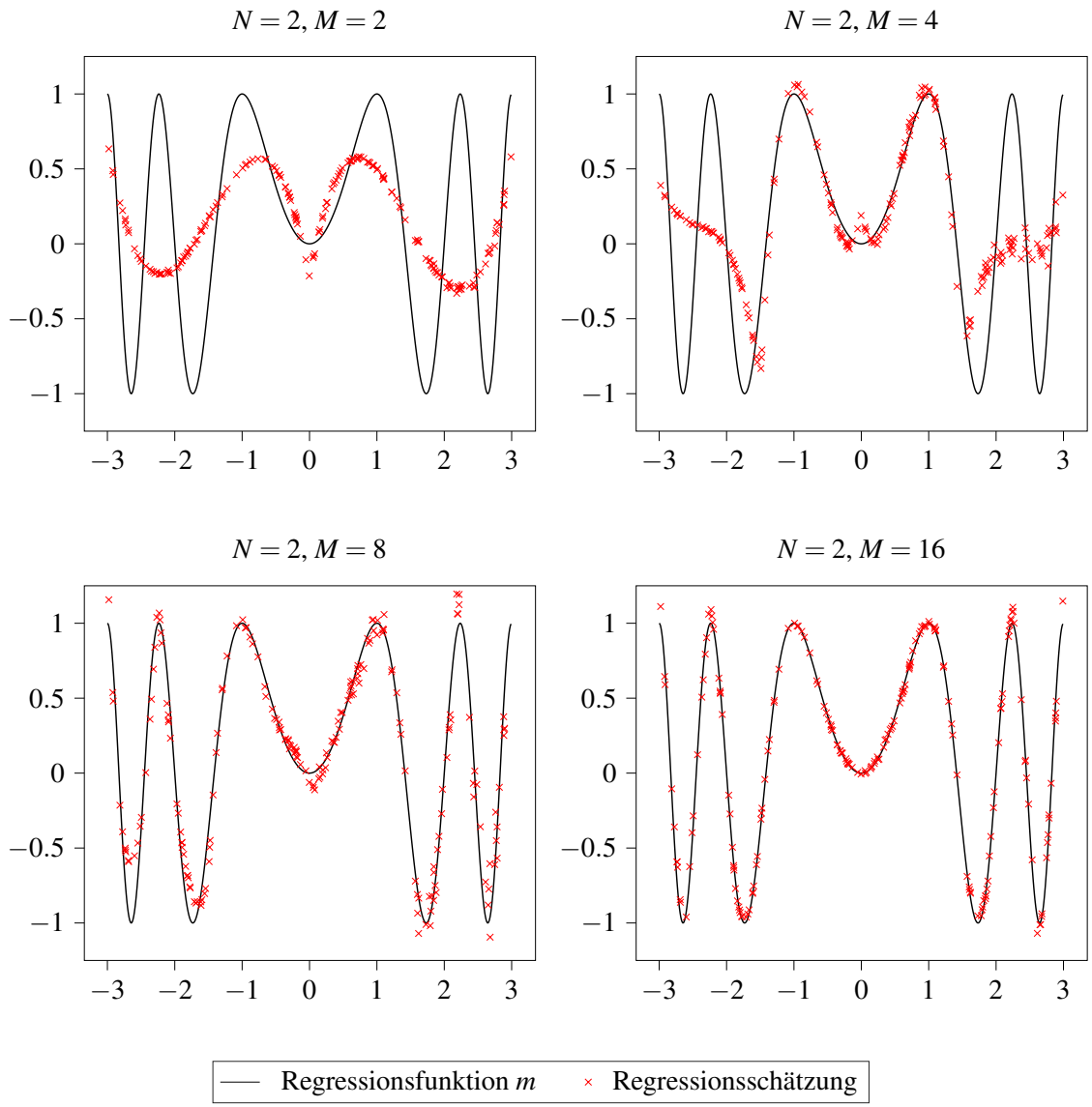

Abbildung 4.2: Approximation der Regressionsfunktion $m(x) = \sin\left(\frac{\pi}{2} \cdot x^2\right)$ durch unseren Neuronale-Netze-Regressionsschätzer mit Parametern $d = 1$, $q = 2$, $R = 10^6$, $a = 3$, $N = 2$ und $M \in \{2, 4, 8, 16\}$.

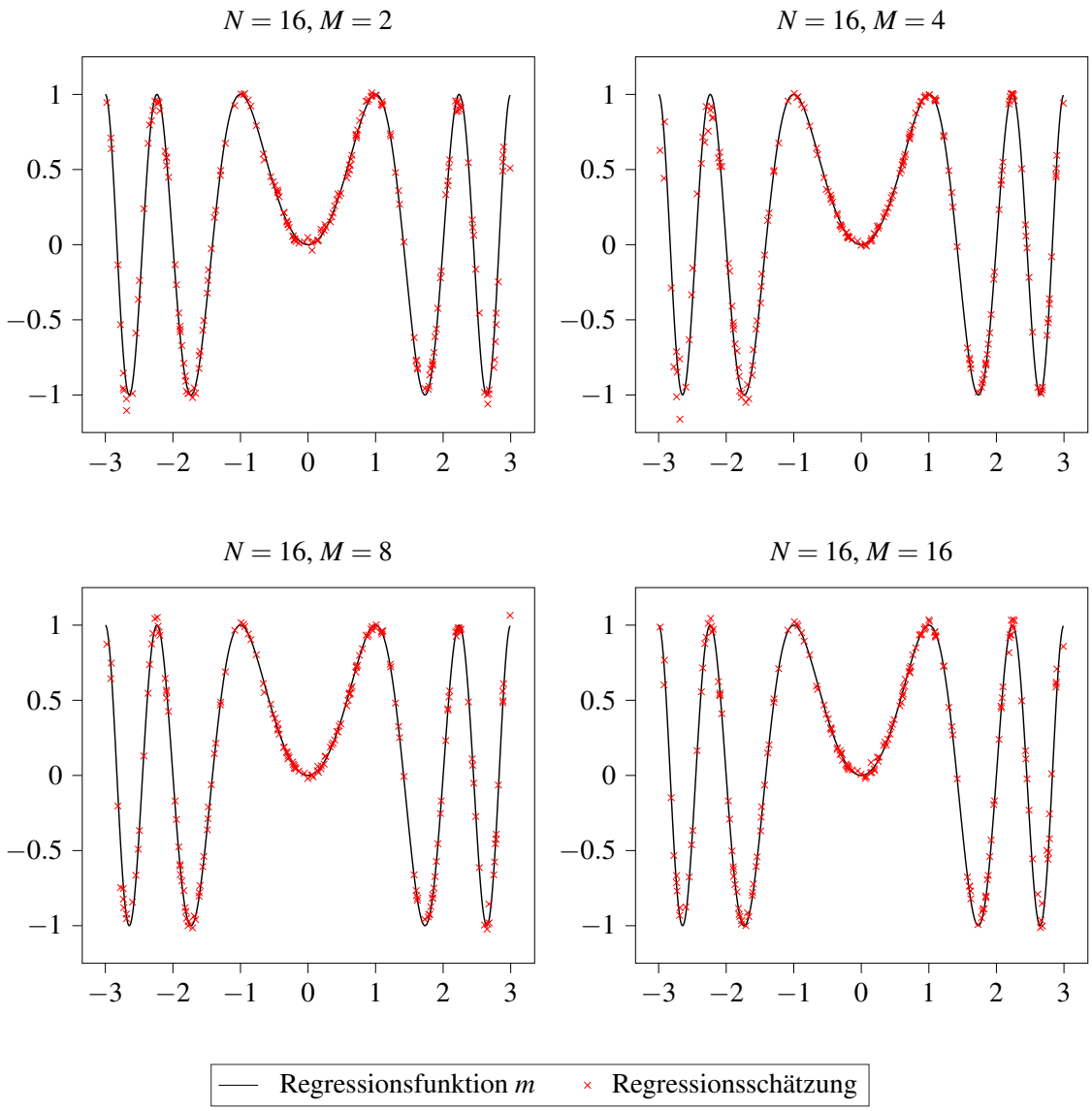

Abbildung 4.3: Approximation der Regressionsfunktion $m(x) = \sin\left(\frac{\pi}{2} \cdot x^2\right)$ durch unseren Neuronale-Netze-Regressionsschätzer mit Parametern $d = 1$, $q = 2$, $R = 10^6$, $a = 3$, $N = 16$ und $M \in \{2, 4, 8, 16\}$.

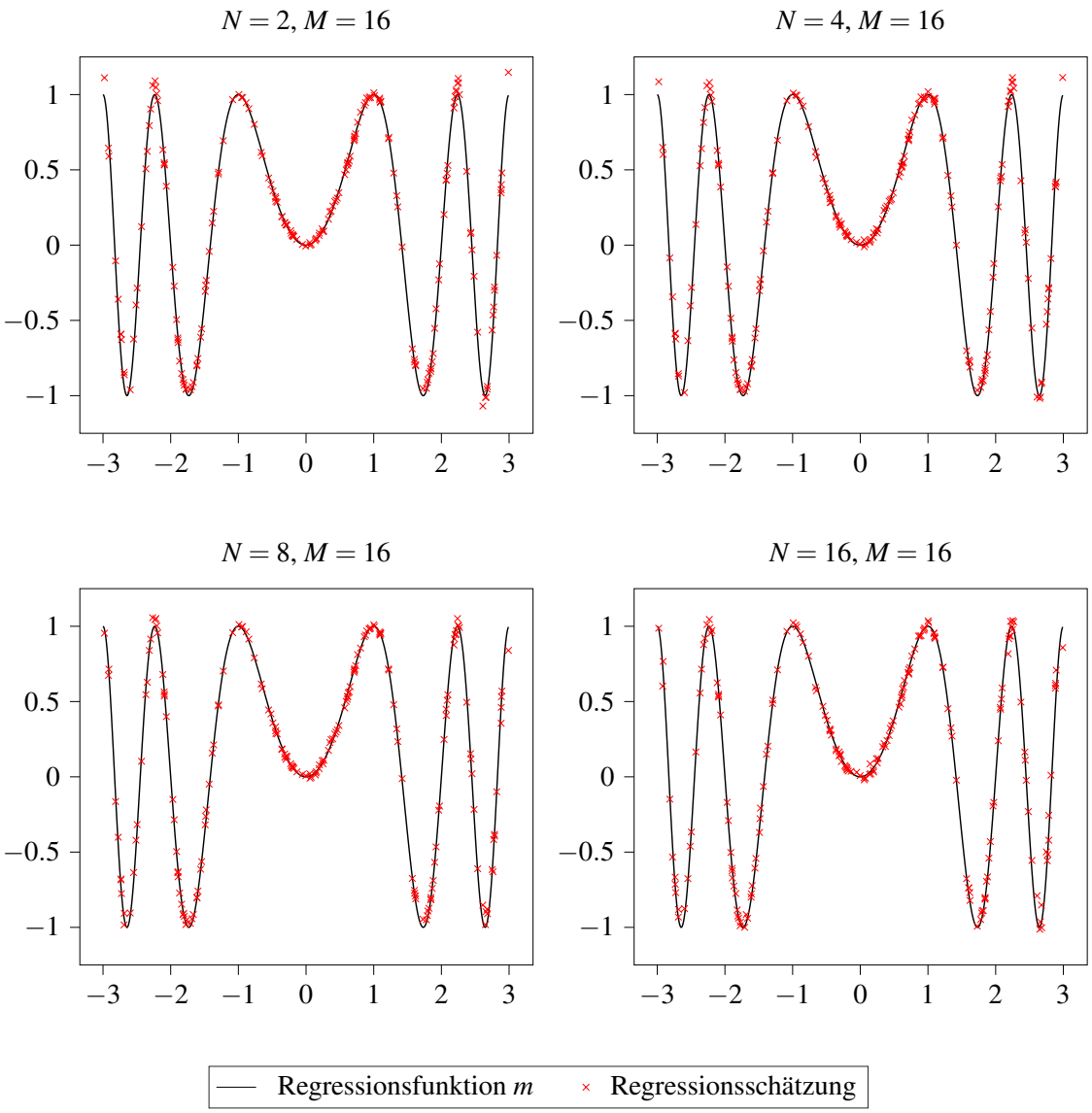

Abbildung 4.4: Approximation der Regressionsfunktion $m(x) = \sin\left(\frac{\pi}{2} \cdot x^2\right)$ durch unseren Neuronale-Netze-Regressionsschätzer mit Parametern $d = 1$, $q = 2$, $R = 10^6$, $a = 3$, $M = 16$ und $N \in \{2, 4, 8, 16\}$.

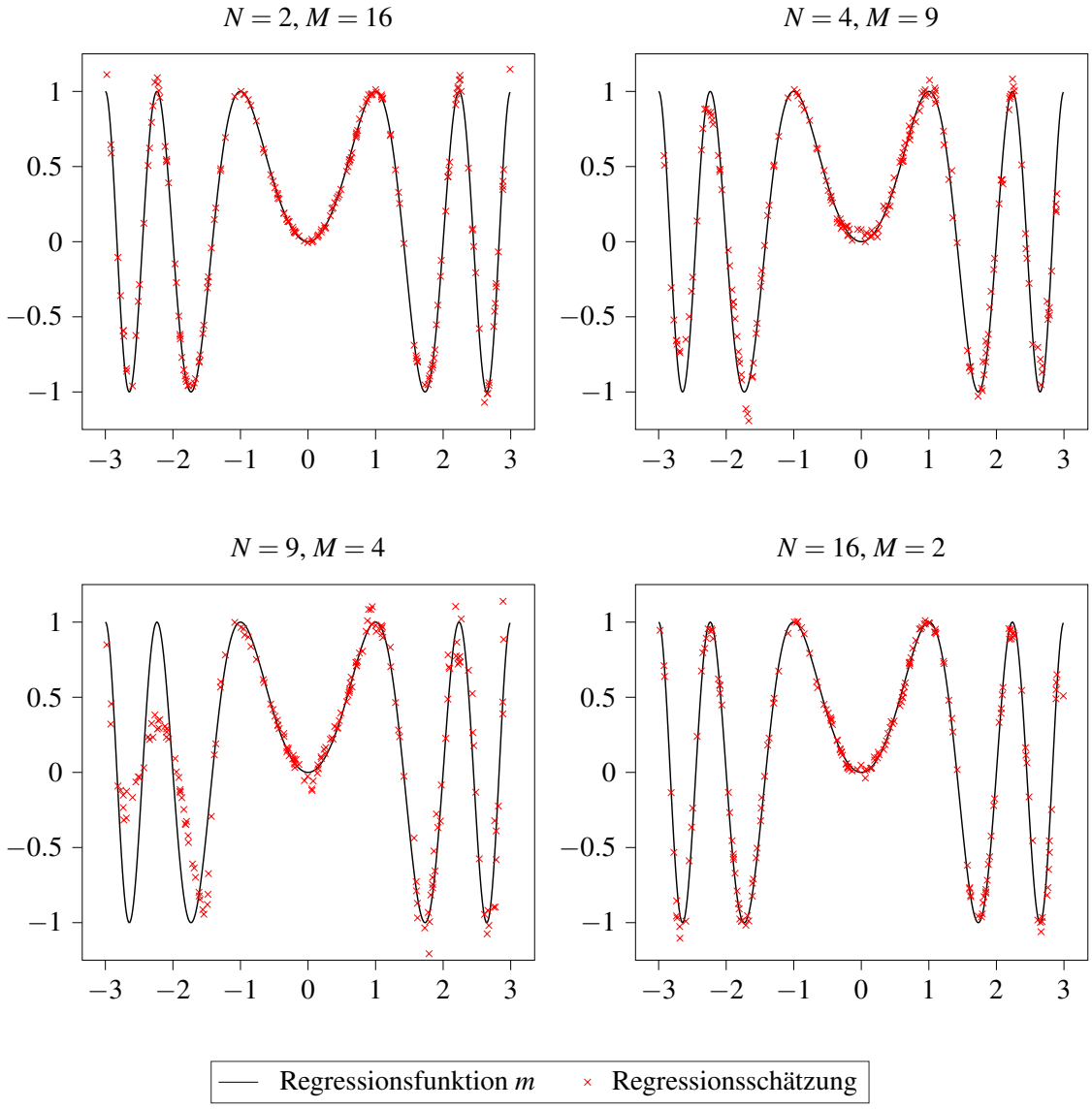

Abbildung 4.5: Approximation der Regressionsfunktion $m(x) = \sin\left(\frac{\pi}{2} \cdot x^2\right)$ durch unseren Neuronale-Netze-Regressionsschätzer mit Parametern $d = 1$, $q = 2$, $R = 10^6$, $a = 3$, $M \in \{2,4,8,9,16\}$ und $N \in \{2,4,8,9,16\}$, wobei wir immer Kombinationen von M und N betrachten, sodass $(M+1) \cdot (N+1) \approx 51$ gilt.

4.2 Vergleich des empirischen L_2-Fehlers

In diesem Abschnitt führen wir nun einen Vergleich des empirischen L_2-Fehlers zwischen dem in Abschnitt 4.1 eingeführten Regressionsschätzer und zwei Standardschätzern durch, die im Verlauf dieses Abschnitts vorgestellt werden. Die simulierten Daten, welche wir verwenden werden, sehen wie folgt aus: Wir wählen X gleichverteilt auf $[-2,2]^d$, wobei $d \in \{1,2\}$ die Dimension des Inputs ist. Mit Gleichung (4.1) können wir Y darstellen, wobei $m = m_d \colon [-2,2]^d \to \mathbb{R}$ den Regressionsfunktionen

$$m_1(x) = \sin\left(0.2 \cdot x^2\right) + \exp(0.5 \cdot x) + x^3$$

und

$$m_2(x_0, x_1) = \sin\left(\sqrt[2]{x_0^2 + x_1^2}\right)$$

entspricht. Den Skalierungsfaktor $\lambda = \lambda_d > 0$ wählen wir als Interquartilsabstand einer Stichprobe von $m_d(X)$. Für den Rauschfaktor σ gilt $\sigma \in \{0.05, 0.1\}$.
Als Nächstes kommen wir zu den Schätzern, die wir in diesem Abschnitt vergleichen möchten. Für die Wahl der Parameter der jeweiligen Schätzer haben wir uns an [BKK19] orientiert. Der Schätzer *fc_neural_1_estimate* ($m_{n,2}$) ist ein neuronales Netz mit Architektur $(1, \mathbf{k})$, wobei die Anzahl an Neuronen $\mathbf{k}$ aus der Menge $\mathscr{A} = \{5, 10, 25, 50, 75\}$ stammt. Die Implementation dieses Schätzers erfolgte mithilfe der *Keras* Bibliothek [C$^+$15]. Keras ist eine *High-Level* Anwendungs-Programmierschnittstelle für neuronale Netze in Python. Wir haben uns für Keras entschieden, da diese Bibliothek einfaches und schnelles Erstellen von neuronalen Netzen durch Benutzerfreundlichkeit, Modularität und Erweiterbarkeit ermöglicht. Für das neuronale Netz haben wir die *ReLU* Aktivierungsfunktion $f(x) = \max\{0, x\}$ verwendet. Die Anzahl der Neuronen aus der Menge $\mathscr{A}$ in der verborgenen Schicht ist so gewählt, dass diese zu einem minimalen empirischen L_2-Fehler des Schätzers führt.
Unser nächster Schätzer *nearest_neighbor_estimate* ($m_{n,3}$) ist ein Nächste-Nachbar-Schätzer [FKL09, Kapitel 7.1], bei dem die Anzahl an nächsten Nachbarn so aus der Menge $\{2, 3, \ldots, 9\}$ ausgewählt wird, dass dieser zu einem minimalen empirischen L_2-Fehler führt. Diesen Schätzer haben wir mithilfe der *Scikit-learn* Bibliothek implementiert [PVG$^+$11]. Scikit-learn ist eine Bibliothek für maschinelles Lernen in Python, die viele effiziente Werkzeuge für maschinelles Lernen, statistische Modellierung einschließlich Klassifikation und Regression enthält.
Um die Qualität der Schätzung mittels Kennzahlen zu quantifizieren und um diese mit anderen Schätzern zu vergleichen, betrachten wir in Tabelle 4.1 und Tabelle 4.2 den Interquartilsabstand und den Median des skalierten empirischen L_2-Fehlers $\varepsilon_{L_2}(m_{n,i})$ der einzelnen Schätzer $m_{n,i}$ von einer Stichprobe von Schätzungen.

Für die Schätzung von m_1 setzen wir: $d = 1, N = 3, q = 2, R = 10^6, a = 2, M = 2$, da diese Wahl der Parameter bereits sehr gute und schnelle Schätzungen liefert. Für die Schätzung von m_2 erhalten wir bereits mit: $d = 2, N = 2, q = 2, R = 10^6, a = 2, M = 2$ sehr gute Schätzungen.
Wir betrachten ein skaliertes Fehlermaß, da wir die zu schätzenden Regressionsfunktionen m_d kennen und der Fehler stark von der Komplexität der zu schätzenden Funktion abhängt. Dieses skalierte Fehlermaß ist so zu verstehen, dass wir den empirischen L_2-Fehler in Verhältnis zum Median des empirischen L_2-Fehlers $\bar{\varepsilon}_{L_2}$ des *konstanten Schätzers* setzen. Dieser konstante Schätzer approximiert die Regressionsfunktion mit dem arithmetischen Mittel der Funktionswerte auf dem Trainingsdatensatz. Die Skalierung führt dazu, dass ein großer Fehler eines Regressionsschätzers im Falle, dass der Fehler des konstanten Schätzers klein ist, auf eine noch schlechtere Leistung hindeutet.
Unser Vorgehen zum Vergleich der drei hier betrachteten Regressionsschätzer gestaltet sich wie folgt: Da die resultierenden skalierten Fehler noch von der Stichprobe von (X,Y) abhängen und um diese Werte besser vergleichen zu können, führen wir die Fehlerberechnung jeweils 50-mal durch und geben dann den Median und den Interquartilsabstand für die Schätzung der betrachteten Regressionsschätzer aus. Um diesen skalierten empirischen L_2-Fehler in jeder der 50 Iterationen zu bestimmen, gehen wir folgt vor: Wir erzeugen ein Training-Sample X_{train} aus Realisierungen von X der Größe 800 und ein Testing-Sample X_{test} der Größe $n = 200$. Auf dem Training-Sample werden nun auch die Werte Y_{train} als Realisierung der Zufallsvariable Y über Gleichung (4.1) bestimmt. Jedem einzelnen dieser Schätzer werden nun die Training-Samples X_{train} und Y_{train} zum Lernen bzw. Festlegen der Parameter gegeben. Wir bestimmen als Erstes den L_2-Fehler der einzelnen Schätzer $m_{n,i}$ mit $i = 1,2,3$ approximativ durch den empirischen L_2-Fehler $\varepsilon_{L_2}(m_{n,i})$ auf der unabhängigen Stichprobe X_{test}. Mit dem konstanten Schätzer bestimmen wir nun 25-mal den empirischen L_2-Fehler auf einer unabhängigen Stichprobe von Realisierungen von X der Größe 200. Von dieser Stichprobe von Fehlern nehmen wir nun den Median und erhalten so das skalierte Fehlermaß $\varepsilon_{L_2}(m_{n,i})/\bar{\varepsilon}_{L_2}$.
Wie wir in Tabelle 4.1 und Tabelle 4.2 anhand des Medians und des Interquartilsabstands des skalierten L_2-Fehlers sehen können, übertrifft unserer Neuronale-Netze-Regressionsschätzer in allen Fällen die Leistung der anderen Schätzer. Diese Schlussfolgerung steht im Einklang mit den Resultaten für den zweidimensionalen Fall aus [BKK19, Tabel 1] .

	m_1			
σ	5%		10%	
$\bar{\varepsilon}_{L_2,N}$	13.4482362		13.3925910	
Lageparam. (Streuungsmaß)	Median	(IQA)	Median	(IQA)
new_neural_network_estimate	**2.482e-05**	(**1.612e-05**)	**6.908e-05**	(**3.936e-05**)
fc_neural_1_estimate	4.384e-04	(2.14e-03)	7.261e-04	(4.57e-03)
nearest_neighbor_estimate	2.9527e-04	(9.312e-05)	9.0864e-04	(2.895e-04)

Tabelle 4.1: Median und IQA von 50 skalierten empirischen L_2-Fehlern für Schätzungen von m_1.

	m_2			
σ	5%		10%	
$\bar{\varepsilon}_{L_2,N}$	0.0324		0.0311	
Lageparam. (Streuungsmaß)	Median	(IQA)	Median	(IQA)
new_neural_network_estimate	**0.003961**	(**0.000932**)	**0.00431**	(**0.000973**)
fc_neural_1_estimate	0.0257	(0.3803)	0.0559	(0.52033)
nearest_neighbor_estimate	0.01616	(0.005906)	0.01763	(0.007081)

Tabelle 4.2: Median und IQA von 50 skalierten empirischen L_2-Fehlern für Schätzungen von m_2.

Appendix

Der Programmcode ist wie folgt aufgebaut:

- `main.py` ist das Hauptprogramm, welches alle Schätzer aufruft und die Ouputs generiert.
- `parametersimulation_d1.py` führt eine Parameterstudie für unseren Neuronale-Netze-Regressionschätzer durch.
- `data_gen.py` generiert die Daten, die wir für unsere Simulation benötigen.
- `help_neural_networks.py` fasst alle Hilfsfunktionen zusammen.
- `new_neural_network.py` enthält unseren Neuronale-Netze-Regressionsschätzer.
- `fc_neural_network.py` enthält das neuronale Netz mit einer verborgenen Schicht.
- `nearest_neighbor.py` enthält einen Nächste-Nachbar-Schätzer.
- `constant.py` enthält den konstanten Schätzer.

Listing 4.1: `main.py`

```
"""
Main Datei die die Simulation und damit den Vergleich der implementierten Schaetzer
durchfuehrt.
"""
import numpy as np
import pandas as pd
from scipy.stats import iqr
from data_gen import gen_data_Y
from constant import constant_estimate
from new_neural_network import new_neural_network_estimate
from nearest_neighbor import nearest_neighbor_estimate
from fc_neural_network import fc_neural_1_estimate

n = 1000
n_train = int(n * 0.8)
n_test = int(n * 0.2)

```

```
'''
ein Vergleich des emp. L2 Fehler gemacht fuer d = 1
'''
# Wahl der Parameter fuer unseren neuen Neuronale-Netze-Regressionschaetzer

N = 3
q = 2
R = 10 ** 6
a = 2
M = 2
d = 1

spreads = [0.05, 0.1]

scaled_error = np.empty((50, 3,))
scaled_error[:] = np.nan

e_L2_avg = np.zeros(25)
e_L2_avg[:] = np.nan

for sigma in spreads:
    for i in range(0,np.size(scaled_error,0),1):

        X_train = np.random.uniform(low=-a,high=a,size=(int(n_train),d))
        m_X_train, Y_train = gen_data_Y(X_train,sigma)

        X_test = np.random.uniform(low=-a,high=a,size=(int(n_test),d))

        Y_pred_new_nn = new_neural_network_estimate(X_train, Y_train, X_test, N, q, R, d
            , M, a,)
        Y_pred_fc_nn_1 = fc_neural_1_estimate(X_train, Y_train, X_test)
        Y_pred_nearest_neighbor = nearest_neighbor_estimate(X_train, Y_train, X_test)

        m_X_test, not_needed = gen_data_Y(X_test,sigma)

        e_L2_new_nn = np.mean(abs(Y_pred_new_nn - m_X_test) ** 2)
        e_L2_fc_nn_1 = np.mean(abs(Y_pred_fc_nn_1 - m_X_test) ** 2)
        e_L2_nearest_neighbor = np.mean(abs(Y_pred_nearest_neighbor - m_X_test) ** 2)

        for j in range(0,np.size(e_L2_avg),1):

            X = np.random.uniform(low=-2,high=2,size=(n_test,d))
            m_X, Y = gen_data_Y(X,sigma)
            Y_pred_constant = constant_estimate(Y)

            e_L2_avg[j] = np.mean(abs(Y_pred_constant - m_X) ** 2)

        scaled_error[i,0] = e_L2_new_nn / np.median(e_L2_avg)
        scaled_error[i,1] = e_L2_fc_nn_1 / np.median(e_L2_avg)
        scaled_error[i,2] = e_L2_nearest_neighbor / np.median(e_L2_avg)

    iqr_new_nn = iqr(scaled_error[:,0])
    iqr_fc_nn_1 = iqr(scaled_error[:,1])
    iqr_nearest_neighbor = iqr(scaled_error[:,2])

```

```
72      median_new_nn = np.median(scaled_error[:,0])
73      median_fc_nn_1 = np.median(scaled_error[:,1])
74      median_nearest_neighbor = np.median(scaled_error[:,2])
75
76      rows = ["noise","e_L2_avg","approach","new_nn", "fc_nn_1", "nearest_neighbor"]
77
78      if sigma == 0.05:
79          series_noise_1 = pd.Series([repr(sigma)+'%',np.median(e_L2_avg),"(Median, IQR)"
                ,(median_new_nn, iqr_new_nn), (median_fc_nn_1, iqr_fc_nn_1), (median_nearest
                _neighbor, iqr_nearest_neighbor)], index=rows)
80          series_noise_1.name = ""
81          print("Der empirische L2 Fehler fuer d = 1 und sigma = 0.05 ist berechnet worden
                !")
82
83      else:
84          series_noise_2 = pd.Series([repr(sigma),np.median(e_L2_avg),"(Median, IQR)",(
                median_new_nn, iqr_new_nn), (median_fc_nn_1, iqr_fc_nn_1), (median_nearest_
                neighbor, iqr_nearest_neighbor)], index=rows)
85          series_noise_2.name = ""
86          print("Der empirische L2 Fehler fuer d = 1 und sigma = 0.1 ist berechnet worden!
                ")
87
88
89  error_df = pd.concat([series_noise_1, series_noise_2], axis=1)
90  error_df.to_csv('out_d_1.csv',index = True)
91
92  '''
93  ein Vergleich des emp. L2 Fehler gemacht fuer d = 2
94  '''
95  # Wahl der Parameter fuer unseren neuen Neuronale-Netze-Regressionschaetzer
96
97  N = 2
98  q = 2
99  R = 10 ** 6
100 a = 2
101 M = 2
102 d = 2
103
104 spreads = [0.05,0.1]
105
106 scaled_error = np.empty((50, 3,))
107 scaled_error[:] = np.nan
108
109 e_L2_avg = np.zeros(25)
110 e_L2_avg[:] = np.nan
111
112 for sigma in spreads:
113     for i in range(0,np.size(scaled_error,0),1):
114
115         X_train = np.random.uniform(low=-a,high=a,size=(int(n_train),d))
116         m_X_train, Y_train = gen_data_Y(X_train,sigma)
117
118         X_test = np.random.uniform(low=-a,high=a,size=(int(n_test),d))
119
120         Y_pred_new_nn = new_neural_network_estimate(X_train, Y_train, X_test, N, q, R, d
```

```
            , M, a,)
        Y_pred_fc_nn_1 = fc_neural_1_estimate(X_train, Y_train, X_test)
        Y_pred_nearest_neighbor = nearest_neighbor_estimate(X_train, Y_train, X_test)

        m_X_test, not_needed = gen_data_Y(X_test,sigma)

        e_L2_new_nn = np.mean(abs(Y_pred_new_nn - m_X_test) ** 2)
        e_L2_fc_nn_1 = np.mean(abs(Y_pred_fc_nn_1 - m_X_test) ** 2)
        e_L2_nearest_neighbor = np.mean(abs(Y_pred_nearest_neighbor - m_X_test) ** 2)

        for j in range(0,np.size(e_L2_avg),1):

            X = np.random.uniform(low=-a,high=a,size=(n_test,d))
            m_X, Y = gen_data_Y(X,sigma)
            Y_pred_constant = constant_estimate(Y)

            e_L2_avg[j] = np.mean(abs(Y_pred_constant - m_X) ** 2)

        scaled_error[i,0] = e_L2_new_nn / np.median(e_L2_avg)
        scaled_error[i,1] = e_L2_fc_nn_1 / np.median(e_L2_avg)
        scaled_error[i,2] = e_L2_nearest_neighbor / np.median(e_L2_avg)

    iqr_new_nn = iqr(scaled_error[:,0])
    iqr_fc_nn_1 = iqr(scaled_error[:,1])
    iqr_nearest_neighbor = iqr(scaled_error[:,2])

    median_new_nn = np.median(scaled_error[:,0])
    median_fc_nn_1 = np.median(scaled_error[:,1])
    median_nearest_neighbor = np.median(scaled_error[:,2])

    rows = ["noise","e_L2_avg","approach","new_nn", "fc_nn_1", "nearest_neighbor"]

    if sigma == 0.05:
        series_noise_1 = pd.Series([repr(sigma),np.median(e_L2_avg),"(Median, IQR)",(
            median_new_nn, iqr_new_nn), (median_fc_nn_1, iqr_fc_nn_1), (median_nearest_
            neighbor, iqr_nearest_neighbor)], index=rows)
        series_noise_1.name = ""
        print("Der empirische L2 Fehler fuer d = 2 und sigma = 0.05 ist berechnet worden
            !")

    else:
        series_noise_2 = pd.Series([repr(sigma)+'%',np.median(e_L2_avg),"(Median, IQR)"
            ,(median_new_nn, iqr_new_nn), (median_fc_nn_1, iqr_fc_nn_1), (median_nearest
            _neighbor, iqr_nearest_neighbor)], index=rows)
        series_noise_2.name = ""
        print("Der empirische L2 Fehler fuer d = 2 und sigma = 0.1 ist berechnet worden!
            ")

error_df = pd.concat([series_noise_1, series_noise_2], axis=1)
error_df.to_csv('out_d_2.csv',index = True)
```

Listing 4.2: `constant.py`

```
"""
Parameterstudie für den eindimensional Fall
"""
```

```
4  import numpy as np
5  import matplotlib . pyplot as plt
6  from scipy.stats import iqr
7  import tikzplotlib
8  from new_neural_network import new_neural_network_estimate
9
10 # Regressionsfunktionen
11 #
12 # x: Ein Vektor x der Dimension d
13 # d: Dimension des Vektors x
14
15 def m (x,d):
16
17     sin = np.sin
18     pi = np.pi
19
20     return sin(pi/2 * x ** 2)
21
22 # Generiert den Vektor Y_1,...,Y_n fuer den Datensatz (X_1,Y_1),...,(X_n,Y_n)
23 #
24 # X: Inputdaten der Form (X_1,...,X_n), wobei X_i \in \R^d fuer i = 1,...,n
25 # sigma: Streuungsfaktor \in \{0.05,0.1\}
26
27 def gen_data_Y (X, sigma):
28
29     n = np.size(X, 0)
30     d = np.size(X, 1)
31
32     m_X = np.zeros((n,1,))
33     m_X[:] = np.nan
34
35     S = np.random.standard_normal(size=(n,1))
36     for t in range(0,n):
37         m_X[t] = m(X[t], d)
38
39     Y = m_X + sigma * iqr(m_X) * S
40     return (m_X, Y)
41
42 n = 1000
43 n_train = int(n * 0.8)
44 n_test = int(n * 0.2)
45
46 # Parametersets
47 # Vergleich 1
48 #N, q, R, M = 2, 2, 10 ** 6, 2
49 #N, q, R, M = 4, 2, 10 ** 6, 2
50 #N, q, R, M = 8, 2, 10 ** 6, 2
51 #N, q, R, M = 16, 2, 10 ** 6, 2
52
53 # Vergleich 2
54 #N, q, R, M = 2, 2, 10 ** 6, 2
55 #N, q, R, M = 2, 2, 10 ** 6, 4
56 #N, q, R, M = 2, 2, 10 ** 6, 8
57 #N, q, R, M = 2, 2, 10 ** 6, 16
58
```

```
# Vergleich 3
#N, q, R, M = 16, 2, 10 ** 6, 2
#N, q, R, M = 16, 2, 10 ** 6, 4
#N, q, R, M = 16, 2, 10 ** 6, 8
#N, q, R, M = 16, 2, 10 ** 6, 16

# Vergleich 4
#N, q, R, M = 2, 2, 10 ** 6, 16
#N, q, R, M = 4, 2, 10 ** 6, 16
#N, q, R, M = 8, 2, 10 ** 6, 16
#N, q, R, M = 16, 2, 10 ** 6, 16

# Vergleich 5
#N, q, R, M = 4, 2, 10 ** 6, 9
N, q, R, M = 9, 2, 10 ** 6, 4

a = 3
d = 1
sigma = 0.05

np.random.seed(1)
X_train = np.random.uniform(low=-a,high=a,size=(int(n_train),d))
m_X_train, Y_train = gen_data_Y(X_train,sigma)

X_test = np.random.uniform(low=-a,high=a,size=(int(n_test),d))

Y_pred_new_nn = new_neural_network_estimate(X_train, Y_train, X_test, N, q, R, d, M, a,)
m_X_test, dummy = gen_data_Y(X_test,sigma)


gridpoints = np.arange(-a,a,0.01)
plt.plot(gridpoints,m(gridpoints,d),color='black')
plt.scatter(X_test, Y_pred_new_nn, color = 'red', alpha=1, marker= "x")
plt.title('N = '+str(N)+', '+'M = '+str(M))
tikzplotlib.save("mytikz_N"+str(N)+"_M"+str(M)+".tex")
```

Listing 4.3: `data_gen.py`

```
"""
Generieren der Daten die wir fuer einen Vergleich von Regressionsschaetzern benoetigen
"""
# Wir waehlen x gleichverteilt auf [-2,2]^d, wobei d die Dimension des Inputs ist
# n is die Groesse der Stichprobe

import numpy as np
from scipy.stats import iqr

# Regressionsfunktionen
#
# x: Ein Vektor x der Dimension d
# d: Dimension des Vektors x

def m_d (x, d):

    sin = np.sin
    exp = np.exp
```

```
19
20      if d == 1:
21          return sin(0.2 * x[0] ** 2) + exp(0.5 * x[0]) + x[0] ** 3
22
23      elif d == 2:
24          return np.sin(np.sqrt(x[0] ** 2 + x[1] ** 2))
25
26      else:
27          print("Your data has the wrong dimension!")
28
29  # Generiert den Vektor Y_1,...,Y_n fuer den Datensatz (X_1,Y_1),...,(X_n,Y_n)
30  #
31  # X: Inputdaten der Form (X_1,...,X_n), wobei X_i \in \R^d fuer i = 1,...,n
32  # sigma: Streuungsfaktor \in \{0.05,0.1\}
33
34  def gen_data_Y (X, sigma):
35
36      n = np.size(X, 0)
37      d = np.size(X, 1)
38
39      m_X = np.zeros((n,1,))
40      m_X[:] = np.nan
41
42      S = np.random.standard_normal(size=(n,1))
43      for t in range(0,n):
44          m_X[t] = m_d(X[t], d)
45
46      Y = m_X + sigma * iqr(m_X) * S
47      return (m_X, Y)
```

Listing 4.4: `help_neural_networks.py`

```
1   """
2   Implementation von neuronalen Netzen welche wir fuer die Konstruktion unseres
3   Neuronale-Netze-Regressionschaetzers benoetigen
4   """
5   import numpy as np
6
7   # Sigmoidfunktion
8   #
9   # x: x \in \R
10
11  def sigmoid (x):
12
13      return 1 / (1 + np.exp(-x))
14
15  # Neuronales Netz welches die Funktion f(x) = x approximiert
16  #
17  # x: reelle Zahl
18  # R: reelle Zahl >= 1
19
20  def f_id (x, R):
21
22      return 4 * R * sigmoid(x / R) - 2 * R
23
24  # Neuronales Netz welches die Funktion f(x, y) = x * y approximiert
```

```
25 #
26 # x: reelle Zahl
27 # y: reelle Zahl
28 # R: reelle Zahl >= 1
29
30 def f_mult (x, y, R):
31
32     return (((R ** 2) / 4) * (((1 + np.exp(-1)) ** 3) / (np.exp(-2) - np.exp(-1)))) \
33     * (sigmoid(((2 * (x + y)) / R) + 1) - 2 * sigmoid(((x + y) / R) + 1) \
34     - sigmoid(((2 * (x - y)) / R) + 1) + 2 * sigmoid(((x - y) / R) + 1))
35
36 # Neuronales Netz welches die Funktion f(x) = max(x,0) approximiert
37 #
38 # x: reelle Zahl
39 # R: reelle Zahl >= 1
40
41 def f_relu (x, R):
42
43     return f_mult(f_id(x, R), sigmoid(R * x),R)
44
45 # Neuronales Netz welches die Funktion f(x) = max(1 - (M/(2 * a)) * abs(x - y),0)
       approximiert
46 #
47 # x: reelle Zahl
48 # y: fixe reelle Zahl
49 # R: reelle Zahl >= 1
50 # M: fixe natuerliche Zahl
51 # a: fixe Zahl > 0
52
53 def f_hat (x, y, R, M, a):
54
55     return f_relu((M / (2 * a)) * (x - y) + 1, R) - 2 * f_relu((M / (2 * a)) * (x - y),
           R) + 2 * f_relu((M / (2 * a)) * (x - y) - 1, R)
```

Listing 4.5: `new_neural_network.py`

```
1  """
2  Implementation unseres Neuronale-Netze-Regressionsschaetzers
3  """
4  import scipy.special
5  import numpy as np
6  import itertools
7  from help_neural_networks import f_id, f_mult, f_hat
8  import math
9
10 # Neuronales Netz welches die Funktion f(x) = (x^(1)- x_ik^(1))^j1 * ... *
11 # (x^(d) - x_ik^(d))^jd * \prod_{j = 1}^d max((1 - (M/2a) * abs(x^(j) - x_ik^(j))),0)
12 #
13 # x: Eingabevektor fuer das neuronale Netz x \in [-a,a]^d
14 # d: Ist die Dimension des Eingabevektors d > 0
15 # j_1_d: Ist ein d-dimensionaler Vektor j_1,...,j_d \in {0,1,...,N}
16 # X_i: Ist eine  d x (M+1)^d  Matrix.
17 # N: Natuerliche Zahl >= q
18 # q:
19 # s: [log_2(N + d)]
20 # R: Zahl >= 1
```

```
# M: M \in N
# a: > 0

def f_net (x, d, j_1_d, X_i, N, q, s, R, M, a):
    #initialize f_l_k

    f_l_k = np.empty((s + 1, (2 ** s) + 1,))
    f_l_k[:] = np.nan

    # Rekursive Definition des neuronalen Netzes f_net nach Kapitel 2

    for k in range(np.sum(j_1_d) + d + 1, (2 ** s) + 1, 1):
        f_l_k[s, k] = 1

    for k in range(1, d + 1, 1):
        f_l_k[s, np.sum(j_1_d) + k] = f_hat(x[k - 1], X_i[k - 1], R, M, a)

    for l in range(1, d + 1, 1):
        k = j_1_d[range(0, l - 1, 1)].sum() + 1
        while k in range(j_1_d[range(0, l - 1, 1)].sum() + 1, j_1_d[range(0, l, 1)].sum
            () + 1, 1):
            f_l_k[s, k] = f_id(f_id(x[l - 1] - X_i[l - 1], R), R)
            k += 1

    for l in range(s - 1, -1, -1):
        for k in range((2 ** l), 0, -1):
            f_l_k[l, k] = f_mult(f_l_k[l + 1, (2 * k) - 1], f_l_k[l + 1, 2 * k], R)

    return f_l_k[0,1]

# Bestimmung der Gewichte der Ausgabeschicht durch loesen eines regularisierten
# Kleineste-Quadrate-Problems
#
# X: Eingabevektoren der Form (X_1,...,X_n) fuer das neuronale Netz aus dem Datensatz (X
    _1,Y_1),...,(X_n,Y_n)
# Y: Eingabevektoren der Form (Y_1,...,Y_n) fuer das neuronale Netz aus dem Datensatz (X
    _1,Y_1),...,(X_n,Y_n)
# N: Natuerliche Zahl >= q
# q:
# R: Zahl >= 1
# d: Ist die Dimension des Eingabevektors d > 0
# M: M \in \N
# a: >0

def output_weights(X, Y, N, q, R, d, M, a):

    s = math.ceil(math.log2(N + d))

    # Anzahl der Eingabevektoren X_1,...,X_n

    n = np.size(X, 0)

    # Eine beliebige Konstante > 0

    c_3 = 0.01
```

```
73
74      # Anzahl der Spalten der Matrix fuer das Kleinste-Quadrate-Problem
75      # In den Spalten sind die Funktionswerte von f_net eingespeichert
76
77      J = int(((1 + M) ** d) * scipy.special.binom(N + d, d))
78
79      # Fuer die Konstruktion der Matrix brauchen wir erstmal alle Inputparameter
80      # fuer f_net, da wir dort nur den Funktionswert fuer einen Vektor j_1,...,j_d
            einsetzen
81      # muessen wir erstmals alle moeglichen Vektoren dieser Art konstruieren die die
            Bedingung 0 <= j_1 + ... + j_d <= N erfuellen
82      # X_ik hat in den Zeilen die Vektoren X_i aus dem Paper
83
84      X_ik = np.transpose(np.empty((d, (1 + M) ** d,)))
85      X_ik[:] = np.nan
86
87      I_k = np.array(list(itertools.product(range(0, M + 1), repeat = d)))
88      X_ik[:] = (I_k[:] * ((2 * a) / M)) - a
89
90      all_j1_jd = np.array(list(itertools.product(range(0, N + 1), repeat = d)))
91      all_j1_jd_by_cond = all_j1_jd[all_j1_jd.sum(axis=1) <= N]
92
93      B = np.empty((n, J,))
94      B[:] = np.nan
95
96      for i in range(0, n):
97          j = 0
98          for k in range(0, ((M + 1) ** d)):
99              for z in range(0, int(scipy.special.binom(N + d, d))):
100                 B[i,j] = f_net(X[i], d, all_j1_jd_by_cond[z], X_ik[k], N, q, s, R, M, a)
101                 j += 1
102
103     weights = np.linalg.solve(np.dot(np.transpose(B),B) + (c_3) * np.identity(J), np.dot
            (np.transpose(B),Y))
104
105     return (weights, J, all_j1_jd_by_cond, X_ik)
106
107 # Bestimmung des Funktionswerts des Neuronale-Netze-Regressionsschaetzers.
108 #
109 # x: Eingabe fuer einen Vektor der Form [-a,a]^d fuer den eine Schaetzung bestimmt
        werden soll
110 # X: Eingabevektoren der Form (X_1,...,X_n) fuer das neuronale Netz aus dem Datensatz (X
        _1,Y_1),...,(X_n,Y_n)
111 # Y: Eingabevektoren der Form (Y_1,...,Y_n) fuer das neuronale Netz aus dem Datensatz (X
        _1,Y_1),...,(X_n,Y_n)
112 # N: Natuerliche Zahl >= q
113 # q:
114 # s: [log_2(N + d)]
115 # R: Zahl >= 1
116 # d: Ist die Dimension des Eingabevektors d > 0
117 # M: M \in \N
118 # a: >0
119
120 def new_neural_network_estimate(X_train, Y_train, X_test, N, q, R, d, M, a):
121
```

```
122     Y_pred = np.empty((len(X_test), 1,))
123     Y_pred[:] = np.nan
124 
125     s = math.ceil(math.log2(N + d))
126 
127     weights, J, all_j1_jd_by_cond, X_ik = output_weights(X_train, Y_train, N, q, R, d, M
            , a)
128 
129     F_net = np.empty((1, J,))
130     F_net[:] = np.nan
131 
132     for u in range (0,len(X_test),1):
133         j = 0
134         while j < J:
135             for k in range(0, ((M + 1) ** d)):
136                 for z in range(0, int(scipy.special.binom(N + d, d))):
137                     F_net[0,j] = f_net(X_test[u], d, all_j1_jd_by_cond[z], X_ik[k], N, q
                            , s, R, M, a)
138                     j += 1
139 
140         Y_pred[u] = np.sum(np.transpose(weights) * F_net)
141 
142     return Y_pred
```

Listing 4.6: `fc_neural_network.py`

```
1  """
2  Implementation eines fully connected neuronalen Netzes mit einer verborgenen Schicht.
3  """
4  import numpy as np
5  from keras.models import Sequential
6  from keras.layers import Dense
7 
8  # Fully connected neuronales Netz mit einer verborgenen Schicht welches die
9  # Anzahl der Neuronen adaptiv, durch Minimierung des emp. L2-Fehlers, aus der Menge \{5,
       10, 25, 50, 75\} waehlt.
10 #
11 # X: Eingabevektor der Form (X_1,...,X_n) fuer das neuronale Netz aus dem Datensatz (X
       _1,Y_1),...,(X_n,Y_n)
12 # Y: Eingabevektor der Form (Y_1,...,Y_n) fuer das neuronale Netz aus dem Datensatz (X
       _1,Y_1),...,(X_n,Y_n)
13 
14 def fc_neural_1_estimate (X_train,Y_train,X_test):
15 
16     Ynew = np.empty((len(X_train), len([5,10,25,50,75]),))
17     Ynew[:] = np.nan
18 
19     count = 0
20     n_neurons = [5,10,25,50,75]
21 
22     d = np.size(X_train, 1)
23 
24     for j in n_neurons:
25         model = Sequential()
26         model.add(Dense(j, input_dim=d, activation='relu'))
27         model.add(Dense(1, activation='linear'))
```

```
28          model.compile(loss='mse', optimizer='adam')
29          model.fit(X_train, Y_train, epochs=1000, verbose=0)
30
31          Ynew[:,count] = model.predict(X_train)[:,0]
32          count += 1
33
34      Diff = Ynew[:] - Y_train[:]
35      best_n_neurons = n_neurons[(1/len(X_train) *(Diff.sum(axis=0) ** 2)).argmin()]
36
37      model = Sequential()
38      model.add(Dense(best_n_neurons, input_dim=d, activation='relu'))
39      model.add(Dense(1, activation='linear'))
40      model.compile(loss='mse', optimizer='adam')
41      model.fit(X_train, Y_train, epochs=1000, verbose=1)
42
43      return model.predict(X_test)
```

Listing 4.7: `nearest_neighbor.py`

```
1  """
2  Implementation eines Naechste-Nachbarn-Schaetzer
3  """
4
5  from sklearn import neighbors
6  from sklearn.model_selection import GridSearchCV
7  import warnings
8
9  warnings.simplefilter(action='ignore', category=FutureWarning)
10
11 # Implementierung des k-Naechste-Nachbarn-Schaetzer. Dieser bestimmt auch selber bei
       einer Liste von Anzahlen an Nachbarn die betrachtet werden
12 # sollen welches die beste Wahl ist. Dieser gibt die Schaetzung fuer X_test aus.
13 #
14 # X_train: Inputvektor fuer das Training des Schaetzers
15 # Y_train: Inputvektor fuer das Training des Schaetzers
16 # X_test: Inputvektor der geschaetzt werden soll
17
18 def nearest_neighbor_estimate (X_train,Y_train,X_test):
19
20      params = {'n_neighbors':[2,3,4,5,6,7,8,9], 'weights': ['uniform', 'distance']}
21
22      knn = neighbors.KNeighborsRegressor()
23
24      knn_gridsearch_model = GridSearchCV(knn, params, cv=5)
25      knn_gridsearch_model.fit(X_train,Y_train)
26
27      return knn_gridsearch_model.predict(X_test)
```

Listing 4.8: `constant.py`

```
1  """
2  Implementation des konstanten Schaetzers.
3  """
4  import numpy as np
5  from scipy import mean
```

```
6
7  # Gibt den Mittelwert der Funktionswerte einer Funktion als Schaetzung zurueck
8  #
9  # Y: Datensatz der Form (Y_1,...) wobei Y_i \in \R fuer i = 1,...
10
11 def constant_estimate(Y):
12     m = np.zeros((len(Y),1,))
13     m[:] = mean(Y)
14      return m
```

Literaturverzeichnis

[Bar15] R. Bartsch. *Allgemeine Topologie*. De Gruyter, Berlin, 2. Auflage, 2015.

[BKK19] A. Braun, M. Kohler, und A. Krzyzak. Analysis of the rate of convergence of neural network regression estimates which are easy to implement. arXiv:1912.05436, 2019.

[Bos13] S. Bosch. *Algebra*. Springer-Lehrbuch. Springer, Berlin, 8. Auflage, 2013.

[C+15] F. Chollet et al. Keras. `https://keras.io`, 2015.

[DGL96] L. Devroye, L. Györfi, und G. Lugosi. *A Probabilistic Theory of Pattern Recognition*. Springer, New York, 1. Auflage, 1996.

[DR08] W. Dahmen und A. Reusken. *Numerik für Ingenieure und Naturwissenschaftler*. Springer, Berlin, 2. Auflage, 2008.

[DW80] L. Devroye und T.J. Wagner. Distribution-free consistency results in nonparametric discrimination and regression function estimation. *Annals of Statistics*, 8:231–239, 1980.

[FKL09] L. Fahrmeir, T. Kneib, und S. Lang. *Regression: Modelle, Methoden und Anwendungen*. Statistik und ihre Anwendungen. Springer, Berlin, 2009.

[For16] O. Forster. *Analysis 1 - Differential- und Integralrechnung einer Veränderlichen*. Springer, Berlin, 12. Auflage, 2016.

[GBC16] I. Goodfellow, Y. Bengio, und A. Courville. *Deep Learning*. Adaptive Computation and Machine Learning series. MIT Press, 2016.

[GKKW02] L. Györfi, M. Kohler, A. Krzyzak, und H. Walk. *A Distribution-Free Theory of Nonparametric Regression.* Springer series in statistics. Springer, Berlin, 1. Auflage, 2002.

[Kle13] A. Klenke. *Wahrscheinlichkeitstheorie*. Springer, Berlin, 2013.

[Kre98] R. Kress. *Numerical analysis*. Springer, Berlin, 1. Auflage, 1998.

[Kö04] K. Königsberger. *Analysis 2*. Springer, Berlin, 5. Auflage, 2004.

[PVG+11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, und E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[RN09] S. Russell und P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall Press, USA, 3. Auflage, 2009.

[TEC19] A. Toller, F. Edholm, und D. Chen. *Proofs in Competition Math: Volume 2*. Selbstverlag, 1. Auflage, 2019.

[VRDJ95] G. Van Rossum und F. L. Drake Jr. *Python tutorial*. Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.