TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Mathematik

Masterarbeit

Analyse der Konvergenzgeschwindigkeit eines einfach berechenbaren Neuronale-Netze-Regressionsschätzers

Adrian Gabel

XX.03.2020

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.

Darmstadt, 22.02.2020

Adrian Gabel

Inhaltsverzeichnis

Einleitung

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat.

Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi.

Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer. Diese Arbeit orientiert sich an [EKT+07] und [Koh10].

Kapitel 1

Grundlagen und Hilfsresultate

Der Zweck dieses Kapitels ist es, grundlegende Definitionen zu sammeln, die in den folgenden Kapiteln verwendet werden. Weiterhin werden wir Hilfsresultate darstellen und beweisen welche wir vor allem für das Resultat der Konvergenzgeschwindigkeit des einfach berechenbaren Neuronale Netze Regressionsschätzer benötigt werden.
In dieser Arbeit behandeln wir Neuronale-Netze-Regressionsschätzer im Kontext der nichtparametrischen Regression mit zufälligem Design. Im Gegensatz zur parametrischen Regression ist bei der nichtparametrischen, die Bauart der schätzenden Funktion komplett unbekannt, was von Vorteil hat dass weniger Annahmen getroffen werden müssen, man aber dadurch noch mehr Daten benötigt um eine Funktion zu schätzen.
Bei der nichtparametrischen Regressionsschätzung ist seien $(X,Y),(X_1,Y_1),(X_2,Y_2),\dots$ u.i.v $\mathbb{R}^d \times \mathbb{R}$-wertige Zufallsvariablen mit $\mathbb{E}[Y^2] < \infty$. Zudem sei $m\colon \mathbb{R}^d \to R$ definiert durch $m(x) = \mathbb{E}[Y \mid X = x]$ die zugehörige Regressionsfunktion. Ausgehend von

$$(X_1,Y_1),\dots,(X_n,Y_n)$$

soll m geschätzt werden.
Das Problem der Regressionsschätzung bei zufälligem Design lässt sich wie gefolgt erläutern. In Anwerndung ist üblicherweise die Verteilung von (X,Y) unbekannt, daher kann $m(x) = \mathbb{E}[Y \mid X = x]$ nicht berechnet werden. Oft ist es aber möglich, Werte von (X,Y) zu beobachten. Ziel ist es dann, daraus die Regressionsfunktion zu schätzen. Im Hinblick auf die Minimierung des L_2-Risikos sollte dabei der L_2-Fehler der Schätzfunktion möglichst klein sein.
Für das L_2-Risiko einer beliebigen messbaren Funktion $f\colon \mathbb{R}^d \to \mathbb{R}$ gilt:

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

d.h. der mittlere quadratische Vorhersagefehler einer Funktion ist darstellbar als Summe des L_2-Risikos der Regressionsfunktion (unvermeidbarer Fehler) und des L_2-Fehlers

der entsteht aufgrund der Verwendung von f anstelle von m bei der Vorhersage bzw. Approximation des Wertes von Y.
Formal führt das daher auf folgende Problemstellung: $(X,Y),(X_1,Y_1),(X_2,Y_2),\ldots$ seien unabhängig identisch verteilte $\mathbb{R}^d \times \mathbb{R}$ 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 ist die Datenmenge

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

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. (Referenz Györfi (2002))

1.1 Definitionen

Es ist bekannt, dass man Glattheitsvoraussetzungen an die Regressionsfunktion haben muss um nichttriviale Konvergenzresultate für nichtparametrische Regressionsschätzer herzuleiten. Dafür verwenden wir die folgende Definition.

Definition 1.1.1 ((p,C)-Glattheit)**.** Sei $p = q + s$ mit $q \in \mathbb{N}_0$ und $s \in (0,1]$ (also $p \in (0,\infty)$ und sei $C > 0$. Eine Funktion $f\colon \mathbb{R}^d \to \mathbb{R}$ heißt (p,C)-glatt, falls für alle $\alpha = (\alpha_1,\ldots,\alpha_d) \in \mathbb{N}_0^d$ 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 $x,z \in \mathbb{R}^d$ gilt

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

wobei $\|\cdot\|$ die euklidische Norm ist.

Bemerkung 1.1.2. Im Falle von $p \leq 1$ ist keine Funktion (p,C)-glatt genau dann, wenn sie Hölder-stetig ist mit Exponent p und Hölder-Konstante C.

Der Ausgangspunkt für die Definition eines neuronalen Netzes ist die Wahl einer Aktivierungsfunktion $\sigma\colon \mathbb{R} \to \mathbb{R}$. Wir haben uns in dieser Arbeit für die sogenannten „squashing functions"entschieden, welche eine monoton wachsend ist und für die $\lim_{x\to-\infty}\sigma(x) = 0$ und $\lim_{x\to\infty}\sigma(x) = 1$ gilt. Ein Beispiel für eine squashing function ist der sogenannte sigmoidal bzw. logisitsche squasher

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

Definition 1.1.3. Sei $n \in \mathbb{N}_0$. Eine Funktion $\sigma\colon \mathbb{R} \to [0,1]$ wird N-zulässig genannt, wenn monoton wachsend und Lipschitz stetig (REFERENZ) 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 $t_\sigma \in \mathbb{R}$, in welchem alle Ableitungen bis hin zur N-ten Ableitung von σ ungleich Null sind.

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

In Lemma 1.2.1 werden wir zudem zeigen, dass der logistische squasher (1.1) N-zulässig ist für beliebiges $N \in \mathbb{N}$.

1.2 Hilfsresultate

Lemma 1.2.1. *Sei $N \in \mathbb{N}$ beliebig, dann erfüllt der logistische squasher $\sigma\colon \mathbb{R} \to [0,1], \sigma(x) = \frac{1}{1+\exp(-x)}$ die Bedingungen aus Definition 1.1.*

Beweis. Sei $N \in \mathbb{N}$ beliebig. 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 bei der Ungleichung die Monotonie der Exponentialfunktion verwendet haben und die obige Ungleichung aus

$$\begin{aligned}
& \exp(s) \leq \exp(t) \\
\Leftrightarrow \quad & \exp(-s) \geq \exp(-t) \\
\Leftrightarrow \quad & 1+\exp(-s) \geq 1+\exp(-t) \\
\Leftrightarrow \quad & \frac{1}{1+\exp(-s)} \leq \frac{1}{1+\exp(-t)}
\end{aligned}$$

folgt. Zudem ist σ als Komposition $N+1$ mal stetig differenzierbarer Funktionen selber auch $N+1$ mal stetig differenzierbar. Die Ableitungen von σ haben die Form:

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

Da wir bei weiterem Ableiten die Produktregel wiederholt anwenden sind alle Ableitungen von σ, Polynome in σ. Dadurch folgt Bedingung (i) aus Definition 1.1, da *sigma* nach Voraussetzung durch 0 und 1 beschränkt ist, und die Ableitungen von σ als Produkt von beschränkten Faktoren daher auch. Da hiermit auch die erste Ableitung von σ beschränkt ist wissen wir nach Satz ... aus (REFERENZ), dass σ Lipschitz stetig ist. Nun kommen wir zum Beweis von Bedingung (ii). Polynome, die nicht das 0-Polynom sind, haben nach Satz ... (REFERENZ) auf $(0,1)$ endlich viele Nullstellen und σ bildet nach Voraussetzung in das Intervall $[0,1] \supseteq (0,1)$ ab. Da die Ableitungen von σ, als Zusammensetzung von Polynome in σ, wieder Polynome sind für die die obere Eigenschaft ebenfalls gilt, existiert ein $t_\sigma \in \mathbb{R}$ mit$\sigma(t_\sigma) \neq 0$ sodass alle Ableitungen bis zum Grad N von σ, aufgrund ihrer Struktur ungleich 0 sind. Daher ist Bedingung (ii) ebenfalls erfüllt. Betrachten wir nun ein beliebiges $x > 0$. Dann wissen wir nach dem Mittelwertsatz (REFERENZ) dass ein $z \in (0,\infty)$ existiert, sodass mit $\exp(z) > 1$, da $z > 0$ ist, gilt:

$$(\exp(x-1) \cdot (x-0) = e^z$$

und da $x \neq 0$ ist, daraus folgt, dass

$$\frac{\exp(x)-1}{x-0} > 1$$

gilt und damit dann auch durch Multiplikation mit x insbesondere

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

Daraus erhalten wir mit Umformungen da $x > 0$ und $1 + \exp(-x) > 0$ ist:

$$\begin{aligned} & x \leq \exp(x) + 1 \\ \Leftrightarrow \quad & x \cdot \exp(-x) \leq 1 + \exp(-x) \\ \Leftrightarrow \quad & \frac{\exp(-x)}{1+\exp(-x)} \leq \frac{1}{x} \\ \Leftrightarrow \quad & 1 - \frac{1}{1+\exp(-x)} \leq \frac{1}{x} \\ \Leftrightarrow \quad & |\sigma(x) - 1| \leq \frac{1}{x}. \end{aligned}$$

Wobei die letzte Ungleichung aus der Eigenschaft des Betrags kommt, da $\frac{1}{1+\exp(-x)} - 1 < 0$ ist, weil $1 + \exp(-x) > 1$, da $\exp(-x) > 0$. Dies zeigt die erste Relation aus Bedingung (iii). Die zweite Relation folgt durch die gleiche Art und Weise, da wir durch

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

wissen, dass σ punktsymmetrisch in $(0, \frac{1}{2})$ ist. Die obige Gleichheit folgt aus

$$\begin{aligned} & \frac{1}{1+\exp(x)} - \frac{1}{2} = -\frac{1}{1+\exp(-x)} + \frac{1}{2} \\ \Leftrightarrow \quad & \frac{1}{1+\exp(x)} + \frac{1}{2} - \frac{1}{1+\exp(-x)} = \frac{1}{2} \\ \Leftrightarrow \quad & \frac{1+\exp(-x)+1+\exp(x)}{(1+\exp(x)) \cdot (1+\exp(-x))} - \frac{1}{2} = \frac{1}{2} \\ \Leftrightarrow \quad & \frac{2+\exp(-x)+\exp(x)}{2+\exp(-x)+\exp(x)} - \frac{1}{2} = \frac{1}{2} \\ \Leftrightarrow \quad & 1 - \frac{1}{2} = \frac{1}{2}. \end{aligned}$$

Aus dieser Eigenschaft folgt mit

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

für $x < 0$ aus der ersten Relation, da $-x > 0$ ist:

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

Damit haben wir alle drei Bedingungen aus Definition 1.1 gezeigt und unsere Aussage bewiesen. □

Lemma 1.2.2. *Sei $\sigma : \mathbb{R} \to \mathbb{R}$ eine Funktion und R, $a > 0$.*

a) *Angenommen σ ist zwei mal stetig differenzierbar und $t_{\sigma,id} \in \mathbb{R}$ so, dass sigma$'(t_{\sigma,id}) \neq 0$ ist. Dann gilt mit*

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

für beliebige $x \in [-a,a]$:

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

b) *Angenommen σ ist drei mal stetig differenzierbar und $t_{\sigma,sq} \in \mathbb{R}$ so, dass sigma$''(t_{\sigma,sq}) \neq 0$ ist. Dann gilt mit*

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

für beliebige $x \in [-a,a]$:

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

Beweis. a) Sei $u = \frac{c}{R} + t_{\sigma,id}$, $\xi = 0$ und $x \in [-a,a]$beliebig. Wir wissen, dass f_{id} 1-mal differenzierbar ist, da nach Vorraussetzung σ 2-mal stetig differenzierbar ist, existiert nach der Restgliedformel von Lagrange (REFERENZ) ein $c \in [\xi, x]$, sodass mit Ausklammern von $\frac{R}{\sigma'(t_{\sigma,id})}$ folgt:

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

Wobei sich die letzte Ungleichung aus den Eigenschaften der Supremumsnorm ergibt und zudem aus $x \in [-a,a] \Leftrightarrow -a \leq x \leq a$ durch Quadrieren der Ungleichung folgt, dass $x^2 \leq a^2$ ist.

b) Folgt analog wie in a) durch 2-maliges Anwenden der Restgliedformel von Lagrange (REFeRENZ) auf die Funktion f die hier nun 2-mal differenzierbar ist, da σ nach Voraussetzung 3-mal stetig differenzierbar ist.

$\square$

Lemma 1.2.3. *Sei $\sigma : \mathbb{R} \to [0,1]$ nach Definition 1, 2-zulässig. Zudem sei $R > 0$ und $a > 0$ beliebig. Dann gilt für das neuronale Netz*

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

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

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

Beweis. Durch Ausmultiplizieren erhalten wir

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

und

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

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

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

wobei bei bei der letzten Ungleichung verwendet haben, dass $a > 0$ nach Lemma 1.2.2b) beliebig gewählt wurde und daher insbesondere für beliebiges $x \in [-2a, 2a]$

$$|f_{sq}(x) - x^2| \leq \frac{40 \cdot \|\sigma'''\|_\infty \cdot a^3}{3 \cdot |\sigma'(t_{\sigma,sq})|} \cdot \frac{1}{R}$$

gilt. $\square$

Lemma 1.2.4. *Sei $\sigma\colon \mathbb{R}\to[0,1]$ nach Definition 1, 2-zulässig. Sei f_{mult} das neuronale Netz aus Lemma 1.2.3 und f_{id} das neuronale Netz aus Lemma 1.2.2. Angenommen es gilt*

$$a \geq 1 \quad \textit{und} \quad R \geq \frac{\|\sigma''\|_\infty \cdot a}{2\cdot|\sigma'(t_{\sigma,id})|}.$$

Dann erfüllt das neuronale Netz

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

für alle $x\in[-a,a]$:

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

Beweis. Da σ nach Voraussetzung 2-zulässig nach Definition 1 ist, gilt für $R\geq 0$, und $x\in\mathbb{R}\setminus\{0\}$:

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

und

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

Damit folgt aus der Homogenität des Betrags

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

Nach Lemma 1.2.2 und Lemma 1.2.3 gilt:

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

und

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

Da nach Voraussetzung $a\geq 1$ ist gilt insbesondere $[0,1]\in[-2a,2a]$ und daher gilt insbesondere $\sigma(x)\in[-2a,2a]$. Zudem erhalten wir durch eine Nulladdition, das Anwenden der Dreiecksungleichung, die Verwendung von Lemma 1.2.2 und der Voraussetzung für R:

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

wobei $x \in [-a,a]$. Daraus folgt insbesondere $f_{id}(x) \in [-2a,2a]$. Mithilfe von $\max\{x,0\} = x \cdot \mathbb{1}_{[0,\infty)}(x)$, der Voraussetzung, zweier Nulladdition und dem zweifachen Anwenden der Dreiecksungleichung erhalten wir:

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

Daraus ergibt sich mithilfe der obigen Eigenschaften und $a^3 \geq 1$

$$
\begin{aligned}
&\leq \frac{160 \cdot \|\sigma'''\|_\infty \cdot a^3}{3 \cdot |\sigma''(t_\sigma)|} \cdot \frac{1}{R} + \frac{\|\sigma''\|_\infty \cdot a^3}{2 \cdot |\sigma'(t_{\sigma,id})|} \cdot \frac{1}{R} \cdot 1 + \frac{1}{R} \\
&\leq \left(\frac{160}{3} \cdot \frac{\|\sigma'''\|_\infty \cdot a^3}{|\sigma''(t_\sigma)|} + \frac{\|\sigma''\|_\infty \cdot a^3}{2 \cdot |\sigma'(t_{\sigma,id})|} + \frac{a^3}{a^3} \right) \cdot \frac{1}{R} \\
&\leq \left(\frac{160 \cdot \|\sigma'''\|_\infty \cdot a^3 + 3 \cdot \|\sigma''\|_\infty \cdot a^3 + 3 \cdot a^3}{3 \cdot \min\{2 \cdot \sigma'(t_{\sigma,id})|, |\sigma''(t_\sigma)|, 1\}} \right) \cdot \frac{1}{R} \\
&\leq \frac{166}{3} \cdot \left(\frac{\max\{\|\sigma'''\|_\infty, \|\sigma''\|_\infty, 1\}}{\min\{2 \cdot \sigma'(t_{\sigma,id})|, |\sigma''(t_\sigma)|, 1\}} \right) \cdot a^3 \cdot \frac{1}{R} \\
&\leq 56 \cdot \frac{\max\{|\sigma''\|_\infty, \|\sigma'''\|_\infty, 1\}}{\min\{2 \cdot |\sigma'(t_{\sigma,id}), |\sigma''(t_\sigma)|, 1\}} \cdot a^3 \cdot \frac{1}{R}.
\end{aligned}
$$

□

Lemma 1.2.5. *Sei $M \in \mathbb{N}$ und sei $\sigma\colon \mathbb{R} \to [0,1]$ 2-zulässig nahc Definition .... Sei $a > 0$ und*

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

sei $y \in [-a,a]$ und f_{ReLU} das neuronale Netz aus Lemma 1.2.4. Dann erfüllt das neuronale Netz

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

für alle $x \in [-a,a]$:

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

Beweis. Als erstes zeigen wir

$$(1 - \frac{M}{2a} \cdot |x|)_+ = \max\{\frac{M}{2a} \cdot x + 1, 0\} - 2 \cdot \max\{\frac{M}{2a} \cdot x, 0\} + \max\{\frac{M}{2a} \cdot x - 1, 0\}, \qquad (x \in \mathbb{R})$$

damit wir das Resultat mithilfe von Lemma 1.2.4 beweisen können. Um die obige Gleichung zu zeigen unterscheiden wir vier Fälle.

Fall 1 $(x < -\frac{M}{2a})$ In diesem Fall hat die linke Seite durch $z_+ = \max\{z,0\} \quad (z \in \mathbb{R})$ und nach der Definition des Betrags die Gestalt

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

und die rechte Seite die Form

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

da $x < 0$ und damit die letzten zwei Summanden 0 sind. Damit sind stimmt die rechte Seite mit der linken überein. $(\square)$

Fall 2 $(-\frac{M}{2a} \leq x \leq 0)$ Dieser Fall liefert aufgrund der nicht Positivität von x analog das selbe Resultat wie Fall 1. $(\square)$

Fall 3 $(0 < x \leq \frac{M}{2a})$ In diesem Fall hat die linke Seite nach der Definition des Betrags die Gestalt

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

und die rechte Seite die Form

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

und erfordert daher eine weitere Fallunterscheidung.

Fall 3.1 $(x \cdot \frac{M}{2a} \leq 1)$ In diesem Fall gilt für die linke Seite:

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

und für die rechte Seite:

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

und stimmt daher mit der linken Seite überein.

Fall 3.2 $(x \cdot \frac{M}{2a} > 1)$ In diesem Fall gilt für die linke Seite:

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

und für die rechte Seite:

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

und stimmt daher mit der linken Seite überein. Damit ist Fall 3 gezeigt. $(\square)$

Fall 4 ($\frac{M}{2a} < x$) In diesem Fall hat die linke Seite nach der Definition des Betrags die Gestalt

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

und die rechte Seite die Form

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

und erfordert daher eine weitere Fallunterscheidung.

Fall 4.1 ($\frac{M}{2a} \cdot x \leq 1$) In diesem Fall gilt für die linke Seite:

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

und für die rechte Seite:

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

und stimmt daher mit der linken Seite überein.

Fall 4.2 ($\frac{M}{2a} \cdot x > 1$) In diesem Fall gilt für die linke Seite:

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

und für die rechte Seite:

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

und stimmt daher mit der linken Seite überein. Damit ist Fall 4 gezeigt. ($\square$)

Durch diese Fallunterscheidung wurde die obige Gleichung (REFERENZ) bewiesen. Daraus folgt mit der Definition von $f_{hat,y}(x)$ und zwei mal der Dreiecksungleichung

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

wobei die letzte Ungleichung daraus folgt, dass wir auf jeden Summanden mit $1 \leq a = M + 1$ Lemma 1.2.4 angewendet haben und die Abschätzung

$$(M+1)^3 = M^3 + 3 \cdot M^2 + 3 \cdot M + 1 \leq M^3 + 3 \cdot M^3 + 3 \cdot M^3 + M^3 = 8 \cdot M^3 \quad (M \in \mathbb{N})$$

verwendet haben. $\square$

Die nächsten Lemmata benötigen wir für den Beweis unseres Hauptresultats, einer Aussage über die Konvergenzgeschwindigkeit unseres neuronale Netze Schätzers.. Diese Lemmata werden hier nur der Vollständigkeit halber und ohne Beweis aufgeführt.

Lemma 1.2.6. *Sei $M \in \mathbb{N}$ und $\sigma \colon \mathbb{R} \to [0,1]$ 2-zulässig nach Definition 1.1.3. (AUF SCHREIBWEISE VON FNET AUFPASSEN) Sei $a \geq 1$ und*

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

und sei $y \in [-a,a]^d$. Sei $N \in \mathbb{N}$ und $j_1, \dots, j_d \in \mathbb{N}_0$ so, dass $j_1 + \cdots + j_d \leq N$ gilt und wir setzen $s = \lceil \log_2(N+d) \rceil$. Sei f_{id}, f_{mult} und $f_{hat,z}$ (für $z \in \mathbb{R}$) die neuronalen Netze wir in Lemma 1.2.2, Lemma 1.2.3 und Lemma 1.2.5. Wir definieren das Netz $f_{net,j_1,\dots,j_d,y}$ durch:

$$f_{net,j_1,\dots,j_d,y}(x) = f_1^{(0)}(x).$$

wobei

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

für $k \in \{1,2,\dots,2^l\}$ und $l \in \{0,\dots,s-1\}$, und

$$f_k^{(s)}(x) = f_{id}(f_{id}(x^{(l)} - y^{(l)}))$$

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

$$f_{j_1+j_2+\cdots+j_d+k}^{(s)}(x) = f_{hat,y^{(k)}}(x^{(k)})$$

für $k = 1,\dots,d$ und

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

für $k = j_1 + j_2 + \cdots + j_d + d + 1, j_1 + j_2 + \cdots + j_d + d + 2, \dots, 2^s$.

Dann erhalten wir für $x \in [-a,a]^d$:

$$\begin{aligned}
&\left| f_{net,y}(x) - (x^{(1)} - y^{(1)})^{j_1} \cdots (x^{(d)} - y^{(d)})^{j_d} \prod_{j=1}^{d} (1 - \frac{M}{2a} \cdot |x^{(j)} - y^{(j)}|)_+ \right| \\
&\leq c_{12} \cdot 3^{3 \cdot 3^s} \cdot a^{3 \cdot 2^s} \cdot M^3 \cdot \frac{1}{R}.
\end{aligned}$$

Lemma 1.2.7. *Sei $\beta_n = c_6 \cdot \log(n)$ für eine hinreichend große Konstante $c_6 > 0$. Angenommen die Verteilung von (X,Y) erfüllt*

$$\mathbb{E}\left(\mathrm{e}^{c_4 \cdot |Y|^2}\right) < \infty$$

für eine Konstante $c_4 > 0$ und dass der Betrag der Regressionsfunktion m beschränkt ist. Sei $\mathscr{F}_n$ eine Menge von Funktionen $f\colon \mathbb{R}^d \to \mathbb{R}$ und wir nehmen an, dass der Schätzer m_n

$$m_n = T_{\beta_n} \tilde{m}_n$$

erfüllt, mit

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

und

$$\frac{1}{n} \sum_{i=1}^{n} |Y_i - \tilde{m}_n(X_i)|^2 \leq \min_{l \in \Theta_n} \left(\frac{1}{n} \sum_{i=1}^{n} |Y_i - g_{n,l}(X_i)|^2 + pen_n(g_n, l) \right)$$

mit einer nichtleeren Parametermenge Θ_n, zufällige Funktionen $g_{n,l}\colon \mathbb{R}^d \to \mathbb{R}$ und deterministischen penalty Termen $pen_n(g_{n,l}) \geq 0$, wobei die zufälligen Funktionen $g_{n,l}\colon \mathbb{R}^d \to \mathbb{R}$ nur von den Zufallsvariablen

$$\mathbf{b}_1^{(1)}, \ldots, \mathbf{b}_r^{(1)}, \ldots, \mathbf{b}_1^{(I_n)}, \ldots, \mathbf{b}_r^{(I_n)},$$

abhängen, die unabhängig von $(X_1,Y_1),(X_2,Y_2),\ldots$ sind. Dann erfüllt m_n:

$$\begin{aligned}
&\mathbb{E} \int |m_n(x) - m(x)|^2 \mathbb{P}_X(dx) \\
&\leq \frac{c_{13} \cdot \log(n)^2 \cdot \left(\log\left(\sup_{x_1^n \in (supp(X))^n} \mathscr{N}_1\left(\frac{1}{n \cdot \beta_n}, \mathscr{F}_n, x_1^n\right)\right) + 1\right)}{n} \\
&\quad + 2 \cdot \mathbb{E}\left(\min_{l \in \Theta_n} \int |g_{n,l}(x) - m(x)|^2 \mathbb{P}_X(dx) + pen_n(g_{n,l}) \right),
\end{aligned}$$

für $n > 1$ und einer Konstante $c_{13} > 0$ welche nicht von n abhängt. (DEFINITION VON LP-e-ÜBERDECKUNGSZAHLEN)

Das nächste Lemma benötigen wir um eine Schranke für die Überdeckungszahl $\mathscr{N}_1\left(\frac{1}{n \cdot \beta_n}, \mathscr{F}_n, x_1^n\right)$ zu finden.

Lemma 1.2.8. *Sei $a > 0$ und $d, N, J_n \in \mathbb{N}$ so, dass $J_n \leq n^{c_{14}}$ und setze $\beta_n = c_6 \cdot \log(n)$. Sei σ 2-zulässig nach Definition 1.1.3. Sei $\mathscr{F}$ die Menge aller Funktionen die durch (2.1) definiert sind mit $k_1 = k_2 = \cdots = k_L = 24 \cdot (N + d)$ und dass der Betrag der Gewichte durch $c_{15} \cdot n^{c_{16}}$ beschränkt ist. Sei*

$$\mathscr{F}^{(J_n)} = \left\{ \sum_{j=1}^{J_n} a_j \cdot f_j : f_j \in \mathscr{F} \quad \textit{und} \quad \sum_{j=1}^{J_n} a_j^2 \leq c_{17} \cdot n^{c_{18}} \right\}.$$

Dann gilt für $n > 1$:

$$\log\left(supp_{x_1^n \in [-a,a]^{d \cdot n}} \mathscr{N}_1 \left(\frac{1}{n \cdot \beta_n}, \mathscr{F}^{(J_n)}, x_1^n \right) \right) \leq c_{19} \cdot \log(n) \cdot J_n,$$

für eine Konstante c_{19} die nur von L, N, a und d abhängt.

Kapitel 2

Konstruktion des Neuronale Netze Regressionsschätzers

In diesem Kapitel werden wir mithilfe von unseren gegebenen Datenmenge

$$\mathscr{D}_n = \{(X_1,Y_1),\dots,(X_n,Y_n)\},$$

unseren Regressionsschätzer konstruieren.
Die Netzwerkarchitektur (L,k) hängt von einer positiven ganzen Zahl L, die der Anzahl der verborgenen Schichten ist und einem Vektor $k = (k_1,\dots,k_L) \in \mathbb{N}^L$, der mit jeder Kombonente die Anzahl der Neuronen in der jeweiligen verborgen Schichte angibt.
Ein mehrschichtiges feedforward neuronales Netz mit Architektur (L,k) und dem logistischen squasher (1.1) als Aktivierungsfunktion, ist eine reelwertige Funktion $f\colon \mathbb{R}^d \to \mathbb{R}$ definiert durch:

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

für $c_0^{(L)},\dots,c_{k_L}^{(L)} \in \mathbb{R}$ und für $f_i^{(L)}$ rekursiv definiert durch:

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

für $c_{i,0}^{(r-1)},\dots,c_{i,k_{r-1}}^{(r-1)} \in \mathbb{R}(r = 2,\dots,L)$ und:

$$f_i^{(1)}(x) = \sigma\left(\sum_{j=1}^{d} c_{i,j}^{(0)} \cdot x^{(j)} + c_{i,0}^{(0)}\right) \tag{2.1}$$

für $c_{i,0}^{(0)}, \dots, c_{i,d}^{(0)} \in \mathbb{R}$.

Bei neuronale Netze Regressionsschätzer wählt man keine Aktivierungsfunktion mehr, da wir einen Funktionswert schätzen wollen nichts mit einer Wahrscheinlichkeit klassifizieren möchten. (REFERENZ)
Für die Konstruktion unseren Schätzers verwenden wir die gegebene Datenmenge $\mathscr{D}$ und wählen die Gewichte des neuronalen Netzes so, dass die resultierende Funktion aus (2.1) eine gute Schätzungen für die Regressionsfunktion ist. Dafür wählen wir die Gewichte bis auf die in der Ausgabeschicht fest und schätzen die Gewichte in der Ausgabeschicht in dem wir mit unserer Datenmenge ein regularisiertes Kleinste-Quadrate-Problem (REFERENZ) lösen.

2.1 Definition der Netzwerkarchitektur

Sei $a > 0$ fest. Die Wahl der Netzwerkarchitektur und der Werte aller Gewichte bis auf die aus der Ausgabeschicht ist durch folgendes Approximationsresultat durch eine lokale Konvexkombination von Taylorpolynomen für (p,C)-glatte Funktionen für $x \in [-a,a]^d$ motiviert. Sei dafür $M \in \mathbb{N}$ und $\mathbf{i} = (i^{(1)}, \dots, i^{(d)}) \in \{0, \dots, M\}^d$, sei

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

und sei

$$\{\mathbf{i}_1, \dots, \mathbf{i}_{(M+1)^d}\} = \{0, \dots, M\}^d,$$

d.h. $\mathbf{i}_1, \dots, \mathbf{i}_{(M+1)^d}$ sind insgesamt $M+1$ Vektoren der Dimension d, wobei jede Komponente aus der Menge $\{0, \dots, M\}$ ausgewählt wurde. Für $k \in \{1, \dots, (M+1)^d\}$ sei

$$p_{\mathbf{i}_k}(x) = \sum_{\substack{j_1, \dots, j_d \in \{0, \dots, q\} \\ j_1 + \dots + j_d \leq q}} \frac{1}{j_1! \cdots j_d!} \cdot \frac{\partial^{j_1 + \dots + j_d} m}{\partial^{j_1} x^{(1)} \dots \partial^{j_d} x^{(d)}} (x_{\mathbf{i}_k}) \cdot (x^{(1)} - x_{\mathbf{i}_k}^{(1)})^{j_1} \cdots (x^{(d)} - x_{\mathbf{i}_k}^{(d)})^{j_d}$$

das Taylorpolynom von m der Ordnung q im Entwicklungspunkt $x_{\mathbf{i}_k}$ und sei

$$P(x) = \sum_{k=1}^{(M+1)^d} p_{\mathbf{i}_k}(x) \prod_{j=1}^{d} \left(1 - \frac{M}{2a} \cdot |x^{(j)} - x_{\mathbf{i}_k}^{(j)}| \right)_+, \tag{2.2}$$

mit $z_+ = \max\{z, 0\} (z \in \mathbb{R})$. Wir zeigen im folgenden Lemma dass $P(x)$ eine lokale Konvexkombination von Taylorpolynomen von m ist.

Lemma 2.1.1. *Sei $a > 0, M \in \mathbb{N}$ und*

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

mit $p_{\mathbf{i}_k}(x)$ wie oben, dann ist $P(x)$ eine lokale Konvexkombination von Taylorpolynomen von m.

Beweis. Es sind drei Bedingungen zu überprüfen. Als erstes geben wir aber für $d = 2$ und $M = 3$ eine Skizze an um die Idee des Beweises zu veranschaulichen. (SKIZZE EINFÜGEN) Es ist ein Gitter mit $(M+1)^d$ Gitterpunkten die den $x_{\mathbf{i}_k}$ entsprechen. Der Abstand zwischen zwei Gitterpunkten beträgt $\frac{2a}{M}$. Man betrachtet immer den Abstand zu den nähesten 2^d Gitterpunkten, da $(1 - \frac{M}{2a} \cdot |x^{(j)} - x_{\mathbf{i}_k}^{(j)}|)_+ = 0$ immer dann gilt, wenn der Abstand zwischen $x^{(j)}$ und $x_{\mathbf{i}_k}^{(j)}$ größer als $\frac{2a}{M}$ ist.
i) Im folgenden wollen wir

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

per Induktion über d zeigen.
Induktionsanfang (IA): Für $d = 1$ kann nur zwischen zwei Gitterpunkten liegen und mit der obigen Begründung ist der Rest gleich Null, daher nehmen wir oBdA an, dass x zwischen $x_{\mathbf{i}_1}$ und $x_{\mathbf{i}_2}$ liegt. Damit folgt

$$\begin{aligned}\sum_{k=1}^{(M+1)} (1 - \frac{M}{2a} \cdot |x - x_{\mathbf{i}_k}|)_+ &= (1 - \frac{M}{2a} \cdot |x - x_{\mathbf{i}_1}|)_+ + (1 - \frac{M}{2a} \cdot |x - x_{\mathbf{i}_2}|)_+ \\ &= 1 + 1 - \frac{M}{2a} \cdot |x - x_{\mathbf{i}_1} + x_{\mathbf{i}_2} - x| \\ &= 1 + 1 - \frac{M}{2a} \cdot \frac{2a}{M} \\ &= 1,\end{aligned}$$

wobei wir unter anderem verwendet haben, dass beide Summenden 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 **i)** gilt für eine beliebiges aber festes $d \in \mathbb{N}$.
Induktionsschritt (IS): Wir nehmen oBdA. an, dass $x_{(0,\ldots,0) \le x \le x_{(1,\ldots,1}}$ gilt, mit $\mathbf{i}_1 = (0,\ldots,0)$ und $\mathbf{i}_{(M+1)}^{d+1} = (1,\ldots,1)$. Das heißt also, dass $x \in [-a, -a + \frac{2a}{M}]^{d+1}$ gilt. Im folgenden zeigen wir

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

Alle Summanden sind Null, wenn $|x^{(j)} - x^{(j)}_{\mathbf{i}_k}| \geq \frac{2a}{M}$ ist. Zudem haben wir oBdA angenommen dass $x \in [-a, -a + \frac{2a}{M}]^{d+1}$ gilt, damit haben wir also nur noch 2^{d+1} Summanden, nämlich die Anzahl der Gitterpunkte die am nähesten zu x sind. Zudem wissen wir, dass alle Gitterpunkte, die in der $(d+1)$ Komponente den selben Wert haben, sind in dieser Dimension gleich weit von $x^{(d+1)}$ entfernt. Das heißt, in jedem Summanden kommt der Faktor $(1 - \frac{M}{2a} \cdot |x^{(d+1)} - x^{(d+1)}_{(0,\dots,0)}|)$ bzw. $(1 - \frac{M}{2a} \cdot |x^{(d+1)} - x^{(d+1)}_{(1,\dots,1)}|)$ vor, da

$$(1 - \frac{M}{2a} \cdot |x^{(d+1)} - x^{(d+1)}_i|) = \begin{cases} (1 - \frac{M}{2a} \cdot |x^{(d+1)} - x^{(d+1)}_{(0,\dots,0)}|) & i \in \{0,1\}^d \times \{0\} \\ (1 - \frac{M}{2a} \cdot |x^{(d+1)} - x^{(d+1)}_{(1,\dots,1)}|) & i \in \{0,1\}^d \times \{1\} \end{cases}$$

daraus ergibt sich:

$$\begin{aligned}
&\sum_{k=1}^{(M+1)^{d+1}} \prod_{j=1}^{d+1} \left(1 - \frac{M}{2a} \cdot |x^{(j)} - x^{(j)}_{\mathbf{i}_k}|\right)_+ \\
&= \sum_{i \in \{0,1\}^{d+1}} \prod_{j=1}^{d+1} \left(1 - \frac{M}{2a} \cdot |x^{(j)} - x^{(j)}_i|\right) \\
&= \left(\sum_{i \in \{0,1\}^d \times \{0\}} \prod_{j=1}^{d} \left(1 - \frac{M}{2a} \cdot |x^{(j)} - x^{(j)}_i|\right)\right) \cdot \left(1 - \frac{M}{2a} \cdot |x^{(d+1)} - x^{(d+1)}_{(0,\dots,0)}|\right) \\
&\quad + \left(\sum_{i \in \{0,1\}^d \times \{1\}} \prod_{j=1}^{d} \left(1 - \frac{M}{2a} \cdot |x^{(j)} - x^{(j)}_i|\right)\right) \cdot \left(1 - \frac{M}{2a} \cdot |x^{(d+1)} - x^{(d+1)}_{(1,\dots,1)}|\right) \\
&\overset{(IV)}{=} 1 \cdot \left(1 - \frac{M}{2a} \cdot |x^{(d+1)} - x^{(d+1)}_{(0,\dots,0)}|\right) + 1 \cdot \left(1 - \frac{M}{2a} \cdot |x^{(d+1)} - x^{(d+1)}_{(1,\dots,1)}|\right) \\
&= 1 + 1 - \frac{M}{2a} \cdot |x^{(d+1)} - x^{(d+1)}_{(0,\dots,0)} + x^{(d+1)}_{(1,\dots,1)} - x^{(d+1)}| \\
&= 1 + 1 - 1 \\
&= 1,
\end{aligned}$$

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

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

$$z_+ = \max\{z, 0\} \geq 0 (z \in \mathbb{R})$$

gilt. Damit wäre die Nichtnegativität der Koeffizienten der Linearkombination gezeigt. Damit ist jeder Summand in

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

größer gleich Null und wegen i) muss dann auch

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

gelten.

iii) Es handelt sich hierbei um eine lokale Konvexität, da die Bedingungen **i)** und **ii)** für alle $x\in[-a,a]$ gelten. □

Als nächstes zeigen wir ein Resultat für (p,C)-glatte Funktion welches wir im weiteren Verlauf dieser Arbeit wieder benötigen werden.

Lemma 2.1.2. *Sei $M\in\mathbb{N}$, c_1 eine Konstante, $a>0$ und m eine (p,C)-glatte Funktion, wobei $p=q+s$ mit $q\in\mathbb{N}_0$ und $s\in(0,1]$. Sei zudem $P(x)$ wie in (2.2) eine lokale Konvexkombination von Taylorpolynomen von m. Dann gilt*:

$$\sup_{x\in[-a,a]^d}|m(x)-P(x)|\leq c_1\cdot\frac{1}{M^p}.$$

Beweis. Nach dem Satz über die Lagrange Form des Restglieds (REFERENZ) existiert ein $\xi\in[x,x_{\mathbf{i}_k}]$, so, dass

$$\begin{aligned}
m(x) &= T_{x_{\mathbf{i}_k},q-1}[m(x)]\\
&= \sum_{\substack{j_1,\dots,j_d\in\{0,\dots,q-1\}\\ j_1+\dots+j_d\leq q-1}}\frac{1}{j_1!\cdots j_d!}\cdot\frac{\partial^{j_1+\dots+j_d}m(x_{\mathbf{i}_k})}{\partial^{j_1}x^{(1)}\dots\partial^{j_d}x^{(d)}}\cdot(x^{(1)}-x_{\mathbf{i}_k}^{(1)})^{j_1}\cdots(x^{(d)}-x_{\mathbf{i}_k}^{(d)})^{j_d}\\
&+\sum_{\substack{q-1<j_1,\dots,j_d\leq q\\ j_1+\dots+j_d\leq q}}\frac{1}{j_1!\cdots j_d!}\cdot\frac{\partial^{j_1+\dots+j_d}m(\xi)}{\partial^{j_1}x^{(1)}\dots\partial^{j_d}x^{(d)}}\cdot(x^{(1)}-x_{\mathbf{i}_k}^{(1)})^{j_1}\cdots(x^{(d)}-x_{\mathbf{i}_k}^{(d)})^{j_d}.
\end{aligned}$$

Nach dem Beweis von Lemma 2.2 **i)** erhalten wir

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

Zudem wissen wir dass man immer den Abstand zu den nähesten 2^d Gitterpunkten betrachtet, da $(1-\frac{M}{2a}\cdot|x^{(j)}-x_{\mathbf{i}_k}^{(j)}|)_+=0$ immer dann gilt, wenn der Abstand zwischen $x^{(j)}$ und $x_{\mathbf{i}_k}^{(j)}$ größer als $\frac{2a}{M}$ ist, daher ergibt sich:

$$\begin{aligned}
&\sum_{\substack{q-1<j_1,\dots,j_d\leq q\\ j_1+\dots+j_d\leq q}}\frac{1}{j_1!\cdots j_d!}\cdot\frac{\partial^{j_1+\dots+j_d}m(\xi)}{\partial^{j_1}x^{(1)}\dots\partial^{j_d}x^{(d)}}\cdot(x^{(1)}-x_{\mathbf{i}_k}^{(1)})^{j_1}\cdots(x^{(d)}-x_{\mathbf{i}_k}^{(d)})^{j_d}\\
&\leq\sum_{\substack{q-1<j_1,\dots,j_d\leq q\\ j_1+\dots+j_d\leq q}}\frac{1}{j_1!\cdots j_d!}\cdot\frac{\partial^{j_1+\dots+j_d}m(\xi)}{\partial^{j_1}x^{(1)}\dots\partial^{j_d}x^{(d)}}\cdot\left(\frac{2a}{M}\right)^q
\end{aligned}$$

und folgern mithilfe der Dreiecksungleichung und der (p,C)-Glattheit von m:

$$\begin{aligned}
|m(x)-P(x)| &\le \sum_{k=1}^{(M+1)^d} |m(x)-p_{\mathbf{i}_k}| \prod_{j=1}^{d} \left(1-\frac{M}{2a}\cdot|x^{(j)}-x_{\mathbf{i}_k}^{(j)}|\right)_+ \\
&\le \left(\frac{2a}{M}\right)^q \|\xi - x_{\mathbf{i}_k}\|^s \cdot C \cdot \sum_{k=1}^{(M+1)^d} \prod_{j=1}^{d} \left(1-\frac{M}{2a}\cdot|x^{(j)}-x_{\mathbf{i}_k}^{(j)}|\right)_+ \\
&= C\cdot\left(\frac{2a}{M}\right)^p \\
&= c_1 \cdot \frac{1}{M^p},
\end{aligned}$$

wobei wir bei der letzten Gleichung Bedingung **i)** aus dem Beweis von Lemma 2.2 und $q+s=p$ verwendet haben. □

$P(x)$ lässt sich in die Form

$$\sum_{k=1}^{(M+1)^d} \sum_{\substack{j_1,\dots,j_d\in\{0,\dots,q\}\\ j_1+\dots+j_d\le q}} a_{\mathbf{i}_k,j_1,\dots,j_d}\cdot(x^{(1)}-x_{\mathbf{i}_k}^{(1)})^{j_1}\cdots(x^{(d)}-x_{\mathbf{i}_k}^{(d)})^{j_d} \prod_{j=1}^{d}\left(1-\frac{M}{2a}\cdot|x^{(j)}-x_{\mathbf{i}_k}^{(j)}|\right)_+$$

durch geeignet gewählte $a_{\mathbf{i}_k,j_1,\dots,j_d}\in\mathbb{R}$ bringen. Als nächstes wollen wir geeignete neuronale Netze $f_{net,j_1,\dots,j_d,\mathbf{i}_k}$ definieren, die die Funktionen

$$x\mapsto (x^{(1)}-x_{\mathbf{i}_k}^{(1)})^{j_1}\cdots(x^{(d)}-x_{\mathbf{i}_k}^{(d)})^{j_d} \prod_{j=1}^{d}\left(1-\frac{M}{2a}\cdot|x^{(j)}-x_{\mathbf{i}_k}^{(j)}|\right)_+$$

approximieren. Zudem möchten wir die Netzwerkarchitektur so wählen, dass neuronale Netze der Form

$$\sum_{k=1}^{(M+1)^d} \sum_{\substack{j_1,\dots,j_d\in\{0,\dots,q\}\\ j_1+\dots+j_d\le q}} a_{\mathbf{i}_k,j_1,\dots,j_d}\cdot f_{net,j_1,\dots,j_d,\mathbf{i}_k}(x) \qquad (a_{\mathbf{i}_k,j_1,\dots,j_d}\in\mathbb{R})$$

in ihr enthalten sind. Um dies zu erreichen, sei $\sigma(x)=\frac{1}{(1+\exp(-x))}\quad(x\in\mathbb{R})$ der logistische Squasher (1.1), wählen $R\ge 1$ und definieren die folgenden neuronale Netze: Das neuronale Netz

$$f_{id}(x) = 4R\cdot\sigma\left(\frac{x}{R}\right)-2R, \tag{2.3}$$

welches, wie in Lemma 1.2.2 gezeigt, die Funktion $f(x)=x$ approximiert. Das neuronale Netz

$$\begin{aligned}
f_{mult}(x,y) = \frac{R^2}{4}\cdot\frac{(1+\exp(-1))^3}{\exp(-2)-\exp(-1)}\cdot\Bigg(&\sigma\Big(\frac{2(x+y)}{R}+1\Big)-2\cdot\sigma\Big(\frac{x+y}{R}+1\Big) \\
&-\sigma\Big(\frac{2(x-y)}{R}+1\Big)+2\cdot\sigma\Big(\frac{x-y}{R}+1\Big)\Bigg),
\end{aligned}\tag{2.4}$$

welches, wie in Lemma 1.2.3 gezeigt, die Funktion $f(x,y) = x \cdot y$ approximiert. Das neuronale Netz

$$f_{ReLu}(x) = f_{mult}(f_{id}(x), \sigma(R \cdot x)), \tag{2.5}$$

welches, wie in Lemma 1.2.4 gezeigt, die Funktion $f(x) = x_+$ approximiert und schließlich noch das neuronale Netz

$$f_{hat,y}(x) = f_{ReLu}\left(\frac{M}{2a} \cdot (x-y) + 1\right) - 2 \cdot f_{ReLu}\left(\frac{M}{2a} \cdot (x-y)\right) + f_{ReLu}\left(\frac{M}{2a} \cdot (x-y) - 1\right), \tag{2.6}$$

welches, wie in Lemma 1.2.5 gezeigt, für fixes $y \in \mathbb{R}$ die Funktion

$$f(x) = \left(1 - \left(\frac{M}{2a}\right) \cdot |x-y|\right)_+$$

approximiert. Mit diesen neuronalen Netzen können wir nun $f_{net,j_1,\ldots,j_d,\mathbf{i}_k}$ rekursiv definieren. Dafür wählen wir $N \geq q$, setzen $s = \lceil \log_2(N+d) \rceil$ und definieren für $j_1, \ldots, j_d \in \{0, 1, \ldots, N\}$ und $k \in \{1, \ldots, (M+1)^d\}$:

$$f_{net,j_1,\ldots,j_d,\mathbf{i}_k}(x) = f_1^{(0)}(x). \tag{2.7}$$

wobei

$$\tag{2.8}$$

$$f_k^{(l)}(x) = f_{mult}\left(f_{2k-1}^{(l+1)}(x), f_{2k}^{(l+1)}(x)\right) \tag{2.9}$$

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

$$\tag{2.10}$$

$$f_k^{(s)}(x) = f_{id}(f_{id}(x^{(l)} - x_{\mathbf{i}_k}^{(l)})) \tag{2.11}$$

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

$$\tag{2.12}$$

$$f_{j_1+j_2+\cdots+j_d+k}^{(s)}(x) = f_{hat,x_{\mathbf{i}_k}^{(k)}}(x^{(k)}) \tag{2.13}$$

für $k = 1, \ldots, d$ und

$$\tag{2.14}$$

$$f_k^{(s)}(x) = 1 \tag{2.15}$$

für $k = j_1 + j_2 + \cdots + j_d + d + 1, j_1 + j_2 + \cdots + j_d + d + 2, \ldots, 2^s$.

$$\tag{2.16}$$

ANZAHL DER SCHICHTEN UND NEURONEN PRO SCHICHT ERLÄUTERN. Da man bei fully-connencted neuronalen Netzen die Gewichte der Verbindungen zwischen zwei Neuronen auf Null setzen kann, sind auch nicht fully-connected neuronale Netze in der der Klasse aller fully-connected neuronaler Netze, mit $s+2$ verbogenen Schichten mit jeweils $24 \cdot (N+d)$ Neuronen pro Schicht, enthalten und damit auch insbesondere $f_{net,j_1,\dots,j_d,\mathbf{i}_k}$.

2.2 Definition der Gewichte der Ausgabeschicht

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

$$\tilde{m}_n(x) = \sum_{k=1}^{(M+1)^d} \sum_{\substack{j_1,\dots,j_d \in \{0,\dots,N\} \\ j_1+\dots+j_d \leq N}} a_{\mathbf{i}_k,j_1,\dots,j_d} \cdot f_{net,j_1,\dots,j_d,\mathbf{i}_k}(x), \tag{2.17}$$

wobei wir die Koeffizienten $a_{\mathbf{i}_k,j_1,\dots,j_d}$ durch Minimierung von

$$\frac{1}{n}\sum_{i=1}^{n} |Y_i - \tilde{m}_n(X_i)|^2 + \frac{c_3}{n} \cdot \sum_{k=1}^{(M+1)^d} \sum_{\substack{j_1,\dots,j_d \in \{0,\dots,N\} \\ j_1+\dots+j_d \leq N}} a^2_{\mathbf{i}_k,j_1,\dots,j_d} \tag{2.18}$$

für eine Konstante $c_3 > 0$. Dieses regularisierte lineare Kleinste-Quadrate Schätzung erhalten wir durch die Lösung eine linearen Gleichungssystems. Dafür definieren wir uns

$$\{B_j \mid j = 1,\dots,J\} = \left\{ f_{net,j_1,\dots,j_d,\mathbf{i}_k}(x) \mid 1 \leq k \leq (1+M)^d \quad \text{und} \quad 0 \leq j_1 + \dots + j_d \leq N \right\}$$

wobei

$$J = (M+1)^d \cdot \binom{N+d}{d}$$

die Kardinalität der Menge ist. Dies erhält man durch TBD. Wir setzen nun

$$\mathbf{B} = (B_j(X_i))_{1 \leq i \leq n, 1 \leq j \leq J} \quad \text{und} \quad \mathbf{Y} = (Y_i)_{i=1,\dots,n}.$$

Wir zeigen im Folgenden, dass der Koeffizientenvektor unseres Schätzers 2.17 die eindeutige Lösung des linearen Gleichungssystems

$$\left(\frac{1}{n}\mathbf{B}^T\mathbf{B} + \frac{c_3}{n} \cdot \mathbf{1} \right) \mathbf{a} = \frac{1}{n}\mathbf{B}^T\mathbf{Y}, \tag{2.19}$$

wobei $\mathbf{1}$ eine JxJ-Einheitsmatrix ist. Den Schätzer aus 2.17 kann man umschreiben zu

$$\tilde{m}_n(x) = \sum_{j=1}^{J} a_j \cdot B_j(x)$$

wobei $\mathbf{a} = (a_j)_{j=1,\dots,J} \in \mathbb{R}^J$ wie in 2.18 den Ausdruck

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n}|Y_i - \tilde{m}_n(X_i)|^2 + \frac{c_3}{n} \cdot \sum_{\substack{j_1,\dots,j_d \in \{0,\dots,q\} \\ j_1+\dots+j_d \le q}} a^2_{\mathbf{i}_k,j_1,\dots,j_d} \\
&= \frac{1}{n}(\mathbf{Y}-\mathbf{Ba})^T(\mathbf{Y}-\mathbf{Ba}) + \frac{c_3}{n}\cdot \mathbf{a}^T\mathbf{a} \\
&= \frac{1}{n}(\mathbf{Y}^T\mathbf{Y} - \mathbf{Y}^T\mathbf{Ba} - \mathbf{a}^T\mathbf{B}^T\mathbf{Y} + \mathbf{a}^T\mathbf{B}^T\mathbf{Ba}) + \frac{c_3}{n}\cdot \mathbf{a}^T\mathbf{a} \\
&= \frac{1}{n}(\mathbf{Y}^T\mathbf{Y} - 2\mathbf{Y}^T\mathbf{Ba}) + \mathbf{a}^T\left(\frac{1}{n}\mathbf{B}^T\mathbf{B} + \frac{c_3}{n}\cdot \mathbf{1}\right)\mathbf{a},
\end{aligned} \tag{2.20}$$

minimiert. In der vorletzten Gleichung haben wir verwendet das $\mathbf{Y}^T\mathbf{Ba} = \mathbf{a}^T\mathbf{B}^T\mathbf{Y}$ gilt, da dieser Ausdruck eine reelle Zahl und damit insbesondere symmetrisch ist. Die Matrix $\mathbf{B}^T\mathbf{B} \in \mathbb{R}^{JxJ}$ ist positiv semidefinit, denn aufgrund der Verschiebungseigenschaft des Standardskalarprodukts gilt für alle $x \in \mathbb{R}$:

$$\langle x, \mathbf{B^T B}x\rangle = \langle \mathbf{B}x, \mathbf{B}x\rangle \ge 0.$$

Zudem wissen wir dass $\frac{c_3}{n}\mathbf{1}$ durch die Wahl von c_3 nur positive Eigenwerte besitzt und damit positiv definit ist. Daher wissen wir, dass die Matrix

$$\mathbf{A} = \frac{1}{n}\mathbf{B}^T\mathbf{B} + \frac{c_3}{n}\cdot \mathbf{1}$$

ebenfalls nur positive Eigenwerte besitzt (REFERENZ), damit also positiv definit ist und eine inverse Matrix $\mathbf{A}^{-1}$ existiert. Zudem ist die Matrix $\mathbf{A}$ symmetrisch. Mit $\mathbf{b} = \frac{1}{n}\cdot \mathbf{A}^{-1}\mathbf{B}^T\mathbf{Y} \in \mathbb{R}^J$ und $\mathbf{b}^T\mathbf{Aa} = \mathbf{aAb} = \mathbf{Y}^T\mathbf{Ba}$, was aus der Symmetrie von $\mathbf{A}$ folgt, erhalten wir:

$$\begin{aligned}
&\frac{1}{n}(\mathbf{Y}^T\mathbf{Y} - 2\mathbf{Y}^T\mathbf{Ba}) + \mathbf{a}^T\left(\frac{1}{n}\mathbf{B}^T\mathbf{B} + \frac{c_3}{n}\cdot \mathbf{1}\right)\mathbf{a} \\
&= \mathbf{a}^T\mathbf{Aa} + \mathbf{b}^T\mathbf{Aa} + \mathbf{a}^T\mathbf{Ab} + \mathbf{b}^T\mathbf{Ab} + \frac{1}{n}\mathbf{Y}^T\mathbf{Y} - \frac{1}{n^2}\mathbf{Y}^T\mathbf{BA}^{-1}\mathbf{B}^T\mathbf{Y} \\
&= (\mathbf{a} + \frac{1}{n}\cdot \mathbf{A}^{-1}\mathbf{B}^T\mathbf{Y})^T\mathbf{A}(\mathbf{a} - \frac{1}{n}\cdot \mathbf{A}^{-1}\mathbf{B}^T\mathbf{Y}) - \frac{1}{n}\mathbf{Y}^T\mathbf{Y} - \frac{1}{n^2}\mathbf{Y}^T\mathbf{BA}^{-1}\mathbf{B}^T\mathbf{Y}.
\end{aligned}$$

Die letzte Gleichung wird für $\mathbf{a} = \frac{1}{n}\cdot \mathbf{A}^{-1}\mathbf{B}^T\mathbf{Y}$ minimal, da wir wissen dass $\mathbf{A}$ positiv definit ist und damit $x^T\mathbf{A}x > 0$ für alle $x \in \mathbb{R}^J$ mit $x \neq 0$ gilt und $(\mathbf{a}-\mathbf{b})^T\mathbf{A}(\mathbf{a}-\mathbf{b}) = 0$ ist für $\mathbf{a} = \mathbf{b}$. Dies zeigt also, dass der Koeffizientenvektor unseres Schätzers 2.17 die eindeutige Lösung des linearen Gleichungssystems 2.19 ist. Da der Koeffizientenvektor die Gleichung 2.18 minimiert, erhalten wir wenn wir den Koeffizientenvektor mit dem Nullvektor gleichsetzen:

$$\frac{1}{n}(\mathbf{Y}-\mathbf{Ba})^T(\mathbf{Y}-\mathbf{Ba}) + \frac{c_3}{n}\cdot \mathbf{a}^T\mathbf{a} \le \frac{1}{n}\sum_{i=1}^{n} Y_i^2,$$

was uns erlaubt eine obere Schranke für den absoluten Wert unsere Koeffizienten abzuleiten.

Kapitel 3

Resultat zur Konvergenzgeschwindigkeit

In diesem Kapitel stellen wir das Hauptresultat dieser Arbeit vor, ein Resultat zur Konvergenzgeschwindigkeit unseres neuronale Netze Regresssionsschätzers 2.17.
Ziel im Folgenden ist die Abschätzung des erwarteten L_2-Fehlers

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

im Falle des sogenannte ... Schätzers mit ...

Satz 3.0.1. *Angenommen die Verteilung von (X,Y) erfüllt*

$$\mathbb{E}\left(\mathrm{e}^{c_4\cdot|Y|^2}\right)<\infty$$

für eine Konstante $c_4>0$ und die Verteilung von X hat einen beschränkten Träger $supp(\mathbb{P}_X)$ und sei $m(x)=\mathbb{E}[Y \mid X=x]$ die entsprechende Regressionsfunktion. Angenommen m ist (p,C)-glatt, mit $p=q+s$ für $q\in\mathbb{N}_0$ und $s\in(0,1]$. Wir betrachten unseren neuronale Netze Regressionsschätzer $\tilde{m}_n$ aus 2.17, wobei σ der logistische Squasher ist und $N\geq q, M=M_n=\lceil c_5\cdot n^{1/(2p+d)}\rceil, R=R_n=n^{d+4}$ und $a=a_n=(\log n)^{1/(6(N+d))}$. Sei $\beta_n=c_6\cdot\log(n)$ für eine hinreichend große Konstante $c_6>0$ und sei m_n gegeben durch

$$m_n(x)=T_{\beta_n}\tilde{m}_n(x)$$

mit $T_{\beta z}=\max\{\min\{z,\beta\},-\beta\}$ für $z\in\mathbb{R}$ und $\beta>0$. Dann gilt für m_n für hinreichend großes n

$$\mathbb{E}\int |m_n(x)-m(x)|^2\mathbb{P}_X(dx)\leq c_7\cdot(log n)^3\cdot n^{-\frac{2p}{2p+d}},$$

wobei $c_7>0$ nicht von n abhängt.

Beweis. Nach Voraussetzung wissen wir, dass $supp(\mathbb{P}_X)$ beschränkt ist. Da wir angenommen haben, dass m (p,C)-glatt und damit insbesondere hölderstetig mit $q=0$ ist, können wir daraus auf die gleichmäßige Stetigkeit von m schließen. Da wir nur über den beschränkten $supp(\mathbb{P}_X)$ integrieren, wissen wir dass m als gleichmäßig stetige Funktion auf einer beschränkten Menge auch beschränkt ist Wir können daher auch oBdA folgern, dass $\|m\|_\infty \leq \beta_n$ gilt, weil wir aufgrund der Bechränktheit von m (IST DER BETRAG VON M AUCH BESCHRÄNKT) einfach eine Skalierung von m nehmen können. Zudem nehmen wir oBdA an, dass $supp(X) = \{x \mid \mathbb{P}_X(x) > 0\} \supseteq [-a_n, a_n]^d$ ist, da wir ansonsten die Zufallsvariable X auf Nullmengen abändern können.
Sei $\mathscr{F}$ die durch 2.1 definierte Menge von Funktionen mit $L = s+2 = \lceil \log_2(N+d) \rceil + 2$, mit $k_1 = k_2 = \cdots = k_L = 24 \cdot (N+d)$ und wo der Betrag der Gewichte durch $n^{c_{20}}$ beschränkt ist. Sei

$$\mathscr{F}^{(J_n)} = \left\{ \sum_{j=1}^{J_n} a_j \cdot f_j \mid f_j \in \mathscr{F} \quad \text{und} \quad \sum_{j=1}^{J_n} a_j^2 \leq c_{21} \cdot n \right\}$$

wobei c_{21} wie unten gewählt wird und für die Kardinalität der Menge gilt

$$J_n = (M_n+1)^d \cdot |\{(j_1, \ldots, j_d) \mid j_1, \ldots, j_d \in \{0, \ldots, N\}, j_1 + \cdots j_d \leq N\}|.$$

Ohne Berücksichtigung der Restriktion $j_1 + \cdots j_d \leq N$ lässt sich

$$|\{(j_1, \ldots, j_d) \mid j_1, \ldots, j_d \in \{0, \ldots, N\}|$$

durch eine Analogie zu der Anzahl an Möglichkeiten, die man in einem Urnenexperiment berechnen in welchem man d-Mal mit Zurücklegen und mit Beachtung der Reihenfolge aus $(N+1)$ Kugeln zieht. Durch das Weglassen der Restriktion erhalten wir

$$J_n \leq (M_n+1)^d \cdot (N+1)^d. \tag{3.1}$$

Sei

$$g_n(x) = \sum_{k=1}^{(M_n+1)^d} \sum_{\substack{j_1,\ldots,j_d \in \{0,\ldots,q\} \\ j_1+\cdots+j_d \leq q}} \frac{1}{j_1! \cdots j_d!} \cdot \frac{\partial^{j_1+\cdots+j_d} m}{\partial^{j_1} x^{(1)} \ldots \partial^{j_d} x^{(d)}}(x_{\mathbf{i}_k}) \cdot f_{net, j_1, \ldots, j_d, \mathbf{i}_k}(x).$$

Nach Konstruktion liegt g_n daher auch in $\mathscr{F}^{(J_n)}$. Da nach Voraussetzung m (p,C)-glatt ist, und $x_{\mathbf{i}_k} \in \mathbb{R}^d$ für alle $k = 1, \ldots, (M_n+1)^d$, gilt:

$$\max_{k \in \{1,\ldots,(M_n+1)^d\}, j_1,\ldots,j_d \in \{0,\ldots,q\}, j_1+\cdots+j_d \leq q} \left| \frac{\partial^{j_1+\cdots+j_d} m}{\partial^{j_1} x^{(1)} \ldots \partial^{j_d} x^{(d)}}(x_{\mathbf{i}_k}) \right| < \infty, \tag{3.2}$$

denn das größte Elemente muss auch beschränkt sein, wenn der Abstand zweier beliebiger Elemente immer beschränkt ist. Da wir uns in der nichtparametrischen Regressionsschätzung befinden und dafür als Bedingung $\mathbb{E}[Y^2] < \infty$ gelten muss, mit wählen wir mit dem bisher gezeigten:

$$c_{21} = \max\left\{\frac{1+\mathbb{E}[Y^2]}{c_3}, (N+1)^d \cdot \max\left\{\left|\frac{1}{j_1!\cdots j_d!} \cdot \frac{\partial^{j_1+\cdots+j_d} m}{\partial^{j_1} x^{(1)} \ldots \partial^{j_d} x^{(d)}}(x_{\mathbf{i}_k})\right|^2 : \right.\right.$$
$$\left.\left. j_1,\ldots,j_d \in \{0,\ldots,q\}, j_1+\cdots j_d \le q\right\}\right\}. \tag{3.3}$$

Sei A_n das Event, dass

$$\frac{1}{n}\sum_{i=1}^{n} Y_i^2 \le 1+\mathbb{E}[Y^2] \tag{3.4}$$

gilt. Wir wissen, dass aufgrund der Unabhängigkeit der $\mathbb{R}^d \times \mathbb{R}$-wertigen Zufallsvariablen $(X,Y),(X_1,Y_1),(X_2,Y_2),\ldots$ nach

$$\begin{aligned}
\mathbb{P}(Y_1 \in \mathbb{R},\ldots,Y_n \in R) &= \mathbb{P}((X_1,Y_1) \in \mathbb{R}^d \times \mathbb{R},\ldots,(X_n,Y_n) \in \mathbb{R}^d \times \mathbb{R}) \\
&= \mathbb{P}((X_1,Y_1) \in \mathbb{R}^d \times \mathbb{R}) \cdots \mathbb{P}((X_n,Y_n) \in \mathbb{R}^d \times \mathbb{R}) \\
&= \mathbb{P}(Y_1 \in \mathbb{R}) \cdots \mathbb{P}(Y_n \in R),
\end{aligned}$$

und durch

$$\begin{aligned}
\mathbb{P}(Y_1 \in \mathbb{R},\ldots,Y_n \in R) &= \mathbb{P}((X_1,Y_1) \in \mathbb{R}^d \times \mathbb{R},\ldots,(X_n,Y_n) \in \mathbb{R}^d \times \mathbb{R}) \\
&= \mathbb{P}((X_1,Y_1) \in \mathbb{R}^d \times \mathbb{R}) \cdots \mathbb{P}((X_n,Y_n) \in \mathbb{R}^d \times \mathbb{R}) \\
&= \mathbb{P}((X,Y) \in \mathbb{R}^d \times \mathbb{R})^n \\
&= \mathbb{P}(Y \in \mathbb{R})^n
\end{aligned}$$

dass die Zufallsvariablen $Y_1,\ldots,Y_n$ auch unabhängig und identisch verteilt sind. Daraus folgern wir mit Hilfe der Linearität des Erwartungswerts, dass $\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^{n} Y_i^2\right] = \mathbb{E}[Y^2]$ gilt. Mit Hilfe der Monotonie und Homogenität der Wahrscheinlichkeitsfunktion $\mathbb{P}$ und der Tschebyscheff Ungleichung für $\varepsilon = 1$ (REFERENZ) erhalten wir

$$\begin{aligned}
\mathbb{P}(A_n^{\mathsf{c}}) &= \mathbb{P}(\frac{1}{n}\sum_{i=1}^{n} Y_i^2 - \mathbb{E}[Y^2] \ge 1) \\
&\le \mathbb{P}(\left|\frac{1}{n}\sum_{i=1}^{n} Y_i^2 - \mathbb{E}[Y^2]\right| \ge 1) \\
&\le \mathbb{V}[\frac{1}{n}\sum_{i=1}^{n} Y_i^2] \\
&= \frac{n \cdot \mathbb{V}[Y^2]}{n^2} \\
&= \frac{\mathbb{V}[Y^2]}{n} \\
&\le \frac{c_{22}}{n},
\end{aligned} \tag{3.5}$$

wobei wir bei der letzten Gleichheit die identische Verteiltheit der $Y_1, \ldots, Y_n$ und Rechenregeln für die Varianz verwendet haben welche wir unter anderem aufgrund der Unabhängigkeit der $Y_1, \ldots, Y_n$ verwenden durften. Sei $\hat{m}_n = T_{\beta_n}\tilde{m}_n = m_n$ im Falle dass Ereignis A_n gilt und andernfalls $\hat{m}_n = T_{\beta_n} g_n$. Durch die Unabhängigkeit von A_n zu den Zufallsvariablen $X, X_1, \ldots, X_n$ und der Jensenschen Ungleichung (REFERENZ) erhalten wir:

$$\begin{aligned}
\mathbb{E}\left[\int |m_n(x) - m(x)|^2 \mathbb{P}_X(dx) \mathbb{1}_{A_n^c}\right] &= \mathbb{E}\left[\int |m_n(x) - m(x)|^2 \mathbb{P}_X(dx)\right] \cdot \mathbb{P}(A_n^c) \\
&\leq \mathbb{E}\left[2m_n(x)^2 + 2m(x)^2 \mathbb{P}_X(dx)\right] \cdot \mathbb{P}(A_n^c) \\
&\leq \mathbb{E}\left[2\beta_n^2 + 2\beta_n^2 \mathbb{P}_X(dx)\right] \cdot \mathbb{P}(A_n^c) \\
&= 4\beta_n^2 \cdot \mathbb{P}(A_n^c),
\end{aligned} \tag{3.6}$$

wobei wir bei der letzten Ungleichung verwendet haben dass wir anfangs angenommen haben, dass $\|m\|_\infty < \beta_n$ und damit für c_6 und n hinreichend groß auch $\tilde{m}_n \leq \beta_n$ und nach der Definition von m_n zudem $m_n \leq \beta_n$ gilt. Bei der letzten Gleichung habe wir dann schließlich noch verwendet dass β_n deterministisch und $\mathbb{P}(X \in supp(\mathbb{P}_X)) = 1$ ist. Durch unsere Definition von $\hat{m}_n$ erhalten wir durch die Monotonie des Erwartungswert und der Abschätzung durch den ganzen Raum:

$$\begin{aligned}
\mathbb{E}\left[\int |m_n(x) - m(x)|^2 \mathbb{P}_X(dx) \mathbb{1}_{A_n}\right] &= \mathbb{E}\left[\int |\hat{m}_n(x) - m(x)|^2 \mathbb{P}_X(dx) \mathbb{1}_{A_n}\right] \\
&\leq \mathbb{E}\left[\int |\hat{m}_n(x) - m(x)|^2 \mathbb{P}_X(dx)\right].
\end{aligned} \tag{3.7}$$

Zusammen mit (3.5), (3.6), (3.7) und der Linearität des Erwartungswerts erhalten wir dann:

$$\begin{aligned}
\mathbb{E}\int |m_n(x) - m(x)|^2 \mathbb{P}_X(dx) &= \mathbb{E}\left[\int |m_n(x) - m(x)|^2 \mathbb{P}_X(dx) \cdot (\mathbb{1}_{A_n^c} + \mathbb{1}_{A_n})\right] \\
&= \mathbb{E}\left[\int |m_n(x) - m(x)|^2 \mathbb{P}_X(dx) \mathbb{1}_{A_n^c}\right] \\
&\quad + \mathbb{E}\left[\int |m_n(x) - m(x)|^2 \mathbb{P}_X(dx) \mathbb{1}_{A_n}\right] \\
&\leq 4\beta_n^2 \cdot \mathbb{P}(A_n^c) + \mathbb{E}\left[\int |\hat{m}_n(x) - m(x)|^2 \mathbb{P}_X(dx)\right] \\
&\leq \frac{4 \cdot c_{22} \cdot \beta_n^2}{n} + \mathbb{E}\left[\int |\hat{m}_n(x) - m(x)|^2 \mathbb{P}_X(dx)\right].
\end{aligned}$$

Nach (2.17) können wir die 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

$$\begin{aligned}
\frac{c_3}{n}\sum_{j=1}^{J_n}\hat{a}_j^2 &= \frac{c_3}{n}\sum_{k=1}^{(M+1)^d}\sum_{\substack{j_1,\dots,j_d\in\{0,\dots,q\}\\ j_1+\dots+j_d\le q}} a_{\mathbf{i}_k,j_1,\dots,j_d}^2\\
&\le \frac{1}{n}\sum_{i=1}^{n}|Y_i-\tilde{m}_n(X_i)|^2+\frac{c_3}{n}\cdot\sum_{\substack{j_1,\dots,j_d\in\{0,\dots,q\}\\ j_1+\dots+j_d\le q}} a_{\mathbf{i}_k,j_1,\dots,j_d}^2\\
&\le \sum_{i=1}^{n}Y_i^2,
\end{aligned}$$

erfüllen, wobei wir bei der letzten Ungleichung wie in (2.18) die minimierende Eigenschaft von $a_{\mathbf{i}_k,j_1,\dots,j_d}$ verwendet haben und die Koeffizienten Null gesetzt haben. Da $c_3 > 0$ ist, erhalten wir dass die $\hat{a}_j$ die Eigenschaft

$$\sum_{j=1}^{J_n}\hat{a}_j^2 \le \frac{1}{n}\sum_{i=1}^{n}Y_i^2\cdot\frac{n}{c_3}$$

erfüllen müssen. Auf A_n erhalten wir dann:

$$\sum_{j=1}^{J_n}\hat{a}_j^2 \overset{(3.4)}{\le} \frac{1+\mathbb{E}[Y^2]}{c_3}\cdot n \overset{(3.3)}{\le} c_{21}\cdot n,$$

woraus dann insbesondere $\tilde{m}_n \in \mathscr{F}^{(J_n)}$ folgt, da $f_j \in \mathscr{F}$. Deswegen nehmen wir nun oBdA. an, dass $\hat{m}_n = T_{\beta_n}\bar{m}_n$ für $\bar{m}_n = \tilde{m}_n, g_n \in \mathscr{F}^{(J_n)}$. Da $c_3 > 0$ ist erhalten wir:

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n}|Y_i-g_n(X_i)|^2\\
&\le \frac{1}{n}\sum_{i=1}^{n}|Y_i-g_n(X_i)|^2\\
&\quad+\frac{c_3}{n}\cdot\sum_{k=1}^{(M+1)^d}\sum_{\substack{j_1,\dots,j_d\in\{0,\dots,q\}\\ j_1+\dots+j_d\le q}}\left|\frac{1}{j_1!\cdots j_d!}\cdot\frac{\partial^{j_1+\dots+j_d}m}{\partial^{j_1}x^{(1)}\dots\partial^{j_d}x^{(d)}}(x_{\mathbf{i}_k})\right|^2 \qquad (3.8)\\
&\overset{(3.3)}{\le} \frac{1}{n}\sum_{i=1}^{n}|Y_i-g_n(X_i)|^2+\frac{c_3}{n}\cdot\sum_{k=1}^{(M+1)^d}c_{21}\\
&= \frac{1}{n}\sum_{i=1}^{n}|Y_i-g_n(X_i)|^2+\frac{c_3\cdot c_{21}\cdot(M_n+1)^d}{n}\\
&= \frac{1}{n}\sum_{i=1}^{n}|Y_i-g_n(X_i)|^2+c_{23}\cdot\frac{(M_n+1)^d}{n}.
\end{aligned}$$

Die Funktionen $\tilde{m}_n$ und g_n unterscheiden sich in den Koeffizienten und da wir die Koeffizienten $a_{\mathbf{i}_k,j_1,\dots,j_d}$ von $\tilde{m}_n$ durch Minimierung von (2.18) erhalten haben und nach

Voraussetzung $N \geq q$ ist, damit dann $\{0,\dots,q\} \subseteq \{0,\dots,N\}$ und wir bei der Minimierung daher auch insbesondere die Koeffizienten von g_n betrachtet haben, erhalten wir:

$$\begin{aligned}
&\frac{1}{n}\sum_{i=1}^{n}|Y_i - \tilde{m}_n(X_i)|^2 \\
&\leq \frac{1}{n}\sum_{i=1}^{n}|Y_i - \tilde{m}_n(X_i)|^2 + \frac{c_3}{n}\cdot \sum_{k=1}^{(M+1)^d} \sum_{\substack{j_1,\dots,j_d\in\{0,\dots,N\}\\ j_1+\dots+j_d\leq N}} a^2_{\mathbf{i}_k,j_1,\dots,j_d} \\
&\leq \frac{1}{n}\sum_{i=1}^{n}|Y_i - g_n(X_i)|^2 \qquad (3.9)\\
&\quad + \frac{c_3}{n}\cdot \sum_{k=1}^{(M+1)^d} \sum_{\substack{j_1,\dots,j_d\in\{0,\dots,q\}\\ j_1+\dots+j_d\leq q}} \left|\frac{1}{j_1!\cdots j_d!}\cdot\frac{\partial^{j_1+\dots+j_d} m}{\partial^{j_1}x^{(1)}\dots\partial^{j_d}x^{(d)}}(x_{\mathbf{i}_k})\right|^2 \\
&= \frac{1}{n}\sum_{i=1}^{n}|Y_i - g_n(X_i)|^2 + c_{23}\cdot\frac{(M_n+1)^d}{n}
\end{aligned}$$

Mit (3.8) und (3.9) erhalten wir zusammen:

$$\frac{1}{n}\sum_{i=1}^{n}|Y_i - \bar{m}_n(X_i)|^2 \leq \frac{1}{n}\sum_{i=1}^{n}|Y_i - g_n(X_i)|^2 + c_{23}\cdot\frac{(M_n+1)^d}{n}. \qquad (3.10)$$

Da g_n nach Definition deterministisch, damit also unabhängig von $(X_1,Y_1),(X_2,Y_2),\dots$ ist, sind mit $|\Theta_n| = 1$, $g_{n,1} = g_n$, der Abschätzung (3.10) für $\hat{m}_n = \mathscr{T}_{\beta_n}\bar{m}_n$ mit $\hat{m}_n \in \mathscr{F}^{(J_n)}$ und dem penalty Term $pen_n(g_{n,1}) = c_{23}\cdot\frac{(M_n+1)^d}{n} \geq 0$ die Voraussetzungen für Lemma 1.2.7 erfüllt und wir erhalten durch dessen Anwendung:

$$\begin{aligned}
&\mathbb{E}\int|\hat{m}_n(x) - m(x)|^2\mathbb{P}_X(dx) \\
&\leq \frac{c_{23}\cdot\log(n)^2\cdot\left(\log\left(\sup_{x_1^n\in(supp(X))^n}\mathscr{N}_1\left(\frac{1}{n\cdot\beta_n},\mathscr{F}^{(J_n)},x_1^n\right)\right)+1\right)}{n} \\
&\quad + 2\cdot\mathbb{E}\left(\min_{l\in\Theta_n}\int|g_{n,l}(x) - m(x)|^2\mathbb{P}_X(dx) + c_{23}\cdot\frac{(M_n+1)^d}{n}\right) \\
&= \frac{c_{23}\cdot\log(n)^2\cdot\left(\log\left(\sup_{x_1^n\in(supp(X))^n}\mathscr{N}_1\left(\frac{1}{n\cdot\beta_n},\mathscr{F}^{(J_n)},x_1^n\right)\right)+1\right)}{n} \\
&\quad + 2\int|g_n(x) - m(x)|^2\mathbb{P}_X(dx) + 2\cdot c_{23}\cdot\frac{(M_n+1)^d}{n},
\end{aligned}$$

wobei wir bei der letzten Gleichheit verwendet haben, dass der letzte Summand deterministisch ist. Zudem wissen wir, dass c_{23} auch unabhängig von n ist und zudem $n > 1$ gilt, da wir n hinreichend groß wählen. Als nächstes überprüfen wir die Voraussetzungen von Lemma 1.2.8 um damit dann die letzte Gleichung weiter abzuschätzen. Nach Voraussetzung ist $\beta_n = c_6\cdot\log(n)$ und $a = (\log n)^{1/(6(N+d))}$. Damit ist $a > 0$ für hinreichend großes

n. Nach Voraussetzung ist zudem $d, N, J_n \in \mathbb{N}$ und es gilt

$$J_n \leq (M_n+1)^d \cdot (N+1)^d \leq n^{c_{14}}$$

nach (3.1) für hinreichend großes n. Wir betrachten hier den logistischen Squasher welcher nach Lemma 1.2.1 insbesondere 2-zulässig nach Definition 1.1.3 ist. Da die hier betrachtete Menge von Funktionen $\mathscr{F}^{(J_n)}$ identisch mit der aus Lemma 1.2.8 ist, sind nun alle Voraussetzungen erfüllt. Da wir oBdA angenommen haben, dass $supp(X) \subseteq [-a,a]^d$ ist, erhalten wir mit Lemma 1.2.8 für hinreichend großes n:

$$\begin{aligned}
&\frac{c_{23} \cdot \log(n)^2 \cdot \left(\log\left(\sup_{x_1^n \in (supp(X))^n} \mathscr{N}_1\left(\frac{1}{n \cdot \beta_n}, \mathscr{F}^{(J_n)}, x_1^n\right)\right) + 1\right)}{n} \\
&\leq \frac{c_{23} \cdot \log(n)^2 \cdot \left(\left(c_{19} \cdot \log(n) \cdot (M_n+1)^d \cdot (N+1)^d\right) + 1\right)}{n} \\
&\leq \frac{c_{23} \cdot \log(n)^2 \cdot \left(2 \cdot c_{19} \cdot \log(n) \cdot (M_n+1)^d \cdot (N+1)^d\right)}{n} \\
&\leq c_{24} \cdot \frac{\log(n)^3 \cdot (M_n+1)^d \cdot (N+1)^d}{n}.
\end{aligned} \tag{3.11}$$

Sei

$$\begin{aligned}
P_n(x) = \sum_{k=1}^{(M_n+1)^d} \sum_{\substack{j_1,\dots,j_d \in \{0,\dots,q\} \\ j_1+\dots+j_d \leq q}} &\frac{1}{j_1! \cdots j_d!} \cdot \frac{\partial^{j_1+\dots+j_d} m}{\partial^{j_1} x^{(1)} \dots \partial^{j_d} x^{(d)}}(x_{\mathbf{i}_k}) \\
&\cdot (x^{(1)} - x_{\mathbf{i}_k}^{(1)})^{j_1} \cdots (x^{(d)} - x_{\mathbf{i}_k}^{(d)})^{j_d} \cdot \prod_{j=1}^{d} (1 - \frac{M_n}{2a} \cdot |x^{(j)} - x_{\mathbf{i}_k}^{(j)}|)_+.
\end{aligned}$$

Für zwei beliebige reelle Zahlen $u, v \in \mathbb{R}$ gilt durch $0 \leq (u-v)^2 = u^2 + v^2 - 2uv$:

$$v^2 + v^2 \geq 2uv$$

und zusammen mit einer Nulladdition, der Linearität des Integral und der Supremumseigenschaft erhalten wir:

$$\begin{aligned}
&\int |g_n(x) - m(x)|^2 \mathbb{P}_X(dx) \\
&= \int |g_n(x) - P_n(x) + P_n(x) - m(x)|^2 \mathbb{P}_X(dx) \\
&= \int |g_n(x) - P_n(x)|^2 + 2(g_n - P_n(x))(P_n(x) - m(x)) + |P_n(x) - m(x)|^2 \mathbb{P}_X(dx) \\
&\leq \int 2|g_n(x) - P_n(x)|^2 + 2|P_n(x) - m(x)|^2 \mathbb{P}_X(dx) \\
&= 2 \int \sup_{x \in [-a,a]^d} |g_n(x) - P_n(x)|^2 \mathbb{P}_X(dx) + 2 \int \sup_{x \in [-a,a]^d} |P_n(x) - m(x)|^2 \mathbb{P}_X(dx) \\
&= 2 \sup_{x \in [-a,a]^d} |g_n(x) - P_n(x)|^2 + 2 \sup_{x \in [-a,a]^d} |P_n(x) - m(x)|^2,
\end{aligned} \tag{3.12}$$

wobei wir im letzten Schritt $supp(X) \subseteq [-a_n, a_n]^d$ und $\mathbb{P}(X \in supp(\mathbb{P}_X)) = 1$ verwendet haben. Um die letzten beiden Summanden weiterhin abzuschätzen möchten wir Lemma 1.2.6 anwenden. Dafür überprüfen wir ob dafür alle Voraussetzungen erfüllt sind. Wir betrachten den logistischen squasher, welcher nach Lemma 1.2.3 insbesondere 2-zulässig ist. Zudem ist für hinreichend großes n die Bedingung für R erfüllt und da unser neuronales Netz (2.7) mit $x_{\mathbf{i}_k} \in [-a_n, a_n]^d$ identisch mit der Definition aus Lemma 1.2.6 ist, sind alle Voraussetzungen erfüllt. Wir erhalten damit für $x \in [-a_n, a_n]^d$ und n hinreichend groß:

$$
\begin{aligned}
&|g_n(x) - P_n(x)| \\
&= \left| \sum_{k=1}^{(M_n+1)^d} \sum_{\substack{j_1,\dots,j_d \in \{0,\dots,q\} \\ j_1+\dots+j_d \leq q}} \frac{1}{j_1! \cdots j_d!} \cdot \frac{\partial^{j_1+\dots+j_d} m}{\partial^{j_1} x^{(1)} \dots \partial^{j_d} x^{(d)}}(x_{\mathbf{i}_k}) \right| \\
&\cdot \left| f_{net,j_1,\dots,j_d,\mathbf{i}_k}(x) - (x^{(1)} - x_{\mathbf{i}_k}^{(1)})^{j_1} \cdots (x^{(d)} - x_{\mathbf{i}_k}^{(d)})^{j_d} \cdot \prod_{j=1}^{d} (1 - \frac{M_n}{2a} \cdot |x^{(j)} - x_{\mathbf{i}_k}^{(j)}|)_+ \right| \\
&\leq (M_n+1)^d \cdot (q+1)^d \cdot e \\
&\cdot \left| f_{net,j_1,\dots,j_d,\mathbf{i}_k}(x) - (x^{(1)} - x_{\mathbf{i}_k}^{(1)})^{j_1} \cdots (x^{(d)} - x_{\mathbf{i}_k}^{(d)})^{j_d} \cdot \prod_{j=1}^{d} (1 - \frac{M_n}{2a} \cdot |x^{(j)} - x_{\mathbf{i}_k}^{(j)}|)_+ \right| \\
&\leq (M_n+1)^d \cdot (q+1)^d \cdot e \cdot \hat{c}_{12} \cdot 3^{3 \cdot 3^s} \cdot a_n^{3 \cdot 2^s} \cdot M_n^3 \cdot \frac{1}{R_n} \\
&\leq (M_n+1)^d \cdot (q+1)^d \cdot c_{25} \cdot a_n^{3 \cdot (N+d) \cdot 2} \cdot \frac{M_n^3}{R_n} \\
&= (M_n+1)^d \cdot (q+1)^d \cdot c_{25} \cdot \log(n) \cdot \frac{M_n^3}{R_n},
\end{aligned}
\tag{3.13}
$$

wobei wir unter anderem verwendet haben, dass für hinreichend großes n:

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

gilt. Im letzten Schritt haben wir dann noch die Definition von a_n eingesetzt. Da nach Konstruktion $a > 0$, m (p,C)-glatt und $P_n(x)$ nach Lemma 2.1.1 eine lokale Konvexkombination von Taylorpolynomen von m, erhalten wir mit Lemma 2.1.2:

$$
|P_n(x) - m(x)| \leq c_{26} \cdot \frac{a_n^p}{M_n^p} \leq c_{26} \cdot \log(n) \cdot \frac{1}{M_n^p}, \tag{3.14}
$$

wobei wir verwendet haben, dass für $p = q + s$ für hinreichend großes n gilt:

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

da nach Voraussetzung $N \geq q$ und $d \geq s$ mit $s \in (0,1]$. Da die rechten Seiten von Ungleichung (3.13) und (3.14) nicht von x abhängen, gelten die Ungleichung ebenfalls für das Supremum. Durch Quadrieren bleiben die Ungleichungen auch erhalten und durch einsetzen der Definitionen von M_n und R_n erhalten wir für n hinreichend groß:

$$
\begin{aligned}
\sup_{x\in[-a_n,a_n]^d} |g_n(x)-P_n(x)|^2 &\leq \left((M_n+1)^d\cdot(q+1)^d\cdot c_{25}\cdot\log(n)\cdot\frac{M_n^3}{R_n}\right)^2 \\
&\leq c_{25}^2\cdot(M_n+1)^{2d}\cdot\log(n)^2\cdot\frac{M_n^6}{R_n^2} \\
&\leq c_{25}^2\cdot(M_n+1)^{2d}\cdot\log(n)^2\cdot\frac{(M_n+1)^{6d}}{R_n^2} \\
&\leq c_{25}^2\cdot\log(n)^2\cdot\frac{(M_n+1)^{8d}}{R_n^2} \\
&\leq c_{25}^2\cdot\frac{n^{\frac{8d}{2p+d}}}{n^{2d+8}}\cdot\log(n)^2 \\
&= c_{25}^2\cdot n^{\frac{8d}{2p+d}-2d-8}\cdot\log(n)^2 \\
&\leq c_{25}^2\cdot n^{\frac{8d}{2p+d}-8\frac{2p+d}{2p+d}}\cdot\log(n)^2 \\
&= c_{25}^2\cdot n^{-\frac{16p}{2p+d}}\cdot\log(n)^2 \\
&\leq c_{25}^2\cdot n^{-\frac{2p}{2p+d}}\cdot\log(n^3,
\end{aligned}
\tag{3.15}
$$

wobei wir bei der letzten Ungleichung verwendet haben, dass $\frac{16p}{2p+d} > \frac{2p}{2p+d}$, da $p > 0$ ist und dass $\log(n)^2 < \log(n)^3$ für n hinreichend groß. Ebenfalls erhalten wir:

$$
\begin{aligned}
\sup_{x\in[-a_n,a_n]^d} |P_n(x)-m(x)|^2 &\leq \left(c_{26}\cdot\log(n)\cdot\frac{1}{M_n^p}\right)^2 \\
&\leq c_{26}^2\cdot\log(n)^2\cdot c_5^{-2p}\cdot n^{-\frac{2p}{2p+d}} \\
&\leq (c_{16}^2\cdot c_5^{-2p})\cdot\log(n)^3\cdot n^{-\frac{2p}{2p+d}}.
\end{aligned}
\tag{3.16}
$$

Mit analogem Vorgehen erhalten wir für (3.11):

$$
\begin{aligned}
&\frac{c_{23}\cdot\log(n)^2\cdot\left(\log\left(\sup_{x_1^n\in(supp(X))^n}\mathcal{N}_1\left(\frac{1}{n\cdot\beta_n},\mathscr{F}^{(J_n)},x_1^n\right)\right)+1\right)}{n} \\
&\leq c_{24}\cdot\frac{\log(n)^3\cdot(M_n+1)^d\cdot(N+1)^d}{n} \\
&\leq \hat{c}_{24}\cdot\log(n)^3\cdot\frac{(c_5\cdot n^{\frac{1}{2p+d}}+2)^d}{n} \\
&\leq \hat{c}_{24}\cdot\log(n)^3\cdot\frac{c_5^d\cdot n^{\frac{d}{2p+d}}}{n} \\
&= \tilde{c}_{24}\cdot\log(n)^3\cdot n^{\frac{d}{2p+d}-1} \\
&= \tilde{c}_{24}\cdot\log(n)^3\cdot n^{-\frac{2p}{2p+d}}.
\end{aligned}
\tag{3.17}
$$

Zudem erhalten wir, da für n hinreichend groß $\log(n)^3 > 1$ gilt, mit analogem Vorgehen:

$$\begin{aligned} c_{21} \cdot \frac{(M_n+1)^d}{n} &\leq \tilde{c}_{21} \cdot \frac{n^{\frac{d}{2p+d}}}{n} \\ &\leq \tilde{c}_{21} \cdot \log(n)^3 \cdot n^{-\frac{2p}{2p+d}} \end{aligned}$$

und

$$\begin{aligned} \frac{4 \cdot c_{22} \cdot \beta_n^2}{n} &\leq \tilde{c}_{22} \cdot \log(n)^3 \cdot n^{-1} \\ &\leq \tilde{c}_{22} \cdot \log(n)^3 \cdot n^{-\frac{2p}{2p+d}}. \end{aligned}$$

$\square$

Kapitel 4

Anwendungsbeispiel auf Simulierte Daten

Literaturverzeichnis

[AHM09] ALBRECHER H., BINDER, A. und P. MAYER: *Einführung in die Finanzmathematik*. Birkhäuser, Basel, 2009.

[Car96] CARRIERE, J.: *Valuation of the early-exercise price for options using simulations and nonparametric regression*. Insurance: mathematics and Economics, 19(1):19–30, 1996.

[EKT+07] EGLOFF, D., M. KOHLER, N. TODOROVIC et al.: *A dynamic look-ahead Monte Carlo algorithm for pricing Bermudan options*. The Annals of Applied Probability, 17(4):1138–1171, 2007.

[Gla13] GLASSERMAN, P.: *Monte Carlo methods in financial engineering*, Band 53 der Reihe *Stochastic Modelling and Applied Probability*. Springer, New York, 2013.

[Kle13] KLENKE, A.: *Wahrscheinlichkeitstheorie*. Springer, Berlin, 2013.

[Koh10] KOHLER, M.: *A review on regression-based Monte Carlo methods for pricing American options*. In: DEVROYE L., KARASÖZEN, B. KOHLER M. und R. KORN (Herausgeber): *Recent Developments in Applied Probability and Statistics*, Seiten 37–58. Physica, Heidelberg, 2010.

[KS98] KARATZAS, I. und S. SHREVE: *Methods of mathematical finance*, Band 39 der Reihe *Stochastic Modelling and Applied Probability*. Springer, New York, 1998.

[LS01] LONGSTAFF, F. und E. SCHWARTZ: *Valuing American options by simulation: a simple least-squares approach*. The review of financial studies, 14(1):113–147, 2001.

[Mal00] MALKIEL, B.: *Börsenerfolg ist kein Zufall - die besten Investmentstrategien für das neue Jahrtausend*. FinanzBuch, München, 2000.

[R D17] R DEVELOPMENT CORE TEAM: *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, 2017.

[SB07] STOER, J. und R. BULIRSCH: *Numerische Mathematik 1*. Springer, Berlin, 2007.

[TVR99] TSITSIKLIS, J. N. und B. VAN ROY: *Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives*. IEEE Transactions on Automatic Control, 44(10):1840–1851, 1999.

Appendix

Der Programmcode ist wie folgt aufgebaut:

- `main.py` TBD.
- `data_gen.py` TBD.
- `help_neural_networks.py` fasst alle Hilfsfunktion zusammen.
- `new_neural_network.py` TBD.
- `fc_neural_network.py` TBD.
- `nearest_neighbor.py` TBD.
- `constant.py` TBD.

Listing 4.1: `main.py`

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 23 15:08:26 2019

@author: adrian

Main Datei die die Simulation und damit den Vergleich der implementierten Schätzer
durchführt.
"""
import numpy as np
from mpl_toolkits import mplot3d
import matplotlib . pyplot as plt
import pandas as pd
from scipy.stats import iqr
from data_gen import gen_data_Y, error_limit
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

'''
EINDIMENSIONALER FALL (d = 1) wird geplottet
'''
N = 3
q = 2
R = 10 ** 6
a = 2
M = 2
d = 1
```

```

sigma = 0.05

n_train = 100
n_test = 2000
# Parameter für unseren neuen Neuronale-Netze-Regressionschätzer
X_train = np.random.uniform(low=-2,high=2,size=(int(n_train),d))
m_X_train, Y_train = gen_data_Y(X_train,sigma)

X_test = np.random.uniform(low=-2,high=2,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, dummy = gen_data_Y(X_test,sigma)


#plt.plot(X_test, Y_pred_nearest_neighbor, '-r', label='nearest_neigbhor')
#plt.plot(X_test, Y_pred_fc_nn_1, '-g', label='fc_nn_1')
#colors = (0,0,0)
area = 4
plt.scatter(X_test, Y_pred_new_nn, s=area, color = 'red', alpha=0.5)
plt.title('Scatter plot pythonspot.com')
plt.xlabel('x')
plt.ylabel('y')
plt.show()
#plt.plot(X_test, Y_pred_new_nn, 'ro-', label='new_nn')
#plt.plot(X_test, m_X_test, '-b', label='m_d')
#plt.legend(loc='upper left')
#plt.xlim(-2, 2)
#plt.xlim(-2,2)
#plt.show()
#plt.savefig('foo.png')

'''
ZEIDIMENSIONALER FALL (d = 2) wird geplottet
'''
N = 2
q = 2
R = 10 ** 6
a = 2
M = 2
d = 2

sigma = 0.05

n_train = 100
n_test = 2000
# Parameter für unseren neuen Neuronale-Netze-Regressionschätzer
X_train = np.random.uniform(low=-2,high=2,size=(int(n_train),d))
m_X_train, Y_train = gen_data_Y(X_train,sigma)

X_test = np.random.uniform(low=-2,high=2,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, dummy = gen_data_Y(X_test,sigma)



x = np.ravel(X_test[:,0])
y = np.ravel(X_test[:,1])

# so wie es sein soll
#z = m_X_test[:,0]
# was der SChätzer auswirft
z = Y_pred_new_nn[:,0]

ax = plt.axes(projection='3d')
ax.scatter(x, y, z, c=z, cmap='viridis', linewidth=0.5);
ax.view_init(40, 20)
fig

'''
```

```
ein Vergleich des emp. L2 Fehler gemacht für d = 1
'''
n_train = 100
n_test = 2000
# Parameter für unseren neuen Neuronale-Netze-Regressionschätzer

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

sigma = 0.05

scaled_error = np.empty((5, 3,))
scaled_error[:] = np.nan

e_L2_avg = np.zeros(5)
e_L2_avg[:] = np.nan

for i in range(0,5,1):

    X_train = np.sort(np.random.uniform(low=-2,high=2,size=(int(n_train),d)), axis = 0)
    m_X_train, Y_train = gen_data_Y(X_train,sigma)

    X_test = np.sort(np.random.uniform(low=-2,high=2,size=(int(n_test),d)), axis = 0)

    #Y_pred_constant = constant_estimate(Y_train)
    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(sum (abs(Y_pred_new_nn - m_X_test) ** 2))
    e_L2_fc_nn_1 = np.mean(sum (abs(Y_pred_fc_nn_1 - m_X_test) ** 2))
    e_L2_nearest_neighbor = np.mean(sum (abs(Y_pred_nearest_neighbor - m_X_test) ** 2))

    for j in range(0,5,1):

        X = np.sort(np.random.uniform(low=-2,high=2,size=(n_test,d)), axis = 0)
        m_X, Y = gen_data_Y(X,sigma)
        Y_pred_constant = constant_estimate(Y)

        e_L2_avg[j] = np.mean(sum(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"]

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 = ""
#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 = ""

error_df = pd.concat([series_noise_1], axis=1)
print(error_df)
#error_df.to_csv('out.csv',index = True)

'''
ein Vergleich des emp. L2 Fehler gemacht für d = 2
'''
```

```
n_train = 100
n_test = 2000
# Parameter für unseren neuen Neuronale−Netze−Regressionschätzer

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

sigma = 0.05

scaled_error = np.empty((5, 3,))
scaled_error[:] = np.nan

e_L2_avg = np.zeros(5)
e_L2_avg[:] = np.nan

for i in range(0,5,1):

    X_train = np.sort(np.random.uniform(low=−2,high=2,size=(int(n_train),d)), axis = 0)
    m_X_train, Y_train = gen_data_Y(X_train,sigma)

    X_test = np.sort(np.random.uniform(low=−2,high=2,size=(int(n_test),d)), axis = 0)

    #Y_pred_constant = constant_estimate(Y_train)
    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(sum (abs(Y_pred_new_nn − m_X_test) ** 2))
    e_L2_fc_nn_1 = np.mean(sum (abs(Y_pred_fc_nn_1 − m_X_test) ** 2))
    e_L2_nearest_neighbor = np.mean(sum (abs(Y_pred_nearest_neighbor − m_X_test) ** 2))

    for j in range(0,5,1):

        X = np.sort(np.random.uniform(low=−2,high=2,size=(n_test,d)), axis = 0)
        m_X, Y = gen_data_Y(X,sigma)
        Y_pred_constant = constant_estimate(Y)

        e_L2_avg[j] = np.mean(sum(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"]

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 = ""
#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 = ""

error_df = pd.concat([series_noise_1], axis=1)
print(error_df)
#error_df.to_csv('out.csv',index = True)
```

Listing 4.2: `data_gen.py`

```
#!/usr/bin/env python3
# −*− coding: utf−8 −*−
```

```
"""
Created on Fri Oct 11 12:01:42 2019

@author: adrian

Generieren der Daten die wir für einen Vergleich von Regressionsschätzern benötigen
"""
# Wir wählen x gleichverteilt auf [-1,1]^d, wobei d die dimension des Inputs ist
# n is die Größe der Stichprobe

import numpy as np
from scipy.stats import iqr

# Regressionsfunktionen
#
# x: Ein Vektor x \in [-1,-1]^d
# d: Dimension des Vektors x

def m_d (x, d):

    pi = np.pi
    cos = np.cos
    sin = np.sin
    exp = np.exp

    if d == 1:
        return sin(0.2 * x[0] ** 2) + exp(0.5 * x[0]) + x[0] ** 3

    elif d == 2:
        return np.sin(np.sqrt(x[0] ** 2 + x[1] ** 2))

    else:
        print("Your data has the wrong dimension!")

def error_limit (x, p, c, d):

        return c * (np.log(x) ** 3) * (x ** (-(2 * p)/(2 * p + d)))

# Generiert den Vektor Y_1,...,Y_n für den Datensatz (X_1,Y_1),...,(X_n,Y_n)
#
# X: Inputdaten der Form (X_1,...,X_n), wobei X_i \in [-1,-1]^d für i = 1,...,n
# sigma: Schwankung in den Werten (Noise) \in \{0.05,0.1\}

def gen_data_Y (X, sigma):

    n = np.size(X, 0)
    d = np.size(X, 1)

    m_X = np.zeros((n,1,))
    m_X[:] = np.nan

    S = np.random.standard_normal(size=(n,1))
    for t in range(0,n):
        m_X[t] = m_d(X[t], d)

    Y = m_X + sigma * iqr(m_X) * S
    return (m_X, Y)
```

Listing 4.3: `help_neural_networks.py`

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Tue Oct 15 11:22:02 2019

@author: adrian

Implementation von Neuronalen-Netzen welche wir für die Konstruktion unseres
Neuronale-Netze-Regressionschätzers benötigen
"""
import numpy as np

# Sigmoidfunktion
#
```

```
# x: x \in \R

def sigmoid (x):

    return 1 / (1 + np.exp(-x))

# Neuronales Netz welches die Funktion f(x) = x approximiert
#
# x: reelle Zahl
# R: reelle Zahl >= 1

def f_id (x, R):

    return 4 * R * sigmoid(x / R) - 2 * R

# Neuronales Netz welches die Funktion f(x, y) = x * y approximiert
#
# x: reelle Zahl
# y: reelle Zahl
# R: reelle Zahl >= 1

def f_mult (x, y, R):

    return (((R ** 2) / 4) * (((1 + np.exp(-1)) ** 3) / (np.exp(-2) - np.exp(-1)))) \
    * (sigmoid(((2 * (x + y)) / R) + 1) - 2 * sigmoid(((x + y) / R) + 1) \
    - sigmoid(((2 * (x - y)) / R) + 1) + 2 * sigmoid(((x - y) / R) + 1))

# Neuronales Netz welches die Funktion f(x) = max(x,0) approximiert
#
# x: reelle Zahl
# R: reelle Zahl >= 1

def f_relu (x, R):

    return f_mult(f_id(x, R), sigmoid(R * x),R)

# Neuronales Netz welches die Funktion f(x) = max(1 - (M/(2 * a)) * abs(x - y),0) approximiert
#
# x: reelle Zahl
# y: fixe reelle Zahl
# R: reelle Zahl >= 1
# M: fixe natürliche Zahl
# a: fixe Zahl > 0

def f_hat (x, y, R, M, a):

    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 4.4: `new_neural_network.py`

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 16 15:40:14 2019

@author: adrian

Um die Gewichte der Ausgabeschicht zu bestimmen lösen wir ein regularisiertes
Kleinste-Quadrate Problem.

"""
import scipy.special
import numpy as np
import itertools
from help_neural_networks import f_id, f_mult, f_hat
import math

# Neuronales Netz welches die Funktion f(x) = (x^(1)- x_ik^(1))^j1 * ... *
# (x^(d) - x_ik^(d))^jd * \prod_{j = 1}^d max((1 - (M/2a) * abs(x^(j) - x_ik^(j))),0)
#
# x: Eingabevektor für das Neuronale Netz x \in [-a,a]^d
# d: Ist die Dimension des Eingabevektors d > 0
# j_1_d: Ist ein d-dimensionaler Vektor j_1,...,j_d \in {0,1,...,N}
```

```
# X_i: Ist eine d x (M+1)^d Matrix.
# N: Natürliche Zahl >= q
# q:
# s: [log_2(N + d)]
# 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
    #(2 ** s) + 1
    #((1 + M) ** d) + 1
    f_l_k = np.empty((s + 1, (2 ** s) + 1,))
    f_l_k[:] = np.nan


    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 lösen eines regularisierten
# Kleineste-Quadrate Problems
#
# X: Eingabevektoren der Form (X_1,...,X_n) für das Neuronale Netz aus dem Datensatz (X_1,Y_1),...,(X_n,Y_n)
# Y: Eingabevektoren der Form (Y_1,...,Y_n) für das Neuronale Netz aus dem Datensatz (X_1,Y_1),...,(X_n,Y_n)
# N: Natürliche 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 constante > 0

    #c_3 = np.random.randint(1,10)
    c_3 = 0.01

    # Anzahl der Spalten der Matrix für das Kleinste-Quadrate Problem
    # In den Spalten sind die Funktionswerte von f_net eingespeichert

    J = int(((1 + M) ** d) * scipy.special.binom(N + d, d))

    # Für die Konstruktion der Matrix brauchen wir erstmal alle Inputparameter
    # für f_net, da wir dort nur den Funktionswert für einen Vektor j_1,...,j_d einsetzen
    # müssen wir erstmals alle möglichen Vektoren dieser Art konstruieren die die Bedingung 0 <= j_1 + ... + j_d <= N
        erfüllen
    # X_ik hat in den Zeilen die Vektoren X_i aus dem Paper

    X_ik = np.transpose(np.empty((d, (1 + M) ** d,)))
    X_ik[:] = np.nan

    I_k = np.array(list(itertools.product(range(0, M + 1), repeat = d)))
    X_ik[:] = (I_k[:] * ((2 * a) / M)) - a
```

```

    all_j1_jd = np.array(list(itertools.product(range(0, N + 1), repeat = d)))
    all_j1_jd_by_cond = all_j1_jd[all_j1_jd.sum(axis=1) <= N]

    B = np.empty((n, J,))
    B[:] = np.nan

    for i in range(0, n):
        j = 0
        for k in range(0, ((M + 1) ** d)):
            for z in range(0, int(scipy.special.binom(N + d, d))):
                B[i,j] = f_net(X[i], d, all_j1_jd_by_cond[z], X_ik[k], N, q, s, R, M, a)
                j += 1

    #weights = np.linalg.solve((1 / n) * np.dot(np.transpose(B),B) + (c_3 / n) * np.identity(J), (1 / n) * np.dot(np.
        transpose(B),Y))
    weights = np.linalg.solve(np.dot(np.transpose(B),B) + (c_3) * np.identity(J), np.dot(np.transpose(B),Y))

    return (weights, J, all_j1_jd_by_cond, X_ik)

# Bestimmung des Funktionswert des Neuronale-Netzte-Regressionsschätzers
#
# x: Eingabe für einen Vektor der Form [-a,a]^d für den eine Schätzung bestimmt werden soll
# X: Eingabevektoren der Form (X_1,...,X_n) für das Neuronale Netz aus dem Datensatz (X_1,Y_1),...,(X_n,Y_n)
# Y: Eingabevektoren der Form (Y_1,...,Y_n) für das Neuronale Netz aus dem Datensatz (X_1,Y_1),...,(X_n,Y_n)
# N: Natürliche Zahl >= q
# q:
# s: [log_2(N + d)]
# R: Zahl >= 1
# d: Ist die Dimension des Eingabevektors d > 0
# M: M \in \N
# a: >0

def new_neural_network_estimate(X_train, Y_train, X_test, N, q, R, d, M, a):

    Y_pred = np.empty((len(X_test), 1,))
    Y_pred[:] = np.nan

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

    weights, J, all_j1_jd_by_cond, X_ik = output_weights(X_train, Y_train, N, q, R, d, M, a)

    F_net = np.empty((1, J,))
    F_net[:] = np.nan

    for u in range (0,len(X_test),1):
        j = 0
        while j < J:
            for k in range(0, ((M + 1) ** d)):
                for z in range(0, int(scipy.special.binom(N + d, d))):
                    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)
                    j += 1

        Y_pred[u] = np.sum(np.transpose(weights) * F_net)

    return Y_pred
```

Listing 4.5: `fc_neural_network.py`

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Oct 11 14:23:15 2019

@author: adrian
"""
import numpy as np
from keras.models import Sequential
from keras.layers import Dense

# Fully-Connected Neuronales Netzt mit einer Verborgenen schicht welches die
# Anzahl der Neuronen adaptiv, durch minimierung des L2 fehlers, aus der Menge \{5, 10, 25, 50, 75\} auswählt.
#
# X: Eingabevektoren der Form (X_1,...,X_n) für das Neuronale Netz aus dem Datensatz (X_1,Y_1),...,(X_n,Y_n)
```

```python
# Y: Eingabevektoren der Form (Y_1,...,Y_n) für das Neuronale Netz aus dem Datensatz (X_1,Y_1),...,(X_n,Y_n)

def fc_neural_1_estimate (X_train,Y_train,X_test):

    Ynew = np.empty((len(X_train), len([5,10,25,50,75]),))
    Ynew[:] = np.nan

    count = 0
    n_neurons = [5,10,25,50,75]

    d = np.size(X_train, 1)

    for j in n_neurons:
        model = Sequential()
        model.add(Dense(j, input_dim=d, activation='relu'))
        model.add(Dense(1, activation='linear'))
        model.compile(loss='mse', optimizer='adam')
        model.fit(X_train, Y_train, epochs=1000, verbose=0)

        Ynew[:,count] = model.predict(X_train)[:,0]
        count += 1

    Diff = Ynew[:] - Y_train[:]
    best_n_neurons = n_neurons[(1/len(X_train) *(Diff.sum(axis=0) ** 2)).argmax()]

    model = Sequential()
    model.add(Dense(best_n_neurons, input_dim=d, activation='relu'))
    model.add(Dense(1, activation='linear'))
    model.compile(loss='mse', optimizer='adam')
    model.fit(X_train, Y_train, epochs=1000, verbose=1)

    return model.predict(X_test)
```

Listing 4.6: `nearest_neighbor.py`

```python
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Fri Oct 11 11:14:47 2019

@author: adrian
K-Nearest-Neighbors Algorithm
"""

# Generate sample data
import numpy as np
from sklearn import neighbors
from sklearn.model_selection import GridSearchCV
import warnings

warnings.simplefilter(action='ignore', category=FutureWarning)

# Implementierung des k-Nächste-Nachbarn-Algorithmus. Dieser bestimmt auch selber bei einer Liste von Anzahlen an Nachbarn die betrachtet werden
# sollen welches die beste Wahl ist.
#
# X: Inputvektor für das Kalibirieren des Modells
# Y: Inputvektor für das Kalibirieren des Modells (Zielvektor an den die Gewichte angepasst werden)
# T: Inputvektor für den eine Vorhersage bestimmte werden soll

def nearest_neighbor_estimate (X_train,Y_train,X_test):

    params = {'n_neighbors':[2,3,4,5,6,7,8,9], 'weights': ['uniform', 'distance']}

    knn = neighbors.KNeighborsRegressor()

    knn_gridsearch_model = GridSearchCV(knn, params, cv=5)
    knn_gridsearch_model.fit(X_train,Y_train)

    return knn_gridsearch_model.predict(X_test)
```

Listing 4.7: `constant.py`

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Wed Oct 23 15:26:19 2019

@author: adrian
Constant Estimator
"""
import numpy as np
from scipy import mean

# Gibt den Mittelwert der Funktionswerte einer Funktion als Schätzer zurück
#
# Y: Datensatz der Form (Y_1,...) wobei Y_i \in \R für i = 1,...

def constant_estimate(Y):
    m = np.zeros((len(Y),1,))
    m[:] = mean(Y)
    return m
```