JUSTUS-LIEBIG-
UNIVERSITÄT
GIESSEN

Justus-Liebig-Universität Gießen

Mathematisches Institut

Neuronale Netzwerke und Anwendungen von Ridge-Funktionen

Dissertation

zur Erlangung des Doktorgrades
der Naturwissenschaftlichen Fachbereiche
der Justus-Liebig-Universität Gießen

vorgelegt von

Philipp Johannes Junk

März 2023

Betreuer: Prof. Dr. Martin Buhmann

Für meine geliebten Eltern.

Danksagung

Vielen Dank an meinen Betreuer Herrn Prof. Martin Buhmann für die zahlreichen fachkundigen Hilfen und Anleitungen zu meiner Arbeit. Besonders für die konstruktive Kritik und die Geduld, die mir geholfen haben, diese Arbeit fertigzustellen.
Ganz besonderen Dank für meine lieben Kollegen aus dem Mathematischen Institut für zahlreiche hilfreiche Anmerkungen und positiven Erfahrungen und Erinnerungen aus der gemeinsamen Arbeit.
Für die Kollegen Constantin Greif, Prof. Dr. Karsten Urban und Hendrik Kleikamp für die gemeinsame Forschungsarbeit zu Ridge-Funktionen.
Meiner Familie für die immerwährende Unterstützung.
Der d-fine GmbH für die Ermöglichung meiner Disputation.

Inhaltsverzeichnis

Zusammenfassung

Diese Arbeit beschäftigt sich mit der Approximationstheorie von Neuronalen Netzwerken und einer Methode zur nichtlinearen Modellreduktion mit Ridge-Funktionen.
Neuronale Netzwerke sind als Teil künstlicher Intelligenz in der modernen Forschung und Anwendung von kaum zu überschätzender Wichtigkeit. Die Forschung zum mathematischen Verständnis dieser Strukturen ist jedoch erst in den letzten Jahren gewachsen.
Zunächst untersuchen wir im ersten Kapitel die Approximation mit klassischen Feedforward Netzwerken und dabei insbesondere solche Netzwerke mit stückweise polynomiellen Aktivierungsfunktion, für welche wir einen engen Zusammenhang zu Splines herstellen. Über diese Verbindung lassen sich verschiedene Aproximationsraten herleiten und die Vorteile von mehrschichtigen Netzwerken dieser Art erklären.
Im nächsten Kapitel betrachten wir den Netzwerktyp der Convolutional Neural Networks und untersuchen diese durch die Theorie der Scattering Networks. Dabei untersuchen wir eine Verallgemeinerung auf diskrete Frames, mithilfe deren wir Stabilitätsaussagen treffen, auch anhand numerischer Beispiele.
Zuletzt nutzen wir Ridge-Funktionen, um aufbauend auf einer Reduced Basis Method eine nichtlineare Methode zur Lösung von parametrischen partiellen Differentialgleichungen im Rahmen der nichtlinearen Modellreduktion zu entwickeln und anhand umfangreicher numerischer Beispiele zu evaluieren. Wir geben konkrete Algorithmen zur Lösung der Probleme der Rekonstruktion von Ridge-Funktionen auf gewissen Gebieten an und wenden diese auf praktische Beispiele an. Außerdem betrachten im Rahmen der theoretischen Untersuchung dieser Methode Fragestellungen zur Approximation mit Ridge-Funktionen.

Abstract

Neural Networks have become ubiquitous in many applications, yet the effectiveness of their application is still not fully understood. This thesis will discuss several topics in approximations by neural networks of different architectures and develop a new method for nonlinear model reduction with ridge functions.

In the first chapter we will review the theory of classical feedforward networks with special regard to piecewise polynomial activation functions and show its connection to the theory of splines.

Next we consider as a special and modern type of networks - the convolutional neural networks and espescially the theory of scattering networks, which we generalize to certain discrete frames. We discuss, how the properties of these frames can be maintained under optimization of the underlying network. We conclude this chapter by numerical examples to this theory.

In the last chapter we formulate a new nonlinear method for solving parametrical problems such as parametric partial differential equations using ridge functions. Thereby we investigate questions of approximation and recovery of ridge functions on certain domains and provide algorithms for solving these problems. We provide numerous computational examples and discuss further possibilities for optimizing the method in the context of reduced basis methods and other model reduction approaches.

Abbildungsverzeichnis

Tabellenverzeichnis

Einleitung

Die Entwicklung von künstlichen Neuronalen Netzwerken hat unseren Alltag und den Umgang mit komplexen Problemen und umfangreichen Datenmengen revolutioniert und ist eine Triebfeder vieler moderner Anwendungen die unter Schlagworten wie künstlicher Intelligenz (AI), Maschinelles Lernen oder Deep Learning ihren Weg in Handys, Autos und viele weitere alltägliche Gegenstände gefunden hat. Seit der Entwicklung erster digitaler Computer in den 1940er Jahren war es Forschungsgegenstand, die Fähigkeiten menschlicher Intelligenz auf diese Rechenmaschinen zu übertragen. Als Pioniere in dieser Entwicklung sind McCulloch und Pitts [54] sowie Hebb [37] zu nennen, welche mit ihrem Modell eines künstlichen Neurons und zugehörigen Lernregeln die Grundlagen für das Feld der Neuroinformatik schufen und das erste mathematische Modell des menschlichen Gehirns. Erste erfolgreiche Anwendungen folgten in den 1950er Jahren mit der Entwicklung des Perzeptrons, eines Neuronalen Netzwerkmodells mit zugehöriger Hardware, durch Rosenblatt [73], welches in der Lage war einfache Bilder wie Ziffern zu klassifizieren. Nach verschiedenen Rückschlägen mit den zu der Zeit verwendeten einschichtigen Netzen entwickelten sich ab den 1980er Jahren weitere Varianten wie die Convolutional Neural Networks [48] (siehe Kapitel 2) und der Backpropagation-Algorithmus erlaubte es, auch mehrschichtige Netzwerke effektiv zu trainieren. Diese Erkenntnisse, zusammen mit der schnell wachsenden Rechenleistung, führten zu einem größerem Fokus auf diese Netzwerke und ihre Theorie und Praxis.
Die mathematische Untersuchung zu den Grundlagen der Approximationstheorie von Neuronalen Netzen wurde gleichzeitig von Cybenko[18], Mhaskar[55], Maiorov, Pinkus[51][65] und weiteren vorangetrieben. Dabei standen ebenfalls zunächst einfache einschichtige Netzwerke im Fokus.
Der kommerzielle Erfolg der neuen Netzwerkarchitekturen bei Aufgaben wie Klassifikation von Bildern, Spracherkennung und -verarbeitung sowie Steuerung von komplexen Prozessen wie bspw. autonomes Fahren haben im Weiteren zu einer stetigen Beschleunigung der Forschung zu Anwendungen von verschiedenen, oftmals unter Deep Learning zusammengefassten, Netzwerken geführt. So sind beispielsweise allein aus dem Jahr 2020 mehr als 11000 Veröffentlichungen auf arxiv mit dem relevanten Themengebiet Machine Learning (cs.LG) verzeichnet. Gleichzeitig sind durch die Vielzahl der verschiedenen Architekturen und Methoden und deren schnelle Abfolge eine mathematisch rigorose Untersuchung der Vorteile und Eigenschaften dieser Netzwerke zu kurz gekommen. Dabei sind beweisbare Eigenschaf-

ten von solchen Netzwerken im Bezug auf kritische oder sicherheitsrelevante Anwendungen dringend gewünscht. Dabei sei zum Beispiel auf sog. adversarial attacks verwiesen, bei denen in tiefen Neuronalen Netzwerken zur Klassifikation von Bildern durch kleine, für den menschlichen Betrachter unsichtbare, Änderungen an der Eingabe die Klassifikationsergebnisse verfälscht werden. Da solche Änderungen beispielsweise bei Anwendungen wie selbstfahrenden Autos und der Erkennung von Passanten zu fatalen Unfällen führen können, besteht dort ein großer Bedarf an Netzwerken, die beweisbar resistent gegen solche Angriffe sind.
Weiterhin sind auch bei einfachen Neuronalen Netzwerken, die als Ridge-Funktionen in vielen Teilgebieten der Mathematik natürlicherweise auftreten, nicht alle Anwendungen erschöpft. Im letzten Kapitel 3 werden wir einschichtige Neuronale Netzwerke im Kontext der nichtlinearen Modellreduktion wiedersehen mit einer an die Lösung von partiellen Differentialgleichungen angepassten Terminologie. Insbesondere werden wir in diesem Zusammenhang Fragestellungen der Identifizierung und Rekonstruktion von Ridge-Funktionen untersuchen.

Notation und Konventionen

Wir führen zunächst einige Notationen und Konventionen ein. Im folgenden sei $\mathbb{N} = \{0, 1, 2, \dots\}$ die Menge der natürlichen Zahlen inklusive 0, $\mathbb{N}_+$ die Menge der positiven natürlichen Zahlen.
Die Kardinalität einer Menge X bezeichnen wir mit $|X|$. Die Rundungsfunktion sei $[x] = \lfloor x \rfloor = \max\{z \in \mathbb{Z} \mid x \geq z\}$. Sei $\mathbb{P}_d^n$ der Raum der Polynome in d Veränderlichen mit Totalgrad kleiner gleich n.
Die Variation oder totale Variation einer Funktion f auf einem Intervall A (oder auch $A = \mathbb{R}$) ist gegeben durch $\operatorname{Var}_A f := \sup_{(x_i)_{i=0}^n \subset A, n \in \mathbb{N}_+} \sum_{i=1}^n |f(x_i) - f(x_{i-1})|$. Ist $\operatorname{Var}_A f < \infty$, so nennen wir f von *beschränkter Variation*.
Für $x, y \in \mathbb{R}^d$ ist $\langle x, y \rangle = \sum_{i=1}^d x_i y_i$ das Standard-Skalarprodukt.
Sei $\Omega \subseteq \mathbb{R}^d$. Wir bezeichnen mit $\mathcal{C}^r(\Omega)$ den Raum der r-mal stetig differenzierbaren Funktionen auf Ω und für $1 \leq p < \infty$ mit $L_p(\Omega)$ den Raum der p-fach integrierbaren Funktionen mit Norm $\|f\|_{L_p(\Omega)} = \left(\int_\Omega |f(x)|^p \, dx\right)^{1/p}$. Weiter ist $\|f\|_\infty = \sup_{x \in \Omega} |f(x)|$.
Wir bezeichnen mit $\operatorname{Lip}(X)$ den Raum der Lipschitz-stetigen Funktionen auf X und mit $\operatorname{Lip}(\alpha, X)$ den Raum der α-Hölder-stetigen Funktionen.
Die Fourier-Transformation einer Funktion $f \in L_1(\mathbb{R}^d)$ sei definiert durch $\mathcal{F}_d f(\xi) = \int_{\mathbb{R}^d} f(x) e^{-2\pi i \langle x, \xi \rangle} \, dx$ und die inverse Transformation durch $\mathcal{F}_d^{-1} g(x) = \int_{\mathbb{R}^d} g(\xi) e^{2\pi i \langle x, \xi \rangle} \, d\xi$.
Mit $\mathbb{1}_A$ bezeichnen wir die Indikatorfunktion einer Menge A.
Wir bezeichnen sowohl die Architektur, gemeint ist die Struktur einer Abbildung, als Neuronales Netzwerk, wie auch die daraus abgeleiteten Funktionen mit gewählten Gewichten etc. Der Leser sei an dieser Stelle besonders darauf hingewiesen, auch wenn die Unterscheidung aus dem Kontext stets ersichtlich sein sollte.

Kapitel 1

Klassische Neuronale Netzwerke

Neuronale Netzwerke sind, wie wir bereits in der Einleitung zu dieser Arbeit sehen konnten, ein zentrales Mittel der modernen, datengestützten und vernetzten Welt. Dabei kommt eine Vielzahl unterschiedlicher Netzwerkarchitekturen und -modellen zum Einsatz. Die Untersuchung von Neuronalen Netzwerken in einem mathematischen und konkret approximationstheoretischen Kontext ist in der Vergangenheit stark auf sogenannte klassische „feed-forward" Netzwerke fokussiert gewesen. Wir möchten in diesem Kapitel zunächst solche klassischen Neuronalen Netzwerke definieren und bekannte Eigenschaften darstellen.
Anschließend betrachten wir insbesondere die Fragestellung, welchen Einfluss die Aktivierungsfunktionen als Teil eines Netzwerks auf dessen Eigenschaften im Bezug auf die Approximation wie Dichtheit in Räumen wie $\mathcal{C}(K)$ und Raten der Approximation in verschiedenen Räumen haben. Besonders untersuchen wir stückweise polynomielle Aktivierungsfunktionen und stellen einen Zusammenhang zur Theorie von Splines her, über welchen wir durch explizite Konstruktionen Approximanten an viele Funktionsklassen wie bspw. Polynome erhalten. Dieser Ansatz erlaubt es dann, einige der klassischen Theoreme über Neuronale Netzwerke auf neue Weise aus den Eigenschaften der Splines abzuleiten. Weiter interessieren wir uns ebenfalls für Fragen der Stabilität solcher Netzwerke, wofür wir Eigenschaften wie Nullstellen und die Geometrie der polynomiellen Gebiete untersuchen. Diese Fragestellung wird für das praktische Problem der Vermeidung von adversarial attacks, welche bereits in der Einleitung erwähnt wurden, benutzt. Zuletzt möchten wir weitere Ansätze zur Approximation wie den Satz von Kolmogorov-Arnold oder als Anwendung wie rationale Approximation mit Neuronalen Netzwerken betrachten.
Beginnen wir nun mit einer ersten Definition des Forschungsgegenstands. Verschiedene Ansätze und Modelle zur Behandlung von Netzwerken führen zu jeweils leicht unterschiedlichen Definitionen für Neuronale Netzwerke. Wir geben zunächst eine möglichst allgemeine Definition aus der Sicht der Approximationstheorie, die alle weiteren von uns betrachteten Fälle abdecken wird. Es sei aber darauf hingewiesen, dass in anderen Zusammenhängen weitere allgemeinere oder andere speziellere Definitionen zielführender sein können. Wir gehen in der folgenden Definition von einem in Schichten angeordnetem Netzwerk mit eindeutigem Informationsfluss aus. Dies umschließt damit zunächst keine rückgekoppelten

Modelle wie sogenannte „recurrent networks" oder ähnliche Ansätze. Die hier definierten Netzwerke bilden jedoch die klassische und am häufigsten verwendete Netzwerktopologie der MLP (multilayer perceptrons) wie auch Convolutional Networks (siehe Kapitel 2) ab.

Definition 1.1 Ein *neuronales Netzwerk* ist eine Funktion $\mathcal{N} : \mathbb{R}^d \to \mathbb{R}^m$ der Form

$$\mathcal{N} = \Sigma \circ f_N \circ \cdots \circ f_2 \circ f_1 \circ F.$$

Wir nennen die Funktionen $f_1, \ldots, f_1$ mit $f_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i+1}}$ die *Schichten* des Netzwerks und definieren die *Tiefe* des Netzwerks $\mathcal{N}$ als N und $L := \max_i n_i$ als die *Breite* eines Netzwerks. Weiter nennen wir $\Sigma : \mathbb{R}^{n_N} \to \mathbb{R}^m$ die *Ausgabe-* und $F : \mathbb{R}^d \to \mathbb{R}^{n_1}$ die *Eingabeschicht* des Netzwerkes $\mathcal{N}$.

$$\mathbb{R}^d \ni \mathbf{x} \longmapsto \boxed{F} \longrightarrow \boxed{f_1} \longrightarrow \cdots \longrightarrow \boxed{f_{N-1}} \longrightarrow \boxed{f_N} \longrightarrow \boxed{\Sigma}$$

Durch diese Definition ist eine natürliche Strukturierung eines neuronalen Netzwerks in Schichten, bestehend aus einer Eingabeschicht F, den inneren Schichten (hidden layers) entsprechend den Funktionen f_i und der Ausgabeschicht Σ, gegeben. Die speziellen Eingabe- und Ausgabeschichten werden je nach Anwendungsfall gewählt. Zunächst möchten wir nun Netzwerke des klassischen MLP-Modells (engl. für multilayer perceptron) definieren. Dazu benötigen wir Ridge-Funktionen:

Definition 1.2 Eine Funktion $f : \mathbb{R}^N \to \mathbb{R}$ der Form

$$f(x) = \sigma(\langle a, x \rangle + b)$$

mit $\sigma : \mathbb{R} \to \mathbb{R}$, $a \in \mathbb{R}^N, b \in \mathbb{R}$, heißt *Ridge-Funktion*.

Ridge-Funktionen treten natürlicherweise in vielen Anwendungen auf, wie in der Fourier-Transformation als $e^{\langle \xi, t \rangle}$ oder als sogenannte Ebene Wellen in der Physik.

Definition 1.3 Ein Neuronales Netzwerk im MLP-Modell oder ein *Perzeptron* ist ein Neuronales Netzwerk $\mathcal{N}$, dessen Schichten f_i aus Ridge-Funktionen bestehen, d.h.

$$(f_i(x))_j = \sigma_j(\langle w_{i,j}, x \rangle + \beta_{i,j}), \quad j = 1, \ldots, n_i,$$

für alle $i = 1, \ldots, N$. In diesem Fall nennen wir die Funktionen σ_i *Aktivierungsfunktionen* und die Parameter $w_{i,j} \in \mathbb{R}$ *Gewichte*.

Wir können solche Netzwerke damit in sogenannte *Neuronen* oder *Knoten* zerlegen, von denen jeder aus der Auswertung einer Ridge-Funktion besteht und die jeweils schichtweise miteinander verbunden sind, d.h. die Ausgabe aller Knoten einer Schicht ergibt die Eingabe sämtlicher Knoten der nächsten Schicht. Darin liegt die Analogie zu den biologischen Neuronalen Netzen in Gehirnen von Lebewesen, wobei die einzelnen Nervenzellen den Knoten entsprechen.

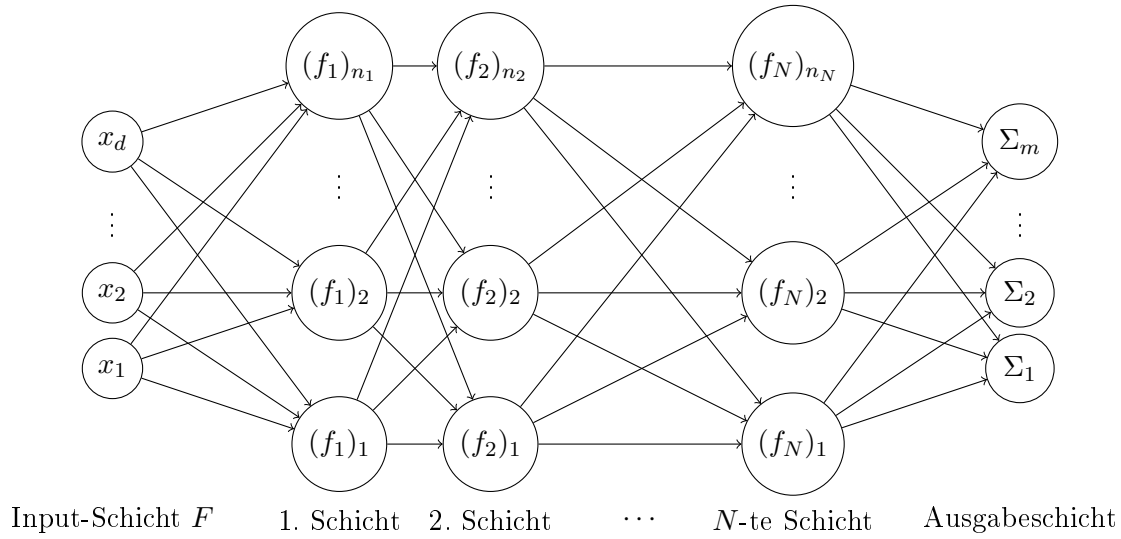

Abbildung 1.1: Ein tiefes Neuronales Netzwerk mit N Schichten

Bemerkung 1.4 Sind alle Aktivierungsfunktionen σ_j in einer Schicht identisch, können wir die Schicht f_j zusammenfassen durch $W_j = (w_{1,j}, \ldots, w_{n_i,j})^T$ und $\mathbf{b}_j = (\beta_{i,j})_{i=1}^{n_i}$ zu

$$f_j(x) = \sigma(W_j x + \mathbf{b}_j),$$

wobei σ komponentenweise angewandt wird.

1.1 Flache Netzwerke

Der einfachste Fall in obiger Definition 1.1 sind Netzwerke mit $d = 1, m = 1$ und $\Sigma = F =$ id. Wir nennen solche Netzwerke *flache, eindimensionale Neuronale Netzwerke*. Zunächst betrachten wir solche Netzwerke mit $\sigma_j = \sigma$ für alle j und σ eine stückweise polynomielle Funktion $\sigma(x) = (x)_+^k$. Damit ist insbesondere die häufig benutzte ReLU-Funktion

$$\mathrm{ReLU}(x) = \max\{0, x\} = (x)_+$$

mit eingeschlossen.

Definition 1.5 Ein flaches NN ist eine Funktion $f : \mathbb{R}^d \to \mathbb{R}$ der Form

$$f(x) = \sum_{i=1}^{n} c_i \sigma(\langle x, w_i \rangle + \beta_i).$$

Ein solches Netzwerk mit einem eindimensionalen Input, d.h. $d = 1$, nennen wir ein *eindimensionales*, *flaches* Netzwerk.

Definition 1.6 Flache, eindimensionale Neuronale Netzwerke zu gegebener Aktivierungsfunktion σ bilden einen linearen Raum, den wir mit

$$\mathcal{N}_1 := \left\{ f \;\middle|\; f(x) = \sum_{i=1}^{n} c_i \sigma(\langle x, w_i \rangle + \beta_i),\ c_i, w_i, \beta_i \in \mathbb{R}, i = 1, \ldots, n,\ n \in \mathbb{N} \right\} \tag{1.1}$$

bezeichnen. Wenn wir uns auf Netzwerke einer maximalen Breite n beziehen wollen, so bezeichnen wir den diesen Raum auch mit $\mathcal{N}_{1,n}$.

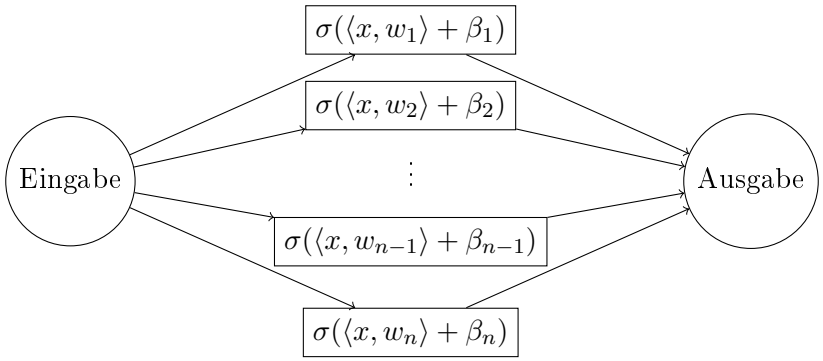

Abbildung 1.2: Struktur eines flachen, eindimensionalen Neuronalen Netzwerks aus dem Raum $\mathcal{N}_{1,n}$ aus Definition 1.1

Bemerkung 1.7 Sei $s : \mathbb{R} \to \mathbb{R}$ eine stückweise lineare Funktion mit Bruchstellen $a_1, \ldots, a_L$. Dann existiert ein eindimensionales, flaches Neuronales Netzwerk $\mathcal{N}$ mit Aktivierungsfunktion $\sigma(x) = \mathrm{ReLU}(x)$, L Knoten und $s = \mathcal{N}$.

Beweis: Jedes eindimensionale, flache ReLU-Netzwerk ist eine stückweise lineare Funktion als Summe von stückweise linearen Funktionen. Umgekehrt sei s stückweise linear mit Bruchstellen $a_1 < \cdots < a_L$. Dann ist $s = \sum_{l=1}^{L} c_l \,\mathrm{ReLU}(x - a_l)$ für geeignete $c_l \in \mathbb{R}$.

Damit sehen wir, dass diese einfachen eindimensionalen ReLU-Netzwerke nur stückweise lineare Funktionen darstellen. Gleichzeitig werden wir sehen, dass bereits diese Netzwerke dicht in $\mathcal{C}(\mathbb{R})$ bzgl. einer geeigneten Norm liegen.

Bemerkung 1.8 Der vorherige Satz gilt in analoger Weise auch für Netzwerke mit stückweise polynomiellen Aktivierungsfunktionen und Splines von höherem Grad. Auf diese werden wir in Abschnitt über Aktivierungsfunktion 1.1.1 weiter eingehen.

Diese flachen Netzwerke bilden die einfachsten Neuronalen Netzwerke und wurden bereits lange unter verschiedenen Bezeichnungen untersucht (siehe bspw. [18], [51], [63],[66], [65] uvm.). Es gibt daher zahlreiche Resultate zur Approximation mit solchen Netzwerken, insbesondere auch scharfe Abschätzungen zur Approximationsrate. Zunächst sehen wir, dass solche Netzwerke beliebige stetige Funktionen auf kompakten Mengen annähern können, wie der folgende bekannte Satz von Cybenko aussagt.

Satz 1.9 (Universal Approximation Theorem [18]) Sei $f \in \mathcal{C}([0,1]^d), \varepsilon > 0$ gegeben. Sei σ eine stetige, messbare Funktion mit

$$\lim_{\xi \to \infty} \sigma(\xi) = 1 \text{ und } \lim_{\xi \to -\infty} \sigma(\xi) = 0.$$

Dann existiert ein flaches Netzwerk $\mathcal{N} \in \mathcal{N}_1$ mit Aktivierungsfunktion σ und

$$\|\mathcal{N} - f\|_\infty \leq \varepsilon.$$

Der Beweis zu diesem Satz ist nicht konstruktiv und benutzt den Satz von Hahn-Banach sowie den Rieszschen Darstellungssatz.
Dieses Theorem wurde in verschiedene Richtungen verallgemeinert, etwa für komplexwertige Netzwerke in [83] und für bestimmte Netzwerkarchitekturen.
Auch die Voraussetzungen an die Aktivierungsfunktion σ können weiter abgeschwächt werden, wie etwa das folgende Resultat von Pinkus [65, Proposition 3.7] besagt.

Satz 1.10 ([65, Proposition 3.7]) Sei $\sigma \in \mathcal{C}(R)$ und kein Polynom. Dann ist die Menge $\mathcal{N}_1$ dicht in $\mathcal{C}(\mathbb{R})$ bezüglich der kompakten Konvergenz, d.h. für jedes $K \subset \mathbb{R}$ kompakt, $f \in \mathcal{C}(K)$ und $\varepsilon > 0$ existiert $\mathcal{N} \in \mathcal{N}_1$ mit $\|f - \mathcal{N}\|_\infty \leq \varepsilon$.

Es gelten analoge Aussagen auch für weitere Räume, insbesondere L_p- und Sobolev-Räume.

Definition 1.11 Sei $k \in \mathbb{N}$, $1 \leq p < \infty$ und $\Omega \subset \mathbb{R}^n$ offen. Dann ist der Sobolev-Raum $W^{k,p}$ der Raum der Funktionen in $L_p(\Omega)$ deren schwache Ableitungen bis zur Ordnung k

existieren und in $L_p(\Omega)$ liegen, d.h.

$$W^{k,p}(\Omega) = \{f \in L_p(\Omega) \mid D^\alpha f \in L_p(\Omega), |\alpha| \leq k\}, \tag{1.2}$$

wobei $\alpha \in \mathbb{N}_0^n$ und $|\alpha| = \sum_{i=0}^n \alpha_i$. Mit der Norm

$$\|f\|_{W^{k,p}(\Omega)} = \|f\|_{k,p} := \left(\sum_{|\alpha| \leq k} \|D^\alpha f\|_{L_p(\Omega)}^p \right)^{1/p} \tag{1.3}$$

ist $W^{k,p}(\Omega)$ ein Banachraum. Für $p = \infty$ definiere für den Raum $W^{k,\infty}$ analog wie oben als $\{f \in L_\infty(\Omega) \mid D^\alpha f \in L_\infty(\Omega), |\alpha| \leq k\}$ mit Norm

$$\|f\|_{W^{k,\infty}} = \|f\|_{k,\infty} := \max_{|\alpha| \leq k} \|D^\alpha f\|_{L_\infty(\Omega)}. \tag{1.4}$$

Definition 1.12 Wir bezeichnen mit $\mathcal{N}_{m,n}$ den Raum der mehrschichtigen Neuronalen Netze wie in Definition 1.1 mit n Schichten und insgesamt $m := \sum_{j=1}^n n_i$ Neuronen.

Wir interessieren uns nun aber zunächst besonders für den Approximationsfehler.

Definition 1.13 Der Fehler der Bestapproximation durch Neuronale Netzwerke ist gegeben durch

$$E_{n,p}(f) = \inf_{f^* \in \mathcal{N}_{1,n}} \|f - f^*\|_{L_p}.$$

Die Approximationsrate für flache Netzwerke ist wohlbekannt, siehe z.B. das folgende Ergebnis von Mhaskar [55]. Diese Rate ist zudem die bestmögliche.

Theorem 1.14 ([55, Theorem 2.1]) Sei $\sigma \in \mathcal{C}^\infty(\mathbb{R})$ kein Polynom auf irgendeinem Teilintervall von $\mathbb{R}$ und B ein offener Ball in $\mathbb{R}^d$. Dann gilt für $f \in W^{r,d}(B)$

$$E_{n,\infty}(f) = \mathcal{O}(n^{-r/d}).$$

Bemerkung 1.15 Die Forderung, dass σ kein Polynom sein darf, ist äquivalent zu $\sigma^{(k)}(x) \neq 0$ für $k \geq 0$ und ein x aus jedem Intervall.

1.2 Aktivierungsfunktionen

Die Wahl der Aktivierungsfunktion σ in der Definition klassischer Neuronaler Netzwerke 1.3 hat einen großen Einfluss auf die Eigenschaften der resultierenden Klasse an Neuronalen Netzwerken. Wir möchten uns zunächst allgemein mit verschiedenen praktisch und theoretisch betrachteten Aktivierungsfunktionen σ auseinandersetzen. In Theorem 1.10 haben wir bereits gesehen, dass Polynome als Aktivierungsfunktionen schlecht geeignet sind. Es bleibt die umgekehrte Fragestellung, ob es besonders günstige Aktivierungsfunktionen gibt, konkret: Gegeben ein festes zu approximierendes f, mit welcher Aktivierungsfunktion σ kann

ein neuronales Netzwerk mit bestimmter Netzwerkstruktur (Tiefe und Breite) die Funktion f am Besten, gemessen in beliebiger Norm, und am Günstigsten, d.h. mit möglichst wenigen nichtverschwindenden Gewichten approximieren? Nach Satz 1.9 und Satz 1.10 ist eine Approximation in $\mathcal{C}(K)$ grundsätzlich mit fast jeder nicht-polynomiellen Funktion möglich, es ist jedoch intuitiv klar, dass die Wahl einer geeigneten Aktivierungsfunktion die Rate der Approximation verbessern kann. Trivialerweise löst etwa $\sigma = f$ mit einem einschichtigen Netzwerk das obige Approximationsproblem exakt, wird jedoch für andere Zielfunktionen f dies nicht erreichen. Auch im Hinblick auf den Prozess des Lernens der optimalen Gewichte lassen sich leicht Unterschiede zwischen verschiedenen Aktivierungsfunktionen feststellen. Bereits in einem einfachen Experiment mit einem Klassifikationsproblem mit dem MNIST-Datensatz [47] erkennen wir, dass der betrachtete Fehler unterschiedlich schnell fällt. Dabei nutzen wir ein einfaches Netzwerk mit 2 Schichten und 768 Neuronen, um eine Klassifikation von handgeschriebenen Ziffern in 10 Kategorien vorzunehmen. Gleichwohl theoretisch alle betrachteten Aktivierungsfunktionen $\mathrm{ReLU}(x) = (x)_+, \mathrm{ReQU}(x) = (x)_+^2$ und $(x)_+^3$ keine Polynome sind und somit beliebige stetige Funktionen approximieren können, fällt das Ergebnis der Approximation bei fester Netzwerkgröße deutlich unterschiedlich aus, wie wir in der folgenden Grafik sehen.

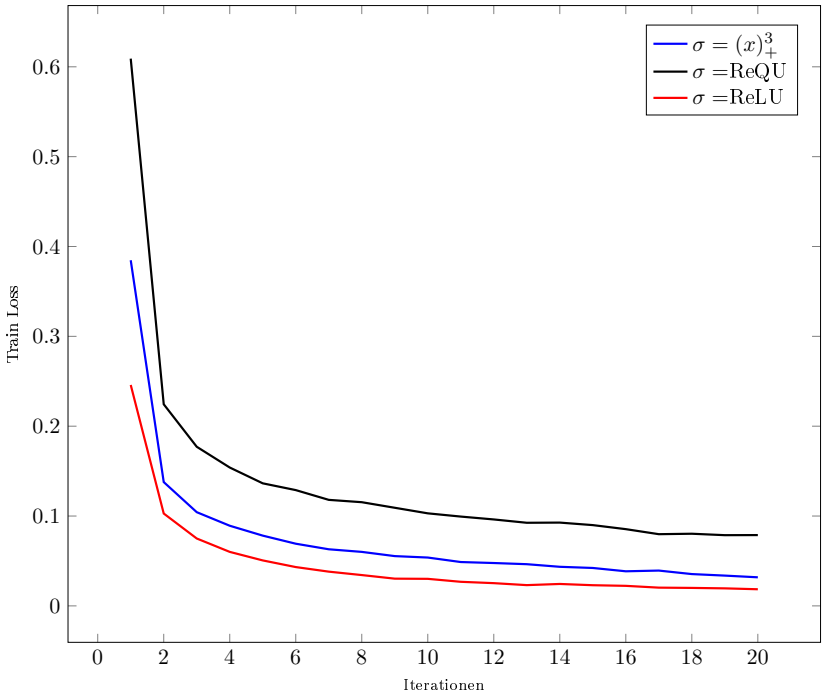

Abbildung 1.3: Fehler bei einem Klassifikationsproblem für verschiedene Aktivierungsfunktionen

Für große Werte von L (das bei unseren voll verbundenen Netzwerken mit fester Tiefe die Zahl der Gewichte widerspiegelt) ist der Fehler nach vielen Iterationen in allen Beispielen ähnlich, denn alle Aktivierungsfunktionen besitzen die universelle Approxima-

tionseigenschaft (vgl. 1.9) und somit konvergiert der Fehler resultierend aus der Wahl der Gewichte gegen 0. Jedoch ist die Geschwindigkeit der Konvergenz der Fehler, insbesondere in den ersten Iterationen, deutlich unterschiedlich was zum Einen mit Problemen bei der Optimierung der Gewichte mit einem Backpropagation-Algorithmus zusammen hängt und zum Anderen mit den unterschiedlichen Approximationsraten bei endlicher Netzwerkgröße. Die Auswirkung auf den Backpropagationsalgorithmus beruhen auf Phänomenen wie zu kleinen oder zu großen Gradienten und lokalen Minima, welche bereits vielfach untersucht wurden, siehe bspw. [70], [26, insb. Kapitel 8, 11].
In der theoretischen Literatur zur Approximation mit neuronalen Netzwerken wurde bisher oftmals mit sogenannten *sigmoidalen* Aktivierungsfunktionen gearbeitet, d.h. Funktionen σ, welche die Voraussetzungen von Satz 1.9 erfüllen mit σ stetig, $\lim_{x\to-\infty}\sigma(x) = 0$ und $\lim_{x\to\infty}\sigma(x) = 1$. In der Praxis werden jedoch vielerlei verschiedene Aktivierungsfunktionen eingesetzt, am häufigsten die sogenannte ReLU-Funktion. Eine Galerie von möglichen Aktivierungsfunktionen ist in Abbildung 1.2 zu sehen. Dabei sind nur die (skalierte) tanh- und Heaviside-Funktion sigmoidal im Sinne der Definition.

Name	Definition $\sigma(x)$	Eigenschaften
ReLU „Rectified Linear Unit"	$(x)_+ = \max\{0, x\}$	stetig, $\frac{d}{dx}\sigma(x) = 0$ für $x < 0$
ReQU „Rectified Quadratic Unit"	$(x)_+^2$	stetig differenzierbar, $\frac{d}{dx}\sigma(x) = 0$ für $x < 0$
ELU „Exponential Linear Unit"	$\begin{cases} \alpha(\exp(x) - 1), & x < 0, \\ x, & x \geq 0. \end{cases}$	stetig, Parameter $\alpha > 0$
CELU „Continuously differentiable ELU"	$\begin{cases} \alpha(\exp(x/\alpha) - 1), & x < 0, \\ x, & x \geq 0. \end{cases}$	stetig differenzierbar, Parameter $\alpha > 0$
LeakyReLU	$\begin{cases} \alpha x, & x < 0, \\ x, & x \geq 0. \end{cases}$	stetig, Parameter $\alpha > 0$
Softplus	$\ln(1 + \exp(x))$	$\mathcal{C}^\infty$
tanh	$\frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)}$	$\mathcal{C}^\infty$, sigmoidal (mit Skalierung)
Heaviside (Schwellenwert-, Stufen-, Einheitssprungfunktion)	$\begin{cases} 0, & x \leq 0, \\ 1, & x > 0 \end{cases}$	sigmoidal

Tabelle 1.1: Acht klassische Aktivierungsfunktionen

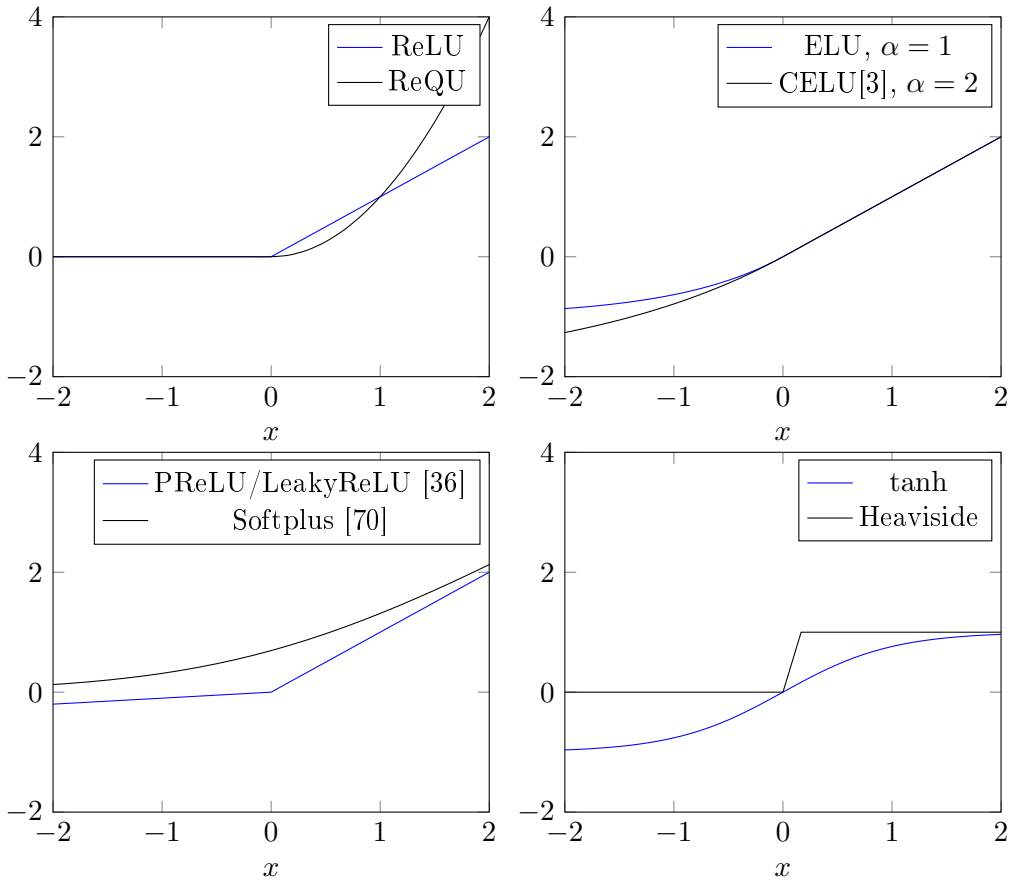

Abbildung 1.4: Verschiedene vorgeschlagene und/oder häufig verwendete Aktivierungsfunktionen

1.2.1 Optimale Aktivierungsfunktionen

Es gibt durchaus in gewissem Sinne „optimale" Aktivierungsfunktionen, so besagt etwa der Satz von Kolmogorov (auch Kolmogorov Superposition Theorem, cf. [20, Kapitel 17]) etwa, dass

Satz 1.16 (Satz von Kolmogorov)
Sei $f \in \mathcal{C}([0,1]^d)$. Es existieren $2d+1$ Lipschitz-stetige Funktionen $\phi_j : [0,1] \to [0,1]$, $j = 1, \dots, 2d+1$ und d Konstanten $\lambda_i > 0$, $i = 1, \dots, d$ mit $\sum_{i=1}^{d} \lambda_i \leq 1$ unabhängig von f sodass

$$f(x_1, \dots, x_d) = \sum_{j=1}^{2d+1} g\left(\sum_{i=1}^{d} \lambda_i \phi_j(x_i)\right) \tag{1.5}$$

für eine Funktion $g \in \mathcal{C}([0,1])$, die nur von f abhängt.

Übersetzt in das Setting der Neuronalen Netzwerke besagt ein Theorem aus [51]

Theorem 1.17 ([51, Theorem 4]) Es existiert eine analytische, monoton wachsende und sigmoidale Aktivierungsfunktion σ, sodass jedes $f \in \mathcal{C}([0,1]^d)$ gleichmäßig durch Ausdrücke

$$\sum_{i=1}^{6d+3} d_i \sigma \left(\sum_{j=1}^{3d} c_{ij} \sigma \left(\sum_{k=1}^{d} w_{ijk} x_k + \theta_{ij} \right) + \gamma_i \right)$$

approximiert werden kann für gewisse Parameter $d_i, c_{ij}, w_{ijk}, \theta_{ij}, \gamma_i$.

Das heißt, dass es Aktivierungsfunktionen gibt, die jede stetige Funktion mit einem Netzwerk von endlicher Tiefe und nur linear von d abhängiger Breite beliebig genau approximieren kann. Solche Aktivierungsfunktionen oder Klassen von Aktivierungsfunktionen heißen *superexpressiv*.
Yarotzky [86, Theorem 3] konnte zeigen, dass es einige einfache Mengen von Aktivierungsfunktionen gibt, die diese Eigenschaft besitzen. So ist etwa $\{\sin, \arcsin\}$ eine solche superexpressive Klasse, und für ρ analytisch und kein Polynom ist $\{\rho, \lfloor \cdot \rfloor\}$ ebenfalls superexpressiv.
Wir zeigen nun, dass keine der häufig verwendeten stückweise polynomiellen Aktivierungsfunktionen dies erfüllen.

Lemma 1.18 Sei $\mathcal{A} = \{(x)_+^k\}_{k=1}^p$ und $N \in \mathbb{N}$ gegeben. Dann existiert eine stetige Funktion $f : [0,1]^d \to \mathbb{R}$ mit

$$\|\mathcal{N} - f\|_\infty > 0$$

für jedes N-schichtige Netzwerk $\mathcal{N} = L_N \circ \cdots \circ L_1$ mit Aktivierungsfunktionen aus $\mathcal{A}$, $L_i : \mathbb{R}^{n_i} \to \mathbb{R}^{n_{i+1}}$ und $n_i \leq n(d) \in \mathcal{O}(d)$ für alle $i = 1, \ldots, N$.

Beweis: Es genügt, die Aussage für $d = 1$ zu zeigen, da aus der Superexpressivität für $d = 1$ die Superexpressivität für beliebige Dimension d aus dem Satz von Kolmogorov 1.16 folgt. Dieses Argument wurde auch bereits von Yarotzky in [86] angewendet. Denn, angenommen, $\Phi = \{\phi_k\}$ sei eine Familie von superexpressiven Aktivierungsfunktionen für $d = 1$. Dann ist nach Satz 1.16 jede Funktion in d Veränderlichen durch eine Summe von $\mathcal{O}(d)$ univariaten Funktionen darstellbar, diese sind nach Voraussetzung der Superexpressivität mit $\mathcal{O}(d)$ Funktionen aus Φ.
Die Aktivierungsfunktionen $(x)_+^k$ sind stückweise Polynome. Damit ist auch

$$L_i(x) = \sum_{j=1}^{n_i} \sigma_j(w_{i,j} x)$$

mit $\sigma_j = (x)_+^k, w_{i,j} \in \mathbb{R}$ für alle $i = 1, \ldots, N$ ein stückweises Polynom vom Grad höchstens k. Damit ist das Netzwerk $\mathcal{N} = L_N \circ \cdots \circ L_1$ ebenfalls ein stückweises Polynom vom Grad höchstens k^N. Sei $I \subset \mathbb{R}$ ein Intervall, auf dem $\mathcal{N}\big|_I$ nicht identisch 0 und ein Polynom vom Grad $\leq p^N$ ist und sei $|I| > 0$ die Länge dieses Intervalls. Dann hat $\mathcal{N}$ auf diesem Intervall höchstens k^N Nullstellen und somit Vorzeichenwechsel. Somit ist $\max_{x \in I} |\mathcal{N}(x) - g(x)| > 0$ für Funktionen g mit mehr Nullstellen, wie etwa $g(x) = \sin(k^{N+1} \pi x)$. □

Auch wenn diese Klassen von Aktivierungsfunktionen nicht superexpressiv sind, wollen wir sie weiter betrachten. Denn diese sind die einfachste Art nichtpolynomieller Funktionen, die somit nach Satz 1.10 beliebige Funktionen annähern können und daher in der Anwendung die mit Abstand am Häufigsten benutzten Aktivierungsfunktionen.

1.2.2 Die RePU Aktivierungsfunktionen

Die in der Praxis am häufigsten verwendete Aktivierungsfunktion ReLU gehört zu einer Gruppe von Aktivierungsfunktionen, die wir Rectified Power Units nennen. Als Rectified Power Unit (RePU) werden die Aktivierungsfunktion $\sigma : \mathbb{R} \to \mathbb{R}^+$ mit $\sigma(x) = (x)_+^k$ bezeichnet. Die ReQU (Rectified Quadratic Unit) ist der interessante Spezialfall $k = 2$.

Approximation mit ReLU-Netzwerken

Die ReLU-Aktivierungsfunktion $\mathrm{ReLU}(x) = (x)_+ = \max\{0, x\}$ ist die mit Abstand am häufigsten in der Praxis eingesetzte Aktivierungsfunktion, da sie jedoch nur stückweise linear ist, sind einige Konstruktionen nötig, um mit ihr andere einfache nichtlineare Funktionen wie Polynome zu approximieren. Eine effizientere Herangehensweise werden wir in Kapitel 2 über den Zusammenhang mit Splines sehen.
Um Polynome approximieren zu können, müssen wir zunächst die elementareren Funktionen $x \mapsto x^n$ und $(x, y) \mapsto xy$ approximieren.

Lemma 1.19 Sei $\varepsilon > 0$ und $\beta > 0$ gegeben. Dann existiert ein Netzwerk $\mathcal{N}_1$ mit $\mathcal{O}(\ln(\frac{1}{\varepsilon}))$ Knoten und Schichten mit

$$\sup_{x,y\in[0,\beta]} |\mathcal{N}_1(x, y) - xy| \leq \varepsilon$$

und $\mathcal{N}(0, y) = \mathcal{N}(x, 0) = 0$ für alle $x, y \in [0, \beta]$.

Beweis: Diese Aussage stammt aus [87, Prop. 2, 3], wir folgen dem dort gegebenen Beweis mit weitere Erläuterungen und approximieren zunächst die Abbildung $x \mapsto x^2$ auf dem Intervall $[0, 1]$ über eine Konstruktion, welche zunächst von Telgarsky [80] als Beispiel für Vorteile von tiefen Netzwerken entwickelt und später von Yarotsky [87] verwendet wurde. Definiere zunächst $s : [0, 1] \to [0, 1]$

$$s(x) := \begin{cases} 2x & , x < \frac{1}{2} \\ 2 - 2x & , x \geq \frac{1}{2} \end{cases} \tag{1.6}$$

als 1-Sägezahn und weiter $s^r := \underbrace{s \circ s \circ \cdots \circ s}_{r}$. Diese Funktion ist ein 2^r-Sägezahn, d.h. s^r ist stückweise linear mit 2^r gleichverteilten Extrema in dem Intervall $[0, 1]$. Konkret ist s^r von der Form

$$s^r(x) = \begin{cases} 2^r x - 2k & , x \in [\frac{2k}{2^r}, \frac{2k+1}{2^r}], k = 0, \ldots, 2^{r-1} - 1, \\ 2k - 2^r x & , x \in [\frac{2k-1}{2^r}, \frac{2k}{2^r}], k = 1, \ldots, 2^{r-1}. \end{cases}$$

Gleichzeitig kann s durch ein ReLU-Netzwerk mit einer Schicht und drei Knoten exakt dargestellt werden, indem $s(x) = 2\,\mathrm{ReLU}(x) - 4\,\mathrm{ReLU}(x-\frac{1}{2}) + 2\,\mathrm{ReLU}(x-1)$ und s^r kann durch ein Netzwerk mit r Schichten dargestellt werden, welche jeweils s berechnen.
Es gilt für eine stückweise lineare Interpolation g_{2^m} von $x \mapsto x - x^2$ an 2^m+1 gleichverteilten Punkten $\frac{k}{2^m}$, $k = 0, \dots, 2^m$, dass zwischen zwei Punkten $\frac{k_0}{2^m}$ und $\frac{k_0+1}{2^m}$ die Interpolante von der Form $g_{2^m}(t) = t(\frac{2^m - 2k_0 + 1}{2^m}) - \frac{k_0^2 - k_0}{2^{2m}}$ ist und damit $\sup_{x\in[0,1]} |x - x^2 - g_{2^m}(x)| = \max_{k=0,\dots,2^m} |\frac{2k+1}{2^{2m+1}} - \frac{(2k+1)^2}{2^{2m}+1} - g_{2^m}(\frac{2k+1}{2^{2m+1}})| = 2^{-2m-2}$.
D.h. $x - \sum_{r=1}^{n} s^r(x)$ ist eine Approximation bzgl. $\|\cdot\|_\infty$ an $x \mapsto x^2$ für alle $x \in [0,1]$ mit Fehler höchstens 2^{-2m-2}. Soll der Fehler also kleiner ε' gehalten werden, so wählen wir $2m \geq \log_2(1/\varepsilon)$.
Wir erhalten die Approximation an die Multiplikation für $\beta \leq 1$ durch $xy = \frac{(x+y)^2 - x^2 - y^2}{2}$.
Falls $\beta > 1$, skaliere die Funktion via $xy = 2\beta \frac{xy}{2\beta}$ und wende entsprechend s^r auf $0 \leq \frac{x}{2\beta} \leq \frac{1}{2}$ bzw. $\frac{y}{2\beta}$ an.
Das Netzwerk $\mathcal{N}_0$ zur Approximation von $(x,y) \mapsto xy$ besteht nach obiger Konstruktion aus $\mathcal{O}(\ln(\frac{1}{\varepsilon}))$ Schichten mit konstanter Breite 4, also aus insgesamt $\mathcal{O}(\ln(\frac{1}{\varepsilon}))$ Knoten. □

Ein Vorteil dieser Konstruktion ist, dass für das resultierende Netzwerk gilt $\mathcal{N}_1(x,y) = 0$ falls $x = 0$ oder $y = 0$. Außerdem ist auch $\|\mathcal{N}_1(x,y) - xy\|_{L_p([0,\beta]^2)} \leq \varepsilon$ für alle $1 \leq p \leq \infty$. Die Funktion s^r aus dem vorherigen Beweis wird als Beispiel für die Komposition von Splines in Kapitel 1.3.6 erneut vorkommen. Aus der Approximation von Polynomen im nächsten Lemma ergeben sich Approximationen für größere Klassen an Funktionen. Eine aktuelle und ausführliche Übersicht liefert [4].

Lemma 1.20 Sei $p \in \mathbb{P}_d^n$ ein Polynom bestehend aus höchstens ℓ Monomen und seien $\varepsilon, \beta > 0$ gegeben, sodass die Beträge der Koeffizienten von p und die Werte der Monome in p kleiner gleich β sind. Dann existiert ein Netzwerk $\mathcal{N}$ mit $\mathcal{O}(\ln(\frac{dn}{\varepsilon})^2)$ Schichten und $\mathcal{O}(\ln(\frac{\ell dn}{\varepsilon})^2)$ Neuronen, sodass

$$\sup_{x\in[0,1]^d} |p(x) - \mathcal{N}(x)| \leq \varepsilon. \tag{1.7}$$

Beweis: Wir kombinieren für diesen Beweis die Ergebnisse des vorherigen Lemmas 1.19. Zunächst konstruieren wir Monome x^s für $s \in \mathbb{N}$ durch Quadrieren und Multiplizieren.
Sei $s = \sum_{i=0}^{\lceil \log_2(s) \rceil} c_i(s) 2^i$ für geeignete $c_i(s) \in \{0,1\}$. Dann ist

$$x^s = x^{\sum_{i=0}^{\lceil \log_2(s) \rceil} c_i(s) 2^i} = \prod_{i=1}^{\lceil \log_2(s) \rceil} x^{c_i(s) 2^i}.$$

Es genügen zur Darstellung von x^s durch ein ReLU-Netzwerk $\mathcal{N}_3$ demnach $\lfloor \log_2(s) \rfloor$ Multiplikationen und $\lfloor \log_2(s) \rfloor$-maliges Quadrieren von x. Nach dem vorherigen Lemma 1.19, benötigt eine Multiplikation mit Genauigkeit ε ein Netzwerk $\mathcal{N}_1$ mit $\mathcal{O}(\ln(\frac{1}{\varepsilon}))$ Knoten und Schichten. Ebenfalls nach dem Beweis dieses Lemma benötigt das Quadrieren mit einem Netzwerk $\mathcal{N}_2$ ebenfalls $\mathcal{O}(\ln(\frac{1}{\varepsilon}))$ Knoten und Schichten. Damit ist der Fehler $\sup_{x\in[0,\beta]} |\mathcal{N}_3 - x^s| = \sup_{x\in[0,\beta]} |\mathcal{N}_1(x, \mathcal{N}_1(\mathcal{N}_2(x), \mathcal{N}_1(\mathcal{N}_2^2(x), \cdots))) - \prod_{i=1}^{\lceil \log_2(s) \rceil} x^{c_i(s) 2^i}|$. Es

gilt für alle $x, y \in [0, \beta]$, dass $|\mathcal{N}_1(x, \mathcal{N}_2(y)) - xy^2| \leq |\mathcal{N}_1(x, y^2) - xy^2| + |\mathcal{N}_1(x, y^2) - \mathcal{N}_1(x, \mathcal{N}_2(y))| \leq \varepsilon + |\mathcal{N}_1(x, \mathcal{N}_2(y)) - x\mathcal{N}_2(y)| + |x\mathcal{N}_2(y) - xy^2| \leq (3 + |x|)\varepsilon$. Induktiv mit $y = \mathcal{N}_1(x, \mathcal{N}_2(\cdots))$ ergibt sich somit ein Fehler von $|\mathcal{N}_3 - x^s| \leq \frac{1}{s^2}\varepsilon$. Um die geforderte Genauigkeit zu erreichen, benötigt das Netzwerk also $\mathcal{O}((\frac{s}{\varepsilon})^2)$ Schichten und Knoten.
Polynome in $\mathbb{P}_d^n$ bestehen aus höchstens $\ell \leq \binom{n+d}{d}$ verschiedenen (multivariaten) Monomen, welche jeweils das Produkt von höchstens d Monomen der Form wie oben vom Grad höchstens n sind. Wir benötigen daher für jedes Monom höchstens $(d-1)$ Multiplikationen. Dann ist für jedes Monom cx^α mit $|\alpha| \leq n$ der Fehler $|cx^\alpha - \mathcal{N}_1(x, \mathcal{N}_1(x^{\alpha_i}, \mathcal{N}_1(x^{\alpha_2}, \cdots)))| \leq n\varepsilon$. Um insgesamt den Fehler ε zu erhalten, müssen wir diese Multiplikation mit Genauigkeit $\frac{\varepsilon}{n}$ und $\mathcal{N}_3$ mit Genauigkeit $\frac{\varepsilon}{dn}$ berechnen. Wir benötigen daher insgesamt ein Netzwerk mit $\mathcal{O}((\frac{\ell dn}{\varepsilon})^2)$ Neuronen und einer Tiefe von $\mathcal{O}((\frac{dn}{\varepsilon})^2)$, da die ℓ Monome in derselben Schicht berechnet werden können. □

Bemerkung 1.21 Die Größe des Netzwerks und die Genauigkeit lässt sich durch eine bessere Darstellung der Polynome durch weniger Multiplikationen verbessern. Gegeben ein multivariates Polynom in d Veränderlichen $p(x) = \sum_{\alpha \in J} c_\alpha x^\alpha$ mit $J \subset \mathbb{N}^d$ existiert eine Darstellung (multivariates Horner-Schema), welches p mit $n_p := (d+1) \max_{\alpha \in J} |\alpha|$ Multiplikationen und $|J|$ Additionen darstellt. Dies lässt sich mit Hilfe von Lemma 1.19 einfach auf ein Neuronales Netzwerk übertragen mit $\mathcal{O}(n_p \ln(\frac{1}{\varepsilon}))$ Schichten. Für die Stabilität dieses Ansatzes siehe Theorem 3 aus Peña, Sauer [61].

Als nächstes betrachten wir Funktionen aus dem Sobolev-Raum (vgl. Definition1.11). Für diese wissen wir bereits aus dem Theorem 1.14, was die optimale Rate der Approximation für flache Netzwerke ist. Mit den oben entwickelten Methoden können wir durch tiefe Netzwerke effizienter Taylorpolynome (univariat wie multivariat) darstellen. Dazu gilt die folgende Aussage aus [87, Theorem 1], wobei wir den Beweis für den multivariaten Fall deutlich vereinfachen.

Lemma 1.22 Sei $\varepsilon > 0$ und $f \in W^{k,\infty}([0,1]^d)$ mit $\|f\|_{k,\infty} \leq 1$ gegeben. Dann existiert ein Netzwerk $\mathcal{N}$ mit $\mathcal{O}(|\ln(\varepsilon)|)$ Schichten und $\mathcal{O}(\varepsilon^{-\frac{d}{k}}(|\ln(\varepsilon)| + 1))$ Knoten sodass $\|f - \mathcal{N}\|_{k,\infty} \leq \varepsilon$.

Beweis: Wir betrachten zuerst den Fall $d = 1$. Sei $N \in \mathbb{N}$. Wir betrachten zunächst das Taylorpolynom zu f vom Grad $k-1$ an der Stelle $x_j = \frac{j}{N}$ der Form

$$T_f(x_j)(x) = \sum_{n=0}^{k-1} \frac{f^{(n)}(x_j)}{n!}(x - x_j)^n$$

Wir schätzen den Fehler der Approximation durch das Taylorpolynom in einer Umgebung von x_j ab. Es gilt nach dem Satz von Taylor

$$|f(x) - T_f(x_j)(x)| = \left| \int_{x_j}^{x} \frac{(x-t)^{k-1}}{k!} f^{(k)}(t)\, dt \right|$$

$$\leq \frac{(x-x_j)^k}{k!}\|f^{(k)}(x)\|_\infty$$
$$\leq \frac{(x-x_j)^k}{k!},$$

da $f \in W^{k,\infty}$ mit $\|f\|_{k,\infty} \leq 1$. Auf der Umgebung $U_{x_j} := \{x \,|\, (x-x_j)^k/k! \leq \varepsilon/4\}$ erhalten wir $\|T_f(x_j)(x)\big|_{U_{x_j}} - f\big|_{U_{x_j}}\| \leq \frac{\varepsilon}{2}$.
Nun approximieren wir $\mathbb{1}_{U_{x_j}}$ durch eine stückweise lineare Funktion ϕ_j. Definiere dazu $x_{j\pm1/2} := \frac{x_{j\pm1} \mp x_j}{2}$. Die Funktion

$$\phi_j(x) := \begin{cases} \frac{x-x_{j-1}}{x_{j-1/2}-x_{j-1}}, & x_{j-1} \leq x \leq x_{j-1/2}, \\ 1, & x_{j-1/2} \leq x \leq x_{j+1/2}, \\ 1 - \frac{x-x_{j+1/2}}{x_{j+1}-x_{j+1/2}}, & x_{j+1/2} \leq x \leq x_{j+1}, \\ 0, & \text{sonst} \end{cases}$$

hat einen Träger der Länge $\leq \frac{2}{N}$ und wird dargestellt durch ein ReLU-Netzwerk mit 4 Knoten in einer Schicht und es gilt $\|\phi_j\|_\infty = 1$ sowie $\sum_{j=0}^N \phi_j(x) = 1$.
Damit setzen wir zusammen $\mathcal{N} = \sum_{j=1}^N \phi_j P_j$ wobei P_j mit $\|P_j - T_f(x_j)\|_\infty \leq \frac{\varepsilon}{2}$ ein Netzwerk wie in Lemma 1.20 mit $\mathcal{O}((\ln \frac{2(k-1)}{\varepsilon})^2)$ Knoten ist und der Multiplikation aus Lemma 1.19 mit Genauigkeit $\frac{\varepsilon}{4N}$. Dann gilt mit $N^k \geq \frac{1}{2k!\varepsilon}$ gewählt, dass

$$\begin{aligned}
\|f-\mathcal{N}\|_{k,\infty} &= \max_{0\leq l\leq k} \sup_{x\in[0,1]} |f^{(l)}(x) - \mathcal{N}^{(l)}(x)| \\
&= \max_{0\leq l\leq k} \sup_{x\in[0,1]} \left| \sum_{j=0}^N \phi_j(x)(f^{(l)}(x) - P_j^{(l)}(x)) \right| \\
&\leq \max_{0\leq l\leq k} \sup_{x\in[0,1]} \sum_{j=0}^N \left(\left| f^{(l)}(x) - P_j^{(l)}(x) \right| + \frac{\varepsilon}{4N} \right) \\
&\leq \max_{0\leq l\leq k} \sup_{x\in[0,1]} \sum_{j=0}^N |f^{(l)}(x) - T_f^{(l)}(x_j)(x)| + |T_f^{(l)}(x) - P_j^{(l)}(x)| + \frac{\varepsilon}{4} \\
&\leq \frac{2\varepsilon}{4} + \frac{\varepsilon}{2} = \varepsilon
\end{aligned}$$

Für $d > 1$ gehen wir analog zum Fall $d = 1$ vor, indem wir das multivariate Taylorpolynom auf kleinen Teilintervallen approximieren. Wir benutzen die Multiindex-Schreibweise mit $\mathbf{j} \in \mathbb{N}_0^d$ und $\mathbf{j}! = \prod_{k=1}^d j_k!, (\mathbf{x})^{\mathbf{j}} = \prod_{k=1}^d x_k^{j_k}$. Definiere als Analogon zu ϕ_j von oben $\phi_{\mathbf{j}}(\mathbf{x}) := \prod_{k=1}^d \phi_{j_k}(x_k)$ und $\mathbf{x_j} = \frac{\mathbf{j}}{N}$. Sei $T_f(\mathbf{x_j})$ das (multivariate) Taylorpolynom zu f an der Stelle $\mathbf{x_j}$. Es ist nach dem mehrdimensionalen Satz von Taylor

$$\begin{aligned}
|f(\mathbf{x}) - T_f(\mathbf{x_j})(\mathbf{x})| &= \left| f(\mathbf{x}) - \sum_{|\mathbf{n}|\leq k} \frac{(\mathbf{x}-\mathbf{x_j})}{\mathbf{n}!} D^{\mathbf{n}} f(\mathbf{x_j}) \right| \\
&\leq \frac{d^k}{k!N^k}\|f\|_{k,\infty} \\
&\leq \frac{d^k}{k!N^k},
\end{aligned}$$

da für $\mathbf{j}, \mathbf{j}'$ mit $|\mathbf{j}-\mathbf{j}'| = 1$ gilt $\|\mathbf{x_j}-\mathbf{x_{j'}}\|_2 = \frac{1}{N}$. Weiterhin überlappen sich nun in d Dimensionen höchstens 2^d der Umgebungen $U_{\mathbf{x_j}} = \{\mathbf{x} | \|\mathbf{x_j}-\mathbf{x}\|_2 \leq \frac{1}{N}\}$. Wählt man also $N^k \geq \frac{2^{d+1}d^k}{k!\varepsilon}$ erhält man $|f(\mathbf{x}) - T_f(\mathbf{x_j})(\mathbf{x})| \leq \frac{\varepsilon}{2}$. Wir setzen wieder zusammen $\mathcal{N} = \sum_{\mathbf{j}\in\{0,\dots,N\}^d} \phi_{\mathbf{j}} P_{\mathbf{j}}$ mit $(N+1)^d$ Multiplikationen wie in Lemma 1.19 und damit $\|P_{\mathbf{j}} - T_f(\mathbf{x_j})\| \leq \frac{\varepsilon}{2}$ gemäß Lemma 1.20 und erhalten $\|f - \mathcal{N}\|_{k,\infty} \leq \varepsilon$.
Damit haben wir insgesamt ein Netzwerk bestehend aus $k\mathcal{O}(|\ln(\varepsilon)|)$ Schichten und Knoten zur Approximation der Multiplikation, $\mathcal{O}((\ln \frac{k-1}{\varepsilon})^2)$ Schichten und $\mathcal{O}(\ln(\frac{k-1}{\varepsilon})^2)$ Knoten zur Approximation der Taylorpolynome und einer Schicht zur Approximation der Funktionen ϕ_j. Wir wählen unsere Netzwerkarchitektur so, dass die Berechnung der Taylorpolynome durch parallele Teilnetzwerke der Tiefe $\mathcal{O}((\ln \frac{k-1}{\varepsilon})^2)$ und Größe $(N+1)^d\mathcal{O}(\ln(\frac{k-1}{\varepsilon})^2)$ erfolgt. Einsetzen des oben gewählten N liefert die behauptete Größe des Gesamtnetzwerkes. □

Bemerkung 1.23 Dieses Lemma hat uns gezeigt, dass ReLU-Netzwerke von endlicher Tiefe Funktionen aus dem Sobolev-Raum $W^{k,\infty}([0,1]^d)$ approximieren können. Der Faktor d in der Anzahl der Knoten des Netzwerks aus dem letzten Lemma weist jedoch darauf hin, dass solche Netzwerke weiter dem Fluch der hohen Dimension unterliegen, d.h. dass die Anzahl der benötigten Knoten und Parameter weiterhin exponentiell von der Dimension abhängt. Die Abschätzungen der benötigten Anzahl an Parametern sind nicht scharf, in der Praxis erzielen Netzwerke mit deutlich weniger gelernten Parametern ähnlich gute Ergebnisse. Jedoch sind bei diesen die Entstehung derselben Parameter nicht erklärbar. Weiter wurde in [87, Thm. 2] für $k, d = 1$ gezeigt, dass sogar ein Netzwerk mit nur 6 Schichten ausreichend ist.

Approximation mit ReQU-Netzwerken

Nachdem wir gesehen haben, wie mithilfe von ReLU-Netzwerken Polynome und weitere Funktionen dargestellt werden können, möchten wir dieselbe Konstruktion auf die Aktivierungsfunktion mit nächsthöherem Grad übertragen. Dies tun wir in den folgenden Lemmas und Korollaren, wobei die Konstruktion in diesem Fall deutlich einfacher ist. Mit der ReQU-Aktivierungsfunktion $\mathrm{ReQU}(x) = (x)_+^2$ lassen sich anders als bei der ReLU die (zunächst univariaten) Funktionen $(x, y) \mapsto x \cdot y$ sehr leicht exakt darstellen. Wir zeigen nun in einigen Lemmas und Korollaren Konstruktionen für ReQU-Netzwerke, welche an die Konstruktionen für ReLU-Netzwerke aus dem vorigen Abschnitt angelehnt sind.

Lemma 1.24 Es existiert ein Netzwerk $\mathcal{N}$ mit Aktivierungsfunktion $\sigma = \mathrm{ReQU}$, einer Schicht und 4 Knoten, so dass

$$\mathcal{N}(x, y) = xy \quad \forall x, y \in \mathbb{R}$$

Beweis: Für $xy \geq 0$ ist $xy = \frac{1}{2}((x+y)^2 - x^2 - y^2)$ und für $xy < 0$ ist $xy = -\frac{1}{2}((x+y)^2 - x^2 - y^2) = \frac{1}{2}((x-y)^2 - x^2 - y^2$. Damit ist insgesamt $xy = \frac{1}{4}((x+y)^2 - (x-y)^2 + (-x-y)^2 - (-x+y)^2)$. □

Im Gegensatz zu ReLU-Netzwerken können wir die Identität $x \mapsto x$ nicht trivial durch ReQU-Operationen darstellen. Bemerke, dass die sog. Residuellen Netzwerke dieses Schritt oftmals nicht benötigen (c.f. Kapitel 1.62).

Korollar 1.25 Sei $\varepsilon > 0$. Es existiert ein Neuronales Netzwerk $\mathcal{N} \in \mathcal{N}_1$ mit ReQU-Aktivierungsfunktion, einer Schicht und vier Neuronen, sodass $\mathcal{N}(x) = x$ für alle $x \in \mathbb{R}$.

Beweis: Wir benutzen das vorherige Lemma 1.24 mit $x = 1x$. □

Dadurch lassen sich Polynome, anders als bei ReLU-Netzwerken, exakt darstellen.

Lemma 1.26 Sei $p \in \mathbb{P}_1^m$ gegeben. Dann existiert ein ReQU-Netzwerk $\mathcal{N}$ von $\lfloor \log_2 m \rfloor + 1$ Schichten und höchstens $5m\lfloor \log_2 m \rfloor + 4$ Neuronen, sodass $p(x) = \mathcal{N}(x)$ für alle $x \in \mathbb{R}$ ist.

Beweis: Wir zeigen wie in Lemma 1.20, dass wir Monome $p(x) = cx^m$ darstellen können. Die Darstellung für beliebige Polynome erfolgt dann durch eine weitere Schicht mit der Identitätsabbildung aus Korollar 1.25.
Sei $m = \sum_{i=0}^{\lfloor \log_2 m \rfloor} m_i 2^i$, $m_i \in \{0, 1\}$, also $cx^m = c \prod_{i=1}^{\lfloor \log_2 m \rfloor} x^{m_i 2^i}$. Damit benötigen wir $\lfloor \log_2 m \rfloor$ Multiplikationen und ebenso viele Iterationen der Aktivierungsfunktion. Nach Lemma 1.24 benötigen wir dafür $\lfloor \log_2 m \rfloor$ Multiplikationen/Schichten mit jeweils 4 Neuronen, zudem ein Neuronen für das Quadrieren, also zusammen ein Netzwerk mit $5\lfloor \log_2 m \rfloor$ Neuronen. Das Polynom ergibt dann aus einem Netzwerk mit $\lfloor \log_2 m \rfloor + 1$ Schichten und höchstens $5m\lfloor \log_2 m \rfloor + 4$ Neuronen. □

Dadurch sehen wir zum einen, dass wir beliebige Funktionen durch solche Netzwerke approximieren können.

Korollar 1.27 Die Menge der Netzwerke mit ReQU-Aktivierungsfunktion ist dicht im Raum $\mathcal{C}(\mathbb{R})$ bzgl. der kompakten Konvergenz.

Beweis: Die Menge der Polynome ist dicht in $\mathcal{C}(\mathbb{R})$ bzgl kompakter Konvergenz nach dem Satz von Weierstraß und wir können nach Lemma 1.26 Polynome genau angeben. Daher existiert zu jedem $\varepsilon > 0$, $f \in \mathcal{C}(\mathbb{R})$ und jeder kompakten Menge K ein ReQU-Netzwerk $\mathcal{N}$ mit $\|\mathcal{N} - f\|_{\mathcal{C}(K)} \leq \|\mathcal{N} - p\|_{\mathcal{C}(K)} + \|p - f\|_{\mathcal{C}(K)} < \varepsilon$. □

Und zum anderen können wir auch konstruktiv diese Polynome angeben um durch Faktorisierung folgenden Satz zu erhalten:

Korollar 1.28 Sei $p \in \mathbb{P}_d^n$ ein Polynom in d Variablen vom Grad höchstens n. Dann existiert ein ReQU-Netzwerk $\mathcal{N}$ mit $\lfloor \log_2 n \rfloor + 1$ Schichten und $5d\binom{n+d}{d}\lfloor \log_2 n \rfloor + 4$ Neuronen mit $p(x) = \mathcal{N}(x)$ für alle $x \in \mathbb{R}^d$.

Beweis: Analog zu den letzten Schritten nach der Approximation von $x \mapsto x^s$ im Beweis zu Lemma 1.20. Da durch ReQU-Netzwerke Multiplikation und Quadrieren exakt dargestellt werden, folgt Gleichheit mit dem Polynom aus der Konstruktion. □

Korollar 1.29 (zu Lemma 1.22) Sei $\varepsilon > 0$ und $f \in W^{k,\infty}([0,1]^d)$ mit $\|f\|_{k,\infty} \leq 1$ gegeben. Dann existiert ein ReQU-Netzwerk $\mathcal{N}$ mit $\mathcal{O}(\log_2 k)$ Schichten und $\mathcal{O}(\varepsilon^{-\frac{d}{k}} \log_2 k)$ Knoten sodass $\|f - \mathcal{N}\|_{k,\infty} \leq \varepsilon$.

Beweis: Verwende die Konstruktion aus 1.22 mit der Konstruktion für die Polynome aus 1.26. Außerdem lassen sich die Indikatorfunktionen ϕ_j als ReQU-Netzwerk mit zwei Schicht und 8 Knoten approximieren, indem wir $\mathrm{ReLU}(x) = \mathrm{ReQU}'(x) = \lim_{h\to 0} \frac{(x)_+^2 - (x+h)_+^2}{2h}$ berechnen. Damit benötigen wir für eine Multiplikation nur 4 Knoten in einer Schicht und für ein Taylorpolynom vom Grad $k-1$ nur $\lfloor \log_2 k - 1 \rfloor + 1$ Schichten und $5(k-1)\lfloor \log_2 k - 1 \rfloor + 4$ Neuronen und somit insgesamt $\lfloor \log_2 k - 1 \rfloor + 3$ Schichten und $5(k-1)N\lfloor \log_2 k - 1 \rfloor + 4 + 8$ Neuronen. Wählen wir $N^k \geq \frac{2^d d^k}{k!\varepsilon}$ wie vorher ergibt sich ein Netzwerk der Größe $\mathcal{O}(\varepsilon^{-\frac{d}{k}} \log_2 k)$. □

Approximation mit RePU-Netzwerken für $k > 2$

Der Fall von Aktivierungsfunktionen $(x)_+^k$ mit $k > 2$ ist bisher nur vereinzelt untersucht worden. Es stellen sich bei diesem Fall neue Probleme. Ist $k \geq 2$, so ist etwa nicht sofort klar, wie man weiterhin beliebige Polynome von geringerem Grad als k approximieren kann. Wir untersuchen einige Fälle und verweisen zudem auf den allgemeinen Ansatz zur Konstruktion von Polynomen durch RePU Netzwerke aus [49].
Zuerst können wir die Abbildung ReLU als (skalierte) $(k-1)$-ste Ableitung von $(x)_+^k$ durch Näherung der Ableitung annähern wie in Lemma 1.25 und auch andere abgeschnittene Potenzfunktionen $(x)_+^\ell$ für $\ell \leq k$.

Lemma 1.30 Sei $\varepsilon > 0, k > 2$. Es existiert ein Neuronales Netzwerk $\mathcal{N} \in \mathcal{N}_1$ mit Aktivierungsfunktion $(x)_+^k$, $(k-2)$ Schichten und vier Neuronen pro Schicht, sodass

$$|\mathcal{N}(x) - x| \leq \varepsilon$$

für alle $x \in \mathbb{R}$.

Beweis: Wir benutzen, dass für $x \geq 0$ gilt $\sigma'(x) = 2x$, d.h. $\lim_{h\to 0} \frac{(x+h)^2 - x^2}{2h} = x$ für alle $x > 0$. Wähle $h \leq 2\varepsilon$, dann ist $\mathcal{N}(x) := \frac{1}{2h}(\sigma(x+h) + \sigma(-x-h) - \sigma(x) - \sigma(-x)) \approx k(x)^{k-1}$. $k-1$-faches Anwenden von $\mathcal{N}$ mit $h^k \leq \varepsilon$ liefert das Geforderte. □

Es gibt ebenfalls exakte Darstellungen der Identität durch andere Konstruktion, die hier benutzte Konstruktion wurde im Kontext der Neuronalen Netzwerke noch nicht verwendet.

Korollar 1.31 Seien $\{x_1, \ldots, x_n\} \subset \mathbb{R}^d$ paarweise verschiedene Punkte gegeben und

$g : \mathbb{R}^d \to \mathbb{R}$ eine beliebige Funktion. Falls $\{0,1\} \subset \sigma(\mathbb{R})$, so existiert ein $\mathcal{N} \in \mathcal{N}_1$, sodass $\mathcal{N}(x_i) = g(x_i)$, $i = 1, \ldots, n$.

Beweis: Sei $h_i \in \mathcal{N}_1$ die Funktion mit $h_i(x_i) = 1$ und $h_i(x_j) = 0$ für $j \neq i$. Es ist $h \in \mathcal{N}_1$. Dann ist $f = \sum_{i=1}^n g(x_i) h_i(\cdot) \in \mathcal{N}_1$ ein Netzwerk mit der behaupteten Interpolationseigenschaft. □

Zur Rate der Approximation mit flachen Netzwerken mit RePU-Aktivierungsfunktion gibt der folgende Satz von Petrushev [63] Auskunft.

Satz 1.32 (Petrushev) Sei σ eine sigmoidale Aktivierungsfunktion und $f \in F^{k,p} = \{f \in W^{k,p}([-1,1]) \mid \|f\|_{k,p} \leq 1\}$. Dann gilt

$$E_{r,2}(f) \leq C r^{-m/d} \tag{1.8}$$

für $m = 1, \ldots, k + 1 + \frac{d-1}{2}$ und eine Konstante C unabhängig von r.

Insbesondere interessant an diesem Ergebnis ist die Konstruktion der Aktivierungsfunktion σ, welche als B-Spline der Ordnung k gewählt werden kann. Wir werden selbst weitere Approximationsraten über den Zusammenhang mit Splines in Abschnitt 1.3 herleiten.

1.2.3 Konstruktion von N-Term-Approximationen mit Neuronalen Netzwerken

Ein typischer Ansatz zur nichtlinearen Approximation ist es, sogenannte N-Term-Approximationen zu betrachten. Dabei geben wir eine Familie (evtl. sogar eine Basis) $\mathcal{F}$ von Funktionen aus einem Raum vor und bilden eine Approximation aus einer N-elementigen Teilmenge von $\mathcal{F}$. Wir können stets eine N-Term-Approximation mit einem Neuronalen Netzwerk bilden, indem wir diese Funktionen aus der Teilmenge durch ein Teilnetzwerk - bspw. über Polynome - approximieren und diese Netzwerke wie in der folgenden Grafik angedeutet zusammensetzen:

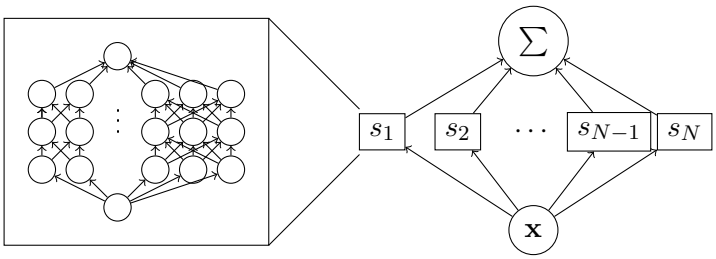

Abbildung 1.5: Struktur eines Netzwerks für eine N-Term-Approximation.

Dadurch lassen sich Kolmogorov-N-Weiten für die Approximation mit solchen Netzwerken berechnen, außerdem können Eigenschaften bestimmter Funktionenklassen auf Neuronale

Netzwerke übertragen. Dabei erhalten wir Netzwerke von endlicher Tiefe, deren Breite von N und der Breite der jeweiligen Teilnetzwerke abhängen.

Beispiel 1.33 Betrachten wir beispielsweise die Familie der Haar-Wavelets

$$\mathcal{F} = \{H_{j,k} \mid k \in \mathbb{Z}, j = 0, \dots, 2^k - 1\}$$

mit

$$H_{j,k}(x) = 2^{k/2} H(2^k x - j) \text{ und } H(x) = \begin{cases} 1, & 0 \le x < 1/2 \\ -1, & 1/2 \le x < 1 \end{cases} \tag{1.9}$$

auf einem Gebiet $\Omega \subset \mathbb{R}^d$ als Teilmenge von $L_2(\Omega)$.
Sei $\mathcal{F}_N$ die Menge aller N-Term-Approximationen aus $\mathcal{F}$, d.h. Funktionen der Form $\sum_{i=1}^N c_i H_{j_i,k_i} + c_0$. Über die N-Term-Approximation mit Haar-Wavelets ist bekannt (siehe bspw. [20]), dass gilt

$$\sigma_N(f)_2 = \sigma_N(f; \mathcal{F})_2 := \inf_{h \in \mathcal{F}_N} \|f - h\|_{L_2(\Omega)} \le M N^{-\alpha}$$

für alle $f \in L_2(\Omega)$ mit $\gamma_N(f) \le M N^{-\alpha - 1/2}$, wobei γ_N den N-größten Haar-Koeffizienten $\langle f, H_{j_i,k_i} \rangle$ bezeichnet.
Wir schreiben kurz $\lambda = (j,k)$. Seien nun ReLU-Netzwerke $\mathcal{N}_{\lambda_i}$ gegeben mit $\|\mathcal{N}_{\lambda_i} - H_{\lambda_i}\|_{L_2(\Omega)} \le \varepsilon_i$ und sei $\mathcal{N} = \sum_{i=1}^N \langle f, H_{\lambda_i} \rangle \mathcal{N}_{\lambda_i}$. Ein solches Netzwerk $\mathcal{N}_{\lambda_i}$ ist bspw. als einschichtiges Netzwerk gegeben durch Verschiebung und Skalierung von $\mathcal{N}_H(x) = \frac{1}{2\delta}\operatorname{ReLU}(x) - \frac{1}{2\delta}\operatorname{ReLU}(x-\delta) - \frac{1}{\delta}\operatorname{ReLU}(x-\frac{1}{2}) + \frac{1}{\delta}\operatorname{ReLU}(x-\frac{1}{2}-\delta) + \frac{1}{2\delta}\operatorname{ReLU}(x-1) - \frac{1}{2\delta}\operatorname{ReLU}(x-1-\delta)$ und Wählen von $\delta \le \frac{4}{3}\varepsilon_i$.
Wir haben damit ein Netzwerk $\mathcal{N}$ mit nur von N abhängender Komplexität (d.h. Breite und Anzahl der Knoten), das einer N-Term-Approximation mit Haar-Wavelets entspricht. Bezeichnen wir mit $\mathcal{H}_N$ den Raum der Netzwerke mit Komplexität N erhalten wir

$$\begin{aligned} \sigma_N(f, \mathcal{H})_2 &= \inf_{\mathcal{N} \in \mathcal{H}} \|f - h\|_{L_2(\Omega)} \\ &\le \inf_{h \in \mathcal{F}_N} \|f - h\|_{L_2(\Omega)} + \|h - \mathcal{N}\|_{L_2(\Omega)} \\ &\le \sigma_N(f, \mathcal{F})_2 + \varepsilon. \end{aligned}$$

Da in unserer Konstruktion ε unabhängig von der Größe N beliebig klein gewählt werden kann, ergeben sich asymptotisch in N, M dieselben Approximationsraten für den Raum $\mathcal{H}_N$ wie für $\mathcal{F}_N$.

1.3 RePU-Netzwerke und Splines

1.3.1 Approximation durch Splines mit freien Knoten

Hier behandeln wir zunächst die bekannten Erkenntnisse der Approximation mit sog. free knot splines, angewandt auf flache Netzwerke mit stückweise polynomiellen Aktivierungsfunktionen wie oben definiert. Wir erinnern an die Definition eines Splines von Grad r zu

einer Knotenfolge $X = \{a = x_0 < x_1 < \cdots < x_n = b\}$. Eine Funktion $s \in \mathcal{C}^{r-1}([a,b])$ heißt Spline, falls $s\big|_{[x_i,x_{i+1}]}$ ein Polynom vom Grad r für $0 \leq i \leq n-1$ ist.

Definition 1.34 Seien $n, r \in \mathbb{N}$ und $[a,b] \subset \mathbb{R}$ gegeben. Dann ist der Raum der Splines mit freien Knoten

$$\Sigma_{n,r} = \{f \mid f \text{ ist Spline vom Grad } r \text{ zu Knotenfolge } X\}$$

für eine beliebige (gültige) Knotenfolge $X \subset [a,b]$ mit $|X| = n$.
Sei $1 \leq p \leq \infty$. Wir bezeichnen weiter mit $\sigma_{n,r}(f)_p$ den Fehler der Approximation mit Splines mit freien Knoten

$$\sigma_{n,r}(f)_p := \inf_{s \in \Sigma_{n,r}} \|f - s\|_p.$$

Eine analoge Definition für multivariate Splines geben wir weiter unten in Abschnitt 1.3.4. Die Approximation aus $\Sigma_{n,r}$ durch Splines mit freien Knoten ist anders als die mit Splines zu gegebener Knotenfolge nichtlinear. Dadurch lassen sich nach DeVore [21, Kapitel 6] unter deutlich schwächeren Annahmen an die zu approximierenden Funktionen als bei fest gewählten Knoten, gute Approximationsraten herleiten. Zunächst bemerken wir, dass eine Bestapproximation bezüglich der meist verwendeten Normen existiert.

Satz 1.35 (aus [72, Kapitel 12, Theorem 4.1])
Zu $f \in L_p([0,1])$ für $0 < p < \infty$ oder $f \in \mathcal{C}([0,1])$ zu $p = \infty$, existiert ein $S^* \in \Sigma_{n,r}$ mit

$$\|f - S^*\|_p = \sigma_{n,r}(f)_p.$$

Es ist zu bemerken, dass die Existenz ebenso für beliebige kompakte Intervalle sichergestellt ist.

Satz 1.36 (aus ebd. [72, Theorem 4.5]) Sei $r \geq 1$ und $f \in \mathcal{C}^{r-1}([0,1])$. Ist $f^{(r-1)}$ von beschränkter Variation, so ist

$$\sigma_{n,r}(f)_\infty \leq \frac{C_r \mathrm{Var}_{[0,1]} f^{(r-1)}}{n^r}$$

für eine Konstante C_r.

Die Ergebnisse über Splines mit freien Knoten können wir auf flache Netzwerke mit ReQU- oder allgemeiner RePU-Aktivierungsfunktion übertragen.

Korollar 1.37 Sei f wie in Satz 1.36 und $\mathcal{N}_1$ wie in Definition 1.1 mit RePU-Aktivierungsfunktion vom Grad r. Dann ist

$$E(f, \mathcal{N}_{1,n})_\infty = \inf_{\mathcal{N} \in \mathcal{N}_{1,n}} \|\mathcal{N} - f\|_\infty \leq \frac{C_r \mathrm{Var}_{[0,1]} f^{(r-1)}}{n^r}. \tag{1.10}$$

Beispiel 1.38 Mithilfe dieser Ergebnisse lassen sich leicht Approximationsraten für viele Funktionen finden. Betrachten wir als Beispiel eine Approximation von $f : x \mapsto x^\alpha$ für $\alpha > 0$ auf $[0,1]$ mit ReQU-Netzwerken aus $\mathcal{N}_{1,n}$.
Dann fällt der Fehler nach Satz 1.36 mit $\sigma_{n,r}(f)_\infty \leq Cn^{-r}$, denn

$$\begin{aligned}\mathrm{Var}_{[0,1]} \frac{d^r}{dx^r} x^\alpha &= \int_0^1 |\frac{d^{r-1}}{dx^{r-1}} x^\alpha| \, dx \\ &= \prod_{k=-r}^{0} (\alpha + k).\end{aligned}$$

Weiter ist für ReQU-Netzwerke $E(f, \mathcal{N}_{1,n})_\infty = \sigma_{n,2}(f)_\infty$, denn wir erhalten für geeignete Gewichte eines Netzwerks mit ReQU-Aktivierungsfunktion einen stückweise quadratischen Spline mit beliebigen Knoten. Damit fällt der Fehler für diese Funktion mit

$$E(x^\alpha, \mathcal{N}_{1,n})_\infty \leq C_2 \prod_{k=-r}^{0} (\alpha + k) n^{-2},$$

wobei C_2 die Konstante aus dem obigen Korollar 1.37 für $r = 2$ ist.

Bemerkung 1.39 Die Approximation mit Splines mit freien Knoten ist eine nichtlineare Approximation, wie auch die Approximation mit Neuronalen Netzwerken mit nichtlinearer Aktivierungsfunktion. Daher ist insbesondere die Wahl der Knoten bzw. Gewichte ein schwieriges Problem. Zudem ist die Zuordnung der zu approximierenden Funktion zu den Gewichten nicht notwendigerweise stetig (bspw. ist die Bestapproximation mit Splines mit freien Knoten nicht eindeutig für viele Probleme). Um also Gewichte stabil bestimmen zu können, sind die Konstruktion von B-Splines und ein sinnvoller Schritt, den wir im nächsten Abschnitt untersuchen wollen.

1.3.2 Konstruktion von Splines durch Neuronale Netzwerke

Wir haben bisher einen engen Zusammenhang zwischen Neuronalen Netzwerken mit RePU-Aktivierungsfunktion und univariaten Splines mit freien Knoten gesehen. Wir zeigen, dass die flache Netzwerke mit Spline-Aktivierungsfunktionen im Eindimensionalen wieder univariate Splines bilden und dass sich tiefe Netzwerke für die Berechnung höherdimensionaler, nicht-separabler Splines eignen. Dadurch können wir direkt Ergebnisse zur Approximation mit Splines auf Neuronale Netzwerke übertragen.

Bemerkung 1.40 Jedes univariate neuronale Netzwerk $\mathcal{N} \in \mathcal{N}_{1,n}$ mit einer versteckten Schicht und einer abgeschnittenen Potenzfunktion als Aktivierungsfunktion kann in der Form

$$\sum_{i=1}^{m} \alpha_i (y_i)_+^s$$

mit $y_i = \omega_i x - \beta_i = \omega_i (x - \frac{\beta_i}{\omega_i})$ geschrieben werden. Für den Fall einer stückweise linearen

Aktivierungsfunktion, sprich der ReLU, ergibt sich daraus

$$\sum_{i=1}^{m} \alpha_i \omega_i (x - \tilde{\beta}_i)_+, \quad \tilde{\beta}_i = \frac{\beta_i}{\omega_i}.$$

Damit erhalten wir eine erste direkte Darstellung von Splines durch Neuronale Netzwerke. Dadurch können wir die vielen Ergebnisse zu Approximationseigenschaften von Splines hier übertragen. Für eine Zusammenfassung zu Splines siehe z.B. [6] oder [20, Kapitel 5, 7, 12]. Insbesondere für einschichtige ReLU-Netzwerke ist klar, dass deren Approximationseigenschaften genau denen von stückweise linearen Splines entsprechen und damit Ergebnisse dazu auf Neuronale Netzwerke übertragbar sind.

Wir können beispielsweise direkt über die Güte der Approximation mit flachen ReLU-Netzwerken aussagen, dass als Folgerung zu einem Resultat zu Splines aus [6]:

Satz 1.41 Sei $\mathcal{N} = \sum_{i=1}^{m} \alpha_i \omega_i (x - \tilde{\beta}_i)_+$ ein flaches ReLU-Netzwerk, mit fest gewählten, monoton wachsenden Gewichten $\tilde{\beta}_i$. Sei weiter $g \in \mathcal{C}^j([a,b])$ für ein Intervall mit $\tilde{\beta}_1 = a < b = \tilde{\beta}_m$ gegeben. Dann lassen sich die Koeffizienten α_i so wählen, dass

$$\|g - \mathcal{N}\|_\infty \leq c B^j \omega(g^{(j)}, B). \tag{1.11}$$

Dabei sei $B := \max_{1 \leq i \leq m-1} |\tilde{\beta}_i - \tilde{\beta}_{i+1}|$ und ω das Stetigkeitsmodul

$$\omega(f,t) = \sup_{\substack{|x-y| \leq t \\ x,y \in [a,b]}} |f(x) - f(y)|.$$

Beweis: Dies ist eine direkte Folgerung aus einem entsprechenden Resultat für Splines, denn für den Splineraum $\mathcal{S}_{k,t}$ gilt nach [6, Theorem XII.(6)], dass für die Knotenfolge $t = \{t_i\}$ und für alle $g \in \mathcal{C}^j([a,b])$

$$\operatorname{dist}(g, \mathcal{S}_{k,t}) \leq C_r |t|^j \omega(D^j g; |t|).$$

Wor wählen in unserer Darstellung $\tilde{\beta}_i = t_i$ und $B = |t|$. Für ReLU-Netzwerke ist $C_r = C_1 = c$ und somit folgt direkt die Aussage. □

Wir möchten nun neben der direkten Darstellung von Splines durch Neuronale Netzwerke auch B-Splines möglichst optimal darstellen. Wir werden je nach Knotenfolge andere Konstruktionen nutzen können. Ein B-Spline zu äquidistanten Knoten ist etwa einfach und exakt durch ein neuronales Netzwerk mit abgeschnittenen Potenzfunktionen als Aktivierungsfunktionen darstellbar: Der B-Spline vom Grad k zur Knotenfolge $\mathbb{Z}$ lässt sich in geschlossener Form angeben als

$$B_{j,k}(x) = \sum_{r=0}^{k} (-1)^{k-r} \binom{k}{r} \frac{1}{(k-1)!} (x - j + r)_+^{k-1}.$$

Dann können wir nun diesen durch folgende Konstruktionen aus abgeschnittenen Potenzfunktionen anderen Grades erhalten. Für $r = 1$ beruhen diese Konstruktionen auf Lemma

1.20, für die anderen Fälle sind diese Konstruktion von B-Splines mit Neuronalen Netzwerken neu.

Satz 1.42 Sei $\varepsilon > 0$. Sei $B^k = B_{0,k}$ der kardinale B-Spline zur Knotenfolge $\mathbb{Z}$ vom Grad k wie oben gegeben. Dann existiert ein Netzwerk $\mathcal{N}$ mit RePU-Aktivierungsfunktion $(x)_+^r$ vom Grad r, sodass gilt:

a) Ist $k - 1 = r$, so ist $B^k = \mathcal{N}$ für ein $\mathcal{N} \in \mathcal{N}_{1,k+1}$.

b) Ist $k < r$, so existiert ein Netzwerk $\mathcal{N}$ mit 2 Schichten und $2r + 2$ Neuronen mit $\|\mathcal{N} - B^k\|_\infty \leq \varepsilon$.

c) Ist $k > 1, r = 1$, so existiert ein Netzwerk $\mathcal{N}$ mit $\mathcal{O}(\ln(\frac{k^2-1}{\varepsilon})^2)$ Schichten und Neuronen, sodass $\|\mathcal{N} - B^k\|_\infty \leq \varepsilon$.

d) Ist $2 \leq r \leq k$, so existiert ein Netzwerk $\mathcal{N}$ mit $r\mathcal{O}(\ln k - r)$ Neuronen, sodass $B^k = \mathcal{N}$.

Beweis: a) Da die abgeschnittenen Potenzfunktionen bereits Grad $k - 1$ haben, ist die Konstruktion trivial: Wir wählen in diesem Fall $\omega_i \equiv 1, \beta_r = j - r$ und $\alpha_r = (-1)^{k-r}\binom{k}{r}\frac{1}{(k-1)!}$.

b) Für $k < r$ und $k \geq 1$ (d.h. keine ReLU-Netzwerke) können wir den binomischen Lehrsatz benutzen. Es gilt

$$(x - b_k)_+^r = \mathbb{1}_{x \geq b_k} \sum_{j=0}^{r} \binom{r}{j} x^j (-1)^{r-j} b_k^{r-j}.$$

Wir suchen Parameter b_k, α_k, sodass $\sum_{k=1}^n \alpha_k (x - b_k)_+^r = (x)_+^k$. Dies führt durch Koeffizientenvergleich auf ein Gleichungssystem mit $(r + 1)$ Gleichungen der Form

$$\begin{pmatrix} 1 & 1 & \cdots & 1 \\ \binom{r}{r-1} b_1 & \binom{r}{r-1} b_2 & \cdots & \binom{r}{r-1} b_n \\ \vdots & \vdots & \ddots & \vdots \\ \binom{r}{0} b_1^r & \binom{r}{0} b_2^r & \cdots & \binom{r}{0} b_n^r \end{pmatrix} \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_N \end{pmatrix} = \begin{pmatrix} \delta_{r,k} \\ \delta_{r-1,k} \\ \vdots \\ \delta_{0,k} \end{pmatrix}.$$

Für $n = r + 1$ paarweise verschiedene b_k ist dieses Vandermonde-System eindeutig lösbar und liefert Koeffizienten für ein Netzwerk $\mathcal{N}$ mit $\mathcal{N}(x) = 0$ für $x \leq b_{\min} := \min_{k=1}^n b_k$ und $\mathcal{N}(x) = x^k$ für $x \geq b_{\max} := \max_{k=1}^n b_k$. Für $x \in [b_{\min}, b_{\max}]$ gilt $\mathcal{N}(x) \leq n \max_{k=1}^n |\alpha_k| \binom{r}{[r/2]} |b_{\max}|$. Wähle $b_k < 0$ nahe an 0, so dass gilt $\|\mathcal{N}(\cdot) - (\cdot)_+^k\|_\infty \leq n \max_{k=1}^n |\alpha_k| \binom{r}{[r/2]} |b_{\min}| = \varepsilon$. Der B-Spline B^k wird dann durch ein Netzwerk mit 2 Schichten und insgesamt $r + 2$ Knoten approximiert wie in Fall a).

c) Wir verwenden die Approximation von Monomen x^{k-1} aus Lemma 1.20 und verwenden fügen eine weitere Schicht hinzu, sodass $(\mathrm{ReLU}(x))^{k-1} = (x)_+^{k-1}$. Nach Lemma 1.20 benötigen wir ein Netzwerk mit $\mathcal{O}(\ln(\frac{k^2-1}{\varepsilon})^2)$ Schichten und $\mathcal{O}(\ln(\frac{k^2-1}{\varepsilon})^2)$ Knoten, sodass x^{k-1} mit Genauigkeit $\frac{\varepsilon}{k+1}$ approximiert wird. Zusammensetzen von $k+1$ solchen Netzwerken ergibt wie in Fall a) den gewünschten B-Spline.

d) Wir benutzen ein veränderte Version der Konstruktion aus Lemma 1.26, indem wir zuerst bemerken, dass $x(x)_+^s = (x)_+^{s+1}$. Sei $\ell = k - 1 - r$. Ist ℓ klein, d.h. $\ell \leq r$, so wenden wir einmal die Aktivierungsfunktion an und multiplizieren ℓ-mal x zu dem Ergebnis, um $x \cdot x \cdot \cdots \cdot x \cdot (x)_+^r = (x)_+^{k-1}$ zu erhalten. Dafür benötigen wir ein Netzwerk mit $\ell+2$ Schichten und $(\ell+2)(2r+4)+4$ Knoten. Falls ℓ größer ist, wird es vorteilhaft Terme $(x)_+^r$ zu multiplizieren, bzw. zu verknüpfen wie im Beweis zu Lemma 1.26. In diesem Fall benötigen wir weniger Neuronen, nämlich $\mathcal{O}(\log_r \ell)$ Multiplikationen mit $(x)_+^r$ und höchstens r Multiplikationen mit x. Die so konstruierten $(x)_+^{k-1}$ setzen wir wie in a) zu B^k zusammen.
Es bleibt zu zeigen, dass die Multiplikation wie oben behauptet durch RePU-Netzwerke darstellbar ist. Für den Fall $r = 2$ ist dies Lemma 1.24. Sei nun $r > 2$. Wir wiederholen die Konstruktion aus b) für $(x)_+^2$ und $(-x)_+^2$ und verwenden dieselbe Formel wie in Lemma 1.24. Wählen wir jeweils b_k symmetrisch erhalten wir eine exakte Darstellung und führen diesen Fall somit durch ein Netzwerk mit 2 Schichten und $2(r+2)$ Neuronen auf den Fall $r = 2$ zurück. □

Dieser Satz lässt sich trivial auf Splines mit anderen äquidistanten Knoten/Bruchstellen übertragen.
Für die Darstellung von allgemeineren Splines mit nicht äquidistanten Knoten können wir ebenfalls analog vorgehen, mit entsprechend gewählten Gewichten. So ist die im Beweis zu b) verwendete Matrix für paarweise verschiedene Knoten stets invertierbar. Für eine numerisch stabilere Darstellung ist uns die bekannte Rekursionsformel für B-Splines hilfreich: Wir überführen diese in eine Baumartige Netzwerkstruktur. Die Rekursionsformel lautet ([6]):

$$B_{j,k}(x) = c(x - t_j)B_{j,k-1}(x) + (1 - c(x - t_{j+1}))B_{j+1,k-1}(x)$$

Das heißt, jeder Knoten muss als Input den Output zweier Neuronen der vorherigen Schicht und den Punkt der Auswertung x als Eingabe haben. Um dies zu bewerkstelligen wird in allen außer der Ausgabeschicht ein Identitäts-Knoten sein, der x weiterpropagiert, um daraus die Terme $c(x-t_j)$ zu berechnen. Zudem wird eine Multiplikation benötigt, diese kann für RePU-Aktivierungsfunktionen mit Grad größer gleich 2 exakt realisiert werden nach Lemma 1.19 bzw. 1.24. Wenn zudem die B-Splines vom kleinsten Grad $k = 1$ dargestellt werden können, erhält man durch folgende neue Netzwerkstruktur eine exakte Darstellung der B-Splines.

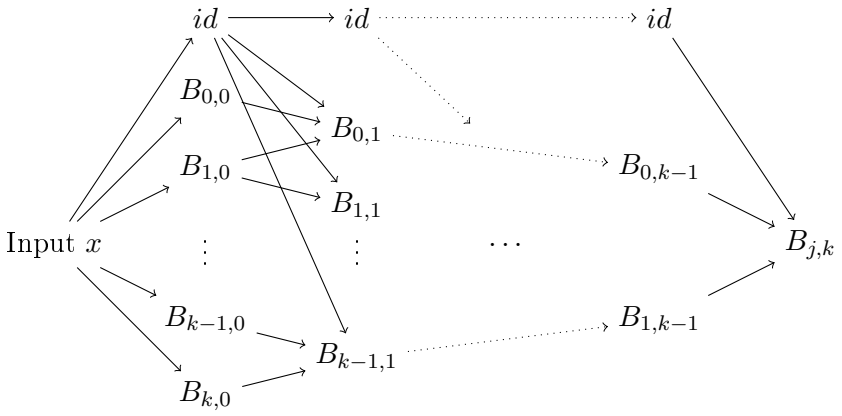

Abbildung 1.6: Eine k-schichtige Netzwerkarchitektur zur rekursiven Darstellung von B-Splines

Bemerkung 1.43 Die in Abbildung 1.6 vorgestellte Netzwerkarchitektur besitzt $\max\{m_b, m_{id}\} + (k-1)\max\{m_m, m_{id}\}$ Schichten und

$$\frac{k^2}{2}n_m + k(n_{id} + n_b - \frac{1}{2}n_m) + n_b \text{ Neuronen},$$

wenn jeder B-Splines $B_{j,0}$ durch ein Netzwerk mit Tiefe m_b und n_b Neuronen berechnet wird, die Identitätsabbildung durch ein Netzwerk der Grüße m_{id}, n_{id} und die Multiplikation durch m_m Schichten und n_m Neuronen.
Für ReQU-Netzwerke ist $m_m = m_{id} = 1$ und $n_m = n_{id} = 4$ nach 1.24, 1.25, während für ReLU-Netzwerke gemäß Lemma 1.19 $n_m = m_m = \mathcal{O}(\ln(\frac{1}{\varepsilon}))$ und $n_{id} = m_{id} = 1$. Wir untersuchen nun zuletzt noch die Stabilität dieser Netzwerke, falls Multiplikation oder $B_{j,0}$ nur approximativ berechnet werden können.
Sei $\mathcal{N}_{b,j}$ ein Netzwerk mit $\|\mathcal{N}_{b,j} - B_{j,0}\|_\infty \leq \varepsilon_1$ und $\mathcal{N}_m$ ein Netzwerk mit $\|\mathcal{N}_m(x,y) - xy\|_\infty \leq \varepsilon_2$ jeweils auf $[0,1]$ oder einem anderen gewählten kompakten Intervall. Dann ist der Fehler des Netzwerkes, das $B_{j,k}$ nähert, gemäß der Analyse aus [6, X.(1)-(3)], kleiner gleich $c(\varepsilon_1 + \varepsilon_2)^k$.

Diese mehrschichtige Architektur ist vorteilhaft, sobald man die abgeschnittenen Potenzfunktionen zu beliebigen Knoten nicht einfach darstellen kann. Für den rekursiven Ansatz müssen nur die einfachsten linearen oder stückweise konstanten B-Splines dargestellt werden. Ist $r = k - 1$, so berechnet man die B-Splines direkt durch eine gewichtete Summe von abgeschnittenen Potenzfunktionen des entsprechenden Grades. In diesem Fall ist ein beliebiger Spline durch ein zweischichtiges Netzwerk darstellbar, indem die erste Schicht die B-Splines darstellt und die zweite Schicht (Ausgabeschicht ohne nichtlineare Aktivierungsfunktion) deren gewichtete Summe.
Zudem können wir eine Darstellung mit Netzen endlicher Breite über die Ableitungen der abgeschnittenen Potenzfunktionen herleiten. Dazu bemerken wir folgendes Lemma mit dem Differenzenoperator $\Delta_h^1(\phi, x) = \phi(x+h) - \phi(x)$ und $\Delta_h^r := \Delta_h^1 \Delta_h^{r-1}$.

Lemma 1.44 ([6, Kapitel 7, §7]) Sei $\phi_k(x) := (x)_+^k$ und $k > r$. Es gilt

$$\phi_k(x) = \frac{1}{r!} \lim_{h \to 0} h^{-r+k} \Delta_h^{r-k}(\phi_r, x) = \frac{1}{r \cdots (r-k+1)} \frac{d^{r-k}}{dx^{r-k}} \phi_r(x)$$

Mit diesem Lemma erhalten wir eine weitere neue Darstellungsmöglichkeit für Splines, indem wir die Ableitungs- bzw. Differenzenoperatoren nähern. So wissen wir, dass wir mit RePU-Netzwerken für $k \geq 2$, die identische Abbildung und Translationen dargestellt werden können. Verbindet man diese, so erhält man den Operator Δ_h^1. Durch die rekursive Konstruktion von Δ_h^{r-k}, kann $\frac{1}{r!} \lim_{h \to 0} h^{-r+k} \Delta_h^{r-k}(\phi_n, x)$ durch ein $(r-k-1)$-schichtiges Netzwerk mit endlicher Breite, das in jeder Schicht Δ_h^1 anwendet, dargestellt werden. Wir formulieren und beweisen dies hier neu als Lemma:

Lemma 1.45 Sei $\varepsilon > 0$, $r \geq k$ und eine Knotenfolge X gegeben. Es existiert ein RePU-Netzwerk $\mathcal{N}$ mit Aktivierungsfunktion vom Grad r und

$$\|\mathcal{N} - B_{j,k}\|_\infty < \varepsilon$$

mit endlicher, nicht von k und ε abhängiger Breite.

Beweis: Nach Lemma 1.44 ist $B_{j,k}(x) = \sum \frac{d_k}{r!} \lim_{h \to 0} h^{-r+k} \Delta_h^{r-k} \phi_r(x - x_k)$ für geeignete Gewichte d_k. Wir können den Operator Δ_h^{r-k} wie oben beschrieben durch ein Netzwerk mit $r-k-1$ Schichten ausdrücken, wobei jede Schicht $(x+h)_+^r - (x)_+^r$ berechnet. Da die Breite nicht von k abhängen soll, berechnen wir die abgeschnittenen Potenzfunktionen nicht parallel, sondern sequentiell wie in Abbildung 1.7. Jede der Summen- und Identitätsknoten benötigt mit der Konstruktion aus dem Beweis von Satz 1.42b) nicht mehr als 2 Schichten und $2r+2$ Neuronen. Um insgesamt Genauigkeit ε zu erhalten muss die genäherte Summation $\tilde{\Sigma} d_j \phi_k(x - x_j)$ erfüllen $|x \tilde{+} y - (x+y)| \leq \frac{\varepsilon}{k}$. Nach der Konstruktion hängt die Größe des Netzwerks nicht von der gewählten Genauigkeit ab. □

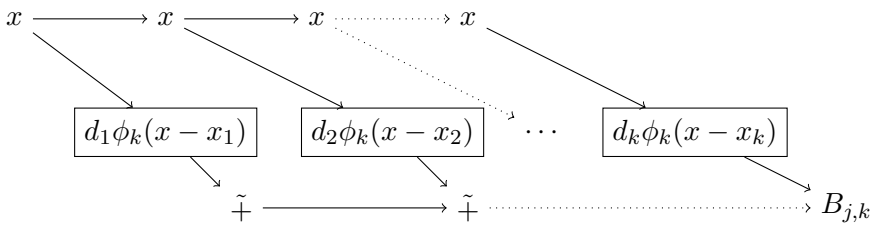

Abbildung 1.7: Struktur des tiefen Netzwerks zur Berechnung von $B_{j,k}$.

Nun können wir durch die exakte Realisierung der Splines mit der in Satz 1.42 bzw. danach gegebenen Konstruktion weitere Ergebnisse zur Splineapproximation auf Neuronale Netzwerke übertragen:
Wir übertragen weitere Ergebnisse der Splineapproximation ([72]) in L_p-Räumen durch

unsere obige Konstruktion auf RePU-Netzwerke. Wir betrachten wieder Splines mit n äquidistanten Knoten vom Grad r und die Approximation mit diesen in L_p-Räumen auf dem Einheitsintervall $[0,1]$. Wir bezeichnen den Fehler der Approximation wie in Definition 1.13 mit

$$E_{n,p}(f) := E(f, \mathcal{N}_{1,n})_p = \min_{g \in \mathcal{N}1,n} \|f - g\|_p$$

und den Fehler der Splineapproximation mit

$$\sigma_{n,r}(f)_p = \min_{g \in \Sigma_{n,r}} \|f - g\|_p.$$

Für Netzwerke mit m Schichten und insgesamt n Neuronen schreiben wir

$$E_{m,n,p}(f) := E(f, \mathcal{N}_{m,n})_p = \min_{g \in \mathcal{N}m,n} \|f - g\|_p. \tag{1.12}$$

Korollar 1.46 Sei $1 \leq p \leq \infty$. Für die Approximationsordnungen der RePU-Netzwerke mit Aktivierungsfunktion von Grad r und gleichmäßigen Bruchstellen der resultierenden stückweise polynomiellen Funktion gilt

1. $E_{n,p}(f) = \mathcal{O}(n^{-r})$ genau dann, wenn $f \in W^{p,r}([0,1])$.
2. $E_{n,p}(f) = \mathcal{O}(n^{-\alpha})$, für $f \in \mathrm{Lip}([0,1])$.

Beweis: Wir zeigen diese Anwendung von Theorem zu Splines neu für Neuronale Netzwerke. Nach Satz 1.42 und folgenden Bemerkungen können wir in dem gegebenen Fall die B-Splines B^k zu beliebiger Knotenfolge exakt durch Neuronale Netzwerke der Breite $k+1$ für den Fall, dass $k-1 = r$ bzw. mit Netzwerken der Tiefe 2 bzw. $4\log_r(k-r-1)$, falls $2 \leq r \leq k$, darstellen. Damit gilt im ersten Fall, dass $\sigma_{n,r}(f)_p = E_{n,p}(f)$ und somit folgt die Aussage aus Lemma 1.45 und den entsprechendem Theorem für Splines [72, Chapter 12, §3, Theorem 2.4].
Im zweiten Fall ist $E_{k+1,4\log_r(k-r-1),p}(f) \sim \sigma_{n,r}(f)_p$ und die Aussage folgt ebenfalls. □

Dadurch haben wir die bekannten Schranken für die Neuronalen Netzwerke auf einem neuen Weg erhalten.

Satz 1.47 Sei $\mathcal{N}_{1,n}$ die Menge der flachen Neuronalen Netzwerke mit RePU-Aktivierungsfunktion der Ordnung k mit n Neuronen. Dann ist

$$\inf_{g \in \mathcal{N}_{1,n}} \|f - g\|_p = \mathcal{O}(n^{-k})$$

für $f \in W^{k,p}([0,1])$ für $1 < p \leq \infty$.

Beweis: Durch die Darstellung von B-Splines der Ordnung $k+1$ als Neuronale Netzwerke aus Satz 1.42 folgt das Resultat direkt aus den Resultaten für Splines in 1.46. □

Dies ist in Übereinstimmung mit anderen Ergebnissen über Neuronale Netzwerke (siehe Pinkus [65], Poggio und Mhaskar[55], [38] uvm.)

Oftmals hängt die Güte der Approximation von der Wahl der Knoten ab, da diese bei Neuronalen Netzwerken durch die Wahl der Parameter β_i gesteuert werden, ist eine Anpassung leicht möglich. So ist beispielsweise die Approximation mit dyadischen Splines mit den für n wachsend geschachtelten Stützstellen $\{t_j\}_1^{2^n-1}, t_j = jw^{-2}$ leicht durch ein tiefes Netzwerk möglich. Bemerke dafür, dass die Funktion s^r aus 1.63 Nullstellen in den Punkten $t_j = \frac{j}{2^r}$, $j = 0, \dots, 2^r$ hat. Daraus können wir eine stückweise polynomielle Funktion mit Knoten in diesen Nullstellen konstruieren.
Die Elemente des nichtlinearen Raums der Splines vom Grad r mit n beliebig gewählten Knoten im Intervall I hängen von $2n + r$ Parametern ab. Wir können über die Beziehung von RePU-Netzwerken zu Splines ebenfalls obere Grenzen zur Approximation aufzeigen. Aus der Theorie über Splines sind die folgenden Ergebnisse bekannt:

Lemma 1.48 ([72, §7, 7.6]) Für $1 \leq p \leq \infty$ und $f \in \mathcal{C}^r(I), I = [0,1]$ gilt für den Approximationsfehler an f durch ein Neuronales Netzwerk mit $2n$ Knoten und Aktivierungsfunktion von Grad r gemessen in der p-Norm, dass

$$\sigma_{2n,r}(f)_p = o(n^{-r})$$

nur dann möglich ist, falls $f \in \mathbb{P}_{r-1}$.

Satz 1.49 ([72, Chapter 12, §4]) Für $f \in \mathcal{C}(A), M > 0$ gilt

$$\sigma_n(f)_\infty \leq \frac{M}{2n+2} \Leftrightarrow \mathrm{Var}_A(f) \leq M$$

Es folgt daraus, dass eine Approximation mit stückweise konstanten Splines, trotz freier Knoten eine stetige Funktion in ∞-Norm nur dann mit Fehler $\sigma_n(f)_\infty = o(n^{-1})$ approximieren kann, wenn diese Funktion bereits konstant ist. Daraus folgt insbesondere in unserem Fall, dass Funktionen wie $\mathbb{1}_{x \geq 0}$ keine geeigneten Aktivierungsfunktionen neuronaler Netzwerke sind. Für die RePU-Aktivierungsfunktion $(x)_+^k$ mit $k \geq 1$ gilt diese starke Einschränkung nicht, diese folgen nach Korollar 1.37 einer optimalen Approximationsrate von $\mathcal{O}(n^{-r})$.
Damit ist ein höherer Grad der stückweise polynomiellen Akitivierungsfunktionen zur Approximation glatter Funktionen von quantifizierbarem Vorteil. Es gilt zusammengefasst für Splines mit [72, Chapter 7, §7, 7.1 und 7.2] und damit in Übertragung auf Neuronale Netzwerke durch Satz 1.42:

Lemma 1.50 Gegeben sei die Approximation im Spline-Raum $S_{T,r}$ von Splines mit Knoten in T und Grad r durch den Linearen Operator zur B-Spline Basis $L_T(f) := \sum_j f(\xi_j) N_j$, wobei die ξ_j aus $\mathrm{supp} N_j$ gewählt sind. Für $f \in \mathcal{C}(I)$ und $\delta := \max_j(t_{j+1} - t_j)$ gilt dann

$$\mathrm{dist}(f, S_{T,r}) \leq \|f - L_T(f)\|_\infty \leq r\omega(f, \delta)_\infty$$

und weiter existiert eine Konstante C_r in Abhängigkeit nur von r mit

$$\mathrm{dist}(f, S_{T,r}) \leq \|f - L_T(f)\|_\infty \leq C_r \delta^k \omega(f, \delta)_\infty.$$

Des weiteren gilt $\operatorname{dist}(f, S_{T,r}) = C_r n^{-r}$ für alle $f \in W^{\infty,r}$.

Approximation durch unvollständige Teilmengen von $\Sigma_{n,r}$ und gemischte Aktivierungsfunktionen

Wir möchten an dieser Stelle auf zwei mögliche Verallgemeinerungen von den bisherigen Ergebnissen hinweisen.
Wir haben gesehen und werden noch sehen, dass durch Neuronale Netzwerke mit RePU-Aktivierungsfunktion vom Grad k Polynome und (B-)Splines von Grad r für sowohl $k \geq r$ als auch $k \leq r$ konstruiert werden können, mit verschiedenen Konstruktionen. Für einige Grade ist diese Konstruktion exakt, während sie für andere nur approximativ ist. So ist zum Beispiel die Konstruktion von stückweise konstanten Splines aus stetigen Aktivierungsfunktionen nicht exakt möglich. Trotzdem ist die Menge der rekonstruierbaren Polynome ohne einzelne Grade dicht in den Räumen $\mathcal{C}([0,1])$ bzw. $L_p([0,1])$, wie sich beispielsweise durch den Satz von Müntz leicht einsehen lässt.
Des weiteren haben wir bei allen Konstruktionen Netzwerke betrachtet, bei den in allen Schichten und Knoten dieselbe Aktivierungsfunktion verwendet wurde. Dies muss nicht notwendigerweise so sein. Es existiert auch eine Entsprechung für Splines, nämlich solchen mit freien Knoten und variablem Grad wie etwa für einen Spezialfall in [19] oder allgemeiner in [17].

Neuronale Netzwerke als Quasi-Interpolationsoperatoren

Die Ansätze von Quasi-Interpolation und der praktischen Anwendung von Neuronalen Netzwerken zeigen einige Übereinstimmungen. Wird eine Approximation einer Funktion aus einer Menge von Daten Ξ erstellt, so ist in beiden Fällen nicht die Erfüllung einer Interpolationsbedingung $\mathcal{N}(\xi)f(\xi)$ für alle $\xi \in \Xi$ gefordert. Für eine ausführliche Beschreibung der Quasi-Interpolation siehe etwa [8]. Allgemein sind Quasi-Interpolationsoperatoren Q für Funktionen f aus einer Klasse $\mathcal{F}$ von der Form

$$Q(f)(x) := \sum_{j \in \Lambda} \gamma_j(f)\phi_j(x),$$

wobei γ_j lineare Funktionale auf $\mathcal{F}$ sind. Bei Neuronalen Netzwerken, insbesondere mehrschichtigen, ist die Approximation und die Bestimmung der Gewichte jedoch inhärent nichtlinear durch einerseits den nichtlinearen Einfluss der Gewichte auf die Berechnung in jeder Schicht durch nichtlineare Aktivierungsfunktionen und andererseits durch die praktische Optimierung dieser Gewichte durch Algorithmen wie Backpropagation aufbauend auf stochastischem Gradientenabstieg oder Varianten davon. Für Neuronale Netzwerke mit fest gewählten Gewichten wie bspw. durch die Konstruktionen $\mathcal{N}_j$ für Polynome oder Splines aus Lemmata 1.20, 1.26, 1.45, 1.55 erhält man analog zu Splines einen Quasi-Interpolationsoperator durch

$$Q_N(f)(x) := \sum_{j} \gamma_j(f)\mathcal{N}_j(x),$$

wobei die γ_j dann lineare Funktionale auf $\mathcal{F}$ gegeben durch etwa Funktionsaufwertungen $\gamma_j(f) = f(x_j)$, an gewissen (bspw. auf einem Gitter liegenden) Datenpunkte x_j sind. Durch die exakte Darstellung der Splines aus den Netzwerken bleiben die Eigenschaften des Operators erhalten.
Weitere tiefergehende Eigenschaften von Quasi-Interpolationsoperatoren oder Verallgemeinerungen davon für Neuronale Netzwerke verschiedener Architekturen sind Gegenstand aktueller Forschung wie bspw. für Bernstein-artige Operatoren in [16] und für sigmoidale Aktivierungsfunktionen in [15].

Nullstellen

Wir möchten einige kurze Beobachtungen zur Anzahl an Nullstellen eines Neuronalen Netzwerks anschließen. Die Zahl der Nullstellen kann als Maß für die Komplexität einer Funktion diesen, bspw. über die VC (Vapnik-Chervonenkis)-Dimension ([78]) oder die Rademacher-Komplexität ([74]). Für Splines lässt sich die Anzahl leicht aus den Eigenschaften der polynomiellen Stücke bestimmen. Es gilt etwa der folgende Satz

Satz 1.51 ([72, Kapitel 5, Theorem 8.2f.]) Sei S eine Splinefunktion mit Intervallen I_j auf denen S ein Polynom ist. Sei $I = [a, b] \subseteq \text{supp}(S)$ gegeben. Dann ist die Anzahl der Nullstellen Z im Intervall (a, b) gleich

$$Z \leq \sigma - L(\Phi) - 2$$

Dabei ist Φ das „Diagramm", d.h. die Menge alle Punkte $(x, y) \in \mathbb{R}^2$ für die $x \in I_j$ und $y \leq j$. Sei dann $L(\Phi)$ die Anzahl dieser Rechtecke und weiter sei σ die Anzahl der singulären Paare.
Wir können dieses für unsere Zwecke einfacher ausdrücken als:
Sei S eine Splinefunktion von Grad r und $I = [c, d]$ ein Intervall, das s Trägerintervall von S enthält. Dann ist

$$Z \leq k(I) - s(r + 1)$$

mit $k(I)$ die Anzahl von Knoten im Intervall I.

Wir betrachten nun als neue Erweiterung die Komposition von Splines als Teil tiefer Neuronaler Netzwerke und leiten daraus eine Schranke für deren Anzahl von Nullstellen her.

Satz 1.52 Seien $r, \ell, n \in \mathbb{N}$ gegeben. Dann besitzt ein (univariates) Neuronales Netzwerk mit RePU-Aktivierungsfunktion vom Grad r, ℓ Schichten und Breite höchstens n

$$(n - r)^\ell$$

Nullstellen auf seinem Träger.

Beweis: Nach Satz 1.51 besitzt jede Schicht des Netzwerks als Spline vom Grad $r-1$ höchstens $n-r$ Nullstellen. Sei $\mathcal{N} = f_\ell \circ \tilde{f}$ und seien $x_1, \ldots, x_{n-r}$ die (nicht notwendigerweise verschiedenen) Nullstellen der ℓ-ten Schicht f_ℓ. Es folgt, dass das Netzwerk $\mathcal{N}$ genau $\sum_{i=1}^{n-r} |\tilde{f}^{-1}(x_i)|$ Nst. besitzt und induktiv folgt, dass die Anzahl der Nullstellen von $\mathcal{N}$ somit $\leq (n-r)^\ell$ ist. □

1.3.3 Tiefe ReQU-Netzwerke

In den bisherigen Abschnitten sind uns durch einige Konstruktionen bereits tiefe Netzwerke begegnet. Für ReLU-Netzwerke ist bekannt (siehe bspw. [41, Theorem 3.2]), dass universelle Approximationen auch mit sehr tiefen Netzwerken bei geringer Breite möglich ist, ein entsprechendes Resultat möchten wir hier erstmals für ReQU-Netzwerke formulieren.

Satz 1.53 Die Menge der tiefen Netzwerke mit beliebig vielen Schichten á höchstens 8 Knoten und ReQU-Aktivierungsfunktion ist dicht in $\mathcal{C}(\mathbb{R})$ bezüglich der Konvergenz auf kompakten Mengen.

Beweis: Wir modifizieren die Konstruktion für Polynome aus 1.26 für Netzwerke von möglichst geringer Breite.
Dazu benutzen wir die Darstellung von Polynomen durch das Horner-Schema, indem wir ein Polynom $p(x) = \sum_{i=0}^{n} a_i x^i$ durch n-maliges Addieren und Multiplizieren darstellen. Nach Lemma 1.24 lässt sich Multiplikation mit ReQU-Netzwerken der Breite 4, das eine Muliplikation darstellt, sowie nach Lemma 1.25 gilt dies ebenso für die identische Abbildung. Dies führt zu einem Netzwerk mit n Schichten und 8 Neuronen pro Schicht, welches nach dem Aufbau in Grafik 1.8 das Polynom berechnet.
Nach dem Satz von Weierstraß (z.B. [67, Theorem 6.1]) sind Polynome dicht in $\mathcal{C}(\mathbb{R})$ bzgl. der kompakten Konvergenz, also gilt dies auch die Menge der ReQU-Netzwerke mit Breite höchstens 8. □

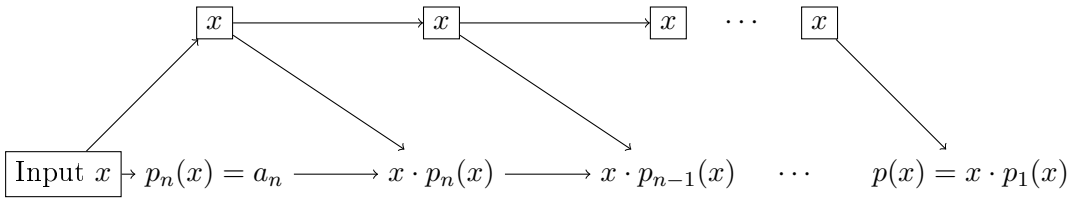

Abbildung 1.8: Struktur eines ReQU-Netzwerks zur Approximation von Polynomen mit möglichst geringer Breite

1.3.4 Mehrdimensionaler Fall

Die einfachsten mehrdimensionalen Splines sind Tensorprodukt-Splines. Sehen wir uns zunächst an, dass sich Neuronale Netzwerke einfach auf mehrere Dimensionen verallgemeinern

lassen: Ein Netzwerk mit einer versteckten Schicht und abgeschnittener Potenzfunktion als Aktivierungsfunktion ergibt sich als

$$\sum_{i=1}^{m} \alpha_i (\langle \boldsymbol{\omega}_i, x \rangle - \beta_i)_+^s,$$

d.h. die Funktion bleibt eindimensional, nur durch die Anwendung eines Skalarprodukts auf den Input und die nun als Vektor gegebenen Gewichte $\boldsymbol{\omega}_i$ wird der höherdimensionale Input zu einem Skalar für die weiterhin univariate Aktivierungsfunktion $(\cdot)_+^s$. Dadurch ergibt sich zunächst ein Unterschied zu den Tensorprodukt-Splines, welche als Produkt der univariaten Splines definiert sind. $T(x) = \prod_{l=1}^{d} B_l(x_l)$. Wir müssen also uns zunächst klar werden, dass durch $\sum_{i=1}^{m} \alpha_i (\langle \boldsymbol{\omega}_i, x \rangle - \beta_i)_+^s$ ein stückweises Polynom definiert wird. Wir können o.B.d.A. annehmen, dass $\beta_i = 0$, da dieser Term wird als Verschiebung nichts an der Aussage ändert.

Lemma 1.54 Die Funktionen der Form

$$\sum_{i=1}^{m} \alpha_i (\langle \boldsymbol{\omega}_i, x \rangle)_+^s$$

mit $\alpha_i \in \mathbb{R}, \boldsymbol{\omega}_i \in \mathbb{R}^d$ sind stückweise Polynome auf Polytopen im $\mathbb{R}^d$.

Beweis: Wir zeigen es für $(\langle \boldsymbol{\omega}, x \rangle)_+^s$. Die Hyperebenen $\langle \boldsymbol{\omega}_i, x \rangle = 0$ teilen den $\mathbb{R}^d$ in konvexe Polytope. In jedem dieser Gebiete ist entweder $(\langle \boldsymbol{\omega}_i, x \rangle)_+^s = 0$ und somit ein Polynom oder $\langle \boldsymbol{\omega}_i, x \rangle \geq 0$ und somit gleich $(\langle \boldsymbol{\omega}_i, x \rangle)^s$ ein Polynom. □

Anders als bei Tensorprodukt-Splines sind hier die Grenzen zwischen den polynomiellen Gebieten Hyperebenen, die nicht unbedingt orthogonal auf den Achsen stehen.

Satz 1.55 Sei $f \in \mathcal{C}^{d+3}([0,1]^d)$. Es existiert ein Neuronales Netzwerk $\mathcal{N}$ mit Breite N, Tiefe unabhängig von N und ReLU-Aktivierungsfunktion, sodass

$$\|f - \mathcal{N}\|_\infty = \mathcal{O}(N^{-2}).$$

Die Gewichte des Netzwerks $\mathcal{N}$ hängen zudem stetig von der Funktion f ab. Die Netzwerkstruktur kann gewählt werden aus $\mathcal{O}(N)$ parallelen Blöcken mit identischer Topologie, die nur von d abhängt.

Beweis: Wir verbessern das Vorgehen vom Beweis zu Lemma 1.22, bei dem wir den Bereich Tensor-artig aufgeteilt haben, indem wir eine natürlichere Aufteilung in Simplizies und Teilnetzwerke wie im Abschnitt zur N-Term-Approximation benutzen. Die benutzte Triangulierung ist bekannt aus [62]. Wir folgen zunächst dem Beweis von [62, A.2-A.6] und gehen in drei Schritten vor, dabei vereinfachen wir das Vorgehen in den Schritten 2 und 3 und erhalten durch Benutzung von Ergebnissen über Splines zusätzlich das Resultat zur Rate der Approximation.

1. Triangulierung und Approximation von f auf Gitter durch $\tilde{f}_1$.
Wir triangulieren das Gebiet $[0,1]^d$ standardmäßig durch Simplizies mit Eckpunkten $x_{\mathbf{n}} = \mathbf{n}/N$ für $\mathbf{n} \in \{0,\dots,N\}^d$, etwa durch $d+1$ jeweils $d-1$-dimensionale Schnitte. (Siehe Grafik 1.9 für ein Beispiel in $d=2$) Sei zu $\mathbf{n} = (n_1,\dots,n_d) \in \mathbb{Z}^d$ und $\rho \in S_d$ einer Permutation der Menge $\{1,\dots,d\}$ der Punkt $\mathbf{n}_\rho$ definiert als $\mathbf{n}_\rho = (n_{\rho(1)},\dots,n_{\rho(d)})$. Eine Menge der Form

$$\Delta_{\mathbf{n},\rho} = \left\{ x \in \mathbb{R}^d \,\middle|\, x = \sum_{j=1}^{d} c_j \mathbf{n}_\rho \text{ und } \sum_{j=1}^{d} c_j = 1 \right\}$$

ist ein Simplex dieser Triangulation. Nenne die daraus resultierende Triangulierung von $[0,1]^d$ P_N.
Dann ist jeder Punkt $\frac{n}{N}$ Ecke von höchstens $(d+1)!$ Simplizes. Die Vereinigung dieser Simplizies ergibt eine konvexe Menge, auf der wir eine Funktion definieren wollen.
Sei $\phi_{\mathbf{n}} : \mathbb{R}^d \to \mathbb{R}$ stetig und stückweise linear auf allen Simplizies der Triangulierung von oben mit $\phi_{\mathbf{n}}(\mathbf{n}) = 1$ und $\phi_{\mathbf{n}}(\mathbf{m}) = 0$ für alle $\mathbf{n} \neq \mathbf{m} \in \mathbb{Z}^d/N$, sowie $\phi_{\mathbf{n}} \equiv 0$ außerhalb der Simplizies. Dadurch ist $\phi_{\mathbf{n}}$ eindeutig festgelegt.
2. Diese Funktion kann durch lineare und ReLU-Operationen ausgedrückt werden. Wir betrachten dazu o.B.d.A. die Funktion ϕ_0 um den Ursprung. Sei $R_1,\dots,R_{(d+1)!}$ die Menge der Simplizes der in 1. konstruierten Zerlegung auf ganz $\mathbb{R}^d$ ausgeweitet mit 0 als einer Ecke. Dann ist für alle Simplizes R_j eindeutig eine affin lineare Abbildung $\ell_j : \mathbb{R}^d \to \mathbb{R}$ gegeben mit $\ell_j(0) = 1$ und $\ell_j = 0$ auf der Facette/Kante des Simplex R_j gegenüber von 0. Diese kann als stückweise affin lineare Funktion durch ein Neuron mit ReLU-Aktivierungsfunktion dargestellt werden.
Dann können wir ϕ_0 definieren als $\phi_0(x) := (\min_j \ell_j(x))_+$.
Diese Funktion lässt sich durch ein Netzwerk mit $2(d+1)! - 1$ Neuronen und $(d+1)! - 1$ Schichten realisieren, denn $\min(x,y) = x - \mathrm{ReLU}(x-y)$. Da $\ell_j(x) \geq 0$, ist $\min_{1\leq j\leq (d+1)!} \ell_j(x) = \ell_1(x) - (\ell_1(x) - (\ell_2(x) - (\cdots)_+ \cdots)_+$.
Das so konstruierte ϕ_0 besitzt die geforderten Eigenschaften: Ist $x \notin \bigcup_{1\leq j\leq (d+1)!} R_j$, so existiert ein R_j mit $\ell_j(x) \leq 0$ (da x außerhalb der Simplizes um 0 liegt), also verschwindet ϕ außerhalb von $S := \bigcup_j R_j$.
Auf S ist ϕ_0 wiederum nichtnegativ, da jedes ℓ_j nichtnegativ ist. Denn S ist konvex und für jedes $1 \leq k \leq (d+1)!$ und $x \in R_k$ ist $\ell_j(x)$ größer gleich der Werte von $\ell_j(v)$ an den Ecken v von R_k.
3. Wir interpolieren die Funktion und schätzen den Fehler ab. Wir definieren die stückweise lineare Interpolante f_1 via

$$f_1(x) = \sum_{\mathbf{n}\in\{0,\dots,N\}^d} f(x_{\mathbf{n}})\, \phi_{\mathbf{n}}(x)$$

Diese Interpolante ist nach Konstruktion linear auf jedem Simplex und interpoliert f an allen Punkten $x_{\mathbf{n}}$.
Zuletzt müssen wir den Fehler der so konstruierten Interpolante abschätzen. Es gilt nach

[5, IV.60 Theorem], dass

$$\|f - f_1\|_\infty \leq cN^{-2}.$$

Da $\mathcal{N}$ die Funktion f_1 exakt darstellt, gilt das Behauptete für den Fehler $\|f - \mathcal{N}\|_\infty$. □

Korollar 1.56 Sei $f : \mathbb{R}^d \to \mathbb{R}$ Lipschitz-stetig. Dann gilt für das Netzwerk $\mathcal{N}$ aus Satz 1.55, dass

$$\|f - \mathcal{N}\|_\infty = \mathcal{O}(N^{-1}).$$

Beweis: Wir betrachten wieder das Netzwerk aus dem Beweis zu Satz 1.55 und schätzen den Fehler der Interpolante f_1 unter der vorausgesetzten Lipschitz-Stetigkeit mit Lipschitz-Konstante L ab.
Mit den Bezeichnungen für die Zerlegung von oben erhalten wir

$$\|f - f_1\|_\infty = \sup_{x \in [0,1]^d} |f(x) - f_1(x)| = \max_{R_j \in P_N} \sup x \in R_j |f(x) - f_1(x)|.$$

f_1 ist linear auf jedem Simplex und der Abstand zwischen zwei Punkten in einem Simplex ist stets kleiner gleich $\frac{\sqrt{d}}{N}$, also ist der Abstand zu einer Ecke kleiner gleich $\frac{\sqrt{d}}{2N}$.
Sei $\mathbf{n}_1$ die Ecke von R_j mit minimalem Abstand zu $x \in R_j$. Damit folgt, dass

$$\sup x \in R_j |f(x) - f_1(x)| \leq \frac{\sqrt{d}}{N} \leq |f(x) - f_1(\mathbf{n}_1)| + |f_1(\mathbf{n}_1) - f_1(x)| \leq 2L\frac{\sqrt{d}}{2N} = \mathcal{O}(\frac{1}{N}).$$

□

Bemerkung 1.57 Ein Beispiel für eine solche Zerlegung in Simplizies in zwei Dimensionen von $[0,1]^2$ mit $N = 4$ ist wie in der folgenden Grafik gezeigt gegeben. Der Träger einer Funktion ϕ zu dem mittleren Punkt $(\frac{1}{2}, \frac{1}{2})$ gehörend ist in rot hervorgehoben.

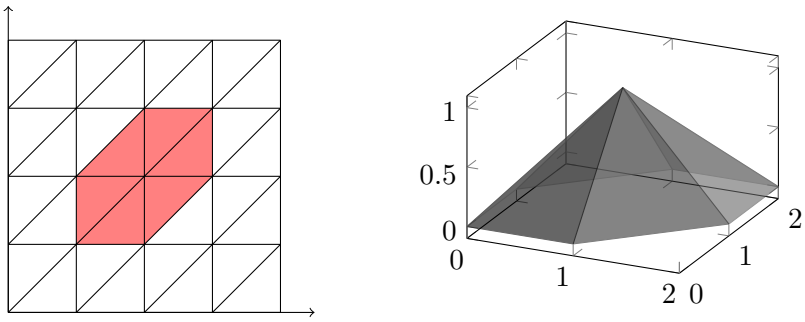

Abbildung 1.9: links: Träger einer Funktion ϕ für $N = 4$, rechts: Funktion ϕ skaliert auf $[0,2]^2$

Die Funktion ϕ ist somit ein linearer Box-Spline und damit ergeben sich die Eigenschaften wie die Partition der Eins aus der Theorie der Box-Splines (siehe etwa [5]).
Dadurch lassen sich analoge Beweise für stückweise Polynome von höherem Grad führen, indem der stückweise lineare Box-Spline durch solche von höherem Grad, entsprechend

der RePU-Aktivierungsfunktion des Netzwerks, ersetzt werden. Dafür benutzen wir eine Triangulierung unseres Bereiches durch mehrere Geraden für die Box-Splines von höherem Grad. Wir benutzen zudem hier eine Approximation in L_p, da wir stückweisen Polynome von höherem Grad mit ReQU-Netzwerken nicht mehr exakt darstellen können. Wir zeigen konkret das folgende Korollar:

Korollar 1.58 Sei $f \in \mathcal{C}^2(\mathbb{R}^2)$. Dann existiert ein Netzwerk $\mathcal{N}$ mit ReQU-Aktivierungsfunktion,

$$\|f - \mathcal{N}\|_{L_p([0,1]^2)} = \mathcal{O}(N^{-2})$$

Beweis: Wir benutzen eine Triangulierung von $[0,1]^2$ durch Linien in 4 Richtungen $(\binom{1}{0}, \binom{1}{1}, \binom{0}{1}, \binom{-1}{1})$, um das ZP-Element zu erhalten, auf dem wir einen stückweise quadratischen Box-Spline definieren.
Dieser lässt sich durch ein ReQU-Netzwerk darstellen, denn jedes Polynom auf jedem Gebiet ist der Form $\ell_j(x_1, x_2) = \alpha_0 + \alpha_1 x_1 + \alpha_2 x_2 + \alpha_3 x_1 x_2 + \alpha_4 x_1^2 + \alpha_5 x_2^2 = \sqrt{\alpha_0}^2 + \frac{\alpha_1}{4}((x_1 + 1)^2 - (x_1 - 1)^2) + \frac{\alpha_2}{4}((x_2 + 1)^2 - (x_2 - 1)^2) \frac{\alpha_3}{4}((x_1 + x_2)^2 (x_1 - x_2)^2) + \alpha_4 x_1^2 + \alpha_5 x_2^2$ mit $\alpha_i \in \mathbb{R}$ (für die exakten Polynome auf den Gebieten siehe bspw. [68, S. 2.1]).
Für diese quadratischen Polynome gilt nicht mehr, dass $\phi_0(x) = \min_j \ell_j(x)$, stattdessen approximieren wir $\mathbb{1}_{x \geq 0}$ und setzen dies verschoben jeweils für alle Teilgebiete ein. Wähle dazu $0 < \delta_1 < \delta_2 \ll 1$. Die Funktion $g(x) := \frac{1}{\delta_1 \delta_2 - \delta_1^2}((x+\delta_1)_+^2 - (x)_+^2 - (x-\delta_2)_+^2 + (x+\delta_2-\delta_1)_+^2)$ ist für $x < -\delta_2$ gleich 0 und für $x \geq 0$ gleich 1 und kann durch ein ReQU-Netzwerk mit 4 Knoten ausgedrückt werden. Zudem ist $\|g - \mathbb{1}_{x \geq 0}\|_{L_p(\mathbb{R})} \leq 6\delta_2^3$ unabhängig von N. Einsetzen der Geraden wie bspw. $\langle \binom{1}{0}, x \rangle \geq 0$ liefert via Multiplikation der Indikatorfunktionen mit dem Polynom die einzelnen Funktionen ℓ_j mit Träger genau R_j.
Dann benutzen wir erneut die Konstruktion von f_1 wie oben und nach [5, III.4 Proposition] folgt, dass $\|f - \mathcal{N}\|_{L_p} \leq cN^{-2}$. □

Die Voraussetzung, dass $f \in \mathcal{C}^2(\mathbb{R}^2)$ liegt, ist sehr großzügig. Eigentlich genügt es, wenn $f \in \mathcal{C}^2([0,1]^2 + B_\epsilon)$ für eine Kugel B_ϵ mit Radius $\epsilon > 0$.
Die Neuronalen Netzwerke sind sogar noch flexibler als die Tensorprodukt- und Box-Splines, bei denen die Unterteilung des Raum in Gebiete, auf denen die Splines stückweise polynomiell sind, gleichmäßig ist. Durch die frei wählbaren Parameter der Netzwerke lassen sich allein durch einschichtige Netzwerke beliebige geradlinig umrandete Gebiete abgrenzen. Gleichzeitig wird die Approximation mit diesen Netzwerken dadurch höchst nichtlinear und entzieht sich somit einer einfachen Analyse. Wir nehmen daher die obigen Ergebnisse als obere Schranke für unsere multivariaten RePU-Netzwerke.

1.3.5 Geometrie tiefer Netzwerke

Es ist nun von Interesse zu wissen, in wie viele polynomielle Gebiete die Splinefunktion unterteilt wird. Durch feinere Unterteilungen lassen sich letztendlich bei Klassifikationsproblemen bessere Lösungen finden und Punkte feiner trennen. Außerdem lassen sich auf

solchen Gebieten das Verhalten eines Netzwerks genau vorhersagen, was im Zusammenhang mit robustem und erklärbarem Deep Learning wichtig ist. Wir beginnen zunächst mit flachen Netzwerken.

Satz 1.59 Seien $n \in \mathbb{N}$ affine Hyperebenen in $\mathbb{R}^d$ für gegeben. Dann teilen diese Hyperebenen in höchstens

$$\sum_{k=0}^{d} \binom{n}{k} \tag{1.13}$$

konvexe Gebiete (Polytope). Hierbei benutzen wir die Konvention, dass $\binom{n}{k} = 0$, falls $n < k$.
Im $\mathbb{R}^2$ können n Geraden den Raum in $\frac{1}{2}(n^2+n)+1$ Gebiete unterteilen.

Beweis: Dieser Satz ist ein bekanntes Resultat der Geometrie, wir geben hier einen Beweis durch kurze eigene Rechnung an. Sei $f(d,n)$ die maximale Anzahl der Gebiete im $\mathbb{R}^d$ nach Schnitten durch n Hyperebenen. Wir sehen, dass durch Schneiden einer weiteren Hyperebene die Anzahl der Gebiete maximal wird, wenn diese $(d-1)$-dimensionale Hyperebene in eine maximale Anzahl von Gebieten durch die n bestehenden Hyperebenen geteilt wird. Die Anzahl der Gebiete unterliegt somit der Rekursionsformel

$$f(d,n+1) = f(d,n) + f(d-1,n),\ n,d \geq 1 \text{ und } f(d,0) = 1, f(0,n) = 1.$$

Es bleibt, die geschlossene Formel zu zeigen. Sei $n = 0$, dann ist $f(d,0) = \sum_{k=0}^{d} \binom{0}{k} = 1$.
Sei weiter $d = 0$, dass gilt $\sum_{k=0}^{0} \binom{n}{k} = \binom{0}{0} = 1$.
Sei nun $n \geq 1$. Es ist

$$\begin{aligned}\sum_{k=0}^{d} \binom{n+1}{k} &= \sum_{k=0}^{d} \binom{n}{k-1} + \binom{n}{k} \\ &= \sum_{k=0}^{d-1} \binom{n}{k} + \sum_{k=0}^{d} \binom{n}{k},\end{aligned}$$

und somit gilt (1.13).
Für $d = 2$ ergibt sich als Spezialfall $\sum_{k=0}^{2} \binom{n}{k} = 1 + n + \frac{n(n-1)}{2}$. □

Diese Anzahl lässt sich nach dem binomischen Lehrsatz abschätzen durch

$$\sum_{k=0}^{d} \binom{n}{k} \leq (1+n)^d,$$

woraus folgt, dass die Anzahl der Gebiete stets nur polynomiell wächst. Die Anzahl der Gebiete alleine sagt noch nichts über deren Form aus. Wir können uns auch anders herum die Frage stellen, wie die Schnitte aussehen.

Bemerkung 1.60 Diese Fragestellung sind insbesondere wichtig im Zusammenhang mit sogenannten „adversarial attacks" (dt.: Feindliche Angriffe). Wir betrachten zunächst nur

ReLU-Netzwerke $\mathcal{N} : \mathbb{R}^d \to \mathbb{R}^m$. Für solche ist aufgrund ihrer stückweise linearen Struktur klar, dass für fast alle $x \in \mathbb{R}^d$ eine Umgebung U_x von x existiert mit $\mathcal{N}(y) = \mathcal{N}(x) + A(y-x)$ für ein $A \in \mathbb{R}^{d\times d}$ und alle $y \in U_x$. Damit ist insbesondere $\|\mathcal{N}(y)\| \leq \|\mathcal{N}(x)\| + \|A\|\|y-x\|$ auf dieser Umgebung für geeignete Operator- und Vektornormen.
Ist die Umgebung U_x um jeden Punkt groß und für die zugehörige lineare Abbildung $\|A\|$ klein, so ist das Netzwerk stabil - konkret Lipschitz-stetig mit Konstante $\|A\|$ - gegenüber kleinen Änderungen der Eingabe.
Besonders bei Klassifikationsproblemen, d.h. bspw. unter Benutzung von Netzwerken mit Ausgabeschicht $\Sigma(x_1, \ldots, x_k) = \max_{1\leq j\leq k} x_j$, welche in typischen Anwendungen der Bilderkennung vorkommen, ist es wichtig, dass die Klassifikation stabil unter solchen kleinen Änderungen ist. Adversarial attacks auf solche Netzwerke zur Bilderkennung zeichnen sich dadurch aus, dass das Eingabebild durch kleine, für den menschlichen Betrachter unsichtbare, Unterschiede so verändert wird, dass das Netzwerk dieses falsch klassifiziert. Solches Verhalten muss in kritischen Bereichen wie autonom fahrenden Automobilen oder der Gesichtserkennung zur Strafverfolgung selbstredend um jeden Preis vermieden werden. Zur Verhinderung dieser Fehler gibt es verschiedene Ansätze wie das adversarial training (siehe [26, Kapitel 7.13]). Für das theoretische Verständnis dieser Angriffe für ReLU-Netzwerke ist es wichtig, die Größe der linearen Gebiete sowie $\|A\|$ abschätzen zu können.

Satz 1.61 (Dyer et al. [23]) Seien N Punkte $x_1, \ldots, x_N \in [0,1]^d$ zufällig gleichverteilt gegeben. Sei $X = \mathrm{conv}(x_1, \ldots, x_N)$ die konvexe Hülle der Punkte und $\mathrm{Vol}(X)$ das Volumen der Menge X. Dann gilt für den Erwartungswert $\mathbb{E}$ des Volumens, dass

$$\mathbb{E}V(X) \to 0,\ d \to \infty,\ \text{falls } N \leq (\lambda + \varepsilon)^d \tag{1.14}$$

mit einer Konstante $\lambda = \exp(\int_{\mathbb{R}_+} (\frac{1}{t} - \frac{1}{\exp(t)-1})^2\, dt) \lessapprox 2,14$.

Wir übertragen dieses Ergebnis auf die von uns betrachteten flachen ReLU-Netzwerke. Die Gewichte des Netzwerks, welche die linearen Gebiete bestimmen können wir zunächst als zufällig annehmen. Solche zufälligen Netzwerke werden zum einen als Vorstufen zu vollständig trainierten benutzt, zum anderen werden auch in der Praxis die Gewichte vieler Netzwerke so regularisiert, dass diese einer einfachen Verteilung folgen.
Dann folgt aus dem Satz 1.61, dass selbst im bestmöglichen Fall, dass die linearen Gebiete eine maximale konvexe Menge (nämlich die konvexe Hülle der Schnittpunkte der Geraden) bilden, deren Volumen für hochdimensionale Probleme gegen 0 konvergiert, solange nicht auch die Menge der Parameter und damit der linearen Regionen exponentiell mit d steigt. Tiefe Netzwerke bilden hier den Vorteil, dass die Anzahl der linearen Gebiete exponentiell mit der Anzahl der Parameter steigen kann, wie wir bereits an Beispiel im Beweis zu Lemma 1.19 sehen konnten und wie im Beispiel 1.63 genauer gezeigt wird. Auch in praktischen Anwendungen wird solches Verhalten beobachtet. (vgl. [85, §5.1])
Ein weiterer Aspekt tiefer Netzwerke ist die Approximierbarkeit von Indikator- bzw. Klassifikatorfunktionen wie $\sum_{i\in I} \mathbb{1}_{A_i}$ für gewisse Mengen A_i, wodurch Netzwerke mit kompaktem Träger entstehen können.

Solche Klassifikatoren sind von einem Neuronalen Netzwerk mit n Knoten in der versteckten Schicht genau dann exakt approximiert werden, wenn die A_i durch Schnitte von Hyperebenen entstehen. Tiefere Netzwerke können auch andere, bspw. nicht geradlinig begrenzte Gebiete approximieren, wie bereits früh bemerkt wurde (siehe bspw. [65, Kapitel 7]).

1.3.6 Approximation durch Komposition von Splines

Wir führen zunächst einen neuen Netzwerktyp ein, der hier von Nutzen sein wird. Die besondere Eigenschaft von diesen residuellen Netzwerken ist, dass bei ihnen der ursprüngliche Input in jeder Schicht als Input vorhanden ist. Netzwerke dieser Form sind Teilmengen der RePU-Netzwerke nach den Überlegungen aus Abschnitt 1.2.1.

Definition 1.62 Ein Netzwerk $\mathcal{N}$ mit Schichten der Form

$$f_i(x) = x + \sum_{j=1}^{n_i} \sigma(\langle w_{i,j}, x\rangle + \beta_{i,j})$$

heißt *residuelles Netzwerk*.

Komposition von Splines

Wir haben es bei tiefen neuronalen Netzwerken mit RePU-Aktivierungsfunktion mit einer mehrfachen Komposition von Spline-Funktionen zu tun. Diese wollen wir nun untersuchen. Wir haben bereits ein Beispiel für Komposition von Splines gesehen.

Beispiel 1.63 (zur Funktion s^r aus 1.19)
Im Beweis zu 1.19 haben wir die Funktion $s : [0,1] \to [0,1]$

$$s(x) = \begin{cases} 2x, & 0 \le x \le \frac{1}{2}, \\ 2 - 2x, & \frac{1}{2} < x \le 1 \end{cases}$$

betrachtet und $s^r = \underbrace{s \circ s \circ \cdots \circ s}_{r}$ benutzt, um $x \mapsto x^2$ zu approximieren. Im Lichte von Splines ist s ein stückweise linearer Spline auf $[0,1]$ mit Knoten $0, \frac{1}{2}$ und 1. Die Funktion s^r ist wie wir gesehen haben dann ein ebenfalls stückweise linearer Spline mit $2^r + 1$ Knoten in $\frac{k}{2^r}$, $k = 0, \ldots, 2^r$.

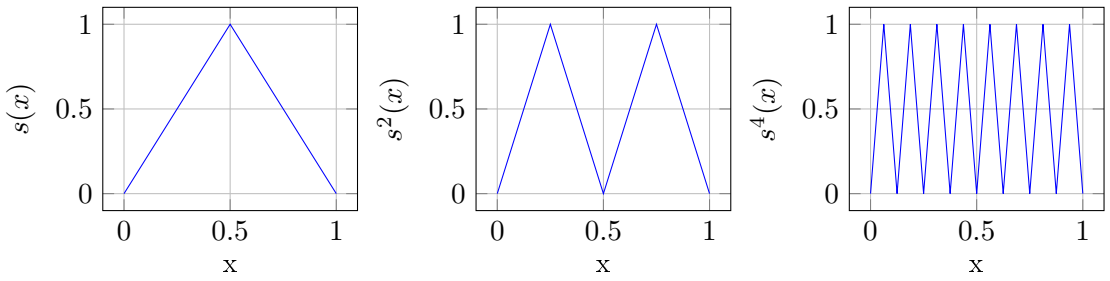

Abbildung 1.10: Die Funktionen s, s^2 und s^4.

Ein weiteres Beispiel für die Vorteile tiefer Netzwerke sind die diversen Konstruktionen von Polynome und B-Splines durch RePU-Netzwerke aus den Lemmata 1.26, 1.42. Mithilfe von tiefen Konstruktionen können wir zudem die Approximationsraten flacher Netzwerke verbessern, vgl. Lemma 1.22 und Korollar. Wir zeigen zuerst, dass wir es weiterhin mit stückweise poynomiellen Funktionen zu tun haben.

Lemma 1.64 Sei $s_1 : [0,1] \to [0,1] \in \mathcal{C}^{k-1}([0,1])$ ein Spline vom Grad $k \geq 1$ zur Knotenfolge $X = \{x_1, \ldots, x_m\} \subset [0,1]$, so dass s_1 keine Fixpunkte besitzt und $s_1(X) \nsubseteq X$ gilt.
Dann ist $s_2 = s_1 \circ s_1 = s_1^2 \in \mathcal{C}^{k-1}([0,1])$ ein stückweises Polynom mit Bruchstellen in $\{s_1^{-1}(x_i) \mid i = 1, \ldots, m\}$ vom Grad höchstens k^2.

Beweis: Es gilt: Sind p_1, p_2 Polynome vom Grad n_1, n_2, so ist $p_1 \circ p_2$ ein Polynom vom Grad höchstens $n_1 n_2$. Seien $I_j = [x_j, x_{j+1}), j = 1, \ldots, x_{m-1}$. Da $k \geq 1$, ist s_1 stetig und bildet Intervalle auf Intervalle ab. Die Bruchstellen der Funktion sind die Punkte $X \cup \{s_1^{-1}(x_i) \mid i = 1, \ldots, m\}$. Wir betrachten nun einen dieser kritischen Punkte x_j für mindestens einmal stetig differenzierbares s_1. Da s_1 fixpunktfrei ist, existiert eine Umgebung von $s_1(x_j)$, auf der s_1 ein Polynom ist. Damit gilt für die Ableitung $\frac{d}{dx} s_1^2(x_j) = s_1'(s_1(x_j)) s_1'(x_j)$. Für $x_j \in X$ gilt: Der linke Teil ist ein Polynom und rechte Teil ist nach Voraussetzung $(k-2)$-mal stetig differenzierbar in dem Punkt. Für $x_j \in \{s_1^{-1}(x_i) \mid i = 1, \ldots, m\}$ gilt, dass der linke Teil $(k-2)$-mal differenzierbar und der rechte Teil ein Polynom ist. Die kritischen Punkte sind für stückweise Polynome der Form $(x - x_j)_+^{k-1} p(x)$ für $p \in \mathbb{P}$ und somit $(k-2)$-mal stetig differenzierbar. □

Allgemeiner gilt: Sind $s_1 : \mathbb{R} \to \mathbb{R}$ und $s_2 : \mathbb{R} \to \mathbb{R}$ stückweise Polynome vom Grad k_1 bzw. k_2. Es gilt für $s = s_2 \circ s_1$: s ist stückweise Polynom auf dem Intervall $s_2(s_1(\mathbb{R}))$ vom Grad höchstens $k_1 k_2$.
Wir erhalten somit durch Komposition von Splines wieder stückweise polynomielle Funktionen, verlieren jedoch Freiheitsgrade, da der stückweise Grad größer ist als die Zahl der Bruchstellen und nicht mehr durch diese allein bestimmt wird. Die Lage der Knoten bestimmt sich ebenfalls aus den Knoten von s_1. Ausgedrückt in Form von Neuronalen Netzwerken folgt daraus, dass durch ein Netzwerk mit identischen Schichten sich automatisch bestimmte polynomielle Gebiete ergeben. Wie in Bemerkung 1.60 interessiert die Größe und Anzahl dieser Gebiete und wir sehen, dass deren Zahl univariat höchstens mit der Anzahl $|\{s_1^{-1}(x_i) \mid i = 1, \ldots, m\}$ wächst. Für die Aktivierungsfunktion $(x)_+^k$ ist dies gleich $k(m+1)$, da auf jedem der Teilintervalle eine Polynom vom Grad k jeden Wert höchstens k-fach annimmt. Auch die Größe der Teilintervalle lassen sich für ein Netzwerk mit gegebenen Gewichten so konkret ausrechnen.

Der Satz von Kolmogorov-Arnold

Einen weiteren Zugang zur Approximation durch Komposition von höherdimensionalen Funktionen liefert der bekannte Satz von Kolmogorov-Arnold (auch Überlagerungssatz von Kolmogorov, siehe [25, Kapitel 17, Theorem 1.1]) den wir bereits unter 1.16 im Zusammenhang mit bestimmten Aktivierungsfunktionen kennen gelernt haben.

Satz 1.65 (Satz von Kolmogorov-Arnold)
Jede Funktion $f \in \mathcal{C}([0,1]^n)$ lässt sich für alle $n \in \mathbb{N}$ darstellen als

$$f(x_1, \ldots, x_n) = \sum_{q=1}^{2n+1} g\left(\sum_{p=1}^{n} \phi_{p,q}(x_p)\right)$$

wobei $g, \phi_{p,q} \in \mathcal{C}([0,1])$ und $\phi_{p,q}$ unabhängig von f sind und nur g abhängig von f ist.

Dieser Satz liefert uns eine natürliche Zerlegung für mehrdimensionale Funktionen in eine Form die unseren betrachteten Neuronalen Netzwerken entspricht. Die verwendeten Funktionen sind sogar explizit angebbar und berechenbar. Dies bietet einen Zugang zur mehrdimensionalen Approximation: Zunächst approximiert man die univariaten Funktionen $\phi_{p,q}$ durch Neuronale Netzwerke und benutzt dann den Satz von Kolmogorov-Arnold. Die Güte dieser Approximation ist bekannt durch die Ergebnisse der Approximation mit Splines. Ein weiteres Theorem zeigt jedoch, dass flache Netzwerke solchen Typs nicht ausreichen:

Satz 1.66 ([25, Kapitel 17, §4, Theorem 4.1]) Für $p \in \mathcal{C}[0,1]$, stetig differenzierbare $\phi_{k,i}$ und beliebiges $g \in \mathcal{C}$ sind die Funktionen der Form

$$f(x_1, \ldots, x_n) = \sum_{k=1}^{N} p_k(x_1, \ldots, x_n) g_k(\phi_{k,1}, \ldots \phi_{k,n-1})$$

eine magere Menge in $\mathcal{C}([0,1]^n)$.

Beachte, dass bei üblichen Netzwerken die $\phi_{k,i}$ eine (multi-)lineare Funktionen in den Gewichten ω_i sind und somit natürlich stetig differenzierbar. Es gilt bekanntlich auch die Widerlegung der Hilbertschen Vermutung:

Satz 1.67 ([72, Theorem 3.1])Sei $r \geq 1$. Dann existiert eine Funktion $f \in \mathcal{C}^r$ in $n \geq 2$ Variablen, die nicht durch Summen von $\mathcal{C}^r$-Funktionen in einer Variablen darstellbar ist.

1.3.7 Rationale Approximation mit Neuronalen Netzwerken

Wir benutzen die obigen Ergebnisse über die Approximation mit Neuronalen Netzwerken mit stückweise polynomiellen Aktivierungsfunktonen, um rationale Funktionen mit ReLU-Netzwerken zu approximieren. Erste Ergebnisse wurden von Yarotsky und anderen in [87],[79] geliefert. Wir möchten diese Approximation durch eine nichtlineare Wahl

der Knoten mit tiefen Netzwerken verbessern und damit eine Anwendung der Ergebnisse über Splines und Neuronale Netzwerke aus dem vorherigen Abschnitt hervorheben. Für die Bestapproximation mit stückweise linearen Splines in L_2 gilt nach [6, Theorem 20]:

Satz 1.68 Sei $g \in \mathcal{C}^2((a,b))$ und $\|g''\|_{L_2} < \infty$. Wähle $\tau_1 = a, \tau_n = b$ und τ_i für $i = 2, \dots, n-1$ gemäß

$$\int_a^{\tau_i} |g''(x)|^{1/2}\, dx = \frac{i-1}{n-1} \int_a^b |g''(x)|^{1/2}\, dx \quad \forall i = 1, \dots, n.$$

Dann gilt

$$\operatorname{dist}(g, S) = \mathcal{O}(n^{-2}).$$

Der Ansatz zur Approximation von rationalen Funktionen bei Telgarsky [79] beruht darauf, $1/x$ über eine geometrische Reihe darzustellen und diese als Polynom mit den Methoden aus dem ersten Abschnitt wie Lemma 1.20 zu nähern. Wir können diesen Ansatz verbessern, denn die indirekte Approximation von $x \mapsto 1/x$ durch Polynome ist nicht die optimale Näherung durch stückweise lineare Funktionen, wie sie ReLU-Netzwerke darstellen. Dadurch verbessern wir die Schranke aus [79, Lemma 3.5] zur Komplexität des Netzwerks. Zunächst benötigen wir eine kleine Beobachtung.

Lemma 1.69 Seien $0 < a < b \in \mathbb{R}$ gegeben. Sei L der Raum der affin linearen Funktionen $[a,b] \to \mathbb{R}$. Dann ist

$$E(L) = \min_{\ell \in L} \max_{x \in [a,b]} \left| \frac{1}{x} - \ell(x) \right| = \frac{1}{2}\left(\frac{1}{a} + \frac{1}{b}\right) - \frac{1}{\sqrt{ab}}.$$

Beweis: Die Approximation bestimmen wir als lineare Interpolation $\ell : x \mapsto \alpha x + \beta$. Nach der Charakterisierung der Minimax-Approximation muss für den Fehler $E := E(\ell)$ und einen Punkt $\theta \in (a,b)$ gelten

$$\frac{1}{a} - \ell(a) = E, \tag{1.15}$$

$$\frac{1}{\theta} - \ell(\theta) = -E, \tag{1.16}$$

$$\frac{1}{b} - \ell(b) = E, \tag{1.17}$$

$$-\frac{1}{\theta^2} - \ell'(\theta) = 0. \tag{1.18}$$

Lösen dieses Gleichungssystems (mit $\theta > 0$ eindeutig gewählt) liefert $\theta^2 = ab, \alpha = \sqrt{ab}, \beta = \frac{1}{\sqrt{ab}} + \frac{1}{2a} + \frac{1}{2b}$ und einen optimalen Fehler in Abhängigkeit der Intervallgrenzen von $E = \frac{1}{2a} + \frac{1}{2b} - \frac{1}{\sqrt{ab}} = \frac{(\sqrt{a}+\sqrt{b})^2}{2ab}$. □

Durch diese explizite Konstruktion können wir nun eine optimale Approximation von $x \mapsto \frac{1}{x}$ durch flache ReLU-Netzwerke angeben, die besser ist als die Konstruktion aus [79] mit Netzwerkgröße $\mathcal{O}(2^k)$ für eine Approximation auf dem Intervall $[2^{-k}, 1]$.

Lemma 1.70 Seien ε und $\mu > 0$ mit $I_\mu = [\mu, 1]$ gegeben. Dann existiert ein ReLU-Netzwerk $N \in \mathcal{N}_{1,n}$ mit $n \in \mathcal{O}((\mu\varepsilon)^{-1/2})$ Knoten, sodass

$$\sup_{x \in I_\mu} \left| N(x) - \frac{1}{x} \right| \leq \varepsilon.$$

Beweis: Wir approximieren die Funktion $x \mapsto \frac{1}{x}$ durch stückweise lineare Splines bezüglich auf Teilintervallen I_j, sodass der Fehler bzgl. $\mathcal{C}(I_j)$ auf allen Teilintervallen gleich ist: Sei dazu S_A der stückweise lineare Spline mit Knoten in $a_1 = \mu$,

$$a_{k+1} = \frac{a_k}{(\sqrt{2\varepsilon a_k} \pm 1)^2},$$

der $x \mapsto \frac{1}{x}$ in den Knoten interpoliert. Dann ist $\|S - \frac{1}{x}\|_\infty < \varepsilon$, da $\sup_{x \in [a_k, a_{k+1}]} |S - \frac{1}{x}| \leq \varepsilon$ nach dem vorherigen Lemma 1.69.
Weiter ist die Folge der a_k so gewählt, dass $a_{k+1} \leq 6a_k - \varepsilon a_k^2$ für $\varepsilon \ll a_k < 1$. Damit liegen $\mathcal{O}((\mu\varepsilon)^{-1/2})$ Punkte im Intervall $[\mu, 1]$. Dann existiert ein flaches Netzwerk mit einer Schicht und $\mathcal{O}((\mu\varepsilon)^{-1/2})$ Knoten, das S realisiert. □

Allgemeiner auch für Funktionen von höherem Grad lassen sich entsprechende optimale Punkte bzgl. einer zweistufigen Minimax-Approximation durch den Remez-Algorithmus errechnen. Dazu sei N als Anzahl der Bruchstellen $a \leq a_1 \leq \cdots \leq a_N \leq b$ und r der Grad der Splines gegeben.

Korollar 1.71 Sei $\varepsilon > 0$ und $\mu > 0$ gegeben und $I_\mu = [\mu, \frac{1}{\mu}]$ wie zuvor. Sei $g(x) = \frac{p(x)}{q(x)}$ eine rationale Funktion mit $\deg p \leq d_1$ und $\deg q \leq d_2$. Dann existiert ein ReLU-Netzwerk f mit $\mathcal{O}((\ln(\frac{c \max\{d_1, d_2\}}{\varepsilon}))^2) + \mathcal{O}(\ln(\frac{1}{\varepsilon}))$ Schichten und $\mathcal{O}((\ln(\frac{c(d_1+d_2+1)}{\varepsilon}))^2 + (\mu\varepsilon)^{-1/2})$ Knoten, sodass

$$\sup_{x \in I_\mu} |f(x) - g(x)| \leq \varepsilon.$$

Dabei hängt die Konstante c nur von μ und $\sup_{x \in I_\mu} |g(x)|, \sup_{x \in I_\mu} |p(x)|$ ab.

Beweis: Wir können nach Lemma 1.70 die Funktion $x \mapsto \frac{1}{x}$ durch ein Netzwerk $\mathcal{N}$ und nach Lemma 1.20 ebenfalls Polynome durch ReLU-Netzwerke approximieren. Wir führen dies nun in einem Netzwerk zusammen. Seien $\tilde{p}$ und $\tilde{q}$ ReLU-Netzwerke mit $\mathcal{O}(\ln(\frac{n}{\varepsilon}))$ Schichten und $\sup |p(x) - \tilde{p}(x)| \leq \varepsilon$ sowie $\sup |q(x) - \tilde{q}(x)| \leq \varepsilon$. Sei weiter $g \in \mathcal{N}$ eine Approximation an die Multiplikation wie in Lemma 1.19, d.h. $\sup |g(x, y) - xy| \leq \varepsilon$.
Dann berechnet $f := g(\tilde{p}, N(\tilde{q}))$ das gewünschte Ergebnis und es gilt mit Standardargumenten (vgl. [67])

$$\begin{aligned} f(x) - \frac{p(x)}{q(x)} &= g(\tilde{p}(x), N(\tilde{q}(x)) - \frac{p(x)}{q(x)} \\ &\leq \tilde{p}(x) N(\tilde{q}(x)) + \varepsilon - \frac{p(x)}{q(x)} \\ &\leq \frac{\tilde{p}(x)}{\tilde{q}(x)} + \tilde{p}(x)\varepsilon + \varepsilon - \frac{p(x)}{q(x)} \end{aligned}$$

$$
\begin{aligned}
&\leq \frac{p(x)+\varepsilon}{q(x)-\varepsilon} + \tilde{p}(x)\varepsilon + \varepsilon - \frac{p(x)}{q(x)} \\
&\leq \frac{q(x)\varepsilon + p(x)\varepsilon}{q(x)(q(x)-\varepsilon)} + \tilde{p}(x)\varepsilon + \varepsilon \\
&= \frac{\varepsilon}{q(x)-\varepsilon} + \frac{p(x)}{q(x)}\frac{\varepsilon}{q(x)-\varepsilon} + \tilde{p}(x)\varepsilon + \varepsilon \\
&\leq 2\mu\varepsilon + \frac{p(x)}{q(x)} 2\mu\varepsilon + \tilde{p}(x)\varepsilon + \varepsilon \\
&:= \tilde{\varepsilon}
\end{aligned}
$$

analog folgt

$$
\begin{aligned}
\frac{p(x)}{q(x)} - f(x) &\leq -\frac{p(x)-\varepsilon}{q(x)+\varepsilon} - \tilde{p}(x)\varepsilon - \varepsilon + \frac{p(x)}{q(x)} \\
&\leq \tilde{\varepsilon}.
\end{aligned}
$$

Durch Wahl von $\tilde{\varepsilon} = \frac{\varepsilon}{1+2\mu(1+R)+P}$ mit $P := \sup_{x\in I_\mu} |p(x)| + \varepsilon$ und $R = \sup_{x\in I_\mu} \left|\frac{p(x)}{q(x)}\right|$ erhalten wir die gewünschte Abschätzung. □

Bemerkung 1.72 Das Netzwerk aus dem vorherigen Lemma 1.70 ist zunächst ein flaches Netzwerk und hat daher nicht die Vorteile tiefer Netzwerke hinsichtlich der Anzahl der benötigten Knoten. Tatsächlich lässt sich es auch durch ein tiefes Netzwerk mit nur $\mathcal{O}((\ln(\frac{c(d_1+d_2)}{\varepsilon}))^2 + \ln(\frac{1}{\mu\varepsilon}))$ Knoten realisieren. Dies ist insbesondere dann einfach möglich, wenn die Knoten gewisse Symmetrien aufweisen. Zur Konstruktion eines solchen aus einer stückweise linearen Approximation gibt das folgende Lemma Auskunft:

Lemma 1.73 Sei S ein stückweise linearer Spline mit Knoten $a_1, \ldots, a_n$ in dem Intervall $[0,1]$ derart, dass $|a_{2k} - a_{2k-1}| = \lambda$ und $|a_{2k} - a_{2k+1}| = 1-\lambda$ für ein $\lambda \in (0,1)$ und alle k. Es gelte weiterhin $S(a_{2k}) = 0$ und $S(a_{2k+1}) = 1$. Dann existiert ein tiefes Neuronales Netzwerk mit ReLU-Aktivierungsfunktion, $\log_2(n)$ Schichten Knoten, das S realisiert.

Beweis: Sei o.B.d.A. $n = 2^k + 1$ und $S(a_1) = 0$. Sei a_m der mittlere Knoten, dann wähle in der ersten Schicht die Koeffizienten so, dass sie einen linearen Spline S_1 mit Knoten $(1-\lambda)^n = a_m$ und $S_1(a_m) = 1$ ergibt. Wähle dann in der nächsten Schicht eine stückweise lineare Funktion bestehend aus zwei Teilstücken mit Bruchstelle in $1-\lambda$, sodass das resultierende zweischichtige Netzwerk abwechselnd Knoten im Abstand λ^{n-1} und $(1-\lambda)^{n-1}$ besitzt. Dies ist möglich, da $S_1([0,a_m]) = [0,1]$ und die Funktion Symmetrie bzgl. a_m aufweist. Wiederholen wir dies n-mal, so erhalten wir die gewünschte Funktion S. □

Als mögliche Verallgemeinerungen dieses Abschnitts können wir Lemma 1.70 auch für Approximation in anderen Räumen wie $L_2([0,1])$ und auch mit anderen abgeschnittenen Potenzfunktionen als Aktivierungsfunktion des Netzwerks formulieren, indem wir die Konstruktionen aus Abschnitt 1.3 nutzen.

Kapitel 2

Convolutional Neural Networks

In diesem Kapitel wollen wir einen besonders wichtigen Typ von Netzwerken betrachten, die Convolutional Neural Networks (CNN, auch Konvolutionsnetz) sind zentral in der Anwendungen zu Bilderkennung und Klassifikation und haben vielen modernen Anwendungen in diesem Bereich neuen Schub gegeben. Zuerst 1989 von LeCun [48] zur Erkennung von Handschriften eingesetzt, wurde es mit ihrer Hilfe möglich, die Erkennungsrate von Bilderkennungssoftware in vielen Bereichen wie Gesichtserkennung, Klassifikation von Bildern und Ähnlichem auf ein vorher unerreichbares, Menschen-ähnliches (siehe bspw. Anwendungen in [60],[82],[13],[91],[11], [40]) Niveau zu heben, oder diese in besonderen Aufgaben sogar zu übertreffen. Die Idee zu diesem Netzwerkmodell kommt aus dem biologischen Abbild des visuellen Kortex im Gehirn von höheren Lebewesen. Die definierende Eigenschaft eines CNN ist, wie der Name bereits sagt, dass es eine Faltung zur Gewichtung der Inputs einsetzt. Dies bedeutet, dass mindestens eine Schicht f des Netzwerks durch

$$(f)_j = \sigma(w_j * x + b)$$

gegeben ist. Eine solche Operation ist ebenfalls durch ein klassisches Netzwerk mit entsprechend vielen Neuronen pro Schicht wie Eingabepunkte darstellbar, indem man Gewichte miteinander verknüpft und auf denselben Wert festlegt. Dadurch erhält man ein Netzwerk mit deutlich weniger Parametern, das zudem lokale Eigenschaften wie etwa Kanten in einem großen Bild erkennen kann. Durch diese Netzwerkarchitektur soll außerdem eine bezüglich häufig vorkommenden Transformationen invariante Darstellung gewonnen werden, die dann zur Klassifikation genutzt werden kann.

Definition 2.1 Ein Convolutional Neural Network (CNN) ist ein neuronales Netzwerk

$$\mathcal{N} = \Sigma \circ f_N \circ \cdots \circ f_1 \circ f_1 \circ F$$

wie in 1.1 mit $f_j(x) = (\sigma_j(w_{i,j} * x + b_i))_{i=1}^{n_j}$ für mindestens ein $j \in \{1, \ldots, N\}$. Hierbei wird σ punktweise auf $w_{i,j} * x + b_i$ angewendet. Die einzelnen Neuronen werden in diesem Zusammenhang auch als „Channels" bezeichnet.

Oftmals sind weitere Schichten des Netzwerks durch Pooling-Layer gegeben. Diese haben erheblichen Einfluss auf die Performance eines CNN, indem Sie die im diskreten Fall die Größe des Inputs deutlich verringern. Es werden in der Praxis verschiedene Pooling-Operatoren eingesetzt, bspw. average- oder max-Pooling. Dabei wird konkret in zwei Dimensionen der diskrete $N \times N$-Input in n^2 Blöcke der Größe $k \times k$ aufgeteilt. Die Ausgabe einer Pooling-Schicht ist dann der Dimension $n \times n$ wobei jeder Eintrag der Ausgabe aus einem Block berechnet wird, bei average-Pooling durch den Mittelwert des Block oder bei max-Pooling durch den maximalen Wert in diesem Block. Für weitere Varianten siehe [26, §9.3]. Für den kontinuierlichen Fall, dass also die Eingabe eine Funktion aus $L_2(\mathbb{R}^d)$ ist, können wir average-Pooling-Schichten durch Operatoren

$$P : L_2(\mathbb{R}^d) \to L_2(\mathbb{R}^d) : f \mapsto p^{d/2} f * g_0(p\,\cdot) \tag{2.1}$$

mit g_0 ein Glättungskern ersetzen.
Wir sehen die verschiedenen Komponenten eines solchen Netzwerks am Beispiel des VGG-19-CNN aus dem Jahr 2015 [77] mit 19 Schichten und insgesamt ca. 144 Millionen Parametern, das besonders für die Bilderkennung im Rahmen des ImageNet-Wettbewerbs entwickelt wurde.

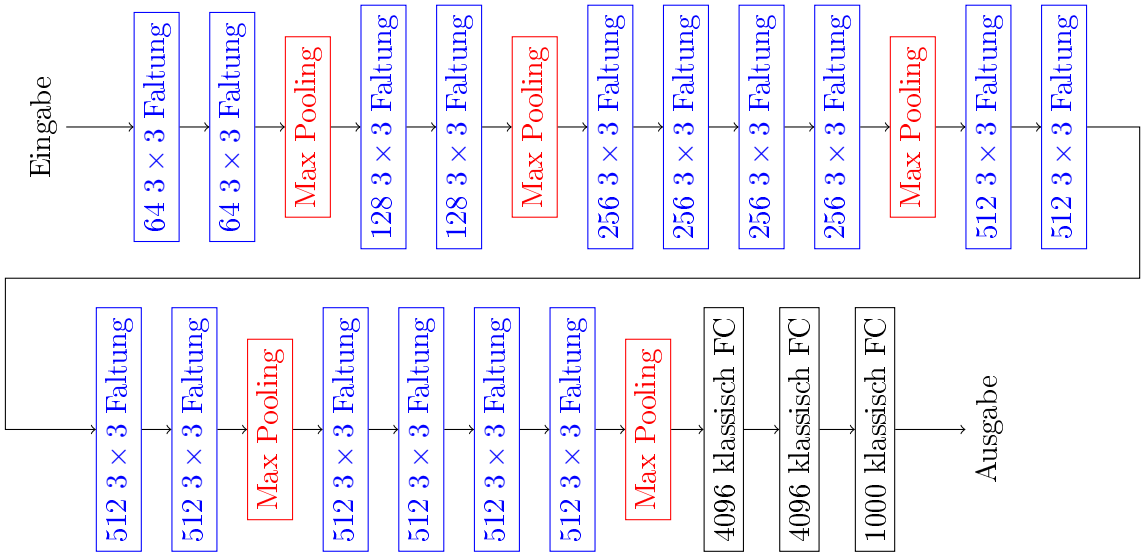

Abbildung 2.1: Struktur des VGG-19 Netzwerks [77], in blau Schichten mit Faltungsoperatoren, in rot Schichten mit Pooling und in schwarz andere Schichten.

Bei diesen wie vielen anderen aktuellen Netzwerken wechseln sich Faltungs- und Pooling-Schichten im Verhältnis von höchstens 4 zu 1 ab mit größer werdender Anzahl von Neuronen pro Schicht. Zudem sind die letzten Schichten des Netzwerks, die für die eigentliche Klassifikation sorgen, klassische Neuronale Netzwerke, wie wir sie bereits in Kapitel 1 untersucht haben. Wir wollen uns daher im Folgenden mit den ersten Schichten von CNNs mit Faltungen und Pooling befassen.
Einen genaueren Überblick über die verschiedenen Komponenten eines solchen Netzwerks

und verschiedener Variationen dieses bildet [32]. Ein erster Ansatz diese Netzwerke zu verstehen, bilden sogenannte Wavelet-Netzwerke, die wir nun einführen.

2.1 Wavelet Networks

Wavelet Networks wurden zuerst durch Zhang und Benveniste (1992) [90] eingeführt und sind eine einfache Erweiterung des klassischen Feedforward Perceptrons wie wir es im ersten Kapitel kennen gelernt haben. Es werden anstelle der klassischen sigmoidalen oder ReLU-Aktivierungsfunktionen Wavelets eingesetzt und weitere Parameter zu Translation oder Dilatation der Wavelets. Idee ist es, die Grundidee einer Wavelet-Zerlegung von Bildern o.ä. in einem Neuronalen Netzwerk zu integrieren, sodass dieses die beweisbaren Eigenschaften der Wavelet-Transformation erhält.
In der Theorie der Signalverarbeitung mit Wavelets wird bei der diskreten Wavelet-Transformation (DWT) eine Zerlegung der Form durchgeführt, dass der Input mit Band- und Tiefpassfiltern, die aus den Wavelet- und Skalierungsfunktionen gewonnen werden, transformiert wird und danach mit Downsampling verkleinert wird.

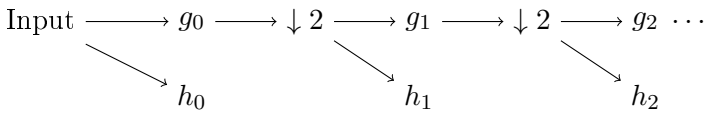

Abbildung 2.2: Struktur der diskreten Wavelet-Transformation

Dieses lässt sich in ein Netzwerk integrieren. Konkret haben diese die Form wie folgt:

Definition 2.2 Eine Funktion der Form

$$\mathcal{N}(x) = \sum_{i=1}^{r} \alpha_i \prod_{j=1}^{d} \psi_j(x)$$

mit $\alpha_i \in \mathbb{R}$ heißt flaches Wavelet Network.

Hierbei verwenden wir den Begriff Wavelet in einem sehr allgemeinen Sinn: Wir fordern nur die Eigenschaften für die stetige Wavelet-Transformation: Ein Wavelet ψ in diesem Sinn ist eine Funktion aus $L_1(\mathbb{R}) \cap L_2(\mathbb{R})$ mit $\int \psi(t)\, dt = 0$ und $\int |\psi(t)|^2\, dt = 1$. Bemerke, dass die erste Forderung äquivalent ist zu $\hat{\psi}(0) = 0$.
In Wavelet Networks (siehe z.B. [1]) werden typischerweise Wavelets der kontinuierlichen Wavelet-Transformation (CWT) wie das Ricker- bzw. Mexican-hat Wavelet $\psi(y) = (1 - y^2)e^{-\frac{1}{2}y^2}$ mit entsprechender Normalisierung oder das Morlet-Wavelet benutzt.

2.2 Scattering Networks

Eine neuere Erweiterung des Gedankens der Wavelet Networks sind die sogenannten Scattering Networks. Diese wurden 2011 zuerst von Mallat [53] eingeführt. Ziel dieser Konstruktion ist es, dadurch das Verhalten und die Stabilität und Effektivität bei vielen praktischen Anwendungen von CNN zu erklären. Anders als bei den im letzten Kapitel vorgestellten Wavelet Networks ist hier die gesamte Struktur und alle Aktivierungsfunktionen fest vorgegeben. Die Gewichte sind festgelegt und nicht gelernt, zudem werden in jeder Schicht Features ausgegeben. Dadurch ist mit diesen Netzwerken nicht direkt eine Klassifikation möglich, der aggregierte Output muss noch durch ein weiteres Netzwerk oder eine andere Klassifikationsfunktion bewertet werden. Der Aufbau und die Koeffizienten der Scattering Transformation ohne Aktivierungsfunktion entsprechen denen einer kontinuierlichen Version der Wavelet-Paket-Transformation. (siehe [14] für diese Transformation)
Wir definieren zunächst die Scattering Networks wie in [53] mit sogenannten *Littlewood-Paley Wavelets*. Diese werden für eine Variante der Wavelet-Transformation ohne Subsampling verwendet, motiviert aus der Littlewood-Paley Theorie, welche mit dyadischen Zerlegungen von L_p Funktionen arbeitet. So ist für ψ Wavelet und $j \in \mathbb{Z}$ die Littlewood-Paley Wavelet-Transformation für eine feste Skalierung j gleich $(W_j f)(x) = \int f(t)\psi(2^{-j}x - t)\, dt$. Damit lässt sich eine Transformation sehr ähnlich zu der stetigen Wavelet-Transformation definieren. Seien dazu $\{\psi_\gamma\}_{\gamma\in\Gamma} \subset L_2(\mathbb{R}^d) \cap L_1(\mathbb{R}^d)$ (stetige) Wavelet-Funktionen und ϕ die zugehörige Skalierungsfunktion für die gilt

$$(1-\delta) \leq |\hat{\phi}|^2 + \frac{1}{2}\sum_{j=-\infty}^{-1}\sum_{\gamma\in\Gamma}\left(|\hat{\psi}_\gamma(2^j\omega)|^2 + |\hat{\psi}_\gamma(-2^j\omega)|^2\right) \leq 1 \quad \text{f.ü.} \tag{2.2}$$

für ein $\delta < 1$. Γ kann hierbei bspw. eine Rotationsgruppe sein und $\psi_\gamma = \psi(\gamma^{-1}\cdot)$, sodass die Transformation später invariant bzgl. Wirkungen dieser Gruppe ist. Wavelets, welche Gleichung (2.2) erfüllen, bezeichnet man als Littlewood-Paley-Wavelets.
Es ist weiter mit der üblichen Notation $\phi_J(u) = 2^{-dJ}\phi(2^{-J}u)$ und $\psi_{j,\gamma}(u) = 2^{-dj}\psi_\gamma(2^{-j}u)$. Im Rahmen der Signalverarbeitung entspricht ϕ_J einem Tiefpass- und $\psi_{j,\gamma}$ einem Bandpassfilter.

Definition 2.3 Seien ψ_γ, ϕ wie oben gegeben. Wir definieren zunächst die Operatoren $A_J, U_\lambda : L_2(\mathbb{R}^d) \to L_2(\mathbb{R}^d)$ durch

$$A_J f(x) := (f * \phi_J)(x) \quad = \int_{\mathbb{R}^d} f(t)\phi_J(x-t)\, dt \quad \text{und} \tag{2.3}$$

$$U_\lambda f := |f * \psi_\lambda| \tag{2.4}$$

wobei wir für $\lambda = (j, \gamma)$ kurz schreiben $\psi_\lambda = \psi_{j,\gamma}$.
Sei nun $p = (\lambda_n = (j_n, \gamma_n))_{n=1}^N$ ein Pfad von Koeffizienten und definiere die Länge dieses Pfads als $|p| := N$. Dann definieren wir den Scattering-Propagationsoperator auf diesen

Pfad durch

$$U[p]f := \prod_{n=1}^{|p|} U_{\lambda_n} f. \tag{2.5}$$

Wir organisieren die Koeffizienten nach Skala in $\Lambda_J = \{\lambda = (j,\gamma) \mid 2^j > 2^{-J}, \gamma \in \Gamma\}$ und erhalten weiter die Scattering-Propagation auf der Menge aller Pfade $\mathcal{P}_J^N := \{p = (\lambda_k)_{k=1}^N \mid \lambda_k \in \Lambda_J\}$ bis zu einer bestimmten Länge N und Skala J durch $U_J^N f := \{U[p]f\}_{p\in\mathcal{P}_J^N}$.
Wir definieren nun den Scattering-Operator für einen Schritt $S_J : L_2(\mathbb{R}^d) \to (L_2(\mathbb{R}^d))^{\tilde{Q}(N)}$ mit

$$S_J f := \{A_J f, (U_\lambda f)_{\lambda\in\Lambda_J}\}. \tag{2.6}$$

Durch erneutes Anwenden von S_J auf $U_\lambda f$ erhalten wir eine netzwerkartige Struktur, wie in der Wavelet-Paket-Transformation.
Als letztes können wir die Ausgabe des Scattering Netzwerks für alle Pfade bis zu einer gewissen Tiefe N als Operator $\Phi_N : L_2 \to (L_2(\mathbb{R}^d))^{Q(N)}$ definieren: $\Phi_J := \{A_J U[p]f\}_{p\in\mathcal{P}_J}$ und $\Phi f := \bigcup_{J\in\mathbb{N}} \Phi_J f$. Sowie für einen einzelnen Pfad $\Phi_J[p] = A_J U[p]$.

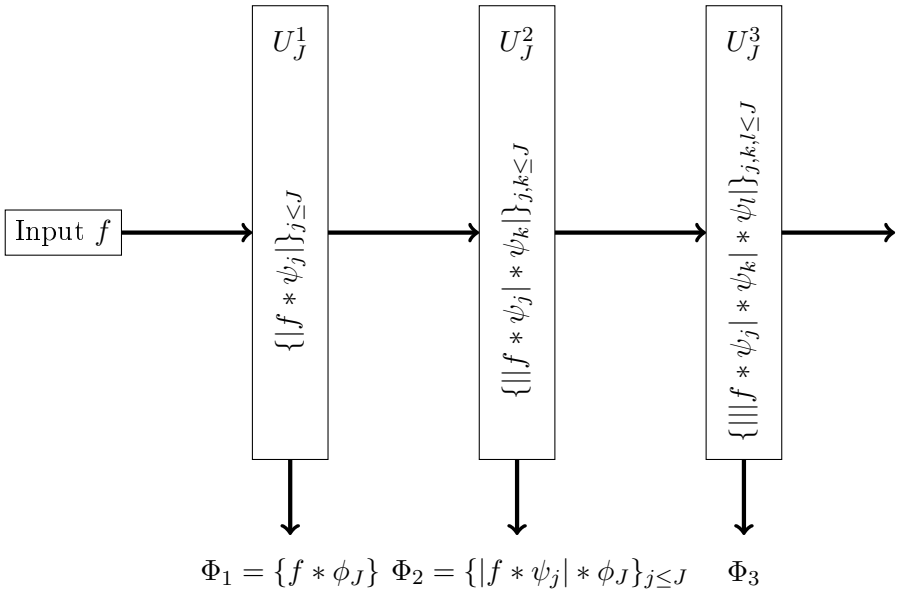

Abbildung 2.3: Struktur eines Scattering Networks

Definition 2.4 Wir definieren eine Norm in dem Raum $(L_2(\mathbb{R}^d))^{Q(N)}$ des Outputs der Operatoren Φ_N. Sei $\Phi_N f = \{s_q\}_{q=1}^{Q(N)} \in (L_2(\mathbb{R}^d))^{Q(N)}$. Dann definiere

$$\|\Phi_N f\|_S := \left(\sum_{q=1}^{Q(N)} \|s_q\|_{L_2(\mathbb{R}^d)}^2\right)^{1/2} \tag{2.7}$$

Mithilfe der Gleichung 2.2 ergibt sich die Normerhaltung durch den Operator U_λ:

Lemma 2.5 (Siehe [53], Proposition 2.1)
Sei $J \in \mathbb{Z}$ und S_J der Scattering-Operator für einen Schritt wie definiert in 2.3. Es gelte

$$\frac{1}{2}\sum_{j\in\mathbb{Z}}\sum_{\gamma\in\Gamma}|\hat{\psi_\gamma}(2^{-j}\omega)|^2 = 1 \quad \text{und} \quad |\hat{\phi}(\omega)|^2 = \frac{1}{2}\sum_{j=-\infty}^{0}\sum_{\gamma\in\Gamma}|\hat{\psi_\gamma}(2^{-j}\omega)|^2 \tag{2.8}$$

für fast alle $\omega \in \mathbb{R}^d$. Dann ist $\|S_J f\|_2 = \|f\|_2$ für alle reellwertigen Funktionen $f \in L_2(\mathbb{R}^d)$.

Beweis: Nach Definition der Norm auf $L_2(\mathbb{R}^d)$ ist $\| \, |f| \, \|_2 = \|f\|_2$ und nach dem Satz von Plancherel gilt $\|S_J f\|^2 = \|A_J f\|^2 + \sum_{\lambda\in\Lambda_J}\|U_\lambda f\|^2 = \|\hat{\phi}\|^2\|f\|^2 + \sum_{\lambda\in\Lambda_J}\|\hat{\psi_\lambda}\|^2\|f\|^2$. Nach Standardargumenten der Wavelet-Analyse ist dies gleichbedeutend zu $|\hat{\phi}(2^J\omega)|^2 + \sum_{\gamma\in\Gamma}\sum_{j\in\mathbb{Z}}|\hat{\psi_\gamma}(2^{-j}\omega)|^2 = 1$ f.ü. Da f reellwertig ist, gilt $|\hat{f}(\omega)| = |\hat{f}(-\omega)|$ und somit folgt die Behauptung. □

Bemerkung 2.6 Zur Wahl der Wavelets ψ_γ: Die von [53] benutzten multivariaten Wavelets ψ_γ werden aus einem Mutter-Wavelet ψ durch Rotationen γ aus einer endlichen Untergruppe G der speziellen orthogonalen Gruppe in d Dimensionen $SO(d)$ bzw. der orthogonalen Gruppe $O(d)$ erzeugt. Konkret ist

$$\psi_\gamma = \psi(\gamma^{-1}\,\cdot)$$

und die daraus wie oben erzeugten Wavelets $\{\psi_\lambda = 2^{dj}\psi_\gamma(2^j\,\cdot)\,|\,\lambda = (j,\gamma), j \in \mathbb{Z}, j < J, \gamma \in G\}$ bilden mit Translationen einen semi-diskreten Parseval-Frame $\Psi_{J,G}$ von $L_2(\mathbb{R}^d)$. Insbesondere werden die Translationen nicht gesampelt, sondern die Wavelet-Transformation wird kontinuierlich als $W_{\psi_\lambda}f(x) = f * \psi_\gamma(x) := \int_{\mathbb{R}^d} f(\mathbf{u})\psi_\lambda(x-\mathbf{u})\,d\mathbf{u}$ betrachtet.
Wir betrachten im Folgenden nur solche Wavelets, welche die Bedingungen von Lemma 2.5 erfüllen.

Bemerkung 2.7 Erfüllen die Wavelets die Bedingungen aus Lemma 2.5, so gilt für den Operator Φ_J

$$\|\Phi_J f - \Phi_J g\| \leq \|f - g\| \tag{2.9}$$

für alle $f, g \in L_2(\mathbb{R}^d)$ mit analogen Argumenten wie im Beweis von 2.5, da $S_J f = \{A_J f, (U_\lambda f)_{\lambda\in\Lambda_J}\}$ und $\Phi_J f$ die Anwendung des Operators A_J auf den letzten Teil von $S_J f$ ist.

Diese Netzwerke besitzen nützliche Eigenschaften wie Translationsinvarianz im folgenden Sinne:

Lemma 2.8 ([53, Proposition 2.10]) Für den Operator Φ_J mit Wavelets, die 2.8 erfüllen, gilt

$$\lim_{J\to\infty}\|\Phi_J(T_t f) - \Phi_J(f)\|_S = 0 \tag{2.10}$$

für alle $f \in L_2(\mathbb{R}^d), t \in \mathbb{R}^d$.

Beweis: Sei $J \in \mathbb{N}$. Es gilt $\|\Phi_J T_t f - \Phi_J f\| \leq \|T_t A_J f - A_J\|_2 \|\Phi_J f\|_S$. Denn die Operatoren A_J und $f \mapsto \psi_\lambda * f$ kommutieren als Faltungsoperatoren mit Translationen, d.h. $A_J(T_t f) = T_t(A_J f)$ und somit gilt auf jedem Pfad p, dass $\Phi_J[p](T_t f) = T_t(\Phi_J f) = T_t A_J U_J[p] f$ für alle J. Aus der Dreiecksungleichung $||x| - |y|| \leq |x - y|$ und deren Anwendung auf $U_\lambda f$ folgt $\|\Phi_J(T_t f) - \Phi_J(f)\|_S \leq \|T_t A_J - A_J\|_2 \|U[p] f\|_S$ nach der Definition der Norm in 2.7.
Es gilt weiter, dass $\|T_t A_J - A_J\| \leq c 2^{-J}|t|$ für eine Konstante c nach [53, Lemma 2.11] und somit folgt die Behauptung. □

Außerdem finden wir eine Stabilitätsaussage bezüglich Diffeomorphismen.

Lemma 2.9 (Theorem 2.12 aus [53]) Sei $\tau \in \mathcal{C}^2(\mathbb{R}^d)$ ein Diffeomorphismus mit $\|\tau\|_\infty \leq \frac{1}{2}$. Wir schreiben F_τ für den Operator $F_\tau f(x) = f(x - \tau(x))$. Dann gilt

$$\|\Phi_J F_\tau f - \Phi_J f\| \leq c \|U_J f\|_1 (2^{-J} \|\tau\|_\infty + K(\tau))$$

mit einer Konstante $K(\tau)$, die nur von τ abhängt.

2.3 Verallgemeinerungen des Scattering für semi-diskrete Frames

Eine natürliche Erweiterung dieses Netzwerktyps besteht darin andere Funktionen als Wavelets zur Konstruktion der Operatoren A_J, U_J zuzulassen. Da in der Praxis die Filter durch Optimierungsalgorithmen aus den gegebenen Daten bestimmt werden, suchen wir so eine Erklärung, wie auch diese Netzwerke gute Eigenschaften bei Klassifikationsproblemen zeigen können. Diese Verallgemeinerung wurde zuerst von Wiatowski und Bölcskei in [84] vorgeschlagen. Zunächst sind die nächsten Kandidaten Curvelets, Shearlets und Ridgelets, welche Teile der Klasse der α-Molecules [30] sind und ähnlich zu Wavelets erzeugt werden. Gleichzeitig wollen wir nun auch nichttriviale Pooling-Operatoren und andere Aktivierungsfunktionen als $|\cdot|$ betrachten. Wir führen zunächst die Klasse von semi-diskreten Frames ein und benutzen dazu eine verallgemeinerte Variante der Littlewood-Paley Gleichung für Wavelets (2.2).

Definition 2.10 Sei Λ eine endliche oder abzählbare Indexmenge und $\{g_\lambda\}_{\lambda \in \Lambda} \subset L_1(\mathbb{R}^d) \cap L_2(\mathbb{R}^d)$. Wir sagen die Menge dieser Funktionen erfüllt die *Littlewood-Paley-Eigenschaft*, falls Konstanten $A, B > 0$ existieren mit

$$A \leq \sum_{\lambda \in \Lambda} |\hat{g_\lambda}(\omega)|^2 \leq B, \quad \text{für fast alle } \omega \in \mathbb{R}^d. \tag{2.11}$$

Definition 2.11 Sei $\{g_\lambda\}_{\lambda\in\Lambda} \subset L_1(\mathbb{R}^d) \cap L_2(\mathbb{R}^d)$ wie oben und $T_a f(x) = f(x-a)$ für $a \in \mathbb{R}^d$ der Translationsoperator. Die Menge $\Psi(\Lambda, \mathbb{R}^d) := \{T_a g_\lambda\}_{\lambda\in\Lambda, a\in\mathbb{R}^d}$ ist ein semi-diskreter Frame des $L_2(\mathbb{R}^d)$ falls Konstanten $A, B > 0$ exisitieren, sodass

$$A\|f\|_2^2 \leq \sum_{\lambda\in\Lambda} \|f * g_\lambda\|_2^2 \leq B\|f\|_2^2, \quad \text{für alle } f \in L_2(\mathbb{R}^d). \tag{2.12}$$

Man nennt die Funktionen g_λ, $\lambda \in \Lambda$ *Atome* des Frames.

Bemerkung 2.12 Ist $\Psi(\Lambda, \mathbb{R}^d)$ ein semi-diskreter Parseval-Frame, d.h. es ist $A = B = 1$, dann gilt sogar

$$\sum_{\lambda\in\Lambda} \|f * g_\lambda\|_2^2 = \sum_{\lambda\in\Lambda} \int_{\mathbb{R}^d} |\langle f, T_a I \psi_\lambda\rangle|^2 \, da. \tag{2.13}$$

Wir bemerken an dieser Stelle den Zusammenhang zwischen Frame-Eigenschaften und der Littlewood-Paley-Ungleichung, viele der Eigenschaften des Scatterings sind Folgerungen aus den Eigenschaften der betrachteten Wavelet-Frames, welche Gleichung (2.2) erfüllen.

Lemma 2.13 (Aus Mallat [52], Theorem 5.11) Die Menge $\Psi(\Lambda, \mathbb{R}^d)$ wie in Definition 2.11 ist ein semi-diskreter Frame genau dann, wenn die Atome $\{g_\lambda\}_{\lambda\in\Lambda}$ die Gleichung (2.11) erfüllen.

Wir können nun verallgemeinerte Scattering Networks zu semi-diskreten Frames mit verschiedenen Aktivierungsfunktionen und Pooling-Operatoren definieren.

Definition 2.14 Seien $\Psi_1, \ldots, \Psi_M$ semi-diskrete Frames mit Frame-Konstanten $(A_j, B_j)_{j=1}^M$.
Wähle $\phi_j \in \Psi_j$ für $j = 1, \ldots, M$ sodass $\Psi_j = \{\phi_j\} \cup \{\psi_{j,k}\}_{k\in\Lambda_j}$. Dieses Element entspricht dem Tiefpassfilter in einer Wavelet-Zerlegung.
Seien zudem $\sigma_j : \mathbb{R} \to \mathbb{R}$, $j = 1, \ldots, M$ Lipschitz-stetige Aktivierungsfunktionen und M_j der Operator

$$M_j : L_2(\mathbb{R}^d) \to L_2(\mathbb{R}^d) : \; M_j f(\mathbf{x}) = (\sigma_j(f(\mathbf{x}))_i)_{i=1}^d$$

mit Lipschitz-Konstanten L_j, d.h. $\|M_j g - M_j h\|_{L_2(\mathbb{R}^d)} \leq L_j \|g - h\|_{L_2(\mathbb{R}^d)}$ für alle $g, h \in L_2(\mathbb{R}^d)$.
Seien $P_j : L_2(\mathbb{R}^d) \to L_2(\mathbb{R}^d)$ Lipschitz-stetige Pooling-Operatoren mit der Eigenschaft $P_j(0) = 0$ und Lipschitz-Konstante R_j. (siehe (2.1) für den prototypischen Pooling-Operator)
Dann können wir analog zu Scattering Networks die Operatoren

$$A_j f = \phi_j * f, \tag{2.14}$$

$$U_\lambda f = P_j M_j (f * g_\lambda) \tag{2.15}$$

definieren. (vgl. 2.3f.) Daraus ergeben sich die verallgemeinerten Scattering-Operatoren auf Pfaden $p \in \mathcal{P}^N = (\lambda_i)_{i=1}^N$ mit $\lambda_j \in \Lambda_j$ durch $U[p]f = \prod_{j=1}^{|p|} U_{\lambda_j} f$ analog zum klassischen

Scattering wie in Definition 2.3.
Zuletzt definieren wir die Ausgabe der n-ten Schicht eines verallgemeinerten Scattering Netzwerks $\Phi_n f = \{A_n U[p]f\}_{p\in\mathcal{P}^n}$ und $\Phi f := \bigcup_{n\in\mathbb{N}} \Phi_n f$.

Es bleiben die wesentlichen Eigenschaften der Scattering Networks wie Translationsinvarianz wie im folgenden Sinne sowie Stabilität gegenüber Diffeomorphismen erhalten. Details dazu finden sich z.B. in [84].
Als Beispiel für eine Anwendung des verallgemeinerten Scatterings benutzen wir einen Frame von Shearlets anstelle von Wavelets.

Ein semi-diskreter Shearlet-Frame für den Raum $L_2(\mathbb{R}^2)$ ist wie folgt gegeben: Shearlets wurden durch Kutyniok [33] eingeführt. Sie sind optimal für Approximation einer Klasse von sogenannten Cartoon-Funktionen und werden in vielen Anwendungen verwendet. Für weitere Details und eine tiefergehende Einführung in Shearlets siehe [45].

Definition 2.15 (Cartoon-Funktion nach [33]) Sei $B \subset \mathbb{R}^d$ eine kompakte Menge mit Rand ∂B eine kompakte $\mathcal{C}^2$-Mannigfaltigkeit. Dann ist die Menge der Cartoon-Funktionen $Cart$ gleich

$$\text{Cart}_B = \left\{ f = f_1 + f_2 \mathbb{1}_B \mid f_i \in L_2(\mathbb{R}^d) \cap \mathcal{C}^1(\mathbb{R}^d), |\nabla f_i(x)| = \mathcal{O}((1+|x|^2)^{-d/2}) \right\} \tag{2.16}$$

Beispiel 2.16 Wir definieren den Scheroperator S durch die Schermatrix $S_\lambda = \begin{pmatrix} 1 & \lambda \\ 0 & 1 \end{pmatrix}$ und die Skalierungsmatrizen $A_\mu = \begin{pmatrix} \mu & 0 \\ 0 & \mu^{1/2} \end{pmatrix}$. Dann ist

$$\Psi = \left\{ c^j T_a S_\lambda A_{2^j} \psi \right\} = \left\{ c^j T_a \psi_{\lambda,j} \right\}_{(\lambda,j)\in\Lambda, a\in\mathbb{R}^2} \tag{2.17}$$

ein semi-diskreter Frame. Dabei ist $c = 2^{3/4}$ und $\psi_{\lambda,j} = S_\lambda A_{2^j} \psi$ für $(\lambda, j) \in \Lambda$ für eine endliche Menge $\Lambda \subset [-2^{j/2}, 2^{j/2}] \times \mathbb{N}$.
Weiterhin besitzen Shearlets die Eigenschaft, dass für eine N-Term-Approximation f_N von Cartoon-Funktionen f mit Elementen aus Ψ gilt

$$\|f - f_N\|_2^2 \leq C N^{-2} (\log N)^3, \;\; N \to \infty,$$

was die optimale Approximationsrate für solche Funktionen ist. Dazu wählt man jeweils die N Elemente mit den größten Shearlet-Koeffizienten. Benutzen wir also den Shearlet-Frame in einem verallgemeinerten Scattering-Netzwerk können wir damit ebenfalls Cartoon-Funktionen besonders gut klassifizieren. Für $M_j = P_j = \text{id}$ ergibt sich etwa gerade eine Shearlet-Transformation, wie sie vielfach untersucht worden ist (siehe auch [45]).

2.4 Grundlagen über endliche Frames

Wir möchten als Nächstes auch Scattering mit diskreten und endlichen Frames definieren und untersuchen. Diese interessieren uns, da in praktischen Anwendungen Filter und

Eingabedaten jeweils diskret sind und damit aus endlichdimensionalen Räumen stammen. Dazu benötigen wir die Theorie von endlichen Frames und Elemente der harmonischen Analysis. Wir beginnen mit der Definition eines endlichen Frames wie in [9].
Notation: Wir schreiben im Folgenden $\phi[k]$ für den k-ten Eintrag von ϕ als Element eines N-dimensionalen Vektorraums. Weiter sei $T(N)$ die Menge der N-periodischen Signale. Dann ist für $\phi, \psi \in T(N)$ die *zyklische Faltung* definiert als $(\phi * \psi)[k] = \sum_{j=0}^{N-1} \phi[j]\psi[k-j]$ und die *diskrete Fouriertransformation* $\hat{\phi}[k] = \sum_{j=0}^{N-1} f[j]e^{-2\pi i \frac{kj}{N}}$.

Definition 2.17 Sei H ein N-dimensionaler Hilbertraum und sei $\Phi := (\phi_i)_{i=1}^m$ eine Familie von Elementen aus H für die Konstanten $0 < A \leq B < \infty$ existieren mit

$$A\|x\|^2 \leq \sum_{i=1}^{m} |\langle x, \phi_i \rangle|^2 \leq B\|x\|^2 \quad \forall x \in H. \tag{2.18}$$

Ist $A = B$, so heißt Φ ein tight finite frame (auch: straffer endliche Frame).
Ist $\Phi \setminus \{\phi_i\}$ für alle $1 \leq i \leq m$ kein Frame mehr, so heißt der Frame Φ exakt.
Haben alle Elemente des Frames Φ gleiche Norm $\|\phi_i\|_H = c$ für $1 \leq i \leq m$, so heißt Φ ein equal norm frame.

Bemerkung 2.18 Einige grundlegende Eigenschaften von (endlichen) Frames sind wie folgt:

1. Φ ist ein Frame von einem Hilbertraum $\mathcal{H}$ genau dann, wenn Φ den Raum $\mathcal{H}$ aufspannt.

2. Gleichung (2.18) ist für die betrachteten Räume von zeitdiskreten komplexen oder reellen Signalen äquivalent dazu, dass eine Konstante B existiert mit $\sum_{i=1}^m |\hat{\phi}_i[k]|^2 \leq B$ für alle $k \in \{0, \ldots, N-1\}$.

Für die Anwendung auf Signalverarbeitung sind insbesondere Frames von $T(N)$ und für die Bildverarbeitung Frames in $\ell(\mathbb{Z}_N^2)$ interessant. Beispiele für Frames lassen sich einfach angeben, denn nach Eigenschaft 1. ist jede Basis und jede Obermenge einer Basis ein Frame. Meist fordern wir jedoch zusätzliche nützliche Eigenschaften, die bspw. zu Wavelet-Frames führen. Zu jedem Frame lässt sich ein Frame-Operator definieren durch die folgende Konstruktion:

Definition 2.19 Sei $\Phi = (\phi_i)_{i=1}^m$ ein Frame im Hilbertraum H. Dann ist der zugehörige Analyse-Operator T_Φ definiert als

$$T_\Phi : H \to \mathbb{C}^M : Tx = (\langle x, \phi_i \rangle)_{i=1}^m$$

und der Synthese-Operator S_Φ als der adjungierte Operator

$$S_\Phi : \mathbb{C}^M \to H : S(a_i)_{i=1}^m = \sum_{i=1}^{m} a_i \phi_i$$

Zusammen ergibt sich der Frame-Operator F_Φ

$$F_\Phi : H \to H : Hx = S_\Phi T_\Phi x = \sum_{i=1}^{m} \langle x, \phi_i \rangle \phi_i$$

Dieser Frame-Operator ist positiv, selbstadjungiert und invertierbar.

Der im Weiteren relevante Hilbertraum ist der Raum der (komplexen oder reellen) N-periodischen Signale $\ell(\mathbb{Z}_N) = \{x = x[n], n \in \mathbb{Z} | x[k+N] = x[k] \text{für alle } k \in \mathbb{Z}\}$ mit innerem Produkt $\langle x, y \rangle = \sum_{k=1}^{N} x[k]\overline{y[k]}$. Dieser N-dimensionale Hilbertraum wird als Modell in der Signalverarbeitung benutzt [9].

2.4.1 Darstellung eines endlichen Frames und Scattering mit endlichen Frames

Wir können den Syntheseoperator S_Φ eines endlichen Frame $\Phi = (\phi_i)_{i=1}^m$ aus $\ell(\mathbb{Z}_m)$ als eine $N \times m$-Matrix darstellen, indem wir die Frame-Atome $\phi_i \in H$ bzgl. der Standardbasis von $\ell(\mathbb{Z}_N)$ aufstellen:

$$A_\Psi = \begin{pmatrix} \vdots & \vdots & & \vdots \\ \phi_1 & \phi_2 & \dots & \phi_m \\ \vdots & \vdots & & \vdots \end{pmatrix}$$

Durch die Damit erhaltene einfache Darstellung, lassen sich weitere Eigenschaften der Frames algebraisch ausdrücken.

Lemma 2.20 Sei $T : H \to \ell_2(\mathbb{Z}_N)$ ein linearer Operator und $\{h_1, \dots, h_d\}$ eine Orthonormalbasis von H. Sei weiter $\{\lambda_m\}_{m=1}^M$ eine Folge positiver Zahlen. Dann sind für eine Darstellung A_Ψ des Synthese-Operators S_Ψ in der Basis $\{h_1, \dots, h_d\}$ äquivalent:

1. $\{S_\Psi e_n\}_{n=1}^N$ ist ein Frame für H mit Eigenvektoren h_i und Eigenwerten λ_i. Hier ist $\{e_n\}_{n=1}^N$ die Standardbasis von $\ell_2(\mathbb{Z}_N)$.

2. Die Zeilen von A_Ψ sind orthogonal und $\sum_{j=1}^m \psi_{j,m}^2 = \lambda_m$ die m-te Zeile ergibt quadratsummiert λ_m.

3. Die Spalten von A_Ψ bilden einen Frame für $\ell_2(\mathbb{Z}_N)$ und $AA^* = \text{diag}(\lambda_1, \dots, \lambda_M)$.

Beweis: Für einen Beweis dieses grundlegende Ergebnis siehe bspw. [9, Proposition 1.12, p.24]

Nun können wir das Scattering für diese endlichen Frames definieren.

Definition 2.21 Seien $\Psi_n = (\psi_{n,i})_{i=0}^m$ für $n = 1, \dots, N$ endliche Frames eines Hilbertraums H.
Sei dann analog zu Definition 2.3 $A_n f := f * \psi_{n,0}$ und $U_{n,j} f := |f * \psi_{n,j}|$, wobei hier mit

$*$ die diskrete Faltung mit „zero-padding" gemeint ist. D.h. die Dimension von f ist gleich der Dimension von $f * \psi$ für beliebigen Filter ψ.

$$(f * \psi_{n,j})(\ell) = \sum_{k \in \mathbb{Z}} f_k \psi_{n,j}(\ell - k) \tag{2.19}$$

Dann ist der Scatteringoperator für endliche Frames über einen Pfad $p = (p_n)_{n=1}^N \in \mathcal{P}_N$ definiert als

$$S_N[p]f = \left\{ A_n f, \left(\prod U_{n,j} \right)_{p_n = j} f \right\}_{n=1}^N$$

Bemerkung 2.22 Sei $N \in \mathbb{N}, \Psi_n$ Frames wie zuvor und p ein Pfad. Der oben definierte Operator $S_N[p]$ ist wohldefiniert, insbesondere ist $\|S_N[p]\| < \infty$. Dies folgt unmittelbar aus den Frame-Eigenschaften von Ψ_n.

Das so erhaltene Scattering mit endlichen Frames erhält viele Eigenschaften des Scattering mit semi-diskreten Frames aus dem vorherigen Abschnitt. Manche speziellen kontinuierlichen Eigenschaften wie die Invarianz gegenüber beliebigen Translationen lassen sich nicht erhalten, werden jedoch durch entsprechende gleichwertige Aussagen für diskrete Translationen und andere Transformationen ersetzt.
Wir benutzen aus der Theorie der Kerndichteschätzer (Kernel Density Estimation) die folgende Abschätzung, [58] indem man eine Dichtefunktion mit einem Filter identifiziert und die Eigenschaften des Erwartungswerts mit der Mittelfrequenz und die Standardabweichung mit der Bandbreite gleichsetzt.

Lemma 2.23 Sei $\psi \in L_2(\mathbb{Z}^N)$ ein diskreter Filter mit $\|\psi\|_2 = 1$ mit Mittenfrequenz ξ und Bandbreite σ definiert als

$$\xi = \frac{1}{(2\pi)^{N/2}} \int_{[-\pi,\pi]^N} \omega |\hat{\psi}(\omega)|^2 \, d\omega, \quad \sigma^2 = \frac{1}{(2\pi)^N} \int_{[-\pi,\pi]^N} |\omega - \xi|^2 \, |\hat{\psi}(\omega)|^2 \, d\omega$$

und $\hat{x}(\omega) = \frac{1}{(2\pi)^{N/2}} \sum_{j \in \mathbb{Z}^N} x(j) e^{-i\langle j, \omega \rangle}$ die diskrete Fouriertransformation von x. Sei $\tau \in \mathbb{Z}^N$ beliebig, dann gilt

$$\|(T_\tau x) * \psi - e^{-i\langle \xi, \tau \rangle}(x * \psi)\|_\infty \leq \sigma |\tau| \|x\|_2. \tag{2.20}$$

Beweis: Die (diskrete) Faltung ist kovariant bzgl. Translationen, es gilt also $(T_\tau x) * \psi = x * (T_\tau \psi)$. Mit der Youngschen Ungleichung folgt dann $\|(T_\tau x) * \psi - e^{-i\langle \xi, \tau \rangle}(x * \psi)\|_\infty \leq \|T_\tau \psi - e^{-i\langle \xi, \tau \rangle} \psi\|_2 \|x\|_2$. Wir schätzen weiter ab

$$\begin{aligned}
\|T_\tau \psi - e^{-i\langle \xi, \tau \rangle} \psi\|_2^2 &= \frac{1}{(2\pi)^N} \int_{[-\pi,\pi]^N} |\widehat{T_\tau \psi}(\omega) - e^{-i\langle \xi, \tau \rangle} \hat{\psi}|^2 \, d\omega \\
&= \frac{1}{(2\pi)^N} \int_{[-\pi,\pi]^N} |e^{-i\langle \omega, \tau \rangle} - e^{-i\langle \xi, \tau \rangle}| \, |\hat{\psi}(\omega)|^2 \, d\omega \\
&\leq \frac{1}{(2\pi)^N} \int_{[-\pi,\pi]^N} |\omega - \xi|^2 |\tau|^2 |\hat{\psi}(\omega)|^2 \, d\omega \\
&= \sigma^2 |\tau|^2,
\end{aligned}$$

und erhalten die gewünschte Abschätzung. □

Dieses Lemma zeigt uns, dass kleine Translationen (d.h. solche mit $\|\tau\| < \frac{1}{\sigma}$) durch kleine Änderungen der Phase $e^{-i\langle\tau,\xi\rangle}$ genähert werden. Durch sukzessive Anwendung von geeigneten Filtern in einem Netzwerk lassen sich also translationsinvariante Netzwerke herstellen, analog zu Satz 2.8 für semi-diskrete Frames.

Lemma 2.24 Sei Ψ_n eine Folge von endlichen Frames mit Frame-Konstanten (A_n, B_n), sei T_a der Translationsoperator zu $a \in \mathbb{Z}^d$. Weiter gelte für die Frame-Atome $\psi_{n,j} \in \Psi_n$, dass σ^2 wie in Lemma 2.23 gilt $|\sigma|a| - B_n| \leq 1$ für alle n. Dann gilt $\|S_N[p](T_a f) - S_N[p]f\| \to 0$ für $N \to \infty$.

Beweis: Aus Lemma 2.23 folgt die Aussage, denn die Norm von $S_N[p]f(T_a f) - S_N[p]f$ ist als Summe der Normen $\|U_{n,j}(T_a f) - U_{n,j}f\| = \| \, |T_a f * \psi_{n,j}| - |f * \psi_{n,j}| \, \| \leq \|T_a f * \psi_{n,j} - f * \psi_{n,j}\| \leq \sigma|a| \|f\|_2 - B_n \|f\|_2$ für $n \leq N$. Damit folgt durch wiederholte Anwendung des Operators $U_{j,n}$ zusammen mit der Voraussetzung an σ, dass $\|S_N[p]f(T_a f) - S_N[p]f\| \leq C^N \|f\|_2$ für ein $C < 1$ in Abhängigkeit von a, σ, B_n. □

2.5 Spectral Tetris

Wie wir gesehen haben bilden endliche Frames die Grundlage zur Untersuchung von Scattering Networks. Nun interessiert uns ein Verfahren zur Konstruktion von solchen endlichen Frames mit gegebenen Eigenschaften. Wir werden es zunächst beschreiben und später erweitern, um damit Frames aus CNNs zu verändern.

Beispiel 2.25 Wie bereits oben in Bemerkung 2.18 beschrieben, gibt es zahlreiche Möglichkeiten, Frames zu konstruieren. Für $T(N) = \ell(\mathbb{Z}_N)$ können wir etwa der Standardbasis $\{e_1, \ldots, e_N\}$ beliebige Elemente hinzufügen und erhalten weiterhin einen Frame. Wir suchen jedoch insbesondere tight finite frames oder solche, die diese Eigenschaft zumindest näherungsweise erfüllen. D.h. es soll für die Frame Grenzen A, B gelten $\frac{B}{A} \approx 1$ (auch die Kondition des Frames genannt). Zudem suchen wir solche, die durch eine regelmäßige Struktur, wie bspw. Translationen eines Elements erzeugt werden.
Ein einfaches Beispiel lässt sich im Zusammenhang der Signalverarbeitung mit Filtern wie folgt konstruieren: Wir identifizieren dazu den Raum $\ell(\mathbb{Z}_N)$ mit $\mathbb{R}^N$. Wir möchten einen Filter, d.h. ein $\phi \in T(N)$ konstruieren, sodass Translationen $\{T_k\phi\}_{k=1}^N$ einen Frame von $T(N)$ bilden. Nach Lemma 2.29 können wir zudem jeden zeitinvarianten Operator auf $T(N)$ so darstellen. Betrachten wir nun einen Filter mit Länge 3 (a_1, a_2, a_3) in $T(N)$. Der Syntheseoperator A_Ψ des zuugehörigen Frames Ψ besitzt dann die Form einer zirkulären

Matrix

$$A_\Psi = \begin{pmatrix} a_1 & & & a_2 \\ a_2 & a_1 & & a_3 \\ a_3 & a_2 & \ddots & \\ & a_3 & \ddots & \\ & & \ddots & a_1 \end{pmatrix}$$

Wählen wir einen Filter eines Wavelets der diskreten Wavelet-Transformation, etwa $(\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}})$ für das diskrete Haar-Wavelet, erhalten wir in der ersten Scattering Schicht (ohne Aktivierungsfunktion) genau die diskreten Wavelet-Koeffizienten auf der gewählten Skala.

Mit dem Verfahren des sogenannten „Spectral Tetris" lassen sich stückweise Frames aufbauen, welche gewünschte Eigenschaften besitzen wie bspw. unit form tight frames. Wir benötigen zunächst den Satz von Schur-Horn zur Konstruktion der Methode. Die Darstellung erfolgt nach [9, Kapitel 2]

Lemma 2.26 (**Satz von Schur-Horn**) Sei S Hilbert-Schmidt-Operator, d.h. ein positiver, selbstadjungierter Operator auf dem Hilbertraum H und seien $\lambda_1 \geq \cdots \geq \lambda_M \geq 0$ die Eigenwerte von S. Weiter sei $N \geq M$ und seien $a_1 \geq \cdots \geq a_N$ positive reelle Zahlen. Dann sind äquivalent:

1. Es existiert ein Frame $\{f_n\}$ von H mit Frame-Operator S und $\|f_n\|_H = a_n$ für alle $1 \leq n \leq N$.

2. Für alle $j \leq k \leq M$ gilt
$$\sum_{j=1}^{k} a_j^2 \leq \sum_{j=1}^{k} \lambda_j$$
und
$$\sum_{j=1}^{N} a_j^2 = \sum_{j=1}^{N} \lambda_j.$$

Nun möchten wir einen endlichen Frame auf einem Hilbertraum H konstruieren. Aus Lemma 2.20 ist der Zusammenhang zur speziellen Matrizen bekannt, die wir nun mithilfe des Satzes von Schur-Horn erzeugen können. Ursprünglich ist Spectral Tetris in [10] eingeführt worden als ein Verfahren um unit norm tight finite frames zu erhalten. Um dies zu erhalten, müssen 3 Voraussetzungen an die Synthese-Matrix A_Φ des Frames Φ wie oben in 2.20 definiert, erfüllt sein.

Korollar 2.27 Die zur Matrix A_Φ gehörende Menge Φ ist ein endlicher unit norm tight Frame, falls gilt

1. Die Spalten von $A_\Phi = (\mathbf{a}_1 | \ldots | \mathbf{a}_m)$ haben Norm 1,

2. die Zeilen von A_Φ sind orthogonal und

3. die Zeilen von A_Φ haben die gleiche Norm.

Aus diesen Forderungen lässt sich ein Verfahren ableiten, solche Matrizen und somit Frames zu erzeugen. Wir zeigen dies zunächst anhand eines Beispiels.

Beispiel 2.28 Nehmen wir an, wir möchten einen Frame mit Norm $\frac{5}{2}$ aus 6 Elementen des $\mathbb{R}^3$ konstruieren, so beginnen wir zunächst mit einer noch ungefüllten 3×6-Matrix A_Φ.

$$A_\Phi = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Geben wir nun zunächst die ersten Elemente aus einem bestehenden Frame vor (z.B. zweifach der erste Vektor der Standardbasis des $\mathbb{R}^3$)

$$\begin{pmatrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Damit ist die Norm der ersten Zeile gleich $2 < \frac{5}{2}$. Um die Norm zu erhöhen, ohne die Orthogonalität zu verletzten benutzen wir eine Matrix der Form

$$B_{x,y} = \begin{pmatrix} \sqrt{\frac{x}{2}} & \sqrt{\frac{x}{2}} \\ \sqrt{\frac{y-x}{2}} & -\sqrt{\frac{y-x}{2}} \end{pmatrix}.$$

Für diese gilt

$$B_{x,y}B_{x,y}^* = \begin{pmatrix} x & 0 \\ 0 & y-x \end{pmatrix}$$

und mit $y = 2, x = \frac{1}{2}$ eingesetzt in unsere Matrix erhalten wir

$$\begin{pmatrix} 1 & 1 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}.$$

Nun ist die Norm der zweiten Zeile gleich $\frac{3}{2}$ und wir können den Frame weiter ergänzen, bspw. zu

$$\begin{pmatrix} 1 & 1 & \frac{1}{2} & \frac{1}{2} & 0 & 0 \\ 0 & 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sqrt{\frac{5}{2}} \end{pmatrix}.$$

Durch iteratives Vorgehen lässt sich so durch Spectral Tetris ein Frame konstruieren. Für weitere Details und Voraussetzungen an die vorgegebene Norm und Größe des Frames siehe auch [9, §13].

Im Rahmen der Optimierung von CNNs ist für die Stabilität ein unit norm tight frame gesucht, der die aus den bisher durch bspw. Backpropagation-Verfahren erhaltenen Filtern und damit Frames erhält und diese nur skaliert. Weiterhin können aber auch zusätzliche Elemente dem Frame hinzugefügt werden.
Diese Idee führt zu dem folgenden Algorithmus:

Input : Matrixdarstellung des Synthese Operators eine Frames $A_\Phi = (\mathbf{a}_1|\dots|\mathbf{a}_m)$, obere Schranke M_1 an die Elementnorm, $\mathbf{d}$ Abstiegsrichtung zu $\mathbf{a}_k = (a_{k1}, \dots, a_{kN})$.
Output : Optimierter Frame $\tilde{A}_\Phi$
Bestimme Normen m_i für $i = 1, \dots, N$ von A_Φ
while $M' < M$ *und* $i \leq N$ **do**
  if $A_d := (\mathbf{a}_1|\dots|\mathbf{a}_k + \mathbf{d}|\dots|\mathbf{a}_m)$ *erfüllt Frame-Eigenschaften und* $\sigma(A_d) \leq B_1, \|\mathbf{a}_k + \mathbf{d}\| \leq M_1$ **then**
    $\tilde{A}_\Phi = A_d$ und **Ende**
  end
  Normiere $\|\mathbf{a}_k + \mathbf{d}\| = 1$
  Wähle zu i mit $d_i \neq 0$ ein $\ell \in \{i+1, \dots, N\}$ sowie $k \neq j \in \{1, \dots, M\}$
  if *Es existiert kein solches* i *und* j **then**
    Füge weitere Spalte $\mathbf{a}_{m+1}$ hinzu und fülle diese mit den Spectral Tetris Algorithmus nach [9, Alg. 13.1, p. 455]
  end
  Sei $x = a_{ki}^2 + a_{ji}^2$
  Sei $\alpha = \left(\frac{-a_{j\ell}^2 \pm \sqrt{a_{j\ell}^4 - 4a_{k\ell}^2 a_{j\ell}^2 a_{ki}^{-2} x}}{2a_{k\ell}^2} \right)^{1/2}$ und $\beta = -a_{ji} - \frac{\alpha^2 a_{ki} a_{k\ell}}{a_{j\ell}}$
  Setze $\tilde{\mathbf{a}}_k := \alpha(\mathbf{a}_k + \mathbf{d})$ und $\tilde{a}_{ji} = a_{ji} + \beta, \tilde{a}_{j\ell} = a_{j\ell}$
  $i \to i+1$, $M' = \max_{k=i}^m \sum_{j=1}^N a_{jk}^2$
end

Algorithmus 1 : Ein modifizierter Spectral Tetris-Algorithmus zur Modifizierung von Frames

Hierbei sei $\|\cdot\|$ die Norm aus einem Hilbertraum. Wie man durch Nachrechnen sieht, besitzt die modifizierte Matrix $\begin{pmatrix} \tilde{a}_{ki} & \tilde{a}_{ji} \\ \tilde{a}_{k\ell} & \tilde{a}_{j\ell} \end{pmatrix}$ orthogonale Zeilen und die Quadratsumme der ersten Zeile ergibt x. Damit erfüllt die resultierende Matrix $\tilde{A}_\Phi$ nach Korollar 2.27 die Eigenschaften der Darstellung eines Frame-Operators.

2.6 Stabilität des Scatterings mit endlichen Frames

Wir haben das Konzept der Scattering Networks auf eine entsprechende Transformation für endliche Frames verallgemeinert. Durch diese Definition wird im Gegensatz zu den semi-diskreten Frames aus [53],[84] nicht mehr die Invarianz bezüglich ganzer Lie-Gruppen

mehr möglich sein, jedoch werden andere Stabilitätsaussagen bestehen bleiben. In CNNs werden wie in den hier definierten Scattering Networks Faltungsoperationen der Form $(\phi_i * x)$ durchgeführt. Wir wollen nun deren Stabilität und den Einfluss von Störungen des Inputs x untersuchen. Zunächst bemerken wir, dass die Faltung von Elementen $a, b, c \in H$ folgende Aussagen erfüllt:

$$a * b = b * a \tag{2.21}$$

$$a * \delta = a \tag{2.22}$$

$$(\alpha a) * b = \alpha(a * b), \alpha \in \mathbb{C} \tag{2.23}$$

$$(a + b) * c = (a * c) + (b * c) \text{ (Distributivität)} \tag{2.24}$$

Aus diesen grundlegenden Eigenschaften können wir unmittelbar herleiten, dass ein flaches CNN nicht translationsinvariant sein kann, wie das folgende Lemma zeigt.

Lemma 2.29 Sei $X \subseteq \ell_2(\mathbb{Z}_N)$ und L ein linearer Operator $X \to X$. Dann sind äquivalent:

1. L ist *zeitinvariant*, d.h. $Lx = x * \phi$ für ein $\phi \in X$ und jedes $x \in X$.
2. L ist linear und kommutiert mit Translationen, d.h. $LT = TL$ für jeden Translationsoperator T.
3. L ist eine Linearkombination der Operatoren $\{T_p\}_{p \in \mathbb{Z}_p}$.

Beweis: Dies ist eine wohlbekannte Aussage aus der Frame-Theorie, siehe z.B. [9, Prop. 10.2].
Sei zunächst $X \subset \ell_2(\mathbb{Z}_N)$ und L zeitinvariant, es gilt $Lx = \phi * x = \sum_p \phi(p) x(\cdot - p) = \sum_{p \in \mathbb{Z}_N} \phi(p)(T_p x)(\cdot)$. Daraus folgt weiter, dass L linear und da $T_p T_q = T_q T_p$ für alle $p, q \in \mathbb{Z}_N$ und es folgt Eigenschaft 2.
Gelte nun 2. für L, und sei $x \in \ell_2(\mathbb{Z}_N)$ dargestellt als Summe $x = \sum_{j=1}^N \delta_j$. Es gilt $L\delta_j = LT_j\delta_0$ und nach Voraussetzung $LT_j\delta_0 = T_j L\delta_0$. Definiere $y := L\delta_0$ und erhalte $Lx = \sum_{j=1}^N T_j y$. Umgekehrt folgt aus 3. die Aussage 2., da Translationsoperatoren untereinander kommutieren. □

Korollar 2.30 Jede Schicht eines Faltungs-Netzwerkes kommutiert als Summe für Faltungsoperatoren nach mit Translationen, d.h. für jede Schicht f_n eines Netzwerkes und einen Translationsoperator T ist $f_n(Tg) = Tf_n(g)$ für alle $g \in X$ für geeignetes $X \subseteq \ell_2(\mathbb{Z}_N)$. Daraus folgt insbesondere, dass ein Convolutional Neural Network, dass keine Aktivierungsfunktion (d.h. $\sigma = \mathrm{id}$) besitzt, unabhängig von der Tiefe, nicht translationsinvariant sein kann.

Invarianz bzgl. Translationen und Modulationen lässt sich jedoch annähernd für tiefe Netze zeigen: Sei dazu $F_{\tau,\omega} f(x) = e^{2\pi i \omega(x)} f(x - \tau(x))$ für eine hinreichend glatte Funktion τ. Sei zudem L_R^2 der Unterraum der R-bandbeschränkten Funktionen von L_2.

Es gilt folgende Aussage für das verallgemeinerte Scattering wie in [84, Theorem 2] für sowohl semi-diskrete, wie auch diskrete Frames.

Lemma 2.31 Es existiert eine Konstante $C > 0$, sodass für alle $f \in L^2_R(\mathbb{R}^d), \omega \in C(\mathbb{R}^d, \mathbb{R})$ und $\tau \in C^1(\mathbb{R}^d, \mathbb{R}^d)$ mit $\|D\tau\|_\infty < \frac{1}{2d}$ gilt

$$\|f - F_{\tau,\omega} f\|_2 \leq C(R\|\tau\|_\infty + \|\omega\|_\infty)\|f\|_2.$$

Der Beweis benötigt ein klassisches Lemma von Schur:

Lemma 2.32 (Lemma von Schur [75]) Sei $C > 0$ und $k : \mathbb{R}^d \to \mathbb{R}^d$ eine lokal integrierbare Funktion mit

$$\sup_{u \in \mathbb{R}^d} \int_{\mathbb{R}^d} |k(x,u)| dx \leq C \text{ und } \sup_{u \in \mathbb{R}^d} \int_{\mathbb{R}^d} |k(x,u)| du \leq C.$$

Dann ist der Integraloperator $K : L^2(\mathbb{R}^d) \to L^2(\mathbb{R}^d)$ gegeben durch

$$Kf(x) = \int_{\mathbb{R}^d} f(u) k(x,u) du$$

ein beschränkter Operator mit Norm $\|K\|_{2,2} \leq C$.

Beweis: Wir zeigen nun Lemma 2.31 analog wie in [84] mit der allgemeineren Variante für endliche Frames. Dazu schreiben $f - F_{\tau,\omega} f$ als einen Integraloperator und wenden dann das Lemma von Schur an.
Wir suchen zunächst einen Integraloperator K mit $Kf = f - F_{\tau,\omega} f$.
Sei dazu $\gamma \in S(\mathbb{R}^d, \mathbb{C})$ so, dass $f = f * \gamma$ für alle R-bandbeschränkten Funktionen $f \in L^2_R(\mathbb{R}^d)$. Dabei ist $L^2_R(\mathbb{R}^d) = \{f \in L^2(\mathbb{R}^d) \mid \text{supp}\hat{f} \subseteq B_R(0)\}$. Dies liefert uns dann einen Operator

$$A_\gamma : L^2(\mathbb{R}^d) \to L^2(\mathbb{R}^d) : A_\gamma f = f * \gamma$$

Dies ist wohldefiniert, da $\|A_\gamma\| \leq \|f\|\|\gamma\|$ und $\gamma \in L^1(\mathbb{R}^d)$ (Youngsche Ungleichung) und es gilt nach Wahl von γ, dass $A_\gamma f = f$ für alle $f \in L^2_R(\mathbb{R}^d)$. Damit ist $\|f - F_{\tau,\omega} f\|_2 = \|A_\gamma f - F_{\tau,\omega} A_\gamma f\|_2$. Wir betrachten also den Operator $F_{\tau,\omega} A_\gamma - A_\gamma$ mit Kern $k(x,u) = e^{2\pi i \omega(x)} \gamma(x - \tau(x) - u) - \gamma(x - u)$. Dies ist lokal integrierbar, da $\gamma \in S$ und $\tau \in \mathcal{C}(\mathbb{R}^d)$. Wir möchten nun das Lemma von Schur anwenden, es bleibt die Beschränktheit des Kerns k zu zeigen. Wir definieren

$$h^{x,u} : \mathbb{R} \to \mathbb{C} : h^{x,u}(t) = e^{2\pi i t \omega(x)} \gamma(x - t\tau(x) - u) - \gamma(x - u)$$

Dann ist $h^{x,u}(t) = h^{x,u}(0) + \int_0^t (\frac{d}{dt} h^{x,u})(s) ds$ für alle $t \in \mathbb{R}$. Bemerke, dass für $t = 1$ $|k(x,u)| = |h^{x,u}(1)| \leq \int_0^1 |(\frac{d}{dt} h^{x,u})(s)| ds$. Es ist

$$\begin{aligned} |(\frac{d}{dt} h^{x,u})(s)| &\leq |\langle \nabla\gamma(x - s\tau(x) - u), \tau(x)\rangle| + |2\pi i \omega(x) \gamma(x - s\tau(x) - u)| \\ &\leq \|\tau\|_\infty |\nabla\gamma(x - s\tau(x) - u)| + 2\pi \|\omega\|_\infty |\gamma(x - s\tau(x) - u)|. \end{aligned}$$

Damit und mit $B_R := \{x \in \mathbb{R}^d, x \leq R\}$

$$\begin{aligned}\int_{\mathbb{R}^d} |k(x,u)|du &\leq \int_{\mathbb{R}^d} \int_0^1 |(\frac{d}{dt} h^{x,u})(s)|ds\, du \\ &\leq \int_0^1 \int_{B_R} \|\tau\|_\infty |\nabla\gamma(x - s\tau(x) - u)| + 2\pi\|\omega\|_\infty |\gamma(x - s\tau(x) - u)|du\, ds \\ &\leq \|\tau\|_\infty \int_0^1 \int_{B_R} |\nabla\gamma(x - s\tau(x) - u)|du\, ds \\ &+ 2\pi\|\omega\|_\infty \int_0^1 \int_{B_R} |\gamma(x - s\tau(x) - u)|du\, ds \\ &\leq \|\tau\|_\infty \|\nabla\gamma\|_1 + 2\pi\|\omega\|_\infty \|\gamma\|_1\end{aligned}$$

und analog

$$\int_{\mathbb{R}^d} |k(x,u)|dx \leq \|\tau\|_\infty \|\nabla\gamma\|_1 + 2\pi\|\omega\|_\infty \|\gamma\|_1.$$

Dann folgt die Aussage mithilfe des Lemmas von Schur und $C = 2\pi(\|\nabla\gamma\|_1 + \|\gamma\|_1)$. □

Proposition 2.33 Sei $\{S_J\}_{J=1}^M$ der Output eines Scattering Networks mit UNTF (unit norm tight frame) Frames Ψ_n wie in Definition 2.14. Dann gilt

$$\|S_J(f) - S_J(F_{\tau,\omega} f)\|_\Upsilon \leq C(R\|\tau\|_\infty + \|\omega\|_\infty)\|f\|_2$$

für alle R-bandbeschränkten Funktionen.

Beweis: Es bleibt nach Lemma 2.31 zu zeigen, dass $\|S_J f - S_J h\| \leq \|f - h\|_2$ für alle $f, h \in L_2(\mathbb{R}^d)$. Dies ist eine Verallgemeinerung von Proposition 2.5 aus [53].
Es gilt nach Definition

$$\|S_J f - S_J h\|^2 = \sum_{p \in \mathcal{P}^J}^{\infty} \|S_J[p]f - S_J[p]h\|^2,$$

für jedes $p = (p_n)_{n=1}^J \in \mathcal{P}^J$ und $p' := (p_n)_{n=1}^{J-1}$ ist $\|S_J[p]f - S_J[p]h\| \leq \|S_J[p'](S_J[p_J]f - S_J[p_J]h)\| \leq \|S_J[p']f - S_J[p']h\|\, \|S_J[p_J]\|$. Der Operator $S_J[p_n]$ ist aufgrund der Eigenschaften der Frames nichtexpansiv: $\|S_J[p_J]f\|^2 = \sum_{n=1}^N \|\psi_{J,n} * f\|^2 \leq 1$. (vgl. auch Lemma 2.5, allgemeiner gilt für Frames mit oberer Schranke B, dass $\|S_J[p_J]f\| \leq \sqrt{B}$) □

Wir sehen mit Proposition 2.33, dass die Anwendung eines Scattering Netzwerks wie in 2.14 definiert stabil ist, in dem Sinn, dass es eine Lipschitz-Eigenschaft in der entsprechenden Norm erfüllt. Gilt für die zugrundeliegenden Frames zu dem Scattering, dass deren obere Frame-Schranke B gleich oder nahe bei 1 ist, so überträgt sich diese Eigenschaft auch auf den Scattering Operator. Typischerweise werden in der Theorie der Bildverarbeitung und -erkennung Störungen der Form wie durch Operatoren $F_{\tau,\omega}$ wie oben betrachtet. Diese bieten Modelle für praktisch vorkommende perspektivische Verschiebungen, Streckungen und Stauchungen und auch Farb- und somit Beleuchtungsänderungen. Nutzt man

Scattering Networks zur Klassifikation, so wird die Klassifikation stabil gegenüber solchen Veränderungen sein. Dies löst nicht im Allgemeinen das Problem der Adversial Attacks wie in Bemerkung 1.60 vorgestellt, da diese in der Regel nicht mit hier betrachteten (stetigen) Veränderungen durch Operatoren $F_{\tau,\omega}$, sondern vielmehr durch pixelweise, kleine Änderungen operieren. Gerade in den oftmals durch die Bildverarbeitung hochauflösender Bilder gegebenen hochdimensionalen Räumen wie $\mathbb{R}^{3\times1000\times1000}$, addieren sich kleine, kaum für das menschliche Auge erkennbare Änderungen pro Bildpunkt schnell zu großen Differenzen auf, sodass auch eine Lipschitz-Eigenschaft wie oben formuliert, solche Attacken nicht alleine verhindern kann.
Im nächsten Abschnitt werden wir untersuchen, welche Eigenschaften in der Praxis verwendete CNNs haben, für die Scattering Networks hier als theoretisches Modell dienen.

2.7 Untersuchung von empirischen Frames aus CNNs

Wir benutzen die CNNs aus der Matlab Deep Learning™Toolbox [81], welche mit dem ImageNet/CIFAR/MNIST-Datensatz vortrainiert wurden. Dabei untersuchen wir insbesondere die Eigenschaften der sich aus den Gewichten der Faltungsschichten ergebenden Frames, bzw. für den Unterraum der Trainingselemente bzw. der Inputs der jeweiligen Netzwerkschicht, die resultierenden Framekonstanten und damit Stabilitätsvorhersagen, also

$$A\|x\|^2 \leq \sum_{i=1}^{m} |\langle x, \phi_i\rangle| \leq B\|x\|^2 \quad \forall x \in \Upsilon$$

mit Υ Trainings-/Input- oder Testmenge. Die Optimierung des Netzwerkes besteht dann darin, diese Menge Υ zu vergrößern ohne dass die Klassifikationsleistung des Netzwerkes abnimmt oder deren Stabilität beeinträchtigt wird (Generalization).

Beispiel 2.34 Betrachten wir zunächst ein einfaches Convolutional Neuronales Netzwerk, das zur Klassifikation des MNIST-Datensatzes verwendet wird. Dieser Datensatz besteht aus 28×28-Pixeln großen Schwarz-Weiß-Bildern welche zu Zahlen aus den 10 Kategorien $0, 1, \ldots, 9$ gehören. Das einfache verwendete Netzwerk besteht aus 3 Schichten sowie Input- und Output-Schicht: Einer Faltungsschicht mit 32 Neuronen mit ReLU-Aktivierungsfunktion und Filter der Größe 3×3, einer Pooling-Schicht mit max-pooling auf 2×2 Pixeln und einer klassischen Schicht mit 100 Neuronen und ReLU-Aktivierungsfunktion. Dieses einfache Modell findet sich bspw. bei Brownlee [7] und wurde mithilfe der Routinen aus der Matlab Deep Learning Toolbox [81] optimiert.
Uns interessieren die Fragestellungen: Gegeben ein trainiertes Netz, wie können wir aus den Filtern etwas über die Aufgabe, auf die dieses Netzwerk trainiert wurde herauslesen. Lassen sich Rückschlüsse auf die Klassifikationsleistung ziehen?
Betrachten wir die Eigenschaften der Betrachten wir nun ein Beispiel zur Klassifikation mit (verallgemeinerten) Scattering Networks. Wir möchten zunächst eine der grundlegendsten Aufgaben der Bilderkennung betrachten und Kanten in einem Bild erkennen. Dies ist eine

Aufgabe die auch durch andere, nicht dem modernen Deep-Learning zugeordnete, Methoden gelöst werden kann.

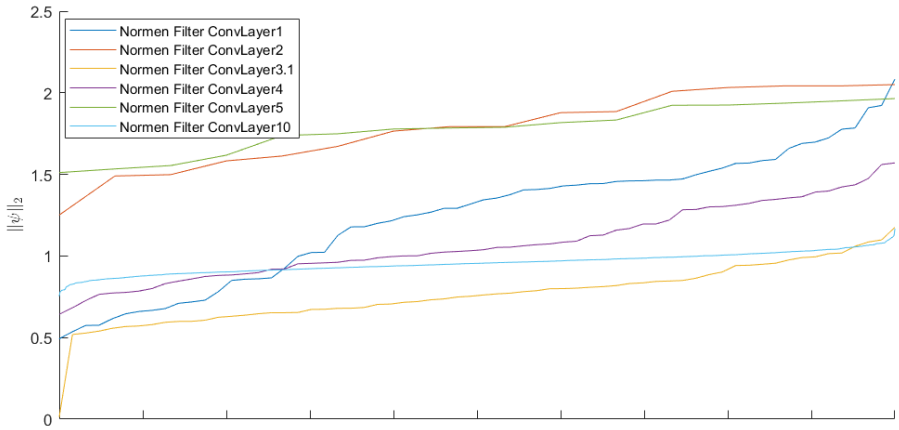

Abbildung 2.4: Normen der Filtern aus trainierten Gewichten für das tiefe CNN „Squeezenet"

Wie in der Grafik 2.4 zu sehen ist, bilden die Filter eines CNNs aus der Anwendung approximativ einen tight frame, denn die Kondition des Frames ergibt sich aus dem Quotienten der maximalen und minimalen Norm der Filter, welche insbesondere in tieferen Schichten nahe bei einander liegen.
Wir haben im vorigen Abschnitten zudem ein Verfahren in Algorithmus 1 gesehen, einen gegebenen Frame so zu verändern, dass

- er weiterhin ein Frame mit Frame-Konstanten A, B möglichst unverändert bleibt,
- sich die Klassifikationsleistung verbessert.

Eine Anwendung auf tiefe Neuronale Netzwerke steht in der Praxis vor dem Problem des hohen Rechenaufwands und der möglichen Änderungen der Netzwerkarchitektur, d.h. insb. Anzahl der Filter, durch die Möglichkeit des Algorithmus, weitere Filter hinzuzufügen.
Wir möchten nun noch kurz auf ein verwandtes Ergebnis zur Stabilität von Frames bei Änderung derer Frame-Konstanten eingehen. Dies ist eine einfache Anwendung des folgenden Lemmas aus der Theorie der Frames.

Definition 2.35 Sei H ein Hilbertraum. Eine Folge $\{f_j\} \subset H$ heißt *Besselfolge* mit Schranke B, falls $\sum_{i=1}^{\infty} |\langle f, f_i \rangle|^2 \leq B\|f\|_H^2$ für alle $f \in H$ gilt.

Es ist wohlbekannt (siehe bspw. [12, Kapitel 15.1]), dass gilt:

Lemma 2.36 Sei $\Phi = \{\phi_j\}$ ein Frame des Hilbertraums H mit Frame-Konstanten A, B.

Sei weiter $\Psi = \{\psi_j\} \subset H$ eine Folge mit $\{\phi_j - \psi_j\}_j$ eine Besselfolge mit Schranke $M < A$. Dann ist Ψ ein Frame mit Konstanten $A' = A(1 - \sqrt{M/A})^2$ und $B' = B(1 + \sqrt{M/B})^2$.

Beweis: Es gilt $\sum_{i=1}^{\infty} |\langle \psi_i, f\rangle|^2 \leq \sum_{i=1}^{\infty} |\langle \psi_i - \phi_i, f\rangle|^2 + |\langle \phi_i, f\rangle|^2 \leq \sqrt{M}\|f\|^2 + B\|f\|^2$.

Seien nun $\Phi_n = (\{\phi_{n,j}\})_n$ und $\Psi_n = (\{\psi_{n,j}\})_n$ zwei Folgen von Frames mit Scattering Propagations-Operatoren $U_{n,j}$ bzw. $U'_{n,j}$ und es gelte für alle n, dass $\{\phi_{n,j} - \psi_{n,j}\}_j$ eine Besselfolge mit Schranke M ist. Seien (A_n, B_n) die Frame-Schranken von Φ_n. Setzen wir dies in eine Filterbank ein, so können wir abschätzen $\|U_{n,j}f - U'_{n,j}f\|_H \leq B_n - B_n(1 + \sqrt{M/B_n})^2$ nach dem vorigen Lemma. Daraus können ebenfalls grob ohne genaueres Wissen über die Beschaffenheit des Frames Stabilitätsaussagen des Scatterings bei Veränderung des Frames getroffen werden.

Kapitel 3

Reduced Basis Method mit Ridgefunktionen

Wir wollen die Möglichkeiten der Approximation mit Ridgefunktionen anhand einer Anwendung zur Lösung von parametrischen partiellen Differentialgleichungen mit einer modifizierten Reduzierten Basis Methode (Reduced Basis Method) zeigen. Dafür entwickeln wir einen Algorithmus, der eine geeignete Basis bildet und online-effizient eine Lösung bestimmt. Die im folgenden Kapitel gezeigten Resultate und Algorithmen sind zum Teil in [27] zu finden. Die Software zu diesem Kapitel ist unter [28] zu finden.

3.1 Einführung und Zielsetzung

Wir möchten eine parametrische partielle Differentialgleichung (PPDE) im schwachen Sinn auf dem Gebiet $\Omega \subset \mathbb{R}^d$ lösen. Solche PPDEs treten häufig im Zusammenhang mit Kontrollproblemen und anderen Optimierungsproblemen auf. Dabei ist eine partielle Differentialgleichung $L_\mu u = f_\mu$ zu vielen verschiedenen Parametern μ möglichst schnell zu lösen, bspw. für in Anwendungen in Echtzeit oder mit beschränkter Rechenleistung. Sei u_μ im Folgenden die Lösung der PPDE zum Parameter μ. Sind die Parameter aus einer kompakten Teilmenge D des $\mathbb{R}^n$ und die Lösungen u_μ aus einem Hilbertraum H, so folgt, dass auch die sog. Lösungsmannigfaltigkeit $F = \{u_\mu \,|\, \mu \in D\}$ kompakt in H ist. Der Ansatz der Methode der reduzierten Basis Methode besteht nun aus zwei Phasen [34]: In der ersten, sogenannten „offline"-Phase wird die PPDE für einige Parameter $\mu_i \in D, i = 1, \ldots, N$ gelöst. Dadurch erhalten wir einen endlichdimensionalen Unterraum von H, den sogenannten „reduzierten" Unterraum $\Phi_N = \{\varphi_1, \ldots, \varphi_N\}$ (im einfachsten Fall durch $\varphi_i = u_{\mu_i}$). Dann kann in einer zweiten „online"-Phase eine Lösung der PPDE für ein Parameter $\mu \notin \{\mu_i\}_{i=1}^N$ sehr effizient aus diesem Raum approximiert werden durch $\sum_{i=1}^N \alpha_i u_{\mu_i}$, wobei nur noch die Koeffizienten α_i durch Petrov-Galerkin-Projektion auf den reduzierten Unterraum durch ein kleines Gleichungssystem bestimmt werden müssen. Eine weitere Einführung in die Methode liefert [69], [35], [34]
Der minimale Fehler einer optimalen N-Term-Approximation kann durch die Kolmogorov-

N-Weite abgeschätzt werden. Diese ist allgemein für eine Klasse $F \subset H$ definiert als (cf. [42])

$$d_N(F) := \inf_{\substack{Y \subset H \\ \dim Y = N}} \sup_{f \in F} \inf_{g \in Y} \|f - g\|_H.$$

Hier setzen wir für F die Lösungsmannigfaltigkeit unser PPDE ein und überprüfen wie schnell die Kolmogorov-N-Weite in Abhängigkeit von N fällt. Die klassische RB-Methode hat den Nachteil, dass für bestimmte Differentialgleichungen wie Transport- und Wellengleichungen die Kolmogorov-N-Weite für $N \to \infty$ nur wie ein Polynom niedrigen Grades abklingt, d.h. $d_N(F) \backsimeq \mathcal{O}(N^{-r})$ für ein $0 < r < 1$, wie u.a. in [29] gezeigt wurde. Da die Kolmogorov-N-Weite eine Eigenschaft der Klasse F und somit der zu lösenden Gleichung ist und nicht von der genauen Wahl der Unterräume $Y \subset H$ durch die RB-Methode abhängt, wird auch jedes andere lineare Schema zur Approximation von Lösungen dieser Gleichungen schlechtes Verhalten zeigen. Wir können jedoch dieses Problem durch die Benutzung eines nichtlinearen Approximationsschemas überwinden. Konkret möchten wir Ridge-Funktionen verwenden.

Ridge-Funktionen sind, wie wir bereits im ersten Kapitel dieser Arbeit sehen konnten, einfache und vielseitige multivariate Funktionen. Sie treten natürlicherweise als Lösung von Transport- und Wellengleichungen auf, wie auch in multivariaten Fourier-Reihen $\{e^{i\langle \mathbf{n}, \mathbf{x}\rangle}\}_{n \in \mathbb{Z}}$ und wurden auch schon direkt zur Lösung von partiellen Differentialgleichungen eingesetzt. [46] Zudem ist viel über die Approximation mit Ridge-Funktionen, insbesondere im Zusammenhang mit Neuronalen Netzwerken bekannt.

Wir möchten den linearen Teil der Approximation beibehalten, da dieser für bestimmte elliptische Gleichungen bereits gut funktioniert (für solche Gleichungen ist $d_N(F) = \mathcal{O}(e^{-N})$), und eine Menge an Ridge-Funktionen $\mathcal{V}_M := \{v_1, \dots, v_M\}$ hinzufügen und damit eine Approximation der Form

$$u_\mu(x) = \sum_{i=1}^{N} \alpha_i \varphi_i(x) + \sum_{j=1}^{M} c_j v_j(\langle a_j, x\rangle + b_j), \quad x \in \Omega \tag{3.1}$$

benutzen. Es ist klar, dass durch das Addieren von Ridge-Funktionen eine nichtlineare Komponente zu der Summe hinzukommt. Dadurch wird die Approximation vermutlich deutlich verbessert, insbesondere da viele Lösungen von Differentialgleichungen aus Ridge-Funktionen bestehen, wie die folgende Bemerkung aussagt.

Bemerkung 3.1 Welche parametrischen partiellen Differentialgleichungen lassen sich durch eine solche Approximation lösen?

Für zwei Dimensionen gilt: Die Summe von Ridge-Funktionen

$$u_\mu(x) = \sum_{j=1}^{M} c_j(\mu) v_j(\langle (a_{j,1}(\mu), a_{j,2}(\mu)), (x_1, x_2)\rangle + b_j(\mu))$$

löst die PPDE

$$\prod_{j=1}^{M} L_j(\mu) := \prod_{j=1}^{M} (a_{j,2}\partial_{x_1} u - a_{j,1}\partial_{x_2}) = 0$$

da für den j-ten Term $w_j := c_j(\mu)v_j((a_{j,1}(\mu)x_1 + a_{j,2}(\mu)x_2 + b_j(\mu))$ gilt

$$\begin{aligned} L_j(\mu)w_j &= (a_{j,2}\partial_{x_1}w_j - a_{j,1}\partial_{x_2}w_j) \\ &= (a_{j,2}a_{j,1} - a_{j,1}a_{j,2})c_j(\mu)v_j'(\dots) \\ &= 0 \end{aligned}$$

Nehmen wir weiter die affine Summe $\sum_{i=1}^{N}\alpha_i(\mu)\varphi_i(x)$ hinzu. Diese löst die PDE

$$\prod_{i=1}^{N} F_j u/\alpha_j(\mu) = 0,$$

wobei F_j lineare Differentialoperatoren mit $F_j\varphi_j = 0$ sind, z.B. $F_j = -\Delta$ für die homogene Poissongleichung.
Damit ergibt sich eine große Menge an Gleichungen die theoretisch exakt mit diesem Ansatz gelöst werden können.

Bemerkung 3.2 Für die Anwendung beschränken wir uns auf Ridge-Funktionen mit Profilfunktion $v \in L_2(\mathbb{R}) \cap \mathcal{C}_{pw}(\mathbb{R})$, d.h. auf quadratintegrierbare und stückweise stetige Funktionen.

Es treten zur Approximation zu einer gegebenen Funktion u, welche meist die Lösung einer PDE ist, verschiedene Szenarien auf:

- 1. Typ: Zu gegebener Profilfunktion $v \in L_2([0,1])$ ist Richtung $a \in \mathbb{R}^d$ gesucht, sodass

  $$\|u(\mathbf{x}) - cv(\langle \mathbf{x}, a\rangle + b_a))\|_2$$

  minimal

- 2. Typ: Zu gegebenen Richtungen $\mathbf{a}_j^* \in \mathbb{R}^d$ finde $v_j \in L^2([0,1])$, $j = 1,\dots,M$, sodass

  $$\left\| u(\mathbf{x}) - \sum_{j=1}^{M} c_j v_j(\langle \mathbf{x}, \mathbf{a}_j^*\rangle + b_{\mathbf{a}_j})) \right\|_2$$

  minimal ist.

- 3. Typ: Eine Kombination der ersten beiden Typen: Finde Richtung $\mathbf{a}^* \in \mathbb{R}^d$ und Profilfunktion $v^* \in L^2(\mathbb{R})$ mit

  $$\|u(\mathbf{x} - cv^*(\langle \mathbf{x}, a^*\rangle + b_a))\|_2$$

  minimal.

Wir beschränken uns hier zunächst auf Funktionen auf dem Einheitswürfel $\Omega = [0,1]^d$. Weiterhin fordern wir, dass die Richtung a bezüglich der 1-Norm normiert ist. Unter diesen Voraussetzungen können wir die Verschiebung b der Profilfunktion in Abhängigkeit von der Richtung a direkt bestimmen und müssen nur Profilfunktionen aus $L_2([0,1])$ betrachten.

Lemma 3.3 Sei $\mathbf{x} \in [0,1]^d$ und $\|\mathbf{a}\|_1 = 1$. Dann ist für

$$b_{\mathbf{a}} := -\sum_{i=1}^{d} \min\{0, a_i\} \tag{3.2}$$

$$\{\langle \mathbf{x}, \mathbf{a}\rangle + b_a \mid \mathbf{x} \in [0,1]^d\} = [0,1].$$

Wir führen nun den Raum der Linear/Ridge-Funktionen und eine einfache Schreibweise für die Funktionen daraus ein.

Definition 3.4 Seien $N, M \in \mathbb{N}_0$ und $\varphi_i \in L_2(\Omega)$ für $i = 1, \ldots, N$ sowie $v_j \in L_2(\mathbb{R})$, $j = 1, \ldots, M$ gegeben. Wir definieren

$$U_{N,M} = \left\{ \sum_{i=1}^{N} \alpha_i \varphi_i(x) + \sum_{j=1}^{M} c_j v_j(\langle \mathbf{a}_j, x\rangle + b_j) \;\middle|\; \alpha_i, b_j, c_j \in \mathbb{R}, \mathbf{a}_j \in \mathbb{R}^d \right\}. \tag{3.3}$$

Dann ist $U_{N,M}$ ein (nichtlinearer) Teilraum von $L_2(\Omega)$.
Mit $\alpha = (\alpha_i)_{i=1}^N, \mathbf{b} = (b_j)_{j=1}^M, \mathbf{c} = (c_j)_{j=1}^M$ und $\mathbf{a} = \{\mathbf{a}_j\}_{j=1}^M$ fassen wir die Variablen zusammen zu

$$\mathbf{y} := (\alpha, \mathbf{b}, \mathbf{c}, \mathbf{a}) \tag{3.4}$$

und schreiben $u_{\mathbf{y}}(x)$ für $\sum_{i=1}^N \alpha_i \varphi_i(x) + \sum_{j=1}^M c_j v_j(\langle \mathbf{a}_j, x\rangle + b_j)$.

Wir nennen im Zusammenhang der Ridge-Funktionen für die RB-Methode v_j die *Profilfunktion*, während wir diese im Zusammenhang mit Neuronalen Netzwerken meist als Aktivierungsfunktion bezeichnet haben und $\mathbf{a}$ die *Richtungen* anstelle von Gewichten bei den Neuronalen Netzwerken. Dies spiegelt die natürliche geometrische Interpretation im Zusammenhang der partiellen Differentialgleichungen wider. Die Bestapproximation aus der Menge $U_{N,M}$ existiert:

Lemma 3.5 Sei $u \in L_2(\Omega)$. Dann existiert ein $\mathbf{y}^* \in \mathbb{R}^{N+(d+2)M}$, das

$$\|u - u_{\mathbf{y}}\|_{L_2(\Omega)}$$

minimiert.

Beweis: $L_2(\Omega)$ ist ein Hilbertraum mit konvexer Norm und $U_{N,M} \subset L_2(\Omega)$ abgeschlossen. Daher folgt die Existenz des optimalen y aus Standardargumenten. (siehe bspw. [67, §2]) □

Analog folgt die Existenz für die Räume $L_p(\Omega)$ für $1 < p < \infty$, wenn man $U_{N,M}$ für geeignete φ_i, v_j als Teilraum dessen auffasst.

Bemerkung 3.6 Die Bestapproximation aus Lemma 3.5 muss nicht eindeutig sein, da der Raum $U_{N,M}$ nicht konvex ist. In Abschnitt 3.0.3 werden wir dies auch anhand mehrerer

Beispiele sehen. Für den später vorgestellten Algorithmus stellt dies jedoch kein Problem dar, da dieser auch mit mehreren Minima gut zurecht kommt.
Ebenfalls kann der L_2-Fehler in Lemma 3.5 in der Anwendung durch eine leichter zu berechnende Ersatzgröße wie die Norm des Residuums oder einen Fehlerschätzer ersetzt werden ohne das Verfahren zu ändern.

Definition 3.7 Sei $U_{N,M}$ und $\mathbf{y}$ wie in (3.4) definiert. Die zugehörige Kostenfunktion $J_u : \mathbb{R}^{N+(d+2)M} \to \mathbb{R}_+$ sei definiert als

$$J_u(\mathbf{y}) := \|u - u_{\mathbf{y}}\|^2_{L_2(\Omega)}.$$

Sei weiter $\mathbf{y}^* = \arg\inf_{\mathbf{y}} J_u(\mathbf{y})$ und $u_{\mathbf{y}^*} = \arg\inf_u \|u - u_{N,M}\|_{L_2(\Omega)}$. (Existenz nach Lemma 3.5)

Weiter ist die Bestimmung der optimalen Koeffizienten α_i und c_j aus (3.1) durch Lösung eines linearen Gleichungssystems möglich:

Lemma 3.8 Sei $u \in L_2(\Omega)$ und $\Phi_N, \mathcal{V}_M$ wie oben zusammen mit $\mathbf{a}_1, \ldots, \mathbf{a}_M, b_1, \ldots, b_M$ gegeben. Sei $G_N = (g_{i,j})_{i,j=1}^N$ mit $g_{i,j} = \langle \varphi_i, \varphi_j \rangle_{L_2(\Omega)}$, $H_M = (h_{i,j})_{i,j=1}^M$ mit $h_{i,j} = \langle v_i(\langle \mathbf{a}_i, \cdot \rangle + b_i), v_j(\langle \mathbf{a}_j, \cdot \rangle + b_j) \rangle_{L_2(\Omega)}$ und $L_{N,M} = (l_{i,j})_{i=1}^N = \langle \varphi_i, v_j(\langle \mathbf{a}_j, \cdot \rangle) \rangle_{L_2(\Omega)}, i = 1, \ldots, N, j = 1, \ldots, M$. Sei weiter $f_N(u) = (\langle u, \varphi_i \rangle_{L_2(\Omega)})_{i=1}^N$ und $g_m = (\langle u, v_j(\langle \mathbf{a}_j, \cdot \rangle + b_j) \rangle_{L_2(\Omega)})_{j=1}^M$.
Dann sind die optimalen Koeffizienten α, c durch das lineare Gleichungssystem

$$\begin{pmatrix} G & L_{N,M} \\ L_{N,M}^T & H_M \end{pmatrix} \begin{pmatrix} \alpha \\ c \end{pmatrix} = \begin{pmatrix} f_N(u) \\ g_m(u) \end{pmatrix} \tag{3.5}$$

bestimmt.

Beweis: Sei o.B.d.A. $b_j \equiv 0$. (Anderenfalls ersetze v_j durch $v_j(\cdot - b_j)$) Es gilt $\|u - u_{\mathbf{y}}\|_2^2 = \|u\|_2^2 + \|u_{\mathbf{y}}\|_2^2 - 2\langle u, u_{\mathbf{y}} \rangle$. Weiter ist

$$\begin{aligned} \langle u, u_{\mathbf{y}} \rangle &= \sum_{i=1}^N \alpha_i \langle u, \varphi_i \rangle + \sum_{j=1}^M c_j \langle u, v_j(\langle a_j, \cdot \rangle + b_j) \rangle \\ &=: f_N(u)^\top \alpha + g_m(u)^\top c. \end{aligned}$$

und

$$\begin{aligned} \|u_{\mathbf{y}}\|_2^2 &= \sum_{i,j=1}^N \alpha_i \alpha_j \langle \varphi_i, \varphi_j \rangle + \sum_{i,j=1}^M c_i c_j \langle v_i(\langle a_i, \cdot \rangle + b_i), v_j(\langle a_j, \cdot \rangle + b_j) \rangle \\ &\quad + 2 \sum_{i=1}^N \sum_{j=1}^M \alpha_i c_j \langle \varphi_i, v_j(\langle a_j, \cdot \rangle + b_j) \\ &=: \alpha^\top G \alpha + c^\top M c + 2\alpha^\top L_{N,M} c \end{aligned}$$

so dass dann $J_u(\mathbf{y}) = 2f_N(u)^\top\alpha - 2g_m(u)^T c + \alpha^\top G\alpha + c^\top Mc + 2\alpha^\top L_{M,N}c$. Die Matrizen G und M sind symmetrisch, daher folgt

$$\begin{aligned}\nabla J_u(\mathbf{y}) &= \begin{pmatrix}\partial_\alpha J_u(\mathbf{y})\\ \partial_c J_u(\mathbf{y})\end{pmatrix} = \begin{pmatrix}-2f_N(u) + G\alpha + G^T\alpha + 2L_{N,M}c\\ -2g_m(u) + Mc + L_{N,M}\alpha\end{pmatrix}\\ &= 2\begin{pmatrix}-f_N(u) + G\alpha + L_{N,M}c\\ -g_m(u) + Mc + L_{N,M}^T\alpha\end{pmatrix}\\ &= -2\begin{pmatrix}f_N(u)\\ g_m(u)\end{pmatrix} + 2\begin{pmatrix}G & L_{N,M}\\ L_{N,M}^\top & M\end{pmatrix}\begin{pmatrix}\alpha\\ c\end{pmatrix}\end{aligned}$$

Damit ist $\nabla J_u(\mathbf{y}) = 0$ genau dann, wenn

$$\begin{pmatrix}G & L_{N,M}\\ L_{N,M}^\top & M\end{pmatrix}\begin{pmatrix}\alpha\\ c\end{pmatrix} = \begin{pmatrix}f_N(u)\\ g_m(u)\end{pmatrix}$$

ist. Zudem ist

$$\nabla^2 J_u(\mathbf{y}) = 2\begin{pmatrix}G & L_{N,M}\\ L_{N,M}^\top & M\end{pmatrix}$$

symmetrisch und positiv semidefinit, da für $\mathbf{x} \in \mathbb{R}^{M+N}$ gilt

$$\mathbf{x}^\top\begin{pmatrix}G & L_{N,M}\\ L_{N,M}^\top & M\end{pmatrix}\mathbf{x} = \left\|\sum_{i=1}^{N} x_i\varphi_i + \sum_{j=N+1}^{M+N} x_j v_j(\langle \mathbf{a}_j, \cdot\rangle + b_j)\right\|^2 \geq 0.$$

Daher ergeben sich die optimalen Koeffizienten wie behauptet. □

3.2 Der Particle-Swarm-Algorithmus

Nun beschäftigen wir uns mit der Bestimmung der optimalen Richtungen zu einer gegebenen Profilfunktion. Das zum 1. Typ gehörige Optimierungsproblem

$$\min!_{a\in\mathbb{R}^d, c\in\mathbb{R}} \ \|u(\cdot) - cv(\langle a, \cdot\rangle + b_a)\|_2 = \min!_{y=(\alpha,b,c,a)} J_u(y)$$

zu einer gegebener Profilfunktion v ist bereits in einfachen Fällen ein höchst nicht-lineares, nicht-konvexes, schwieriges Problem, dessen numerische Behandlung uns vor große Aufgaben stellt. Wir möchten das Problem weiterhin für mehrere Profilfunktionen lösen, d.h. für gegebene Profilfunktionen $v_1, \ldots, v_M$

$$\min!_{\substack{\mathbf{a}_i\in\mathbb{R}^d, i=1,\ldots,M\\ c\in\mathbb{R}^M}} \left\|u(\mathbf{x}) - \sum_{i=1}^{M} c_i v_i(\langle \mathbf{a}_i, \mathbf{x}\rangle + b_{\mathbf{a}_i})\right\|_2. \tag{3.6}$$

In Grafik 3.1 sehen wir in einigen einfachen Beispielen für $M = 2$ Dimensionen das Verhalten des Fehlers. Dafür haben wir via der Standardtransformation

$$z \mapsto \frac{z}{1-|z|}, \quad z \in (-1, 1)$$

das Intervall $(-1, 1)$ mit $\mathbb{R}$ identifiziert und dadurch die Richtungen $\mathbf{a}_i \in \mathbb{R}^2$ durch sogenannte „Partikel" $\mathbf{p}_i$ in $(-1, 1)^2$ ersetzt. Zudem haben wir die erste Komponente aller Richtungen gleich eins gesetzt. Die benutzten Funktionen und Richtungen sind:

	$u(x_1, x_2)$	$v_1(z)$	$v_2(z)$
1.	$1.6\cos(10x_1 + \frac{10}{3}x_2) + 0.8\cos(10x_1 - 5x_2)$	$\cos(10z)$	$\cos(10z)$
2.	$\cos(\frac{1}{2}x_1 + x_2) + \cos(\frac{1}{3}x_1 - \frac{1}{6}x_2)$	$\cos(\frac{1}{2}z)$	$\cos(\frac{1}{3}z)$
3.	$2\lvert x_1 - 3x_2\rvert + 0.1(x_1 - 5x_2 + 1)^2$	$\lvert z\rvert$	$(z+1)^2$

Tabelle 3.1: Drei Beispiele für das Verhalten des Fehlers auf 3.6

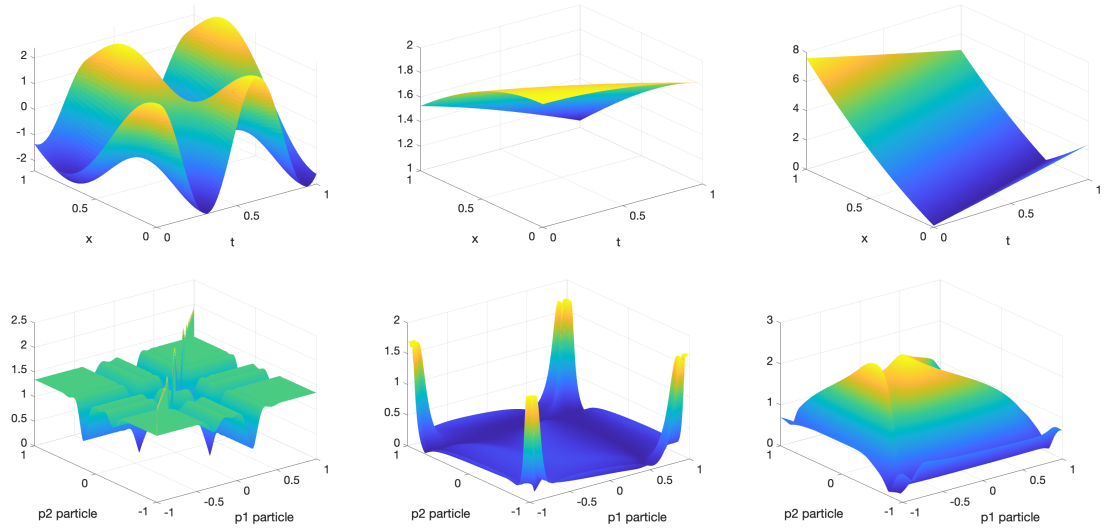

Abbildung 3.1: Drei Beispiele für das Verhalten des Fehler: Obere Zeile die Zielfunktionen, untere Zeile die Fehler in Abhängigkeit der Partikel

Wir sehen in den Beispielen in Grafik 3.1 mehrere lokale (sogar globale) Minima, sowie Plateaus wie auch steile Gradienten. Daher sind Standard-Abstiegsverfahren nicht in der Lage diese Probleme zu lösen. Wir schlagen daher einen sogenannten Particle-swarm-Algorithmus zur Lösung dieses Problems vor. Dieser ist wie folgt aufgebaut: Sei zunächst $\tilde{a} : S^d \to \mathbb{R}^d$ eine Abbildung zwischen einem beschränkten Gebiet, hier $S^d = (-1, 1)^d$, aus dem die Partikel sind und dem Suchgebiet der Richtungen, hier $\mathbb{R}^d$. Dies kann die oben erwähnte Abbildung $z \mapsto \frac{z}{1-|z|}$ punktweise angewandt auf eine Richtung oder auch $z \mapsto \tan(\frac{\pi}{2}z)$ sein.[1]

[1] Die Wahl einer bestimmten Abbildung kann abhängig vom Problem sein, für die Transport- und Wellengleichung mit mittelgroßen Richtungen lieferte $z \mapsto \tan(\frac{\pi}{2}z)$ die besten Ergebnisse. Falls a priori das Suchgebiet für die Richtungen beschränkt werden kann, sollte auch diese Abbildung auf dieses Gebiet zentriert werden.

Wir wählen eine Anzahl an Partikeln $n_{pro} \in \mathbb{N}$ pro Dimension d des Suchraums und initialisieren zunächst ein äquidistantes Gitter $P^{(0)} \subset S^d$. Dieses enthält $m_{par} = n_{par}^d$ Partikel $p = (p_1, \ldots, p_d) \in S^d$.
Nun verändern wir in jedem Schritt dieses Gitter der Partikel so, dass diese sich dem globalen Minimum annähern. Sei dazu

$$P^{(k)} := \{p_{\mathbf{i}}^{(k)} \in S^d \mid \mathbf{i} = (i_1, \ldots, i_d) \in \{1, \ldots, n_{par}\}^d\}$$

das Gitter im k-ten Schritt. Jeder Partikel $p_{\mathbf{i}}^{(k)} \in P^{(k)}$ besitzt $3^d - 1$ benachbarte Partikel $p_{\mathbf{j}}^{(k)}$, d.h. solche mit $\mathbf{i} \neq \mathbf{j}$ und $|i_s - j_s| \leq 1$ für alle $1 \leq s \leq d$. Sei $\mathcal{I}(\mathbf{i})$ die Menge der Indizes der Nachbarn. Dann bestimmen wir für jeden Partikel den Nachbarn mit dem besten Funktionswert als

$$q_{\mathbf{i}}^{(k)} := \arg\min_{\mathbf{j} \in \mathcal{I}(\mathbf{i})} J(\tilde{a}(p_{\mathbf{j}}^{(k)}))$$

wobei wir $\check{J}(p) := J(\tilde{a}(p))$ definieren.
Danach verändern wir jeden Partikel $p_{\mathbf{i}}^{(k)} \in P^{(k)}$ in Abhängigkeit des gewählten Parameters $\gamma \in (0, \frac{1}{2})$ gemäß

$$p_{\mathbf{i}}^{(k+1)} = (1 - \gamma) p_{\mathbf{i}}^{(k)} + \gamma q_{\mathbf{i}}^{(k)}. \tag{3.7}$$

Für die Punkte am Rand des Gitters definieren wir die Nachbarn als die gegenüber auf dem Gitter liegenden Punkte, d.h. topologisch gesprochen wird unser Gitter zu einem Torus. So ist für jedes $1 \leq s \leq d$ das Partikel $p_{\mathbf{j}}^{(k)}, \mathbf{j} = (j_1, \ldots, j_{s-1}, 1, j_{s+1}, \ldots, j_d)$, ein Nachbar von $p_{\mathbf{j}'}^{(k)}$ mit $\mathbf{j}' = (j_1, \ldots, j_{s-1}, n_{par}, j_{s+1}, \ldots, j_d)$ und umgekehrt. Dies ermöglicht es den Partikeln sich auch in Richtung des Randes $\partial(-1, 1)^d$ zu bewegen, die Veränderung der Partikel erfolgt analog auch über den Rand.
Damit erhalten wir das neue Gitter $P^{(k+1)}$ und wiederholen die Schritte von oben.
Nach K vorgegebenen Iterationen beenden wir den Algorithmus und bestimmen aus dem letzten $P^{(J)}$ die optimale Richtung als $a^* = \tilde{a}(p^*)$ und

$$p^* = \arg\min_{p \in P^{(K)}} \check{J}(p). \tag{3.8}$$

Wir fassen nun die Methode in dem folgenden Algorithmus zusammen:

Input : Anzahl der Partikel m_{par}, Anzahl Iterationen K, Schrittweite $\gamma \in (0, \frac{1}{2})$
Output : Optimaler Partikel p^*
Initialisiere das Gitter der Partikel $P^{(0)}$ beliebig
for $j = 0 \dots J-1$ **do**
  Setze neues $P^{(k+1)}$
  for $i \in \{1, \dots, m_{par}\}$ **do**
    Bestimme $q_i^{(k)} = \arg\min_{\mathbf{j} \in \mathcal{I}(\mathbf{i})} \check{J}(p_{\mathbf{j}}^{(k)})$
    Berechne $p_i^{(k+1)}$ nach (3.7)
    Berechne $\check{J}_u(p_i^{(k+1)})$
  end
  Bestimme $p^* = \arg\min_{p \in P^{(k)}} \check{J}(p)$ wie in 3.8
end

Algorithmus 2 : Der Particle-Swarm-Algorithmus

Bemerkung 3.9 Algorithmus 2 erhält die Gitterstruktur der Partikel, d.h. die Indexmenge $I(\mathbf{i})$ ist invariant unter den Iterationen.

Beweis: Da $\gamma \in (0, \frac{1}{2})$ ist, gilt für benachbarte Partikel p_i, p_j, dass $\|p_i^{k+1} - p_j^{k+1}\| = \|(1-\gamma)p_i^k + \gamma p_j^k - (1-\gamma)p_j^k - \gamma p_i^k\| = (1-2\gamma)\|p_i^k - p_j^k\| > 0$ und da das Anfangsgitter $P^{(0)}$ äquidistant initialisiert wurde folgt die Behauptung. □

Zur Visualisierung des Algorithmus zeigt Grafik 3.2 und 3.3 einige Particle Grids. Dabei sieht man, dass sich die Punkte schnell an lokalen Minima häufen und sich später auf das globale Minimum und Linien, die zu diesem hinlaufen, konzentrieren. In der zweiten Grafik sind zur Veranschaulichung der Minima die Höhenlinien der Kostenfunktion im Hintergrund mit eingezeichnet. Im Falle von mehreren globalen Minima bewegen sich die Partikel zu diesen hin, später aber weiterhin zwischen diesen bewegen wie in Grafik 3.3 zu sehen.

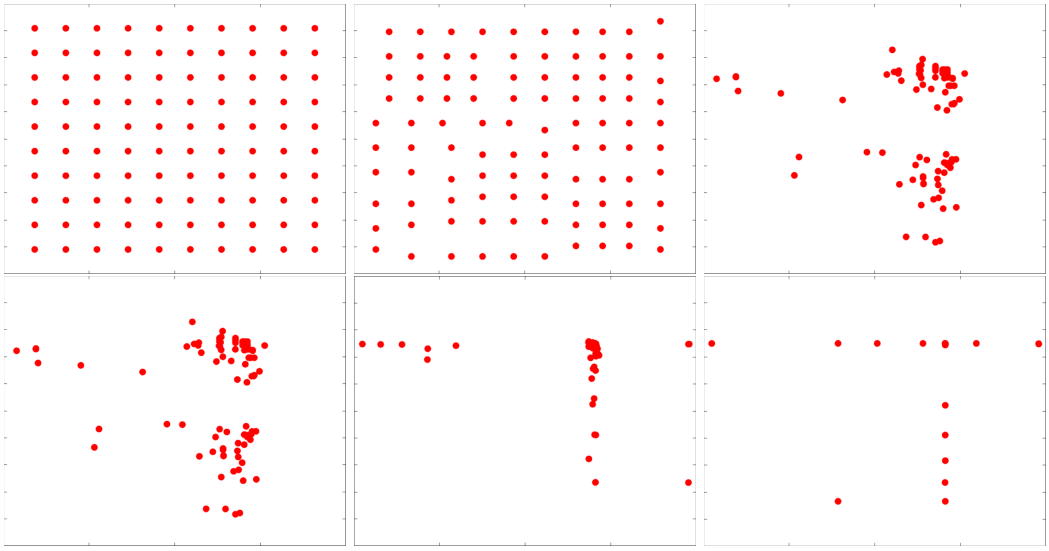

Abbildung 3.2: Mehrere Gitter $P^{(k)}$ aus dem Algorithmus für ein einzelnen Minimum, $d = 2, m_{par} = 100$, Iterationen $k = 0, k = 1, k = 5, k = 25, k = 56, k = 90$.

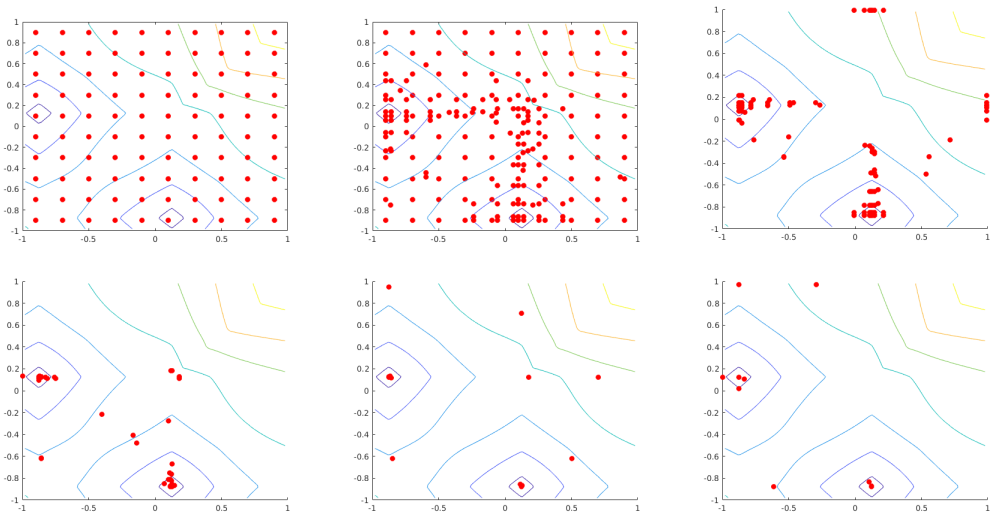

Abbildung 3.3: Visualisierung des Algorithmus bei mehreren globalen Minima, Iterationen $k = 1, k = 5, k = 10, k = 20, k = 50, k = 90$. $(d = 2, n = 129, m_{par} = 100)$

Quantitativ lässt sich anhand von 3.4 sehen, dass weit von einem Minimum entfernte Punkte zunächst mit konstanter Geschwindigkeit gegen dieses konvergieren. Eine genauere Aussage zur Konvergenz liefert Lemma 3.10.

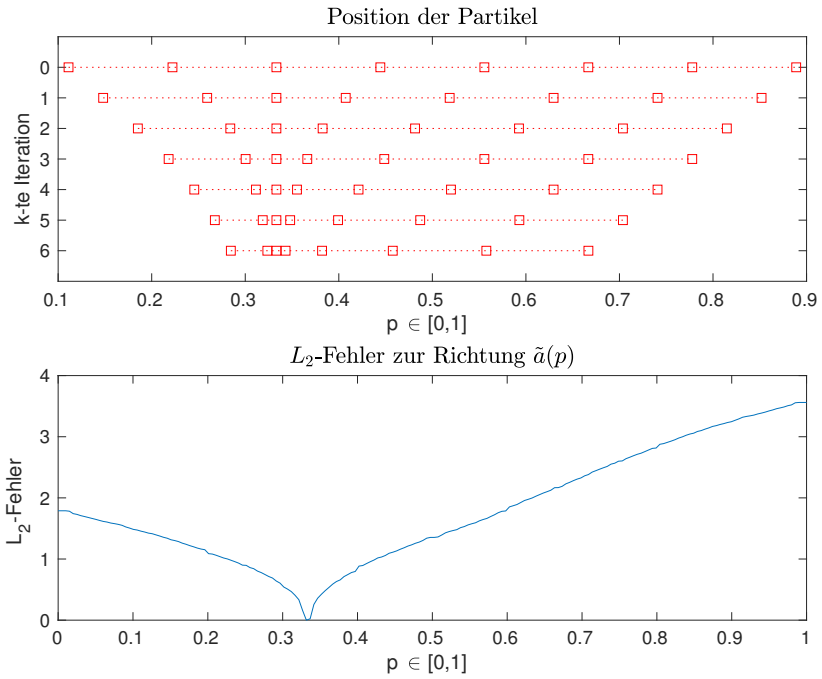

Abbildung 3.4: Mehrere Gitter $P^{(k)}$ für $d = 1$, dazu abgebildet die Fehlerfunktion $\check{J}$ in Abhängigkeit von $p \in (0, 1)$

Der Algorithmus zeigt wie wir später bei den numerischen Experimenten sehen werden gute Konvergenzeigenschaften für die betrachteten Probleme. Zunächst lässt sich zumindest Konvergenz für konvexe Funktionen zeigen.

Lemma 3.10 Sei J eine konvexe Funktion auf $\mathbb{R}^d$ mit J nicht konstant auf der Menge $\tilde{a}(P^{(0)})$. Dann konvergiert die Folge der p^* aus Algorithmus 2 gegen das Minimum auf diesem Intervall.

Beweis: Sei $x^* = \arg\min J(x)$ das Minimum der Funktion. Wir nehmen an, dass $x^* \notin P^{(k)}$ für alle k, ansonsten ist nichts zu zeigen.
Sei $p_k^* \in P^{(k)}$ der Partikel im k-ten Schritt mit dem minimalen Funktionswert. Seien $p_j \in P_I^{(k)}$ die benachbarten Partikel von p_k^*. Da die Funktion J konvex ist, liegt das Minimum x^* in der konvexen Hülle $\mathrm{conv}(P_I^{(k)})$. Damit ist für mindestens ein $p_j \in P_I^{(k)}$ die Konvexkombination $J((1-\gamma)p_k^* + \gamma p_j) < J(p_k^*)$. Und damit ist $\|x^* - p_{k+1}^*\|_2 \leq \|x^* - p_k^*\|_2 - \gamma \frac{\sqrt{d}}{\max_j \|p_k^* - p_j\|_2}$ und der Algorithmus konvergiert gegen das Minimum. □

Wir können weiterhin die Komplexität des Algorithmus angeben

Lemma 3.11 Sei K die Zahl der Iterationen, m die Anzahl der Partikel, und die Kosten zur Auswertung der Funktion J seien $\mathcal{O}(n^d)$. Dann ist die Komplexität des Algorithmus 2

in der Ordnung von

$$\mathcal{O}(K \cdot m \cdot n^d).$$

Bemerkung 3.12 Der Algorithmus 2 kann leicht parallelisiert werden, da die Position der neuen Partikel $p^{(k+1)}$ ausschließlich von dem vorherigen Gitter $P^{(k)}$ abhängen und dort sogar nur von den direkt benachbarten Partikeln. Somit kann eine Parallelisierung der Berechnung der einzelnen $p^{(k+1)}$ parallel ohne Kommunikation zwischen den Prozessen erfolgen.

Bemerkung 3.13 Es sind viele weitere Erweiterungen des Algorithmus denkbar:

- Varianten des Updates der Partikel: Es ist möglich die Partikel, ähnlich wie in anderen Heuristiken mit einer gewissen Trägheit (Momentum) auszustatten. Speichert man zusätzlich zu jedem Partikel $p^{(k)} \in P^{(k)}$ in Schritt k seine Bewegung im Schritt zuvor $d^{(k)} := p^{(k)} - p^{(k-1)}$, so lässt sich für $0 < \alpha < 1$ mit $p^{(k+1)} = p^{(k)} + \alpha d^{(k)} + (1 - \delta)p_{\mathbf{i}}^{(k)} + \delta q_{\mathbf{i}}^{(k)}$ ein Update mit Trägheit definieren. Für die meisten Probleme bringt diese Variante keine größeren Vorteile bzgl. Geschwindigkeit der Konvergenz.

- Verbesserung der Geschwindigkeit durch einfachere Gitter und Nachbarschaftsstrukturen: In dem oben vorgestellten Gitter hat jedes Partikel $3^d - 1$ Nachbarn. Es ist möglich, diese Nachbarschaftsbeziehungen auszudünnen, indem die Diagonalen nicht mehr als Nachbarn in Betracht gezogen werden. Dies führt zu einer Verringerung der Anzahl an Nachbarn auf 2^d. Zu Beginn des Algorithmus führt dies insbesondere bei komplexen Problemen zu einer deutlich langsameren Konvergenz. Zu einem späteren Zeitpunkt wenn sich Partikel bereits um Minima gruppiert haben, ist dies jedoch eine einfache Möglichkeit Rechenzeit zu sparen. Ebenso können nach einer gewissen Zahl von Iterationen die schlechtesten Partikel aussortiert werden. Dies verändert jedoch auch stark die Nachbarschaftsbeziehungen in dem Gitter. Die Behandlung dieser Probleme kann wie in [59] beschrieben durchgeführt werden.

- Zufällige Bewegung der Partikel: Ähnlich wie in einem simulated annealing- oder STUN-Algorithmus kann eine festgelegte Anzahl von Partikel pro Schritt in eine zufällige Richtung mit zufälliger Schrittweite verändert werden um bei besonders schwierigen Problem mit kleinen Minima dem Algorithmus mehr Flexibilität zum Finden dieser zu geben.

3.3 Bestimmung von Profilfunktionen

3.3.1 Profilfunktion zu gegebener Richtung

Wir beschäftigen uns nun mit dem Problem vom 2. Typ. Für $M = 1$ ist zu gegebener Richtung a die optimale Profilfunktion durch folgendes Lemma gegeben:

Lemma 3.14 Sei $a \in S^{d-1}$ und $u \in L_2([0,1]^d)$. Sei $v_a \in L_2([0,1])$ definiert als

$$v_a(y) := \frac{1}{\lambda(I_a(y))} \int_{s \in I_a(y)} u(s)\, ds, \quad y \in [0,1], \tag{3.9}$$

wobei $\lambda(I_a(y))$ das $(d-1)$-dimensionale Lebesguemaß der Menge

$$I_a(y) = \{s \in [0,1]^d \mid \langle s, a\rangle + b_a = y\}$$

ist. Dann ist

$$v_a = \arg \min_{v \in L_2([0,1])} \|u - v(\langle \cdot, a\rangle + b_a)\|_{L_2([0,1]^d)}.$$

Beweis: Zunächst bemerken wir, dass $v_a \in L_2([0,1])$, da I_a beschränkt und $u \in L_2([0,1]^d)$. Durch die Charakterisierung der besten L_2-Approximation genügt es zu zeigen, dass

$$\int_{[0,1]^d} (u(x) - v_a(\langle a, x\rangle + b_a)) w(\langle a, x\rangle + b_a)\, dx = 0 \tag{3.10}$$

für alle $w \in L_2([0,1])$. Durch die festgelegte Richtung a ist $w(\langle x, a\rangle + b_a)$ konstant für alle $x \in I_a(y)$, also $w(\langle x, a\rangle + b_a) = w(y)$.
Weiter ist $\bigcup_{y \in [0,1]} I_a(y) = [0,1]^d$, also müssen wir nur zeigen, dass

$$\int_{[0,1]} \int_{s \in I_a(y)} (u(s) - v_a(y)) w(y)\, ds dy = 0.$$

Da w beliebig aus $L_2([0,1])$ gewählt ist, bedeutet dies

$$\int_{s \in I_a(y)} (u(s) - v_a(y)) w(y)\, ds = 0 \quad \text{f.ü.}$$

und somit

$$\int_{s \in I_a(y)} u(s)\, ds = \mu(I_a(y)) v_a(y) \quad \text{f.ü.},$$

was uns eine (bis auf Identifizierung in $L_2([0,1])$) eindeutige Lösung v_a in der behaupteten Form gibt. □

Wir haben dies numerisch implementiert, indem wir für $n_1 = n^2$ Punkte $0 = y_1 < y_2 < \cdots < y_{n_1 - 1} < y_{n_1} = 1$ jeweils die Länge von $I_a(y_i)$ bestimmt haben und das dazugehörige Integral durch eine einfache Quadraturformel genähert haben. Dabei haben wir folgende Ergebnisse erhalten:

Beispiel 3.15 Wir testen zunächst mit einer Quadraturformel nach Newton-Cotes vom Grad 1 (Mittelpunktregel) mit 10 Punkten. Wir erhalten einen punktweisen Fehler in der Größenordnung 10^{-8}, was unter Berücksichtigung der relativ groben Diskretisierung und einfachen Quadraturmethode sehr gut ist.

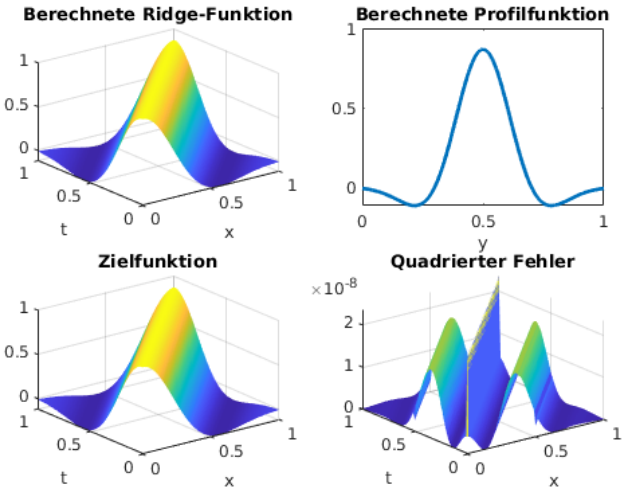

Abbildung 3.5: Rekonstruktion von v als Mexican-hat Wavelet für $n = 129$

Beispiel 3.16 Dies funktioniert ebenfalls für gestörte Zielfunktionen. Dafür haben wir erneut das Mexican-Hat Wavelet aus dem vorherigen Beispiel genommen und zu diesem eine normalverteilte Variable mit $\mu = 0, \sigma = 0.1$ addiert. Das Ergebnis bleibt trotzdem gut mit einem Fehler nicht größer als der durch die zufällige Variable hinzugefügte.

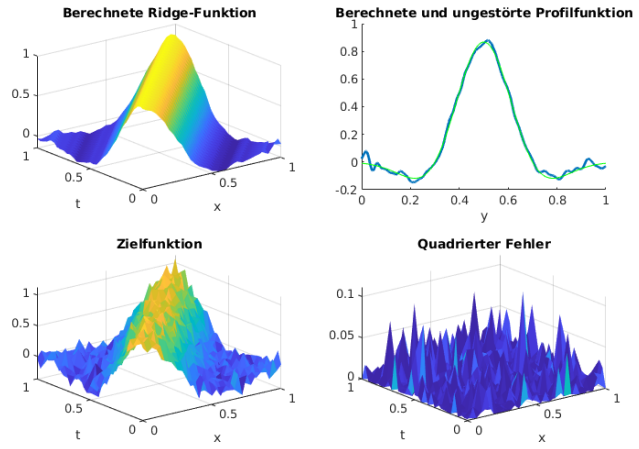

Abbildung 3.6: Rekonstruktion des Mexican-hat Wavelets aus gestörter Zielfunktion mit $n = 129$

Der Diliberto-Straus-Algorithmus

Zur Berechnung zweier optimaler Funktionen in $\mathcal{C}(K)$ für $K \subset \mathbb{R}^d$ kompakt und konvex gibt es den Ansatz eines alternierenden Algorithmus.
Der Algorithmus wurde zuerst von Diliberto, Straus [22], [50] ausgeführt und für Ridge-Funktionen in [64],[66] ausgearbeitet.

Sei dazu $f \in \mathcal{C}(K)$ sowie Richtungen $\mathbf{a}^1, \mathbf{a}^2 \in \mathbb{R}^d$ linear unabhängig gegeben. Wir suchen nun eine Bestapproximation f^* aus dem Raum $\mathcal{R}(\mathbf{a}^1, \mathbf{a}^2) = \{v_1(\langle \mathbf{a}^1, \cdot \rangle) + v_2(\langle \mathbf{a}^2, \cdot \rangle) | v_1, v_2 \in \mathcal{C}(\mathbb{R})\}$. Zunächst ignorieren wir die zweite Richtung und betrachten die Bestapproximation in $\mathcal{C}(K)$ mit einer Ridge-Funktion zur Richtung $\mathbf{a}^1$. Wir wissen, dass Ridge-Funktionen orthogonal zu ihrer Richtung konstant sind und setzen daher $f^*(\mathbf{y})$ als die beste konstante Approximation an f auf der Hyperebene $I_1(\mathbf{y}) = \{\mathbf{x} \in K \mid \langle \mathbf{a}^1, \mathbf{x} \rangle = \langle \mathbf{a}^1, \mathbf{y} \rangle\}$. Nach der Charakterisierung der Bestapproximation in $\mathcal{C}(K)$ ist dies gleich

$$f^*(\mathbf{y}) = B_1 f(\mathbf{y}) = \frac{\max_{\mathbf{x} \in I_1(\mathbf{y})} f(\mathbf{x}) + \min_{\mathbf{x} \in I_1(\mathbf{y})} f(\mathbf{x})}{2}. \tag{3.11}$$

Analog ist für $\mathbf{a}^2$ und die Hyperebene $I_2(\mathbf{y}) = \{\mathbf{x} \in K \mid \langle \mathbf{a}^2, \mathbf{x} \rangle = \langle \mathbf{a}^2, \mathbf{y} \rangle\}$ die Bestapproximation

$$f^*(\mathbf{y}) = B_2 f(\mathbf{y}) = \frac{\max_{\mathbf{x} \in I_2(\mathbf{y})} f(\mathbf{x}) + \min_{\mathbf{x} \in I_2(\mathbf{y})} f(\mathbf{x})}{2}. \tag{3.12}$$

Nun geht der Algorithmus wie folgt vor:

Input : Zielfunktion $f \in \mathcal{C}(K)$, Richtungen $\mathbf{a}^1, \mathbf{a}^2$, Anzahl Iterationen N
Output : Optimale Profilfunktionen $v_1^N, v_2^N \in \mathcal{C}(\mathbb{R})$
Setze $n = 0$, $f^0 := f$;
while $n \leq N$ **do**
  Berechne $f^{n+1} = f^n - B_1 f^n$ mit $v_1^n = B_1 f^n$ gemäß 3.11;
  Berechne $f^{n+2} = f^{n+1} - B_2 f^{n+1}$ mit $v_2^n = B_2 f^{n+1}$ gemäß 3.12 ;
  Setze $n \leftarrow n + 2$;
end

Algorithmus 3 : Der Diliberto-Straus-Algorithmus

Diliberto und Straus konnten in [22], [66, Theorem 9.24] zeigen, dass dieser Algorithmus konvergiert, d.h. dass

$$\lim_{n \to \infty} \|f - v_1^n - v_2^n\|_\infty = \|f - f^*\|_\infty = \inf_{g \in \mathcal{R}(\mathbf{a}^1, \mathbf{a}^2)} \|f - g\|_\infty.$$

Wir möchten im Weiteren jedoch mehr als nur zwei Richtungen und andere Normen als die oben benutzen und verfolgen einen anderen Ansatz.
Für das bis jetzt betrachtete Gebiet $\Omega = [0,1]^d \subset \mathbb{R}^d$ ergeben sich numerische Probleme, wenn $\int_{s \in I_a(y)} u(s)\, ds$ für y nahe dem Rand des Definitionsbereichs von v_a ausgewertet werden soll, da $I_a(y)$ beliebig klein werden kann. Wir wollen daher optimale Profilfunktionen auf einem einfacheren Bereich bestimmen. Es ist nicht sinnvoll optimale Profilfunktionen bzgl. der L_2-Norm auf ganz $\mathbb{R}^d$ zu betrachten, da Ridge-Funktionen i.A. nicht in $L_2(\mathbb{R}^d)$ liegen, wir können aber $\|\cdot\|_\infty$ und/oder andere Bereiche wie $\mathcal{B} = \{\|x\|_2 \leq 1\}$ betrachten. Auf diesen Gebieten ist ein einfacher Ansatz möglich: Seien d linear unabhängige Richtungen $A = \{\mathbf{a}_1, \ldots, \mathbf{a}_d\} \subset \mathbb{R}^d$ gegeben. Wir normalisieren diese bzgl. der euklidischen Norm, sodass $\|\mathbf{a}_i\|_2 = 1$ für $i = 1, \ldots, d$. Diese Menge der Richtungen A bildet eine Basis des $\mathbb{R}^d$, wähle dann $\mathbf{b}_j = \mathbf{a}_j^*$, $j = 1, \ldots, d$, als die zu A duale Basis. D.h. $\langle \mathbf{a}_i, \mathbf{b}_j \rangle = \delta_{ij}$. Wir wählen

dann die Profilfunktion $\tilde{v}_j(\xi) := u(\xi \mathbf{b}_j)$. Dies liefert uns für den Fall, dass u selbst eine Summe von Ridge Funktionen

$$u(x) = \sum_{i=1}^{d} v_i(\langle \mathbf{a}_i, x \rangle)$$

ist eine exakte Rekonstruktion, denn

$$\begin{aligned} u(\xi \mathbf{b}_j) &= \sum_{i=1}^{d} v_i(\langle \mathbf{a}_i, \xi \mathbf{b}_j \rangle) \\ &= \sum_{i=1}^{d} v_i(\xi \langle \mathbf{a}_i, \mathbf{b}_j \rangle) \\ &= v_j(\xi). \end{aligned}$$

Die Darstellung ist nach [66, Corollary 3.2] eindeutig bis auf eine Konstante. Wir können durch Normierung zu $\tilde{v}_j(0) = 0$ für alle j die Darstellung eindeutig machen, wenn wir auch u entsprechend normalisieren.
Haben wir gestörte Richtungen $\tilde{\mathbf{a}}_1, \dots \tilde{\mathbf{a}}_d$ gegeben, so können nach dem folgenden Lemma den dadurch entstehenden Fehler abschätzen.

Lemma 3.17 (Fornasier [24]) Seien Basen $\mathbf{a}_1, \dots, \mathbf{a}_d \in \mathbb{R}^d$ und $\tilde{\mathbf{a}}_1, \dots \tilde{\mathbf{a}}_d \in \mathbb{R}^d$ gegeben mit

$$\sum_{i=1}^{d} \|\mathbf{a}_i - \tilde{\mathbf{a}}_i\|_2^2 \leq \varepsilon \leq 1,$$

dann gilt für die Approximation $\tilde{f} = \sum_{j=1}^{d} v_j(\langle \tilde{\mathbf{a}}_j, \cdot \rangle)$ an $f = \sum_{j=1}^{d} v_j(\langle \mathbf{a}_j, \cdot \rangle)$ mit Profilfunktionen $v_j \in \mathcal{C}^2(\mathbb{R})$

$$\|f - \tilde{f}\|_\infty^2 \leq C\varepsilon \tag{3.13}$$

mit einer Konstante C in Abhängigkeit von den Richtungen.

Beweis: Sei $x \in \mathbb{R}^d$ mit $\|x\|_2 \leq 1$. Wir stellen zuerst mit einfacher linearer Algebra die exakten Richtungen durch die gestörten Richtungen dar. Es sei $\{\mathbf{b}_1, \dots, \mathbf{b}_d\}$ die duale Basis zu $\{\mathbf{a}_1, \dots, \mathbf{a}_d\}$ und $\{\tilde{\mathbf{b}}_1, \dots, \tilde{\mathbf{b}}_d\}$ die duale Basis zu $\{\tilde{\mathbf{a}}_1, \dots, \tilde{\mathbf{a}}_d\}$. Dann gilt

$$\langle \mathbf{a}_i, x \rangle = \left\langle \sum_{j=1}^{d} \langle \mathbf{a}_i, \tilde{\mathbf{b}}_j \rangle \tilde{\mathbf{b}}_j, x \right\rangle = \sum_{j=1}^{d} \langle \tilde{\mathbf{b}}_j, x \rangle \langle \mathbf{a}_i, \tilde{\mathbf{b}}_j \rangle = \sum_{j=1}^{d} \langle \tilde{\mathbf{a}}_j, x \rangle \langle \tilde{\mathbf{b}}_j, \mathbf{a}_i \rangle,$$

da $\sum_{k=1}^{d} \langle \tilde{\mathbf{b}}_j, \tilde{\mathbf{a}}_k \rangle \tilde{\mathbf{a}}_k = \sum_{k=1}^{d} \delta_{jk} \tilde{\mathbf{a}}_k = \tilde{\mathbf{a}}_j$.

Dies setzen wir ein und erhalten

$$\begin{aligned} |f(x) - \tilde{f}(x)| &= \left| \sum_{j=1}^{d} v_j(\langle \mathbf{a}_j, x\rangle) - \sum_{i=1}^{d} \tilde{v}_i(\langle \tilde{\mathbf{a}}_i, x\rangle) \right| \\ &= \left| \sum_{j=1}^{d} v_j(\langle \mathbf{a}_j, x\rangle) - \sum_{i=1}^{d} f(\langle \tilde{\mathbf{a}}_i, x\rangle \tilde{\mathbf{b}}_i) \right| \\ (*) \qquad &\leq \sum_{j=1}^{d} \left| v_j(\langle \mathbf{a}_j, x\rangle) - \sum_{i=1}^{d} v_i(\langle \mathbf{a}_i, \tilde{\mathbf{b}}_j\rangle \langle \tilde{\mathbf{a}}_j, x\rangle) \right| \end{aligned}$$

Wir entwickeln die Terme mit Taylor um 0 und erhalten mit $y := \langle \mathbf{a}_j, x\rangle$ und der Restglieddarstellung

$$v_j(\langle \mathbf{a}_j, x\rangle) = v_j(0) + v_j'(0)y + \int_0^y \frac{y-\xi}{1!} v_j''(\xi)\, d\xi$$

und analog mit $\tilde{y} := \langle \mathbf{a}_i, \tilde{\mathbf{b}}_j\rangle \langle \tilde{\mathbf{a}}_j, x\rangle$

$$\sum_{i=1}^{d} v_i(\langle \mathbf{a}_i, \tilde{\mathbf{b}}_j\rangle \langle \tilde{\mathbf{a}}_j, x\rangle) = \sum_{i=1}^{d} v_i'(0)\tilde{y} + \sum_{i=1}^{d} \int_0^{\tilde{y}} (\tilde{y} - \xi) v_i''(\xi)\, d\xi$$

Wir betrachten nun einzeln die Terme $\int_0^y (y-\xi)v_j''(\xi)\, d\xi - \int_0^{\tilde{y}} (\tilde{y}-\xi)y_j''(\xi)\, d\xi$ und schätzen diese ab.

$$\begin{aligned} \left| \int_0^y (y-\xi)v_j''(\xi)\, d\xi - \int_0^{\tilde{y}} (\tilde{y}-\xi)v_j''(\xi)\, d\xi \right| &= \left| \int_0^y (y-\tilde{y})v_j''(\xi)\, d\xi + \int_y^{\tilde{y}} (\tilde{y}-\xi)v_j''(\xi)\, d\xi \right| \\ &\leq \|v_j''\|_\infty \left(|y||y-\tilde{y}| + \frac{1}{2}|y-\tilde{y}|^2 \right) \end{aligned}$$

Wir setzen wieder die Definition von $y, \tilde{y}$ ein und erhalten mit der Cauchy-Schwarz-Ungleichung

$$\begin{aligned} \sum_{i=1}^{d} \left(|y||y-\tilde{y}| + \frac{1}{2}|y-\tilde{y}|^2 \right) &\leq \left(\sum_{i=1}^{d} |\langle \mathbf{a}_i, x\rangle|^2 \right)^{1/2} \left(\sum_{i=1}^{d} |\langle \mathbf{a}_i, x\rangle - \langle \tilde{\mathbf{a}}_i, x\rangle \langle \tilde{\mathbf{b}}_i, \mathbf{a}_i\rangle|^2 \right)^{1/2} \\ &\quad + \frac{1}{2} \sum_{i=1}^{d} |\langle \mathbf{a}_i, x\rangle - \langle \tilde{\mathbf{a}}_i, x\rangle \langle \tilde{\mathbf{b}}_i, \mathbf{a}_i\rangle|^2 \\ &\leq \left(\sum_{i=1}^{d} \|\mathbf{a}_i\|_2^2 \|x\|_2^2 \right)^{1/2} \left(\sum_{i=1}^{d} |\langle \mathbf{a}_i, x\rangle - \langle \tilde{\mathbf{a}}_i, x\rangle \langle \tilde{\mathbf{b}}_i, \mathbf{a}_i\rangle|^2 \right)^{1/2} \\ &\quad + \frac{1}{2} \sum_{i=1}^{d} |\langle \mathbf{a}_i - \tilde{\mathbf{a}}_i, x\rangle + \langle \tilde{\mathbf{a}}_i, x\rangle + \langle \tilde{\mathbf{a}}_i, x\rangle \langle \tilde{\mathbf{b}}_i, \tilde{\mathbf{a}}_i - \mathbf{a}_i\rangle - \langle \tilde{\mathbf{a}}_i, x\rangle \langle \tilde{\mathbf{b}}_i, \tilde{\mathbf{a}}_i\rangle|^2 \\ &\leq \sqrt{d} \left(\sum_{i=1}^{d} |\langle \mathbf{a}_i - \tilde{\mathbf{a}}_i, x\rangle + \langle \tilde{\mathbf{a}}_i, x\rangle \langle \tilde{\mathbf{b}}_i, \tilde{\mathbf{a}}_i - \mathbf{a}_i\rangle|^2 \right)^{1/2} \\ &\quad + \frac{1}{2} \sum_{i=1}^{d} |\langle \mathbf{a}_i - \tilde{\mathbf{a}}_i, x\rangle + \langle \tilde{\mathbf{a}}_i, x\rangle \langle \tilde{\mathbf{b}}_i, \tilde{\mathbf{a}}_i - \mathbf{a}_i\rangle|^2, \end{aligned}$$

da $\langle \tilde{\mathbf{b}}_i, \tilde{\mathbf{a}}_i \rangle = 1$, die Richtungen normiert sind und $\|x\|_2 \leq 1$ ist. Wir wenden erneut die Cauchy-Schwarz-Ungleichung an und erhalten

$$\begin{aligned}
&\leq \sqrt{d} \left(\sum_{i=1}^{d} \|\mathbf{a}_i - \tilde{\mathbf{a}}_i\|_2^2 + \sum_{i=1}^{d} \|\mathbf{a}_i\|_2^2 \|\tilde{\mathbf{b}}_i\|_2^2 \|\tilde{\mathbf{a}}_i - \mathbf{a}_i\|_2^2 \right)^{1/2} \\
&+ \sum_{i=1}^{d} \|\mathbf{a}_i - \tilde{\mathbf{a}}_i\|_2^2 + \sum_{i=1}^{d} \|\mathbf{a}_i\|_2^2 \|\tilde{\mathbf{b}}_i\|_2^2 \|\tilde{\mathbf{a}}_i - \mathbf{a}_i\|_2^2 \\
&\leq \sqrt{\varepsilon(1 + \max_{i=1}^{d} \|\tilde{\mathbf{b}}_i\|_2^2)} + \varepsilon(1 + \max_{i=1}^{d} \|\tilde{\mathbf{b}}_i\|_2^2 \leq \tilde{C}\sqrt{\varepsilon}.
\end{aligned}$$

Wir setzen nun die Terme aus Gleichung $(*)$ zusammen und erhalten

$$\begin{aligned}
(*) &\leq \sum_{j=1}^{d} \left| \|y_j''\|_\infty \tilde{C} \max\{\sqrt{\varepsilon}, \varepsilon\} \right| + \sum_{j=1}^{d} \sum_{\substack{i=1 \\ i \neq j}}^{d} \left| \langle \tilde{\mathbf{a}}_j, x \rangle \langle \tilde{\mathbf{b}}_j, \mathbf{a}_i - \tilde{\mathbf{a}}_i \rangle \right|^2 \\
&\leq d \max_{j=1}^{d} \|y_j''\|_\infty \tilde{C} \max\{\sqrt{\varepsilon}, \varepsilon\} + d \max_{j=1}^{d} \|y_j''\|_\infty \tilde{C}\varepsilon \leq C\sqrt{\varepsilon},
\end{aligned}$$

mit Konstante C in Abhängigkeit von d, $\max_{j=1}^{d} \|y_j''\|_\infty$ und $\max_{i=1}^{d} \|\tilde{\mathbf{b}}_i\|_2^2$. □

3.3.2 Unbekannte Richtungen

Zu dem Approximationsproblem von Typ 3 mit unbekannter Richtung und unbekannter Profilfunktion kombinieren wir die Ansätze, die wir bisher vorgestellt haben zu einem Algorithmus.

Input : Anzahl der Partikel m_{par}, Zielfunktion u, Anzahl Iterationen K, Schrittweite $\gamma \in (0, \frac{1}{2})$
Output : Optimaler Partikel p^* und zugehörige optimale Profilfunktion v^*
Initialisiere das Gitter der Partikel $P^{(0)}$
for $k = 0 \dots K$ **do**
  Setze neues $P^{(k+1)}$
  for $i \in \{1, \dots, m_{par}\}$ **do**
    Bestimme v_i^* zu $p_i^{(k)}$ gemäß (3.9)
    Berechne Fehler $\check{J}(p_j^{(k)})$ mit $V = \{v_i^*\}$ für alle Nachbarpartikel $p_j^{(k)}, j \in \mathcal{I}(i)$
    Bestimme $q_i^{(k)} = \arg\min \check{J}(p_j^{(k)})$
    Berechne $p_i^{(k+1)}$ nach (3.7)
  end
  Bestimme $p^* = \arg\min_{p \in P^{(k)}} \check{J}(p)$ wie in (3.8) und $v^* = v_{p^*}$ nach (3.9)
end

Algorithmus 4 : Greedy-Algorithmus zur Lösung des Problems von Typ 3

Beispiel 3.18 Gegeben sei die Funktion $u(x) = \exp(\langle a, x \rangle) + \exp(\|x\|_2) \in L_2([0,1]^2)$ mit Richtung $a = (1/2, 1/2)^T$. Diese Funktion ist Summe einer Ridge-Funktion und einer

radialen Funktion, welche sich nicht gut durch Ridge-Funktionen approximieren lassen. Nun wenden wir Algorithmus 4 auf u an, um die optimale Richtung a und die Profilfunktion $v(x) = \exp(x)$ zu rekonstruieren, ohne dass diese durch die radiale Funktion gestört wird. Nach 10 Iterationen findet der Algorithmus, trotz einer groben Diskretisierung von u in nur $n = 33$ Punkte pro Dimension, die Richtung a bis auf einen Fehler von $2,14 \cdot 10^{-3}$, bei einem Gesamt-L_2-Fehler von $1,29 \cdot 10^{-1}$. Weitere Ergebnisse fassen wir in der folgenden Tabelle zusammen. Der minimal mögliche L_2-Fehler für die ersten drei Funktionen ist $1,28 \cdot 10^{-1}$.

Funktion u	n	K	Fehler $\|a - a^*\|_2$	$\|u(\cdot) - v^*(\langle a^*, \cdot \rangle)\|_{L_2([0,1]^2)}$
$\exp(\langle a, x\rangle) + \exp(\|x\|_2)$	33	10	$2,14 \cdot 10^{-3}$	$1,29 \cdot 10^{-1}$
$\exp(\langle a, x\rangle) + \exp(\|x\|_2)$	15	10	$2,14 \cdot 10^{-3}$	$1,38 \cdot 10^{-1}$
$\exp(\langle a, x\rangle)$	15	10	$2,14 \cdot 10^{-3}$	$2,81 \cdot 10^{-3}$
$\exp(\langle a, x\rangle) + \exp(\|x\|_2)$	33	20	$7,47 \cdot 10^{-4}$	$1,29 \cdot 10^{-1}$

Tabelle 3.2: Ergebnisse der Anwendung von Algorithmus 4 mit K Iterationen

Bemerkung 3.19 Der Rechenaufwand zur Bestimmung der optimalen Profilfunktion v^* hängt von der Feinheit der Diskretisierung des Integrals n_{int} und der Diskretisierung n_v der Variablen λ ab. Wir berechnen zur Verbesserung der Genauigkeit meist für eine Diskretisierung von $[0,1]^2$ in $n \times n$ Gitterpunkte n^2 Auswertungen von $v^*(\lambda)$. Damit ist sichergestellt, dass auch für große Richtungen a der Interpolationsfehler zwischen den Gitterpunkten klein bleibt. In diesem Fall ist der Aufwand zur Berechnung von v^* bei $\mathcal{O}(n^2 n_{int})$ in jedem Schritt des Algorithmus, also insgesamt $\mathcal{O}(mn^4 n_{int} J)$ für $d = 2$. Das Problem vom Typ 3 ist also das schwierigste der Vorgestellten.

Mehrere Funktionen

Für mehrere gesuchte Funktionen ist die Situation komplizierter. Es gibt u.U. keine (bis auf Identifikation in L_2) eindeutige Lösung mehr. Dies ist das Ergebnis von [66, Kapitel 3.2 und 4.1]:

Lemma 3.20 [66, Corollary 3.7] Sei $r \in \mathbb{N}$ und seien $v_i : \mathbb{R} \to \mathbb{R}$, $i = 1, \dots, r$ messbare Funktionen und $\mathbf{a}_i \in \mathbb{R}^d$ paarweise linear unabhängige Richtungen. Dann ist

$$\sum_{i=1}^{r} v_i(\langle \mathbf{a}_i, x \rangle) = 0$$

genau dann, wenn v_i ein Polynom vom Grad m_i ist, sodass für jedes homogene Polynom $q_i \in \left\{ \sum_{|k|=m_i} b_k x^k \mid b_k \in \mathbb{R} \right\}$ in d Veränderlichen gilt: Ist $q_i(\mathbf{a}_j) = 0$ für alle $j = 1, \dots, r, j \neq i$, dann ist auch $q_i(\mathbf{a}_i) = 0$.

Zudem ist es in der Praxis nicht ausreichend die optimale Profilfunktion zu den verschiedenen Richtungen nacheinander zu bestimmen wie wir in den folgenden Beispielen sehen:

Beispiel 3.21 Sei $v_1(x) = v_2(x) = \sin(2\pi x)$, $a_1 = (\frac{1}{2}, 1)^T, a_2 = (2, 1)^T$ und $u(x) = v_1(\langle a_1, x\rangle) + v_2(\langle a_2, x\rangle)$. Geben wir die exakten Richtungen vor, liefert 3.9 eine sehr genaue Approximation der Funktionen v_1 und v_2 in beliebiger Reihenfolge. Suchen wir jedoch $a^* = \arg\min_{a\in\mathbb{R}^2} \|v^*(\langle a, \cdot\rangle) - u(\cdot)\|_{L_2([0,1]^2)}$, so ist dies minimal für $a^* = (\frac{2}{3}, \frac{1}{3})^T$ und mit optimaler Profilfunktion v_1^* mit L_2-Fehler von 0.6007. Im nächsten Schritt wenden wir wieder den Algorithmus auf das Residuum $u - v_1^*(\langle a^*, \cdot\rangle)$ an. Dabei erhalten wir eine weitere optimale Richtung bezüglich dieser neuen Funktion und eine optimale Profilfunktion wie in Grafik 3.8 zu sehen. Obwohl die Zielfunktion ursprünglich als eine Summe von zwei Ridge-Funktionen definiert worden ist, gelingt es nicht diese beiden in zwei Schritten des Greedy-Algorithmus zu erhalten. Der Fehler ist nach 2 Schritten noch 0.2479 und fällt relativ langsam weiter zu 0.1347 im dritten und 0.0942 im vierten Schritt.

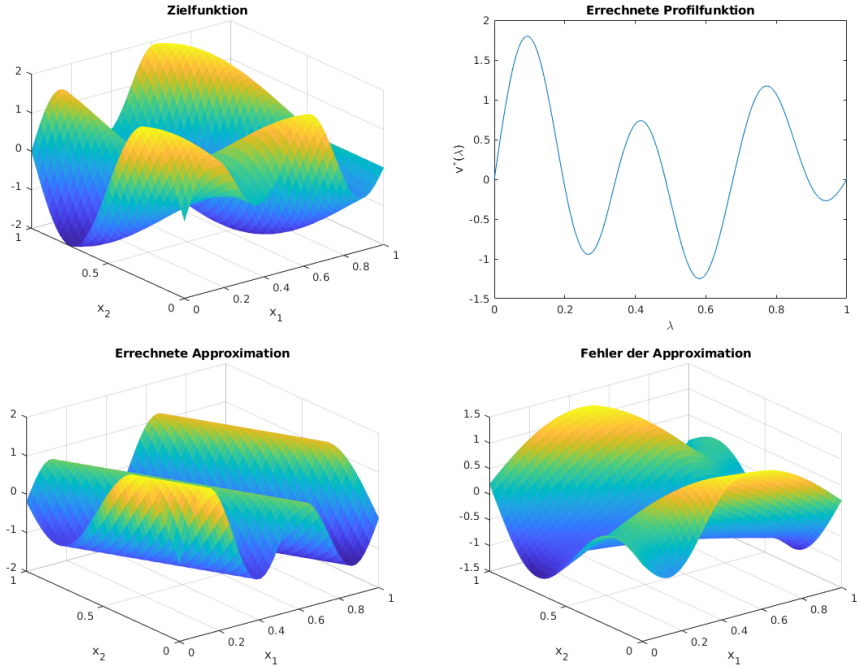

Abbildung 3.7: Ergebnis nach einer Anwendung von Algorithmus 4

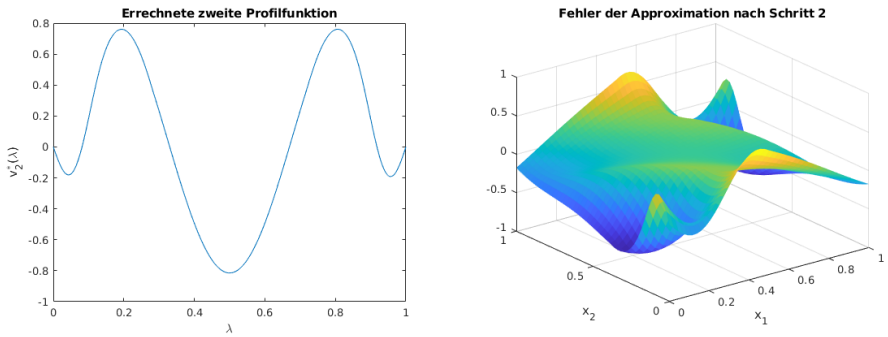

Abbildung 3.8: Ergebnis nach Anwendung von Algorithmus 4 auf $\tilde{u} = u - v_1^*$

Wir sehen weiter, dass bereits bei kleinem Fehler der Richtungen wie sie durch Algorithmus 4 auftreten können zu großen Fehlern nach mehreren Iterationen führen können.

Beispiel 3.22 Das Verhalten liegt nicht an numerischen Ungenauigkeiten in der Berechnung, wie wir durch ein analytisches Beispiel sehen. Sei $\mathbf{a}_1 = (1,0)^T$ und $\mathbf{a}_2 = (0,1)^T$. Wir betrachten die Funktion in zwei Variablen $\mathbf{x} = (x_1, x_2)^T$

$$u(\mathbf{x}) = v_1(\langle \mathbf{a}_1, \mathbf{x} \rangle) + v_2(\langle \mathbf{a}_2, \mathbf{x} \rangle) \tag{3.14}$$

mit Profilfunktionen $v_1 = v_2 = \mathbb{1}_{x \geq \frac{1}{2}}$. Mit vorgegebener exakter Richtung rekonstruiert der Algorithmus die Profilfunktionen v_1, v_2 bis auf eine Konstante (gemäß 3.9 und Lemma 3.20) genau mit Fehler $\frac{1}{2}$ nach einem Schritt und exakter Rekonstruktion nach zwei Schritten. Jedoch ist für $\mathbf{a}_3 = (1,1)^T$ die optimale Profilfunktion gleich

$$\tilde{v} = \begin{cases} 0, & x \leq \frac{1}{4}, \\ 4(x - \frac{1}{4}), & \frac{1}{4} < x \leq \frac{3}{4}, \\ 2, & \frac{3}{4} < x \leq 1. \end{cases}$$

mit Fehler $6 \cdot \frac{1}{8} \cdot \frac{1}{3} = \frac{1}{4} < \frac{1}{2}$ nach einem Schritt. Weiter ist das Residuum $u - c\tilde{v}$ für alle $c \in \mathbb{R}$ keine Ridge-Funktion und daher nicht in einem weiteren Schritt exakt rekonstruierbar. Es zeigt sich erneut, dass ein schrittweises Vorgehen (greedy-type) nicht das optimale Ergebnis liefert.

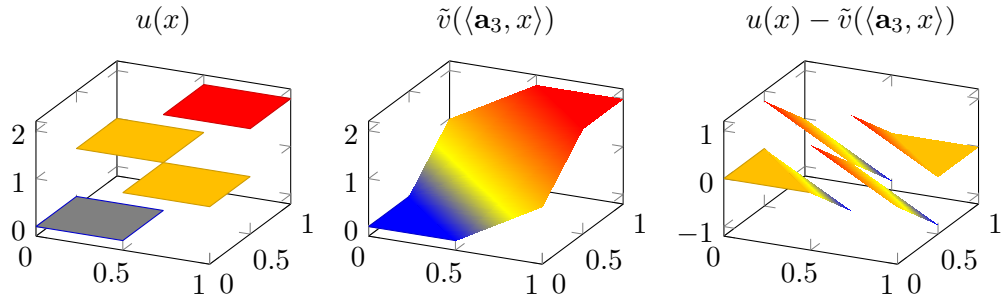

Abbildung 3.9: Die Funktion u, die Approximation $\tilde{v}$ mit Richtung $\mathbf{a}_3$ und das Residuum aus Beispiel 3.22.

Wir benötigen eine stabilere Methode um mehrere Profilfunktionen aus einer gegebenen Es gilt die folgende Charakterisierung aus [66, Theorem 8.4, S.93]:

Lemma 3.23 Sei $u \in L_2([0,1]^d)$, $\mathbf{a}_1, \ldots, \mathbf{a}_r \in \mathbb{R}^d$ linear unabhängig. Dann ist $F^*(x) = \sum_{i=1}^r v_i^*(\langle \mathbf{a}_i, x \rangle)$ eine Bestapproximation aus dem Raum der Ridge-Funktionen an u genau dann, wenn

$$\int_{I(j;\lambda)} u(x) - F^*(x)\, d\sigma(x) = 0, \quad \text{für alle } j = 1, \ldots, r \text{ und fast alle } \lambda \in \mathbb{R}, \tag{3.15}$$

wobei $I(j;\lambda) = \{x \in [0,1]^d \mid \langle \mathbf{a}_j, x \rangle = \lambda\}$ und σ das $(d-1)$-dimensionale Maß auf $I(j;\lambda)$ ist.

Falls die Richtungen $\mathbf{a}_j$ und damit die Mengen $I(j;\lambda)$ senkrecht aufeinander stehen, können wir (3.15) schreiben als

$$\begin{aligned}\int_{I(j;\lambda)} u(x)\,d\sigma(x) &= \int_{I(j;\lambda)} \sum_{i=1}^{r} v_i^*(\langle \mathbf{a}_i, x\rangle)\,d\sigma(x)\\ &= \int_{AI(j;\lambda)} \sum_{i=1}^{r} v_i^*(\langle e_i, x\rangle)\,d\sigma(x)\end{aligned}$$

für $j = 1,\dots,r$ und fast alle $\lambda \in \mathbb{R}$ mit A die Matrix bestehend aus den Richtungen und e_i dem i-ten Einheitsvektor. D.h. $Ae_i = \mathbf{a}_i$. Dann ist

$$v_j^*(\lambda) = \frac{1}{|AI(j;\lambda)|}\int_{AI(j;\lambda)} u(x)\,d\sigma(x) - \sum_{\substack{i=1\\ i\neq j}}^{r}\int_{AI(j;\lambda)} v_i^*(x_i)\,d\sigma(x). \tag{3.16}$$

Wir betrachten im Weiteren den Fall $r = d = 2$ und ohne Verschiebung $b_j = 0$. In diesem Fall sind die Mengen $I(j;\lambda)$ Geraden orthogonal zur Richtung a_j. Die obige Charakterisierung 3.15 wird zu

$$\int_{I(j;\lambda)} u(x)\,d\sigma(x) = \int_{I(j;\lambda)} v_1(\langle \mathbf{a}_1, x\rangle) + v_2(\langle a_2, x\rangle)\,d\sigma(x), \quad j = 1,2, \text{ für fast alle } \lambda \in \mathbb{R},$$

bzw. äquivalent zu

$$\begin{aligned} v_1(y) &= \frac{1}{|I(1;y)|}\int_{I(1;y)} \left(u(x) - v_2(\langle \mathbf{a}_2, x\rangle)\right)\,d\sigma(x), \quad \forall y \in \mathbb{R},\\ v_2(y) &= \frac{1}{|I(2;y)|}\int_{I(2;y)} \left(u(x) - v_1(\langle \mathbf{a}_1, x\rangle)\right)\,d\sigma(x), \quad \forall y \in \mathbb{R}.\end{aligned} \tag{3.17}$$

Sei zudem $\bar{u} := \int_{[0,1]^2} u(x)\,dx$. Dann ist

$$v_j^*(\lambda) = \frac{1}{|I_{\mathbf{a}_1}(\lambda)|}\int_{I(j;\lambda)} (u(x) - v_{j\pm1}(\langle \mathbf{a}_{j\pm1}, x\rangle)\,dx, \quad j = 1,2,$$

und v_j ist eindeutig bis auf Konstanten c_j mit $c_1 + c_2 = \bar{u}$.
Substituieren wir die zweite Gleichung von (3.17) in die Erste, so ergibt sich

$$v_1(\lambda) = \frac{1}{|I(1;\lambda)|}\int_{I(1;\lambda)} \left(u(x) - \frac{1}{|I(2;\langle \mathbf{a}_2, x\rangle)|}\int_{I(2,\langle \mathbf{a}_2,x\rangle)} \left(u(y) - v_1(\langle \mathbf{a}_1, y\rangle)\right)\,d\sigma(y)\right)\,d\sigma(x) \tag{3.18}$$

als die Lösung einer Integralgleichung zweiter Art.
Wir wollen für diesen Fall die Existenz und Eindeutigkeit der Bestapproximation zeigen. Dafür benötigen wir grundlegende Sätze über die Lösbarkeit von Integralgleichungen. Wir formulieren (3.18) um in eine Gleichung der Form

$$v(t) = g(t) + \int_{(0,1)^2} k(x,y,t)v(\langle \mathbf{a}_1, x\rangle)\,dxdy, \tag{3.19}$$

indem wir setzen

$$\begin{aligned} g(t) &= \frac{1}{|I(1;\lambda)|}\int_{I(1;t)} u(x) - \frac{1}{|I(2;\langle \mathbf{a}_2, x\rangle)|}\int_{I(2,\langle \mathbf{a}_2,x\rangle)} u(y)\,d\sigma(y)d\sigma(x),\\ k(x,y,t) &= \frac{\mathbb{1}_{I(1;t)}(x)}{|I(1;t)|}\,\frac{\mathbb{1}_{I(2;\langle \mathbf{a}_2,x\rangle)}(y)}{|I(2;\langle \mathbf{a}_2, x\rangle)|}.\end{aligned}$$

Sei $G \subset \mathbb{R}$ gleich $G = \{y \in \mathbb{R} \mid I(1;y) \cap [0,1]^2 \neq \emptyset\}$. Der Operator

$$L : L_2(G) \to L_2(G) : \; v \mapsto \int_{(0,1)^2} k(x,y,t) v(\langle \mathbf{a}_1, y \rangle)\, dxdy \tag{3.20}$$

ist kompakt. Wir können $|I(j;y)|$ gegen $|I(j;y+h)|$ für hinreichend kleines h durch geometrische Überlegungen abschätzen:

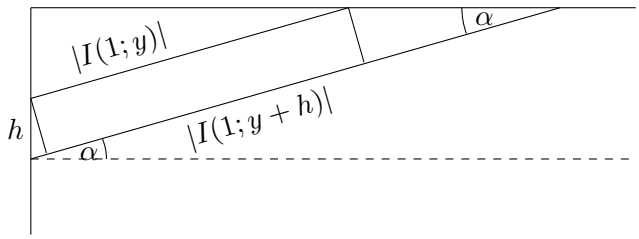

Abbildung 3.10: Geometrische Herleitung der Länge von $I(j;y+h)$

Wollen wir Gleichung (3.18) numerisch lösen, ergibt sich für jedes λ ein lineares Gleichungssystem wie folgt: Sei o.B.d.A. $\mathbf{a}_1$ im ersten Quadranten, $h > 0$ und $|h|$ klein, sodass $I(j;y)$ und $I(j;y+h)$ oberhalb der Geraden $x_1 = x_2$ liegen wie in Abbildung 3.10. Alle weiteren Fälle folgen aufgrund der Symmetrie. Es ist $\alpha = \arccos(\frac{(\mathbf{a}_1)_1}{\|\mathbf{a}_1\|_2})$ und

$$|I(1;y+h)| = |I(1;y)| + h\sin\alpha + h\cos\alpha\cot\alpha. \tag{3.21}$$

Damit hängt die Länge von $I(1;y+h)$ stetig von h ab. Das hilft uns für die numerische Lösung: Wir können damit den Fehler durch die Diskretisierung abschätzen. Außerdem gibt uns dies auch eine theoretische Lösbarkeit.

Lemma 3.24 Seien $\mathbf{a}_1, \mathbf{a}_2 \in \mathbb{R}^2$ linear unabhängig, L der Operator wie oben definiert. Dann besitzt die Gleichung (3.18) eine Lösung.

Der Beweis folgt Standard-Verfahren für Integralgleichungen, siehe bspw. [43].

Beweis: Der Operator A ist linear und beschränkt. Sei $K \subset L_2(G)$ kompakt. Dann ist nach dem Satz von Kolmogorov-Riesz LK kompakt, falls k gleichgradig stetig ist. Der Kern ist $k(x,y,t)$ ist insbesondere Lipschitz-stetig in t, da nach (3.21) $|I(1;t)|$ Lipschitz-stetig in t ist. Weiter ist der Operator L surjektiv, denn der Operator rekonstruiert für eine Richtung nach Lemma 3.14 exakt $v_{\mathbf{a}}$ für ein beliebig gewähltes $u = v_{\mathbf{a}}(\langle \mathbf{a}, \cdot \rangle)$. Damit ist nach der Fredholmschen Alternative der Operator L invertierbar und die Integralgleichung (3.18) besitzt eine Lösung. □

Um diese Integralgleichung praktisch zu lösen, können wir entsprechende numerische Verfahren wie in [44] benutzen. Für spezielle Richtungen können wir auch genauer eine Lösung angeben. Sei etwa $\mathbf{a}_1 = (0,1)^T, \mathbf{a}_2 = (\frac{1}{2}, \frac{1}{2})$ und die Verschiebungen $b_1 = 0, b_2 = \frac{1}{2}$.

Dann können wir $I(1;t)$ und $I(2;\langle \mathbf{a}_2, x\rangle + b_2)$ expliziter angeben als

$$I(1;t) = \{(s_1, s_2) \in [0,1]^2 \mid s_2 = t\}, \tag{3.22}$$

$$I(2;\langle \mathbf{a}_2, x\rangle + b_2) = \{(z, z + x_2 - s_1) \mid z \in [0, 1 + s_1 - x_2]\}. \tag{3.23}$$

Damit können wir den hinteren, von v abhängigen Teil der Integralgleichung (3.19) vereinfachen zu $-\int_0^1 \log(\max\{t,y\} - ty)v(y)\,dy$. Daraus können wir Eigenpolynome dieses Operators bestimmen, d.h. Polynome p mit $p(x) = -\int_0^1 \log(\max\{x,y\} - xy)p(y)\,dy$. Wir erhalten die ersten Eigenpolynome $p_1(x) = x - \frac{1}{2}, p_2(x) = x^2 - x + \frac{1}{6}, p_3(x) = y^3 - \frac{3}{2}y^2 + \frac{3}{5}y - \frac{1}{20}$ und stellen fest, dass es sich bei diesen um verschobene Legendre-Polynome handelt. Indem wir Polynome bzgl. der Basis von Legendre-Polynomen darstellen, können wir den Operator G besonders einfach anwenden und ihn mittels einer Neumann-Reihe $(\mathrm{id} - G)^{-1} = \sum_{k=0}^{\infty} G^k$ invertieren, um eine Lösung der Integralgleichung zu erhalten.
Hierbei ist noch zu klären, ob sich auch für beliebige Richtungen einfache Eigenpolynome bzw. andere Eigenfunktionen des Integraloperators finden lassen. Dies ist Gegenstand laufender Forschung.

Optimale Profilfunktionen bezüglich mehrerer Funktionen

Erweitern wir das Approximationsproblem vom Typ 3 bezüglich mehrerer Zielfunktionen. Gegeben seien N Funktionen $u_i : [0,1]^d \to \mathbb{R}, i = 1, \ldots, N$ und eine gewünschte Anzahl an Ridge-Funktionen M. Nun ist unser Ziel M Profilfunktionen $v_i, i = 1, \ldots, M$ und Richtungen $(\mathbf{a}_i, b_i)$ sowie Koeffizienten $c_i, i = 1, \ldots, M$ zu bestimmen, sodass

$$\max_{i=1}^{N} \left\| u_i(\mathbf{x}) - \sum_{j=1}^{M} c_j v_j(\langle \mathbf{a}_j, \mathbf{x}\rangle + b_j) \right\|_{L_2([0,1]^d)} \tag{3.24}$$

minimal ist. Durch den im vorherigen Abschnitt vorgestellten Algorithmus erhalten wir zu jeder Zielfunktion u_i beliebig viele passende Profilfunktionen. Ziel ist es nun, aus diesen Profilfunktionen M auszuwählen bzw. aus diesen zu errechnen, sodass der Wert in Gleichung (3.24) minimal ist.

Beispiel 3.25 Betrachten wir nun ein einfaches Beispiel, um zu zeigen, dass es optimal sein kann, Funktionen zu kombinieren.
Die Lösung parameterabhängigen Transportgleichung in zwei Dimensionen auf $[0,1]^2$ mit Anfangsbedingung

$$u_{x_1} + \mu u_{x_2} = 0, \quad u(0,y) = u_0(y), y \in \mathbb{R}$$

ist gegeben durch $u_\mu(x_1, x_2) = u_0(x_2 - \mu x_1)$. Lösen wir die Gleichung für verschieden große Parameter μ, so erhalten wir stets nur einen Teil der optimalen Profilfunktion u_0 und nur für $\mu \to 0$ sehen wir die gesamte Funktion u_0. Falls auch die Anfangsbedingung $u(0,y) = u_0(y)$ parameterabhängig ist, lässt sich nur durch Kombinieren von vielen Lösungen u_0 finden. Eine zu große Anzahl an Profilfunktionen M ist insbesondere deshalb ein Problem, da der Particle Swarm Algorithmus online effizient arbeiten soll und der Rechenaufwand mit

$2M \cdot n^2$ wächst. (siehe Lemma 3.11)
Gleichzeitig ist $\hat{u}_\mu$ von einer einfachen Struktur unabhängig vom Parameter μ, wie in Grafik 3.11 zu sehen ist. Dort sehen wir, dass für $u_0(y) = \sin(50y) + \sin(75y)$ die entsprechenden Einträge der 2D-Fourier-Transformierten von $u_0(x_2 - \mu x_1)$ weiter auf einer Geraden liegen, die orthogonal zu der Richtung der Ridge-Funktion (hier $(1,0), (1,1), (1,2)$) ist. Weiter sind die Peaks verschoben durch die verschiedenen Parameter μ, aber weiterhin leicht zu erkennen.

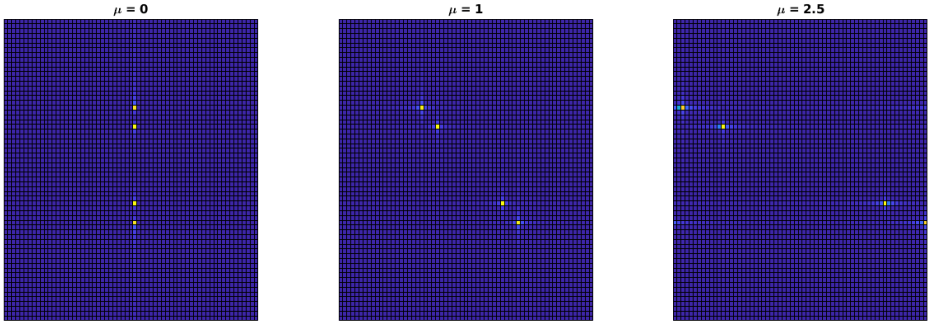

Abbildung 3.11: Magnitude der Fourier-Transformation von u_μ für $\mu = 0, 1, 2.5$

Wir verfolgen diesen Ansatz weiter und formulieren Lemma 3.14 über die Fourier-Transformierte.

Lemma 3.26 Seien $\mathbf{a} \in \mathbb{R}^d$ und $u \in L_2([0,1]^d)$ gegeben. Sei $\mathcal{F}_d : L_2(\mathbb{R}^d) \to L_2(\mathbb{R}^d)$ die Fourier-Transformation in d Dimensionen und $\lambda(y) = \lambda(I_\mathbf{a}(y))$. Dann ist

$$\hat{v}_\mathbf{a} * \hat{\lambda}(\xi) = \mathcal{F}_d(u)(\xi\mathbf{a}). \tag{3.25}$$

Beweis: Sei $\mathcal{F}_1^{-1}$ die univariate Fourier-Rücktransformation, δ die Delta-Distribution und $S_\mathbf{a} : L_2(\mathbb{R}^d) \to L_2(\mathbb{R})$ der Schnittoperator definiert durch $S_\mathbf{a}(f)(x) = f(x\mathbf{a})$. Für die L_2-Bestapproximation $v_\mathbf{a}$ aus dem Raum der Ridge-Funktionen gilt nach Lemma 3.14

$$v_\mathbf{a}(y) = \frac{1}{\lambda(I_\mathbf{a}(y))} \int_{\mathbf{s} \in I_\mathbf{a}(y)} u(\mathbf{s})\, d\mathbf{s} = \int_{I_\mathbf{a}(y)} u(\mathbf{s})\, d\sigma(\mathbf{s}),$$

für σ das Maß auf $I_\mathbf{a}(y)$, d.h. $v_\mathbf{a}$ ist die gewichtete Projektion von u auf die Gerade $\{r\mathbf{a}\}$. Es gilt

$$\begin{aligned}
\mathcal{F}_1^{-1} S_\mathbf{a} \mathcal{F}_d u(y) &= \int_\mathbb{R} e^{2\pi i \xi y} S_\mathbf{a} \mathcal{F}_d u(\xi)\, d\xi \\
&= \int_\mathbb{R} \int_{\mathbb{R}^d} e^{2\pi i \xi y} e^{-2\pi i \langle \mathbf{x}, \xi\mathbf{a} \rangle} u(\mathbf{x})\, d\mathbf{x}\, d\xi \\
&= \int_{\mathbb{R}^d} u(\mathbf{x}) \int_\mathbb{R} e^{2\pi i \xi (y - \langle \mathbf{x}, \mathbf{a} \rangle)} d\xi\, d\mathbf{x} \\
&= \int_{\mathbb{R}^d} u(\mathbf{x}) \delta(y - \langle \mathbf{x}, \mathbf{a} \rangle)\, d\mathbf{x}
\end{aligned}$$

$$= \int_{\mathbf{x} \in I_{\mathbf{a}}(y)} u(\mathbf{x})\, d\mathbf{x}$$
$$= v_{\mathbf{a}}(y)\lambda(I_{\mathbf{a}}(y)).$$

Entsprechend ist $\hat{v}_{\mathbf{a}}(\xi) * \hat{\lambda}(\xi) = \mathcal{F}_d(u)(\xi\mathbf{a})$ wie behauptet. □

3.4 Numerische Experimente

Durch numerische Experimente wollen wir die Leistungsfähigkeit unseres Algorithmus 2 zeigen. Zunächst für ein einfaches Problem, wie das Auffinden einer Richtung bei gegebenen Profilfunktionen:

Beispiel 3.27 Wir betrachten die Funktion

$$v(t,x) = 3\left|x + \frac{1}{4}t\right| + \sin(8x + 8t) + \frac{1}{2}\left(x + \frac{1}{2}t\right)^2$$

welche stetig und bis auf eine Gerade sogar beliebig oft stetig differenzierbar ist. Für solche Funktionen kann unser Algorithmus sehr schnell die entsprechenden Richtungen finden. Hier ergibt sich nach 100 Iterationen ein L_2-Fehler von $8,4501 \cdot 10^{-16}$ bei einer Diskretisierung mit $n = 129$ und unter Verwendung von 125 Partikeln bei einer Schrittweite von $\delta = \frac{1}{3}$.

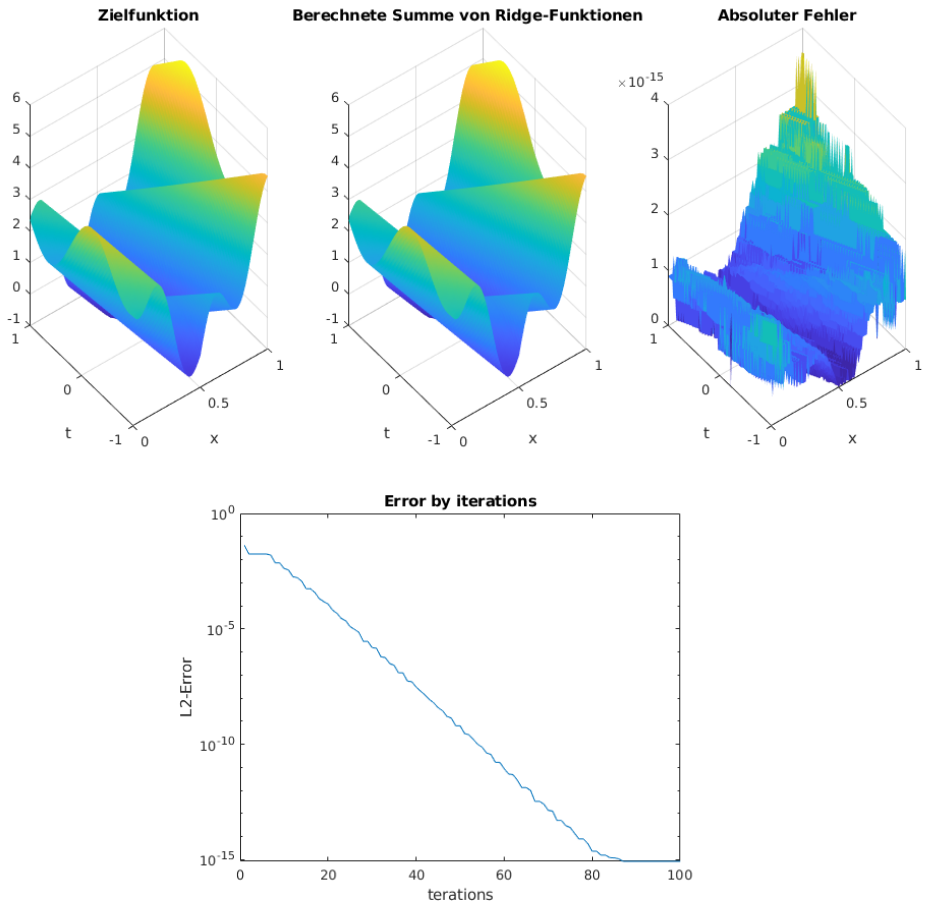

Abbildung 3.12: Ein einfaches Beispiel für die Approximation

Nun betrachten wir ein schwierigeres Problem. Dazu haben wir als Profilfunktionen vier verschiedene Wavelet-Funktionen gewählt, nämlich v_1 als Haar-Wavelet, v_2 als Mexican-hat bzw. Ricker-Wavelet, v_3 als das Morlet-Wavelet, jeweils skaliert und um $0,5$ zentriert. Zusätzlich wählen wir $v_4 = \text{ReLU}$ als die bekannte Hockeystick-Funktion. Da Wavelets durch Translation und Skalierung einen Frame oder eine Basis des L_2 bilden und Translation und Skalierung genau die Aufgaben der Verschiebungen b und der Richtungen a in einer Dimension sind, bedeutet eine gute Performance bei diesen Funktionen dass theoretisch alle dadurch genäherten Funktionen, sprich ganz L_2, gut approximiert werden können. Zusätzlich haben wir 6 verschiedene Polynome als linearen Anteil gewählt, der Einfachheit halber die Standardbasis für Polynome in zwei Dimensionen bis Grad zwei $\varphi_1(x_1, x_2) = 1, \varphi_2(x_1, x_2) = x_1, \varphi_3(x_1, x_2) = x_2, \varphi_4(x_1, x_2) = x_1^2, \varphi_5(x_1, x_2) = x_2^2, \varphi_6(x_1, x_2) = x_1 x_2$. Bemerke, dass diese Basis für Zwecke von Berechnungen nicht optimal stabil ist. Wir wählen die Zielfunktion als

$$u(x_1, x_2) = \sum_{i=1}^{6} \varphi(x_1, x_2) + \sum_{j=1}^{4} v_j(x_1 + a_j x_2)$$

d.h. wir haben zunächst die Verschiebung als 0 gewählt und die Koeffizienten α_i, c_j gleich

1 gesetzt. Wir betrachten die Funktion auf $\Omega = [0,1] \times [-1,1]$ und mit Feinheit der Diskretisierung $n = 129$. Wir haben mit $6^4 = 1296$ Partikeln und Schrittweite $\delta = \frac{1}{3}$ nach 1000 Iterationen einen absoluten Fehler von $0,09 \cdot 10^{-15}$ erreicht. Die Ergebnisse sind in Grafik 3.13 zu finden.

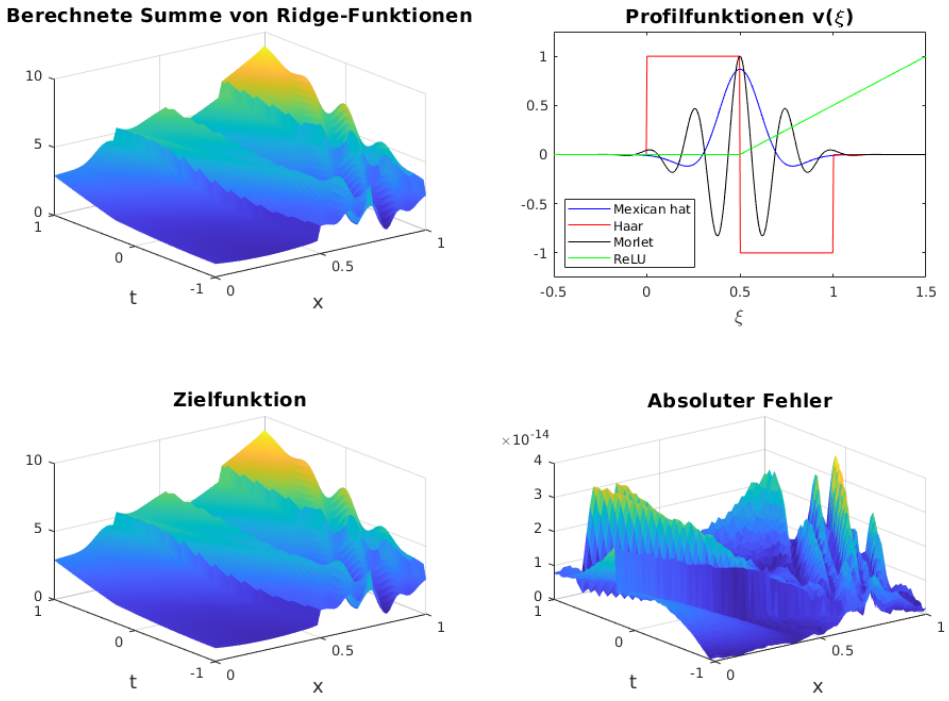

Abbildung 3.13: Approximation von mehreren Wavelet-Funktionen

Setzen wir nun den Algorithmus für die Lösung parametrischer partieller Differentialgleichungen zusammen. Wie genau bestimmt wird, wann ein Fehler „schnell" genug fällt, hängt von Art der Gleichung ab. Für gewisse elliptische Probleme ist bspw. aus dem linearen Teil nur N^{-1} zu erwarten und eine Erweiterung mit Ridge-Funktionen früh sinnvoll.

Input : Eine parametrische Differentialgleichung $L_\mu u_\mu = f_\mu$, kompakte Parametermenge Λ

Output : Eine effiziente online-Methode zur Lösung der Gleichung

Wähle Trainingsmenge $\Lambda' \subseteq \Lambda$ **while** *Fehler e fällt schnell genug* **do**

  Löse die PPDE für μ und schätze Fehler $e = \sup_{\mu\in\Lambda} \|u_\mu - \tilde{u}_\mu\|$ ab für $U_{N,M}$ wie in (3.3)

  $X_{M+1} \leftarrow X_M \cup \{u_\mu\}$

  Wähle neues $\mu \in \Lambda'$

end

Berechne v^* zu u_μ mit Algorithmus 4

$V_{N+1} \leftarrow V_N \cup \{v^*\}$

if *Fehler e klein* **then**

  Interpoliere $a(\mu)$

  Gebe $V_N, X_M, a(\cdot)$ aus

end

Berechne weiteres v^* zum Residuum von u_μ

Gehe zur Berechnung von v^*

Algorithmus 5 : Algorithmus zur Lösung parametrischer Probleme

Wir zeigen dessen Leistungsfähigkeit anhand einiger PPDEs:

Beispiel 3.28 Die homogene lineare Transportgleichung in zwei Dimensionen mit Parameter $\mu > 0$ ist gegeben durch

$$\partial_t u(t,x) + \mu \partial_x u(t,x) = 0 \text{ auf } (0,T) \times \mathbb{R}$$
$$u(0,x) = u_0(x), x \in \mathbb{R}.$$

Die analytische Lösung des Problems lautet

$$u(t,x) = u_0(x - \mu t)$$

und ist somit eine Ridge-Funktion.

Wir wenden nun unseren Algorithmus auf diese Gleichung an. Die Approximationsgüte hängt bei dieser einfachen Gleichung stark von den Randbedingungen ab. Im Falle einer homogenen Gleichung mit glattem u_0 konnte das exakte Ergebnis sogar häufig exakt reproduziert werden.

Beispiel 3.29 Wir betrachten die (eindimensionale) lineare Wellengleichung

$$\partial_{tt}^2 u - \mu^2 \partial_{xx}^2 u = 0, \quad t > 0, x \in \mathbb{R}$$

mit Anfangswert $u(0) = f$ und $\dot{u} = 0$. Die analytische Lösung ist mit der Formel von d'Alembert gegeben als

$$u_\mu(t,x) = \frac{1}{2}(f(x - \mu t) + f(x + \mu t)),$$

was eine Summe von zwei Ridge-Funktionen ist. Weiterhin ist auch für allgemeinere Randbedingungen mit $\dot{u} \neq 0$ die Lösung eine Summe von Ridge-Funktionen.

Beispiel 3.30 (Ebene Wellen) Ein Beispiel in höherer Raumdimension ist die ebene Wellengleichung.

$$\nabla^2 u - \frac{1}{c^2} u_{tt} = 0$$

Die Lösungen dieser PPDE sind Funktionen der Form

$$u(t, x) = v(\vec{n}^T - ct)$$

für eine Funktion $v : \mathbb{R} \to \mathbb{R}$. Diese beschreibt die Ausbreitung einer mehrdimensionalen Welle in Richtung $\vec{n}$ mit Geschwindigkeit c. Existieren bspw. mehrere Lichtquellen kann es ein Ziel sein, eine Funktion der Form

$$u(t, x) = \sum_{i=1}^{\nu} v_i(\vec{n}^T x - ct)$$

zu approximieren. Wir können o.B.d.A. annehmen, dass $t = 1$ fest ist und müssen somit nur noch die Richtungen $a_i = (-c, n_1^i, \dots n_d^i), i = 1, \dots, \nu$ rekonstruieren. Daher benötigen wir Partikel der Dimension νd. Für ein dreidimensionales Beispiel erreichen wir einen Fehler von $1.0404 \cdot 10^{-4}$ nach 200 Iterationen.

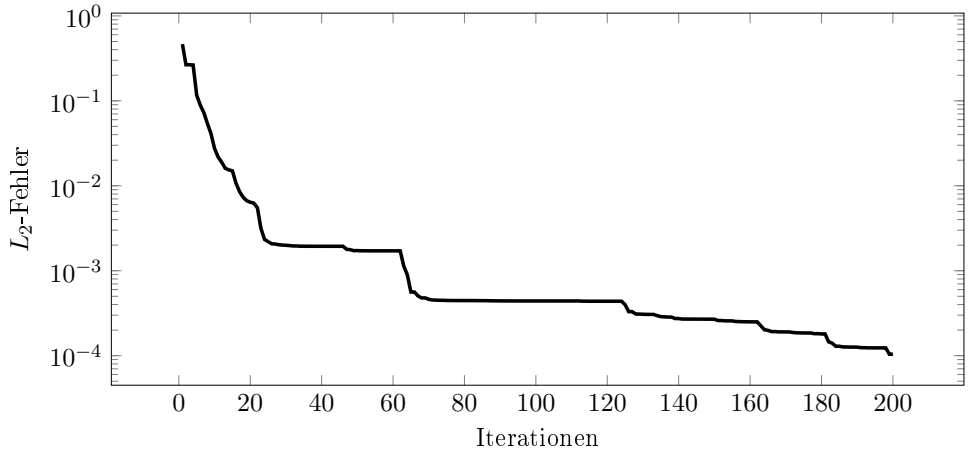

Abbildung 3.14: Fehlerplot zu Beispiel 3.30 für $n = 69$ und $n_p = 4^6$ Partikel

Beispiel 3.31 Wir wollen nun zeigen, dass unser Algorithmus auch für nicht-stetige Funktionen bzw. Anfangswerte eine gute Approximation findet. Dazu betrachten wir eine Wendeltreppen-artige Funktion bestehend aus 10 Sprungfunktionen

$$v(x) := \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases},$$

die als Ridge-Funktionen mit gleichmäßig ansteigendem Winkel summiert werden, wie in Grafik 3.15 zu sehen ist.

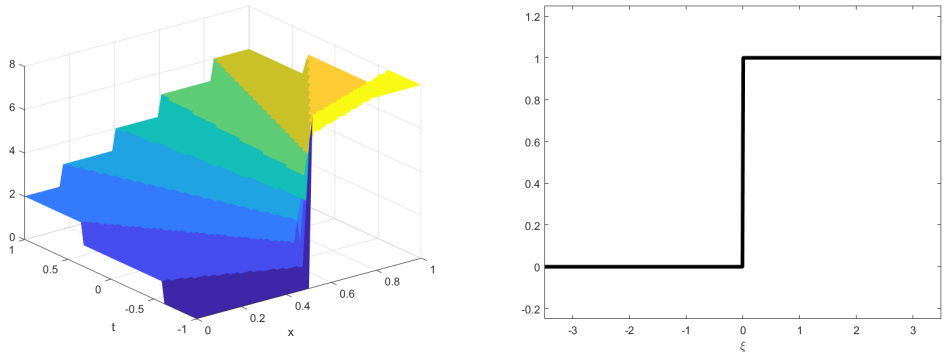

Abbildung 3.15: Die betrachtete Zielfunktion und die Profilfunktion v

Aufgrund der hohen Anzahl an Profilfunktionen können wir nur eine geringe Anzahl an Partikeln pro Dimension benutzen. Glücklicherweise erreicht der Fehler für $n_{par} = 4$ bereits nach weniger als 50 Schritten einen Fehler von exakt 0, für $n_{par} = 3$ konvergierte der Fehler deutlich langsamer und für $n_{par} = 2$ war keine Konvergenz festzustellen. Der Fehlerabfall lässt sich anhand von Grafik 3.31 nachvollziehen.

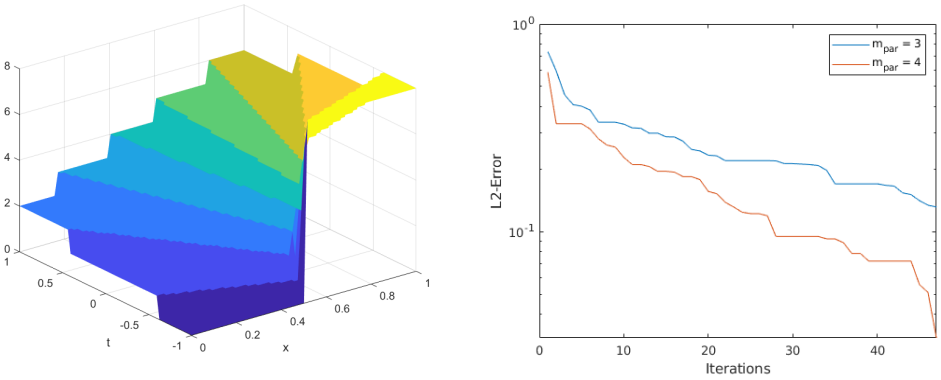

Abbildung 3.16: Berechnete Approximation für $n_{par} = 4$ und Fehler über die Iterationen

Beispiel 3.32 Approximation von Funktionen, die keine Ridge-Funktionen sind. Bisher haben wir Zielfunktionen betrachtet, die Ridge-Funktionen sind, sodass eine exakte Rekonstruktion der Richtung und damit der Funktion möglich war. Um zu zeigen, dass wir auch andere Funktionen approximieren können, betrachten wir zunächst für $c \geq 1$ die Funktion

$$u(t, x; \mu) := \sum_{k=1}^{\infty} \frac{1}{k!} \cos\left(2\pi(k\frac{\sqrt{c}}{10}t + x)\right) \tag{3.26}$$

auf dem Gebiet $\Omega = (0, 1) \times (-1, 1)$. Wir wählen für den linearen Teil unser Approximation

$$\varphi_j(t, x) := u(t, x, j), \quad j = 1, \ldots, N$$

und Profilfunktionen

$$v_i = \cos(2\pi \cdot),\ i = 1, \ldots, M.$$

In Abbildung 3.32 sehen wir den L_2-Fehler für verschiedene Werte von N und M bei fester Diskretisierung $n = 129$ und Schrittweite $\gamma = \frac{1}{3}$. Wir erhalten dabei kein monotones Abklingen des Fehler in N aufgrund von Stabilitätsproblemen bei der Bestimmung der optimalen Koeffizienten nach Lemma 3.8. Wir sehen dafür ein monotones Verhalten für festes N in M wenn man die Daten spaltenweise betrachtet.

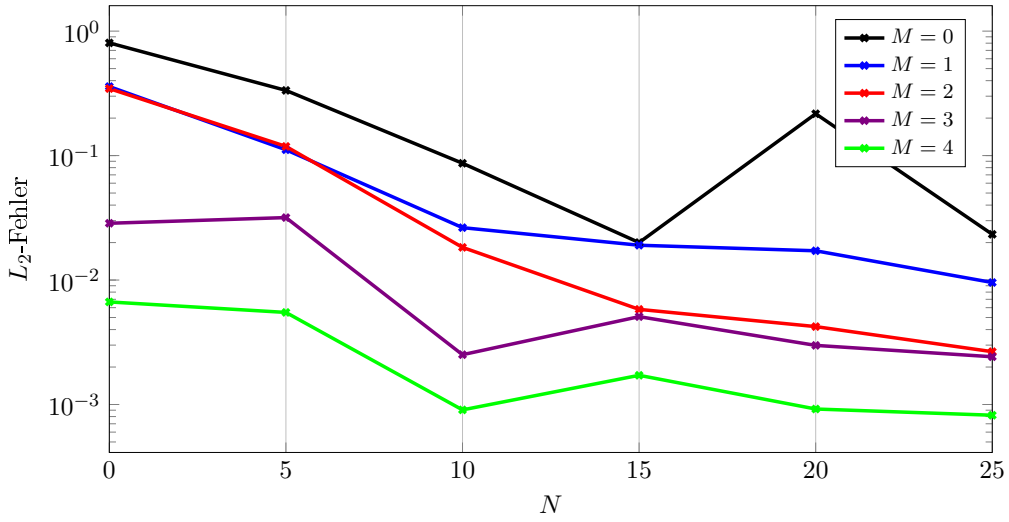

Abbildung 3.17: Approximation einer unendlichen Reihe von Ridge-Funktionen: L_2-Fehler für verschiedene N, M

3.4.1 Verbesserung der Genauigkeit

Der vorgestellte Particle-Swarm-Algorithmus hat den Vorteil, dass er erfolgreich bei der globalen Optimierung dieser komplexen Funktionen ist. Nachteile des Algorithmus sind der relativ große Rechenaufwand und die relativ langsame (lineare) Konvergenz. Die langsame Konvergenz kann durch eine Nachiteration mit einem klassischen Optimierungsverfahren beschleunigt werden. Dazu brechen wir die Iteration des Particle-Swarm-Algorithmus nach einer festgelegten Anzahl von Iterationen oder einer gewissen Rechenzeit ab und nehmen das zu diesem Zeitpunkt beste Partikel als Startpunkt eines anderen Optimierungsverfahrens. Nach den ersten Iterationen des Particle-Swarm-Algorithmus sollte dieses beste Partikel nah an dem globalen Optimum sein, sodass ein normales Verfahren superlinear gegen dies konvergiert. Dazu benötigen wir die Ableitungen der Fehlerfunktion für eine diskretisierte Differentialgleichung, die in dem folgenden Lemma gegeben sind. Hierbei bezeichnen wir für eine Funktion u aus einem Raum V deren Diskretisierung in einem N-dimensionalen Unterraum V^N mit u^N und ebenso die Diskretisierung von Operatoren b mit b^N. Wir verwenden hier eine variationelle Formulierung (vgl. z.B. [2])

Lemma 3.33 Sei $\mathcal{P}$ eine kompakte Parametermenge, $(H, \|\cdot\|_H)$ ein Hilbertraum und $b_\mu : H \times H \to \mathbb{R}$ eine Bilinearform mit $\beta := \inf_{u\in H} \sup_{v\in H} \frac{b_\mu(u,v)}{\|u\|\|v\|} = 1$. Sei weiter H^N eine Diskretisierung des Raums H, sodass $\beta^N = 1$ und u^N die diskretisierte Lösung von $b_\mu(u,v) = f_\mu(v)$ für alle $v \in H$ und eine rechte Seite $f_\mu \in H'$. Dann ist $\|u^N - u_N(\mu;\delta)\|_H = \sqrt{J(\delta)}$ mit u_N einer Lösung aus der Menge $U_{N,M}$ wie in (3.3) und

$$J(\delta) := \|f_\mu - B_\mu u_N(\mu;\delta)\|_H^2.$$

Hier ist B_μ der zur partiellen parametrischen Transport- oder Wellengleichung gehörende Operator $B_\mu u = c_\mu + b_\mu \cdot \nabla u - \nabla \cdot (A_\mu \nabla u)$ und $J(\delta) \to \min!$ ist ein Optimierungsproblem in $N(d+2)$ Veränderlichen.

Beweis: Es gilt

$$\nabla J(\delta) = 2\langle f_\mu - B_\mu u_N(\mu;\delta), \frac{d}{d\delta} B_\mu u_N(\mu;\delta)\rangle$$

Da $\beta = 1$, ist

$$\|u^N - u_N(\mu;\delta)\|_H = \sup_{w\in H^N} \frac{b_\mu(u^N - u_N(\mu), w)}{\|w\|_H} = \sup_{w\in H^N} \frac{f_\mu(w) - b_\mu(u_N(\mu), w)}{\|w\|_H}$$

gleich der Norm des Residuums. □

Beispiel 3.34 Für die parametrisierte lineare Transportgleichung in zwei Dimensionen $B_\mu u = u_{x_1} + \mu u_{x_2} = 0$ auf dem Gebiet Ω mit Anfangswert u_0 und ihre Ridge-Approximation $B_\mu u_N = \sum_{n=1}^N \alpha_n v_n(\langle a_n, \cdot\rangle + b_n)$ fassen wir die Fehlerfunktion zusammen als $J(\delta)$, wobei $\delta = (\alpha_n, a_n, b_n)_{n=1}^N$ die Zusammenfassung der freien Parameter in der Approximation ist. Damit erhält man

$$\nabla J(\delta) = -2 \int_\Omega B_\mu u_N(x) \nabla_\delta B_\mu u_N(x)\, dx.$$

Setzen wir konkret $N = 1$, $\Omega = (0,1)^2$ und $v_1 = u_0$ ein und schreiben $a = (a_1, a_2), \alpha = \alpha_1, b = b_1$, d.h. $d = (\alpha, (a_1, a_2), b)$, erhalten wir für $\nabla J(\delta) = (\nabla J(\delta)_j)_{j=1}^4$

$$\nabla J(\delta)_1 = -2\alpha(a_1 + \mu a_2)^2 \int_\Omega u_0'(\langle a, x\rangle + b)^2\, dx \tag{3.27}$$

$$\begin{aligned}\nabla J(\delta)_2 = &-2\alpha^2(a_1 + \mu a_2)^2 \int_\Omega x_1 u_0'(\langle a, x\rangle + b) u_0''(\langle a, x\rangle + b)\, dx \\ &- 2\alpha^2(a_1 + \mu a_2) \int_\Omega u_0'(\langle a, x\rangle + b)^2\, dx\end{aligned} \tag{3.28}$$

$$\begin{aligned}\nabla J(\delta)_3 = &-2\alpha^2(a_1 + \mu a_2)^2 \int_\Omega x_2 u_0'(\langle a, x\rangle + b_1) u_0''(\langle a, x\rangle + b)\, dx \\ &- 2\alpha^2\mu(a_1 + \mu a_2) \int_\Omega u_0'(\langle a, x\rangle + b)^2\, dx\end{aligned} \tag{3.29}$$

$$\nabla J(\delta)_4 = -2\alpha^2(a_1 + \mu a_2)^2 \int_\Omega u_0'(\langle a, x\rangle + b)^2\, dx \tag{3.30}$$

Damit ist $\nabla J(\delta) = 0$ für beliebiges $\alpha \neq 0$ und $a_1 + \mu a_2 = 0$. Damit lässt sich einfach ein Abstiegsverfahren zur Lösung nutzen.

3.4.2 Anwendung auf die Reduzierte Basis Methode zur Lösung parametrischer partieller Differentialgleichungen

Die vorgestellten Methoden wollen wir nun zur Lösung parametrischer partieller Differentialgleichungen (PPDEs) nutzen.

Definition 3.35 Sei $\mathcal{H}$ ein Hilbertraum, $D \subset \mathbb{R}^p$ kompakt die Parametermenge. Sei zu jedem $\mu \in D$ ein Differentialoperator $L_\mu : \mathcal{H} \to \mathcal{H}$ und ein $f_\mu \in \mathcal{H}$ definiert. Dann ist das parametrische, analytische Problem definiert wie folgt: Finde $u_\mu \in \mathcal{H}$, sodass

$$L_\mu u_\mu = f_\mu \tag{3.31}$$

Wir merken an, dass es uns genügt eine Lösung von 3.31 im schwachen Sinn zu finden. Wir diskretisieren nun das Problem. Sei $X_h \subset \mathcal{H}$ ein endlichdimensionaler Teilraum und sei $L_\mu^h : X_h \to X_h$ ein geeigneter diskreter Differentialoperator für jedes $\mu \in D$. Durch $f_\mu^h \in X_h$ erhalten wir das diskretisierte parametrische Problem: Finde $u_\mu^h \in X_h$, sodass

$$L_\mu^h u_\mu^h = f_\mu^h. \tag{3.32}$$

Sei $P_h : X \to X_h$ die Projektion auf den Raum X_h.
Wir führen nun einen weiteren reduzierten Raum ein. Sei dazu $\tilde{U}_{N,M}$ ein diskreter Teilraum von $U_{N,M}$ wie in Gleichung 3.3 definiert, der gleichzeitig Teilraum von X_h ist. D.h. die Elemente von $\tilde{U}_{N,M}$ seien von der Form

$$\tilde{U}_{N,M} = \left\{ \sum_{i=1}^{N} \alpha_i P_h(\varphi_i) + \sum_{j=1}^{M} c_j P_h(v_j(\langle \mathbf{a}_j, \cdot \rangle + b_j)) \right\}$$

Sei außerdem $P_{N,M} : X_h \to \tilde{U}_{N,M}$ eine Projektion auf den Raum $\tilde{U}_{N,M}$. Diese wird von dem von uns vorgeschlagenen Particle-swarm-Algorithmus bewerkstelligt.

Definition 3.36 (Separabilität) Wir nennen $L^h, f^h, \mathbf{a}^j$ *separierbar parametrisch*, falls $L_\mu^h = \sum_{l=1}^{R} \theta_l(\mu_l) L_{\mu_l}^h$, $f_\mu^h = \sum_{l=1}^{S} \gamma_l(\mu_l) f_{\mu_l}^h$ und ebenso $\mathbf{a}_\mu^j = \sum_{l=1}^{Q} \xi_l(\mu_l) \mathbf{a}_{\mu_l}^j$ für Funktionen $\theta_l, \gamma_l, \xi_l : D \to \mathbb{R}$.

Sind diese Voraussetzungen erfüllt, so erhalten wir für ein neues $\mu \in D$ die Operatoren, Funktionen und Richtungen L_μ^h, f_μ^h und $\mathbf{a}_\mu$ durch eine einfache Interpolation. Dadurch genügt es das reduzierte Problem $L_\mu^{red} u_\mu = f_\mu^{red}$ durch ein kleines lineares Gleichungssystem zu lösen. Dazu berechnen wir $f_\mu^{red} = \sum_{l=1}^{S} \gamma_l(\mu) f_{\mu_l}^h$ und $A = \sum_{l=1}^{R} \theta_l(\mu) P_{N,M}(L_\mu^h(\varphi_i))_{i=1}^N$.

Bemerkung 3.37 Es genügt hier, den Differentialoperator L_μ^h auf den linearen Teil φ_i eines Elements aus $\tilde{U}_{N,M}$ anzuwenden, da die Ableitung einer Ridge-Funktion wieder eine Ridge-Funktion zu derselben Richtung $\mathbf{a}$ ist. Also bleibt die Projektion $P_{N,M}(L_\mu^h(v_j(\langle \mathbf{a}_j, \cdot \rangle)))$ $\subset \left\{ \sum_{j=1}^{M} c_j P_h(v_j(\langle \mathbf{a}_j, \cdot \rangle + b_j)) \right\}$ im Teilraum der Ridge-Funktionen mit variablen Profilfunktionen.

Beispiel 3.38 Als erstes Beispiel dazu betrachten wir die Wärmeleitungsgleichung auf dem sogenannten „Thermal Block". Dies modelliert die Temperaturverteilung in einem aus verschieden stark Wärme leitenden Materialien bestehenden Block. Dies führt auf ein Poisson-Problem mit stückweise konstanten Koeffizienten. Wir möchten effizient für verschiedene Wahlen dieser Koeffizienten Lösungen der Gleichung errechnen. Für unser Beispiel betrachten wir das Gebiet $\Omega = [0,1]^2$, welches wir in 4 Teilgebiete $\Omega_i := [0,1] \times [\frac{i-1}{4}, \frac{i}{4}]$ mit zugehörigen Leitungskonstanten $\mu_i, i = 1, \ldots, 4$ aufteilen. Die Formulierung als Variationsproblem ist gegeben durch

$$a(u, v; \mu) = \sum_{i=1}^{4} \mu_i \int_{\Omega_i} \nabla u(x) \nabla v(x) dx. \tag{3.33}$$

Es ist bekannt aus [35]/[34], dass die exakte Lösung dieses Problems in $x = (x_1, x_2)$

$$u(x; \mu) = \sum_{i=1}^{4} \frac{1}{\mu_i} \varphi_i(x), \tag{3.34}$$

ist, wobei

$$\varphi_i(x) := \begin{cases} \frac{1}{4}, & 0 \leq x_2 < \frac{i-1}{4}, \\ -x_2 + \frac{i}{4}, & x_2 \in [\frac{i-1}{4}, \frac{i}{4}], \\ 0, & \frac{i}{4} < x_2 \leq 1 \end{cases}$$

ist. Wir betrachten nun Parameter $\mu \in [0.1, 10]^4$. Für unsere Menge an linearen Funktionen wählen wir $\Phi_4 = \{\varphi_1, \varphi_2, \varphi_2, \varphi_4\}$. Wegen (3.34) sollte der lineare Teil unser Approximation bereits die Lösung sehr gut annähern. Dies erkennt unser Algorithmus auch in der Praxis und findet bereits nach einem Schritt eine Lösung mit sehr kleinem Fehler und Ridge-Koeffizienten $\mathbf{c} = 0$. Dadurch sehen wir, dass unser Algorithmus ohne gegebene Profilfunktionen Lösungen ebenso gut wie die klassische Reduzierte Basis-Methode findet.

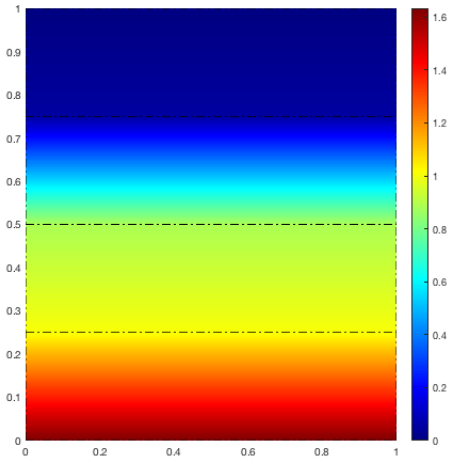

Abbildung 3.18: Eine Lösung zu der PDE aus Beispiel 3.38 zu $\mu = (0.4, 2, 0.3, 5)$.

Beispiel 3.39 Die lineare Transportgleichung aus Beispiel 3.28 hat als Lösung eine Ridge-Funktion, daher setzt unser Algorithmus die Koeffizienten $\alpha_i \equiv 0$ und erreicht mit einer Profilfunktion ein optimales Ergebnis.

Beispiel 3.40 Wir betrachten die Wellengleichung wie in Beispiel 3.29

$$u_{tt}(t,x) - \mu^2 u_{xx}(t,x) = 0, \quad (t,x) \in \Omega = (-1,1) \times (0,1)$$

mit Dirichlet-Randbedingungen. Wir wählen diese so, dass die parametrische Lösung gegeben ist durch

$$u_\mu(t,x) = \frac{1}{2}\sin(10x + 10\mu t) + \frac{1}{2}\sin(10x - 10\mu t).$$

Dies ist, wie bereits in Beispiel 3.29 beschrieben, eine Summe von Ridge-Funktionen. Wir wählen daher $M = 2$ und wählen die Profilfunktionen gleich den Randwerten $v_1 = v_2 = \sin(10\,\cdot)$. Weiter nehmen wir die linearen Funktionen φ_i aus 3.38. Wir sehen, dass dieses Problem schwieriger ist als die vorherigen Beispiele und unser Algorithmus mehr Iterationen benötigt um eine geforderte Genauigkeit zu erreichen.

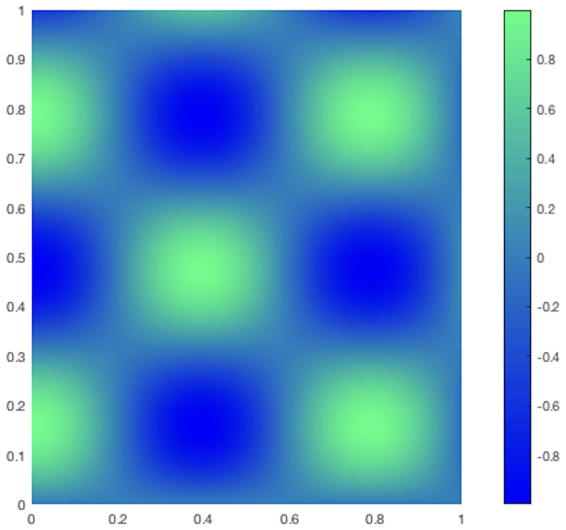

Abbildung 3.19: Lösung der Wellengleichung für den Parameter $\mu = 0.8$.

Wir vergleichen unseren Algorithmus mit einer Lösung durch eine klassische lineare Basis Φ_5 mit $N = 5$ Elementen, die aus einer POD (Hauptkomponentenanalyse) von 20 snapshots u_{μ_i} mit μ_i, $i = 1, \ldots, 20$ gleichverteilt aus $[0.1, 10]$ gewonnen wurde. Wir sehen, dass die POD-Basis sich schlecht zur Approximation einer Lösung eignet und L_2-Fehler nicht unter 0.4767 erzielt, während unser Verfahren für alle getesteten $\mu \in [0.1, 10]$ bessere Ergebnisse erzielt, für einige Parameter sogar fast Maschinengenauigkeit mit einem Fehler von $8,47 \cdot 10^{-18}$.

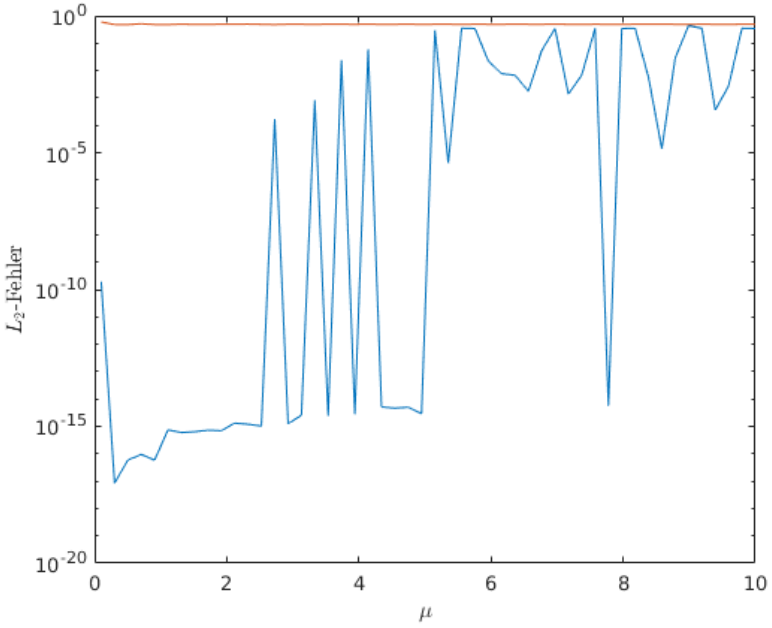

Abbildung 3.20: Fehler der Approximation mit klassischer POD-Basis (rot) und für Erweiterung mit Ridge-Funktionen (blau, unser Verfahren)

Auch bei einer Vergrößerung der POD-Basis auf 20 Elemente Φ_{20} gebildet aus 50 snapshots bleibt der Vorteil der Ridge-Approximation klar. Es ist jedoch zu beobachten, dass das Problem unabhängig von der gewählten linearen Basis für größere Parameter μ schwieriger wird. Dies liegt daran, dass die betrachteten Lösungen u_μ wie $\sin(\mu t)$ stark oszillieren und dadurch die Minima der Kostenfunktion kleiner werden, ähnlich wie im linken Teil der Grafik 3.1. Zum Erreichen guter Ergebnisse sind in diesem Fall mehr Partikel nötig um diese Minima zu erkennen. Für die meisten Parameter μ lässt sich weiterhin eine deutliche Verbesserung der Ergebnisse erreichen.
Unsere Ergebnisse zu den oben beschriebenen Beispielen fassen wir in der folgenden tabellarischen Übersicht zusammen:

Beispiel	Betrachtete PPDE	Parameter μ	K	L_2-Fehler
3.38	Wärmeleitungsgleichung	$(0.1, 10, 1, 0.6)$	1	$5.1019e-15$
3.38	Wärmeleitungsgleichung	$(10, 2, 0.1, 0.5)$	1	$1.4446e-14$
3.38	Wärmeleitungsgleichung	$(0.4, 2, 0.3, 5)$	1	$6.0861e-15$
3.39	Transportgleichung	$1/4$	12	$7.1348e-05$
3.39	Transportgleichung	$1/4$	66	$4.6205e-16$
3.39	Transportgleichung	1	19	$3.0119e-05$
3.39	Transportgleichung	1	70	$9.3829e-17$
3.39	Transportgleichung	4	24	$6.1985e-05$
3.39	Transportgleichung	4	67	0
3.40	Wellengleichung mit Φ_4	$1/4$	20	$7.6682e-05$
3.40	Wellengleichung mit Φ_4	$1/4$	83	$8.9850e-16$
3.40	Wellengleichung mit Φ_{20}	$1/4$	83	$1.8705e-16$
3.40	Wellengleichung mit Φ_4	1	21	$8.2400e-05$
3.40	Wellengleichung mit Φ_4	1	86	$3.1765e-16$
3.40	Wellengleichung mit Φ_{20}	1	86	$2.8851e-16$
3.40	Wellengleichung mit Φ_4	4	28	$4.3012e-05$
3.40	Wellengleichung mit Φ_4	4	83	$8.9850e-16$
3.40	Wellengleichung mit Φ_{20}	4	83	$2.7593e-15$

Tabelle 3.3: Ergebnisse der numerischen Experimente aus diesem Kapitel mit parametrischen PDEs.

3.5 Learning Methoden zur effizienten Bestimmung der Richtungen

In einer praktischen Anwendung können wir nicht für jeden Parameter μ einer PPDE aufwendig die optimale Richtung a^* finden. Wir benötigen insbesondere für die Online-Phase des Algorithmus eine effizientere Methode.
Wir nehmen zunächst die Mengen der Basisfunktionen Φ_N und V_M als fest gegeben an. Wir versuchen dann offline eine Abbildung $\mu \mapsto (\mathbf{a}(\mu), b(\mu))$ zu trainieren. Dazu sei $\mu_{tr} = \{\mu_1, \dots, \mu_T\}$ eine Menge von Trainingsparametern aus dem Parameterraum.
Für niedrige Dimensionen und/oder viele Parameter bietet sich dafür ein einfache Interpolation an, die dann leicht in Echtzeit an weiteren Parameters $\hat{\mu} \notin \mu_{tr}$ ausgewertet werden kann. Wir bezeichnen die interpolierten Richtungen mit $\check{\delta} := (\check{\mathbf{a}}(\hat{\mu}), \check{b}(\hat{\mu}))$. Der Fehler der Approximation ist dann abhängig vom Fehler der Interpolation und der Profilfunktion, wie man durch leichte Rechnung sieht.

Lemma 3.41 Sei J wie in Lemma 3.33 und sei $v \in \mathcal{C}^1(\mathbb{R})$ und Lipschitz-stetig. Dann ist

$$\|J(\check{\delta}) - J(\delta)\| \leq c\|\check{\delta} - \delta(\mu)\| \cdot \|v'(\delta)\|_\infty.$$

Beispiel 3.42 Wir wenden das oben erwähnte Verfahren an auf Lösungen der Transportgleichung

$$u_t + cu_x = 0$$

wobei $c = c(\mu) = \frac{1}{\sqrt{\mu}}$ die relativistische Geschwindigkeit ist und c somit nichtlinear von μ abhängt. Wir samplen äquidistant 10 $\mu \in [0.1, 1]$ und interpolieren die daraus errechneten Richtungen mithilfe von Splines gemäß der MATLAB-Routinen. Wir werten die Fehler für $n = 129$ und 100 Test-Punkte aus $[0.1, 1]$ aus. Dabei verwenden wir drei verschiedene Funktionen als Anfangswerte und sehen die Abhängigkeit, die wir in Lemma 3.33 hergeleitet haben. Zudem können wir gut die Abhängigkeit des Fehlers von v' wie in 3.41 erkennen.

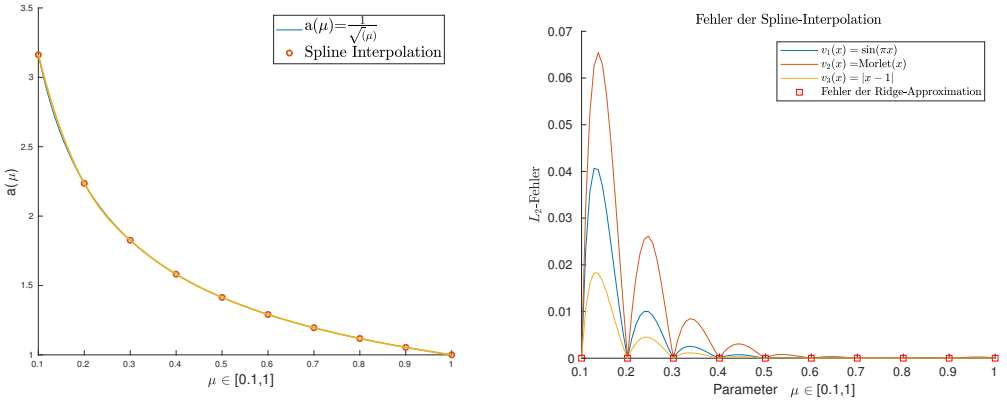

Abbildung 3.21: Fehler durch Interpolation der Richtungen und daraus resultierender Fehler der Ridge-Approximation

3.6 Zusammenfassung und Ausblick

Die in diesem Kapitel vorgestellten Algorithmen bilden ein neues Verfahren zur effizienten Lösung von parametrischen Problemen. Insbesondere konnten wir durch numerische Beispiele zeigen, dass sich das komplizierte Optimierungsproblem vom Typ 1 durch den Particle-Swarm-Algorithmus in vielen Fällen gut lösen lässt. Weiter bietet unser Ansatz eine echte Verbesserung zur klassischen linearen Reduzierten-Basis-Methode, da diese ein Spezialfall unseres Verfahrens ist.
Es bleiben als offene Forschungsfragen zu unseren Verfahren

- Der Particle-Swarm-Algorithmus beruht auf einer gewissen Heuristik, weshalb keine strengen allgemeinen Konvergenzaussagen getroffen werden konnten. Wir untersuchen daher weitere Varianten wie die Kombination des Algorithmus mit Abstiegs- oder Backpropagation-Methoden um entsprechende Aussagen zu erhalten.

- Die Komplexität der Methode wächst nach Lemma 3.11 mit dem Faktor n^d mit d der Raumdimension und ist daher für hochdimensionale Probleme schlecht geeignet. Für die meisten Anwendungen von PPDEs in zwei oder drei Dimensionen ist unser Algorithmus schnell genug, auch wenn für Berechnungen in Echtzeit weitere Optimierungen nötig sind.

- Die Bestimmung optimaler Profilfunktionen im Fall $r = M > 1$ setzt die Lösung einer Integralgleichung, bzw. eines Systems von Integralgleichungen voraus, was rechnerisch aufwendig und von schwer quantifizierbarer Stabilität ist. Durch Finden einer expliziten Darstellung der Lösung einer solchen Integralgleichung hoffen wir, diesen Schritt zu vereinfachen. Für spezielle Richtungen und Gebiete konnten wir bereits solche Darstellungen errechnen, eine allgemeine Lösung ist Teil laufender Forschung. Alternativ ist der Ansatz über die Fourier-Transformation weiter zu untersuchen.

- Eine genaue Quantifizierung des Vorteils durch die Hinzunahme von Ridge-Funktionen zu einer RB-Methode steht noch aus. Aufgrund der komplexen Probleme bei der Bestimmung optimaler Ridge-Funktionen sind die klassischen Greedy-Analysen wie der bei RB-Methode nicht leicht auf diesen Fall übertragbar und es sind neue Ansätze nötig.

Literaturverzeichnis

Die besondere Aktualität und laufende Forschung im Bereich der Neuronalen Netzwerke führt dazu, dass einige Referenzen auf preprints oder kommerzielle Forschungsberichte verweisen. Diese Verweise wurden mit Stand vom 29.05.2023 auf aktuelle Versionen und Veröffentlichungen geprüft.

[1] Antonis Alexandridis und Achilleas Zapranis. "Wavelet Neural Networks: A Practical Guide". In: *Neural Networks* 42 (2013), S. 1 –27. ISSN: 0893-6080.

[2] W. Arendt und K. Urban. *Partielle Differenzialgleichungen: Eine Einführung in analytische und numerische Methoden*. Spektrum Akademischer Verlag Heidelberg, 2010. ISBN: 978-3-8274-2237-8.

[3] Jonathan T. Barron. "Continuously Differentiable Exponential Linear Units". In: (Apr. 2017). arXiv: 1704.07483 [cs.LG].

[4] Julius Berner u. a. "Theory of Deep Learning". In: Hrsg. von P. Grohs und G. Kutyniok. Cambridge University Press, 2022. Kap. The Modern Mathematics of Deep Learning. eprint: https://arxiv.org/abs/2105.04026.

[5] C. de Boor, K. Höllig und S Riemenschneider. *Box Splines*. Bd. 98. Applied Mathematical Sciences. Springer-Verlag, 1993. ISBN: 0387941010.

[6] Carl de Boor. *A Practical Guide To Splines Revised Edition*. Springer, 2001.

[7] Jason Brownlee. *Deep Learning for Computer Vision*. Machine Learning Mastery, 2018. URL: https://machinelearningmastery.com/machine-learning-with-python/.

[8] M. Buhmann und J. Jäger. *Quasi-Interpolation*. The Cambridge Monographs on Applied and Computational Mathematics. Cambridge University Press, 2022.

[9] Peter G. Casazza und Gitta Kutyniok, Hrsg. *Finite Frames*. Applied and Numerical Harmonic Analysis. Birkhäuser, 2013.

[10] P.G. Casazza u. a. "Constructing Tight Fusion Frames". In: *Appl. Comput. Harmon. Anal.* 30.2 (2011), S. 157 –187.

[11] Li Chen u. a. "Beyond human recognition: A CNN-based framework for handwritten character recognition". In: *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*. IEEE, 2015. DOI: 10.1109/acpr.2015.7486592.

[12] Ole Christensen. *An introduction to frames and Riesz bases*. Boston: Birkhäuser, 2003. ISBN: 3764342951.

[13] Joon Son Chung u. a. "Lip Reading Sentences in the Wild". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017. DOI: 10.1109/cvpr.2017.367.

[14] R. R. Coifman u. a. "Signal processing and compression with wavelet packets". In: *Progress in Wavelet Analysis and Applications, Proceedings of the International Conference Wavelets and Applications, Toulouse, 1992* (Jan. 1993), S. 77–93.

[15] Danilo Costarelli und Renato Spigler. "Approximation results for neural network operators activated by sigmoidal functions". In: *Neural Networks* 44 (2013), S. 101–106. DOI: 10.1016/j.neunet.2013.03.015.

[16] Danilo Costarelli und GIanluca Vinti. "Max-product neural network and quasi-interpolation operators activated by sigmoidal functions". In: *Journal of Approimation Theory* 209 (2016), S. 1–16.

[17] Jean-Pierre Crouzeix, Nadezda Sukhorukova und Julien Ugon. "Characterization Theorem for Best Polynomial Spline Approximation with Free Knots, Variable Degree and Fixed Tails". In: *Journal of Optimization Theory and Applications* 172.3 (2017), S. 950–964. DOI: 10.1007/s10957-016-1048-1.

[18] G. Cybenko. "Approximation by superpositions of a sigmoidal function". In: *Mathematics of Control, Signals, and Systems* 2.4 (1989), S. 303–314.

[19] R. DeVore und K. Scherer. "VARIABLE KNOT, VARIABLE DEGREE SPLINE APPROXIMATION TO X". In: *Quantitative Approximation*. Elsevier, 1980, S. 121–131. DOI: 10.1016/b978-0-12-213650-4.50016-7.

[20] Ronald DeVore. *Constructive approximation*. Berlin New York: Springer-Verlag, 1993. ISBN: 0387506276.

[21] Ronald A. DeVore. "Nonlinear approximation". In: *Acta Numerica* (1998), S. 51–150.

[22] S.P. Diliberto und E.G. Straus. "On the approximation of a function of several variables by the sum of functions of fewer variables". In: *Pacific J. Math* 1 (1951), S. 195 –210.

[23] M. Dyer, Z. Fredi und C. McDiarmid. "Volumes spanned by random points in the hypercube". In: *Random Structures & Algorithms* 3 (1992), S. 91 –106.

[24] Massimo Fornasier, Jan Vybíral und Ingrid Daubechies. "Robust and resource efficient identification of shallow neural networks by fewest samples". In: *Information and Inference: A Journal of the IMA* 10.2 (2021), S. 625–695. DOI: 10.1093/imaiai/iaaa036.

[25] Yuly Makovoz George G. Lorentz Manfred v. Golitschek. *Constructive Approximation: Advanced Problems*. Grundlehren der mathematischen Wissenschaften. Sp, 1996.

[26] Ian Goodfellow, Yoshua Bengio und Aaron Courville. *Deep Learning*. Hrsg. von Thomas Dietterich. MIT Press Ltd, Nov. 2016. ISBN: 9780123743701. URL: http://www.deeplearningbook.org.

[27] Constantin Greif, Philipp Junk und Karsten Urban. "Linear/Ridge Expansions: Enhancing Linear Approximations by Ridge Functions". In: *ACOM* 48.15 (Juni 2022).

[28] Constantin Greif und Philipp Johannes Junk. *Linear-Ridge-Expansions*. online verfügbar. MATLAB-Software. 2021. URL: https://github.com/pjjunk/linear-ridge.

[29] Constantin Greif und Karsten Urban. "Decay of the Kolmogorov N-width for wave problems". In: *Applied Mathematics Letters* 96 (2019), S. 216–222.

[30] Philipp Grohs u. a. *α-Molecules: Curvelets, Shearlets, Ridgelets, and Beyond*. Forschungsber. 2013-24. Seminar für Angewandte Mathematik ETH Zürich, Juli 2013.

[31] Philipp Grohs u. a. "Deep Neural Network Aproximation Theory". In: *CoRR* (2019). URL: http://arxiv.org/abs/1901.02220.

[32] Jiuxiang Gu u. a. "Recent advances in convolutional neural networks". In: *Pattern Recognition* 77 (2018), S. 354–377. DOI: 10.1016/j.patcog.2017.10.013.

[33] K. Guo, G. Kutyniok und D. Labate. "Sparse multidimensional representation using anisotropic dilation and shear operators". In: *Wavelets and Splines, Nashboro Press* (2006), S. 189–201.

[34] B. Haasdonk. "Model Reduction and Approximation: Theory and Algorithms". In: Hrsg. von P. Benner u. a. Philadelphia: SIAM, 2016. Kap. Reduced Basis Methods for Parametrized PDEs - A Tutorial Introduction for Stationary and Instationary Problems.

[35] B. Haasdonk. *Reduced Basis Methods for Parametrized PDEs - A Tutorial*. Hrsg. von P. Benner u. a. SIAM, 2017.

[36] Kaiming He u. a. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: (Feb. 2015). arXiv: 1502.01852 [cs.CV].

[37] Donald O. Hebb. *The Organization of Behavior*. New York: Wiley & sons, 1949.

[38] Tomaso Poggio Hrushikesh Mhaskar Qianli Liao. "Learning Functions: When Is Deep Betten Than Shallow". In: *CBMM Memo* 045 (2016).

[39] Stéphane Mallat Joakim Andén Vincent Lostanlen. "Classification with Joint Time-Frequency Scattering". In: *IEEE Transactions on Signal Processing* 67.14 (2019), S. 3704–3718.

[40] Andrew B. Kahng. "AI system outperforms humans in designing floorplans for microchips". In: *Nature* 594.7862 (2021), S. 183–185. DOI: 10.1038/d41586-021-01515-9.

[41] Patrick Kidger und Terry Lyons. "Universal Approximation with Deep Narrow Networks". In: *Proceedings of Machine Learning Research*. Hrsg. von Jacob Abernethy und Shivani Agarwal. Bd. 124. 33rd Annual Conference on Learning Theory, 2020, S. 1–22.

[42] A. Kolmogorov. "Über die beste Annäherung von Funktionen einer gegebenen Funktionenklasse". In: *Annals of Mathematics* 37.1 (1936), S. 107–110.

[43] Rainer Kress. *Linear Integral Equations*. Springer-Verlag GmbH, Dez. 2013. 412 S. ISBN: 9781461495932. URL: https://www.ebook.de/de/product/23082779/rainer_kress_linear_integral_equations.html.

[44] R.P. Kulkarni. "Approximate Solution of multivariable Integral Equations of the second kind". In: *Journal of Integral Equations and Applications* 16.4 (2004).

[45] Gitta Kutyniok und Demetrio Labate, Hrsg. *Shearlets*. Applied and Numerical Harmonic Analysis. Birkhäuser, 2012. ISBN: 9780817683153.

[46] Gitta Kutyniok u. a. "A Theoretical Analysis of Deep Neural Networks and Parametric PDEs". In: *Constructive Approximation* (2021).

[47] Yann LeCun. *The MNIST database of handwritten digits*. http://yann.lecun.com/exdb/mnist/. abgerufen zuletzt 30.01.2019. 1998. URL: http://yann.lecun.com/exdb/mnist/.

[48] Yann LeCun u. a. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Computation* 1.4 (1989), S. 541–551.

[49] Bo Li, Shanshan Tang und Haijun Yu. "Better Approximations of High Dimensional Smooth Functions by Deep Neural Networks with Rectified Power Units". In: *Commun. Comput. Phys.* 27.2 (Feb. 2020), S. 379 –411.

[50] W.A. Light und E.W. Cheney. *Approximation Theory in Tensor Product Spaces*. LNM 1169. Berlin: Springer-Verlag, 1985.

[51] Vitaly Maiorov und Allan Pinkus. "Lower bounds for approximation by MLP neural networks". In: *Neurocomputing* 25.1-3 (1999), S. 81–91. DOI: 10.1016/s0925-2312(98)00111-8.

[52] S. G. Mallat. *A wavelet tour of signal processing : the sparse way*. Amsterdam Boston: Elsevier/Academic Press, 2009. ISBN: 9780123743701.

[53] Stéphane Mallat. "Group Invariant Scattering". In: *Communications on Pure and Applied Mathematics* 65.10 (2012), S. 1331–1398.

[54] Warren McCulloch und Walter Pitts. "A logical calculus of the ideas immanent in nervous activity". In: *Bulletin of Mathematical Biophysics* 5 (1943), S. 115 –133.

[55] Hrushikesh Mhaskar. "Neural Networks for optimal approximation of smooth and analytic functions". In: *Neural Computation* 8.1 (1996), S. 164–177.

[56] Xiuyuan Cheng Mia Xu Chen Stéphane Georges Mallat. "Unsupervised Learning by Deep Scattering Contractions". In: *Conference on Neural Information Processing Systems (NIPS)* (2014).

[57] Guido Montúfar u. a. “On the Number of Linear Regions of Deep Neural Networks”. In: (2014).

[58] E.A. Nadaraya. *Nonparametric Estimation of Probability Densities and Regression Curves*. Springer Netherlands, Dez. 1988. 228 S. ISBN: 9027727570. URL: https://www.ebook.de/de/product/4203328/nadaraya_nonparametric_estimation_of_probability_densities_and_regression_curves.html.

[59] Andrea Olsson. *Particle swarm optimization : theory, techniques, and applications*. Hauppauge, N.Y: Nova Science Publishers, 2011. ISBN: 9781613247006.

[60] David Ouyang u. a. “Video-based AI for beat-to-beat assessment of cardiac function”. In: *Nature* 580.7802 (2020), S. 252–256. DOI: 10.1038/s41586-020-2145-8.

[61] J.M. Pena und T. Sauer. “On the multivariate Horner scheme”. In: *SIAM Journal of Numerical Analysis* 37.4 (2000), S. 1186 –1197.

[62] Philipp Petersen und Felix Voigtlaender. “Optimal approximation of piecewise smooth functions using deep ReLU neural networks”. In: *Neural Networks* 108 (2018), S. 296–330.

[63] Pencho P. Petrushev. “Approximation by ridge functions and neural networks”. In: *SIAM Journal on Mathematical Analysis* 30.1 (1998), S. 155–189.

[64] A. Pinkus. “Surface Fitting and Multiresolution Methods”. In: Hrsg. von A. Le Méhauté, C. Rabut und L.L. Schumaker. Vanderbild University Press, Nashville, TN, 1997. Kap. Approximating By Ridge Functions, S. 1 –14. ISBN: 0826512941.

[65] Allan Pinkus. “Approximation theory of the MLP model in neural networks”. In: *Acta Numerica* (1999), S. 143–195.

[66] Allan Pinkus. *Ridge Functions*. Cambridge Tracts in Mathematics 205. Cambridge University Press, 2015.

[67] M.J.D. Powell. *Approximation theory and methods*. Cambridge University Press, 1981.

[68] Claudio Procesi. “Splines and index theorem”. In: *Bulletin of Mathematical Sciences* 40.1 (Juni 2012), S. 1–67. DOI: 10.1007/s13373-011-0013-4.

[69] A. Quateroni, A. Manzoni und F. Negri. “Reduced basis methods for partial differential equations: An introduction”. In: *Springer, Cham* (2016).

[70] Prajit Ramachandran, Barret Zoph und Quoc V. Le. “Searching for Activation Functions”. In: (Okt. 2017). arXiv: 1710.05941 [cs.NE].

[71] Richard G. Baraniuk Randall Balestriero. “A Spline Theory of Deep Networks”. In: *Proceedings of the 35th International Conference on Machine Learning, Stockholm, Sweden, PLMR 80* (2018).

[72] George G. Lorentz Ronald A. DeVore. *Constructive Approximation*. Grundlehren der mathematischen Wissenschaften 303. Springer, 1993.

[73] F. Rosenblatt. *The Perceptron – a perceiving and recognizing automaton*. Techn. Ber. 85-460-1. Cornell Aeronautical Laboratory, 1957.

[74] Itay Safran und Ohad Shamir. "Depth-Width Tradeoffs in Approximating Natural Functions with Neural Networks". In: (2017). URL: arXiv:1610.09887v2.

[75] Issai Schur. "Bemerkungen zur Theorie der Beschränkten Bilinearformen mit unendlich vielen Veränderlichen". In: *Journal für die reine und angewandte Mathematik* 140 (1911).

[76] Uri Shaham, Alexander Cloninger und Ronald R. Coifman. "Provable approximation properties for Deep neural networks". In: *Applied and Computational Harmonic Analysis* (2016). DOI: http://dx.doi.org/10.1016/j.acha.2016.04.003.

[77] Karen Simonyan und Andrew Zisserman. "Very Deep Convolutional Networks for Large-Scale Image Recognition". In: (Sep. 2014). arXiv: 1409.1556 [cs.CV].

[78] E.D. Sontag. "VC Dimension of Neural Networks". In: *Neural Networks and Machine Learning* (1998), S. 69–95.

[79] Matus Telgarsky. "Neural Networks and Rational Functions". In: *Proceedings of the 34th International Conference on Machine Learning, Sydney, Australia*. 2017.

[80] Matus Telgarsky. "Representation Benefits ofDeep Feedforward Networks". In: *eprint* (2015). eprint: arXiv:1509.08101v2. URL: arXiv:1509.08101v2.

[81] Inc. The MathWorks. *Deep Learning Toolbox (TM)*. R2020b. 2020. URL: https://de.mathworks.com/help/pdf_doc/deeplearning/index.html.

[82] Oriol Vinyals und et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *Nature* 575.7782 (2019), S. 350–354. DOI: 10.1038/s41586-019-1724-z.

[83] Felix Voigtlaender. "The universal approximation theorem for complex-valued neural networks". In: (Dez. 2020). arXiv: 2012.03351 [math.FA].

[84] Thomas Wiatowski und Helmut Bölcskei. "A Mathematical Theory of Deep Convolutional Neural Networks for Feature Extraction". In: (2016).

[85] Dominic Yang, Prisanna Balaprakash und Sven Leyffer. "Modelling design and control problems involving neural network surrogates". In: *Computational Optimization and Applications* 83 (2022), S. 759–800. DOI: https://doi.org/10.1007/s10589-022-00404-9.

[86] Dmitry Yarotsky. "Elementary superexpressive activations". In: (Feb. 2021). arXiv: 2102.10911 [cs.NE].

[87] Dmitry Yarotsky. "Error bounds for approximations with deep ReLU networks". In: (Okt. 2016). arXiv: 1610.01145 [cs.LG].

[88] Dmitry Yarotsky. "Optimal approximation of continuous functions by very deep ReLU networks". In: *COLT 2018*. 2018. URL: arXiv:1802.03620v2.

[89] Dmitry Yarotsky. "Quantified advantage of discontinuous weight selection in approximations with deep neural networks". In: (Mai 2017). arXiv: 1705.01365 [cs.NE]. URL: arXiv:1705.01365v1.

[90] Q. Zhang und A. Benveniste. "Wavelet networks". In: *IEEE Transactions on Neural Networks* 3.6 (1992), S. 889–898. DOI: 10.1109/72.165591.

[91] Xuan Zhang u. a. "AlignedReID: Surpassing Human-Level Performance in Person Re-Identification". In: (Nov. 2017). arXiv: 1711.08184 [cs.CV].

[92] Ding-Xuan Zhou. "Deep distributed convolutional neural networks: Universality". In: *Analysis and Applications* 16.6 (2018), S. 895–919. DOI: 10.1142/S0219530518500124.