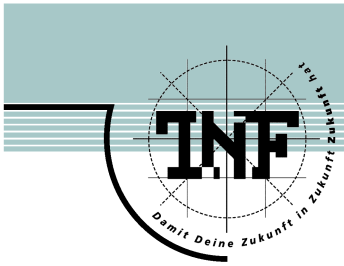


JOHANNES KEPLER
UNIVERSITÄT LINZ
Netzwerk für Forschung, Lehre und Praxis

Iterative Regularization and Training of Neural Networks

Diplomarbeit

zur Erlangung des akademischen Grades

Diplomingenieur

in der Studienrichtung

Technische Mathematik

Eingereicht von

Andreas Hofinger

Angefertigt am *Institut für Industriemathematik*

Betreuung:

o.Univ.-Prof. Dipl.-Ing. Dr. Heinz W. Engl
Dipl.-Ing. Dr. Martin Burger

Linz, Februar 2003

The author dedicates his efforts on this work to his friend
Rainer Nobis (February 11, 1976 – July 8, 2002).

Zusammenfassung

Angefangen bei der Psychologie, die sich Einblick in menschliches Verhalten erhofft, über Biologie und Medizin, bis hin zur Informatik, die Lösungsansätze für Probleme wie Bild- und Spracherkennung sucht, gibt es viele unterschiedliche Motivationen für die Untersuchung neuronaler Netze. Eine besondere Stärke dieser Netze ist es, dass sie mit Hilfe von Beispielen *trainiert* werden können (Kapitel 1).

Eine mathematische Analyse dieses Trainingsprozesses zeigt, dass sowohl im linearen Fall ([BE00] bzw. Kapitel 2.1) als auch im Fall von flexiblen Gewichten ([BBEH02] bzw. Kapitel 2.2), das Lernen eines neuronalen Netzes ein *schlecht gestelltes Problem* ist. Dies bedeutet, dass beliebig kleine Fehler in den Eingabedaten (den Trainingspatterns) zu beliebig großen Fehlern in der Ausgabe (den zu optimierenden Parametern) führen können. Um dennoch vernüftige Resultate erhalten zu können, müssen Regularisierungsverfahren angewendet werden.

In [BN03] und [BN01] wurden *Tikhonov-artige Regularisierungsmethoden* untersucht. Die häufig verwendeten Zugänge "weight decay" und "output smoothing", sowie eine Kombination dieser beiden Verfahren wurden analysiert, für alle drei Methoden konnten Parameterauswahlmethoden entwickelt werden, welche die bislang verwendeten Heuristiken ersetzen können (Kapitel 3).

Ziel dieser Arbeit ist es, *iterative Regularisierungsmethoden*, also Verfahren wie die nichtlineare Landweberiteration oder das Newtonverfahren in Kombination mit einem Abbruchkriterium zu untersuchen. Nach kurzer Untersuchung zeigt sich, dass die "Nichtlinearitätsbedingung", ein hinreichendes Konvergenzkriterium für das Landweberverfahren, nicht erfüllt sein kann, sobald das Netzwerk aus zwei oder mehr Knoten besteht (Kapitel 4.3). Dies ist ein unerwartetes Resultat, denn der vielverwendete Algorithmus *Backpropagation* ist ein naher Verwandter des Landweberverfahrens (Kapitel 4.2).

Im Gegensatz dazu können wir zeigen, dass Landweber- und Newtonverfahren angewandt auf *einzelne* Knoten lokal konvergieren (Kapitel 4.4 bis 4.6). Aufbauend auf diesem Resultat wird in Kapitel 5 ein *Greedy Algorithmus* entwickelt, der die Größe des Netzes Schritt für Schritt um jeweils *einen* Knoten erhöht. Dieser Algorithmus behält die guten Approximationseigenschaften von neuronalen Netzen bei (insbesondere die dimensionsunabhängige Konvergenzrate) und ermöglicht uns gleichzeitig zur Bestimmung der einzelnen Knoten das Landweber- bzw. das Newtonverfahren zu verwenden.

Abschließend werden die theoretischen Resultate numerisch anhand von Beispielen sowohl für "ridge-construction"- als auch für "radial basis function"-Netzwerke verifiziert (Kapitel 6).

Abstract

Starting with psychology, which is interested in a deeper insight to human behaviour, over biology and medicine, to informatics, which is searching for efficient methods for solving problems such as picture- or speech-recognition, there are many different motivations, for the investigation of neural networks. One strength of these networks is, that they can be *trained* using examples.

A mathematical analysis of this training process shows, that in the linear case ([BE00] or Section 2.1) as well as in the case of flexible weights ([BBEH02] or Section 2.2) this process is an *ill-posed problem*. This means, that arbitrary small errors in the input data (the training patterns) can lead to arbitrary large errors in the output (the parameters to be optimized). Methods that can cope with this problem are called *regularization methods*.

In [BN03] and [BN01] *Tikhonov-type regularization methods* have been investigated. The frequently used approaches "weight decay" and "output smoothing", as well as a combination of these two methods have been analyzed. For all three methods parameter choice rules were developed, which can replace heuristics used so far (Chapter 3).

The aim of this thesis is to investigate *iterative regularization methods*, i. e., methods such as the nonlinear Landweber iteration or Newton's method combined with a stopping rule. Short investigation shows, that the "nonlinearity condition", a sufficient condition for convergence of Landweber's method cannot be fulfilled, as soon as the network consist of two or more nodes (Section 4.3). This result is unexpected, since the widely used algorithm *backpropagation* is a near relative to Landweber's method (Section 4.2).

In contrast to this result we can show that Landweber's and Newton's method converge locally when they are applied to *single* nodes (Sections 4.4 to 4.6). Based on this result in Chapter 5 a greedy algorithm is developed, which increases the size of the network step by step by *one* node each. This algorithm preserves the good approximation properties of neural networks (especially the dimension independent convergence rate) and enables us at the same time to utilize Landweber's and Newton's method to find the individual nodes.

Finally the theoretical results are verified numerically by examples for "ridge-construction"- as well as "radial basis function"-networks (Chapter 6).

Dankesworte

Zuerst möchte ich mich bei meiner Familie bedanken, die mich während meines ganzen Studiums unterstützt hat, und die mir zu Hause ein angenehmes, ungestörtes Arbeitsklima geboten hat.

Besonders möchte ich hier meinen Bruder Markus erwähnen der immer ein interessierter Zuhörer war und ihm auf diesem Wege wünschen, dass er diesen Wissensdrang auch in Zukunft beibehält.

Meinen Betreuern Martin Burger und Prof. Heinz Engl danke ich dafür, dass sie mir die Gelegenheit gegeben haben unter ihrer Anleitung diese Arbeit zu verfassen und mir gleichzeitig ermöglichten meinen eng gesteckten Zeitrahmen einzuhalten. Bei dieser Gelegenheit möchte ich auch Martin alles Gute für seinen Aufenthalt in den USA wünschen.

Einen Dank möchte ich auch all jenen Studienkollegen aussprechen die geduldig meinen oft langwierigen Ausführungen gelauscht haben und mir dadurch ermöglicht haben selbst einen besseren Einblick in die Materie zu bekommen.

Abschließend will ich noch Petra erwähnen, die immer bemüht war einen Einblick in meine (doch sehr theoretische) Arbeit zu bekommen und hoffe dass sie auch in Zukunft ein solches Interesse an meinem Berufsleben hat.

Diese Arbeit wurde vom “Fonds zur Förderung der wissenschaftlichen Forschung” im Rahmen des Spezialforschungsbereichs F013, “Numerical and Symbolic Scientific Computing”, unterstützt.

Contents

Chapter 1

Introduction

In the first chapter we will present some of the basic and most important facts about neural networks.

Our starting point is a brief introduction to the invention and development of neural networks and their biological background ([KKN96], [Köh90], [MRS91], [BE92]).

Next we explain by an example how a boolean-valued network can be built using a simple model of nerve cells ([Köh90]). In combination with the preceeding section, this gives an idea of processes such as picture recognition in biological systems.

From this biological point of view we turn to a more mathematical one. A boolean-valued network describes a parameterized input-output relation and can for this reason be used to approximate boolean functions. For approximation of real-valued functions some modifications have to be made. Due to this step, we will lose the connection to processes in biological systems, nevertheless we will keep some notations like *nodes*, *training* and so on, which are common in the literature on neural networks.

First we generalize the ideas for boolean-valued networks to a formula for real valued networks and present different architectures that may be used to approximate functions ([BE00], [BBEH02], [GJP95], [Köh90]). The specific choice of the architecture depends on the type of functions that shall be approximated.

Then the property of limited completeness is investigated. This means that any continuous function can be approximated arbitrarily well using a neural network, provided the size of the network is large enough. This property,

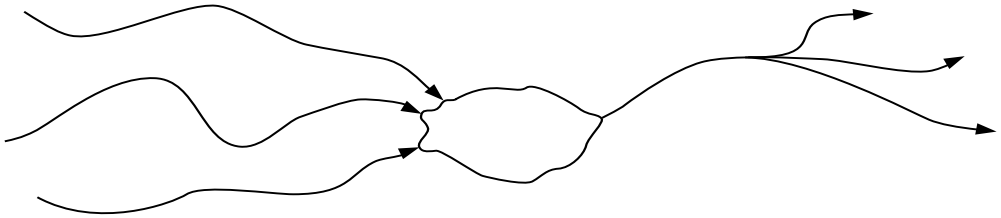


Figure 1.1: A simple neuron

which a large class of network architectures has ([HSW89], [GP90], [GJP95]), is a fundamental requirement for being able to use neural networks for function approximation.

Finally, we investigate the interesting approximation properties of neural networks. As we shall see, by using neural networks a *dimension independent* convergence rate can be achieved ([DS96], [Bar93], [BN03]), whereas in linear function approximation schemes the rate of convergence slows down with increasing spatial dimension of the problem ([NG99], [Pin85], [Lor66]).

1.1 Biological Background

The investigation of artificial neural networks started already 1940 and was motivated by the interest in the neuro-physiological fundamentals of the human brain ([KKN96], [Köh90], [MRS91]). A cooperation between different disciplines like biology, psychology, mathematics and informatics lead to models of processes like picture- or speech-recognition.

It was known that the brain consists of a vast amount of nerve cells - so called *neurons*, each of them being connected with up to several thousands of other neurons. These neurons affect each other by electrical signals which are transported through nerve fibres.

Each neuron has various input channels (dendrites) and one output channel (axon). This output channel can split up into several branches (Figure 1.1) and therefore influence many other neurons.

The structure of the the brain is assumed to be organized hierarchically as we explain briefly with an example (a more detailed explanation can be found e. g. in [BE92, Chapter 3.3]):

> By means of the optic nerve one layer of neurons is connected directly with the sensors in the retina of the human eye and each neuron reports if there is a signal at the corresponding sensor or not. Each of these neurons is connected with many other neurons from another layer, which perform boolean operations on the input signal. For instance a combination of five neurons can realize the XOR-function (see Section 1.2). This structure can recognize if two adjacent sensors have a different input signal. Combining the output of such small networks, a detector for edges can be built, more precisely a detector for *one alignment* of an edge, occuring at a *specific area* in the field of vision. Again, the output of such detectors for specific lines can be combined to form a structure that recognizes for instance arbitrary horizontal lines. In a next step detectors for different angles (acute, right, obtuse) can be formed and so forth, finally followed by layers that recognize letters, words and sentences.

This model of the process of picture recognition has been well confirmed by D. H. Hubel and T. N. Wiesel, who demonstrated the existence of various detectors described above and received the nobel prize in 1981.

The enormous parallelization is the reason why we are able to see in real time even though the response time of neurons is in the range of milliseconds.

1.2 Boolean-valued Functions

In a simple model the input channels can act by inhibiting or activating the neuron. If their weighted sum is above some threshold, the neuron gives output value 1, otherwise 0. Using such simple neurons, boolean-valued functions can be built ([Köh90, Chapter 4.4]). The XOR-function (exclusive OR) takes two input values, and gives output value 1 if exactly one of the input values is equal to 1, otherwise the output is set to zero. This function can be realized, using 5 (artificial) neurons (cf. Fig. 1.2).

To clarify the meaning of the parameters in the picture we explain the behaviour of the network for a specific input signal:

> The input vector shall be given as $x_1 = 1$ and $x_2 = 0$. Since x_1 is larger than the threshold 0.5 and x_2 is smaller than 0.5, the lower left neuron will produce the output signal +1, the lower right will produce output signal 0 respectively.

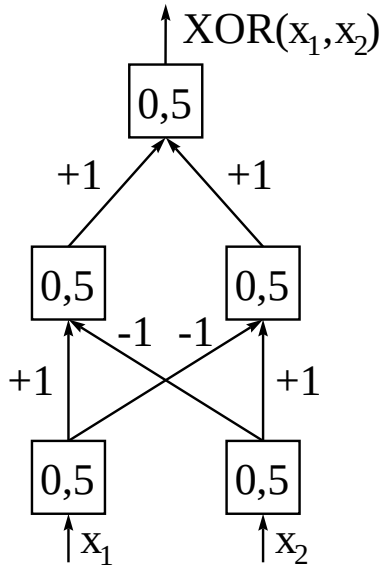


Figure 1.2: Neural network representing the XOR-function. The values inside the nodes represent the threshold of each neuron

The input value for the left neuron in the middle layer is the weighted sum of the outputs from the neurons below, this is $+1$ times the output from the lower left neuron plus -1 times the output from the lower right one and is therefore given as $+1$. This value is again larger than the threshold 0.5 and so this neuron produces the output signal $+1$. The input value for the right neuron can be computed as -1 which is below the threshold, and so no output signal (or better: output signal zero) is produced.

The last neuron, which is also called *output neuron* multiplies both values from the neurons below with $+1$, adds them up, and since the result $(+1)$ is above its threshold it produces output value $+1$. Altogether the input vector $(x_1, x_2) = (1, 0)$ is transferred to the scalar output value $+1$ which is the desired output.

Figure 1.2 motivates the following definition for the parts of a neural network. A boolean neural network consists of:

An input layer (or sensors), preprocessing the input signal and transforming it into a boolean vector.

An output layer (in Fig. 1.2 this is only a single neuron), collecting the output signals from the layers below and converting the weighted sum

into a boolean value.

One or more hidden layers (or processing units). These are called hidden, because they (in terms of real world neurons) have no contact to the outer world, neither their input nor their output channels can be observed from outside.

Furthermore, such a network is called a *feed-forward* network since information only travels in one direction, e. g. in the picture from bottom to top.

In mathematical terms, the neural network in Figure 1.2 can be written as

$$\mathrm{XOR}(x_1, x_2) = \sigma\Big(\sum_{j=1}^{2} c_j \cdot \sigma\big(\sum_{i=1}^{2} w_{ij} \cdot \sigma(x_i - t_{0i}) - t_{1j}\big) - t_2\Big), \tag{1.1}$$

where the parameters t_{ij}, c_j and w_{ij} are given as

$$c = (+1, +1), \quad w = \begin{pmatrix} +1 & -1 \\ -1 & +1 \end{pmatrix}, \quad \text{and} \quad t_{0i} = t_{1i} = t_2 = 0.5\,.$$

The so-called *activation function* σ is commonly chosen either as the Heaviside function

$$\sigma(x) = \begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$$

or a smoothed (and differentiable) variant of it, e. g.,

$$\sigma(x) = \frac{1}{1 + e^{-x}}\,.$$

A plot of the latter function is given in Figure 1.4. This function has the convenient property that the derivative can be easily computed from the value of the function itself as $\sigma'(x) = \sigma(x)(1-\sigma(x))$. The popular backpropagation algorithm takes advantage of this property.

Both activation functions presented above belong to the class of *sigmoidal functions*, these are bounded, measurable functions that fulfill (cf. [Bar93])

$$\lim_{x \to -\infty} \sigma(x) = 0 \quad \text{and} \quad \lim_{x \to +\infty} \sigma(x) = 1\,. \tag{1.2}$$

The structure in (1.1) is of course highly nonlinear but in the next section we will make some simplifications to weaken this nonlinearity.

1.3 Real-valued Functions

For approximation of real-valued functions some modifications have to be made. The main goal of the input layer above was to transform the input vector into boolean values. This step of course has to be skipped, otherwise most of the information included in the input pattern will be lost. Therefore, the input neurons in the input layer are replaced by the identity function.

The task of the output neuron in the network above was to calculate the sum of the outputs from the hidden layer and transform this sum into a boolean value. In a real-valued neural network this conversion is also skipped, the output value is given directly as the weighted sum of the underlying neurons.

What remains is the hidden layer, the nonlinearity in this step is kept. We denote the so-called *activation function* ([Köh90], [MRS91]) in the jth neuron by Φ_j. This activation function is not necessarily a sigmoidal function, but can also be chosen differently (cf. Section 1.4).

In contrast to the boolean-valued network above, where three interlocking nonlinear functions were used, now there remains only one main nonlinearity. The output of such a (single-layer) real-valued network can be represented by the formula

$$\begin{aligned} f_k^j(x) &= \sum_{i=1}^{k} c_{ij}\Phi(x,t_i) \\ &\overset{e.g.}{=} \sum_{i=1}^{k} c_{ij}\sigma(a_i^T x - b_i), \quad \text{where} \quad t_i = (a_i, b_i) \end{aligned}$$

which corresponds to the general case of a neural network with vectorial output (such a network is shown in Figure 1.3). In the following we will only consider networks with scalar output and write f_k instead of f_k^j. Hence, the networks we will deal with have the form

$$f_k(x) = \sum_{i=1}^{k} c_i\Phi(x,t_i) \tag{1.3}$$

and fit into the scheme of feed-forward neural networks with one hidden layer and linear output layer.

This formula is of course far away from any process in the human brain, although we are able to motivate the approach by looking at biological processes. Especially different architectures, like e. g. multilayer networks can

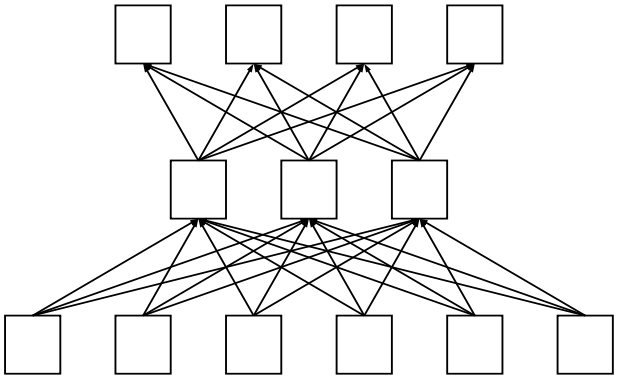


Figure 1.3: Neural network with one hidden layer and vectorial output.

be motivated by imitation of biological neural systems. Nevertheless, the connection that remains is at last not very strong and often overrated.

In Section 1.2 a specific neural network was constructed and it was demonstrated that this network realizes the XOR-function. The question arises how the parameters of a network (in the example c, w and t) can be determined if an arbitrary function shall be realized. This is a so-called *inverse problem*, the search for parameters that give a desired output (cf. e. g. [EHN96]). One possibility to find these parameters is the algorithm *backpropagation* (see Section 4.2). Here the parameters of the neural network are initialized using random values. The output of this "random network" is compared with the desired output and the parameters c and t are adjusted such that the residual decreases. This process, which is referred to as *training* in the neural network community, is a nonlinear parameter identification problem and is, as we shall see later, an ill-posed problem.

We emphasize that the variable parameters t_i appearing in the nonlinear function Φ in Formula (1.3) are not chosen a-priori, but determined by an optimization process, whereas when e. g. Fourier-series or standard splines are used, the approximating functions are fixed and the approximation is only done by adjusting the parameters c_i. As we shall see later, this results in better approximation capabilities, especially if the input space is high-dimensional.

The construction of real-valued networks, starting from boolean-valued ones motivates to choose the functions Φ as sigmoidal shaped function, but there are many different possibilities to choose the activation function as we shall see in the next section.

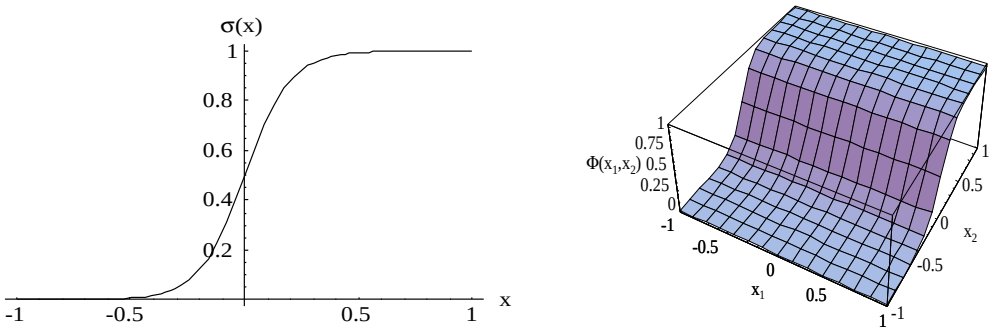


Figure 1.4: Plot of a sigmoidal function σ and a corresponding 2d-ridge-construction Φ for the choice $a = (0, 1)$ and $b = 0$.

1.4 Neural Network Architectures

In the following we review some common ways to choose the activation function inside a neural network. Usually within one network only one type of activation function Φ is used. This function Φ can be for instance (cf. [BE00]):

Ridge-Constructions: Here the activation function is given by

$$\Phi(x, t) = \sigma(a^T x - b), \quad \text{with} \quad t = (a, b)$$

where $a \in \mathbb{R}^n$ is a vector with equal dimension as the input x and $b \in \mathbb{R}$ is a scalar value. There is great freedom to choose the function σ as we shall see in Section 1.5. If σ is chosen as a sigmoidal function, then in the 2-dimensional case the resulting function looks like a ridge (see Figure 1.4).

Radial Basis Functions: In this case the activation function is radially symmetric, the function value depends only on the distance to a center point t, i. e.,

$$\Phi(x, t) = \Xi\left(\|x - t\|^2\right),$$

where the function Ξ is usually chosen to be some kind of peak function. A frequently used basis function is the Gaussian

$$\Xi(z^2) = \Xi_{\sigma_0}(z^2) = \frac{1}{\sqrt{2\pi}\sigma_0} e^{-z^2/(2\sigma_0^2)}, \qquad \sigma_0 \in \mathbb{R},\ \sigma_0 > 0$$

another common choice is the multiquadric function

$$\Xi(z^2) = \Xi_r(z^2) = \frac{\sqrt{r^2 + z^2}}{r}, \qquad r \in \mathbb{R}\backslash\{0\}.$$

Fuzzy Controllers: e. g. Sugeno Controllers can also be seen as neural networks. In [BBEH02] the activation function (*membership function*) is chosen as B-spline, where the knots are used as free parameters that can be adjusted to the particular function being approximated.

Multilayer Networks: These networks have more than one hidden layer. Due to the greater amount of parameters such a network is expected to have better approximation capabilities than a single-layer network, but also greater instabilities might arise during the training process. As we will see in Section 1.5 also very simple networks are universal approximators, therefore our analysis will include only single-layer networks, the investigation of multilayer networks is left for future work.

Several other possibilities for the choice of the activation function are given and motivated in [GJP95]. We will restrict ourselves in our analysis to the cases of radial basis functions and ridge constructions (for a detailed description of the assumptions we impose on Φ see Section 4.4.1).

The main effect of a specific activation function, is a change in the representation condition, which will occur later in Theorems 1.2 and 5.3. In brief, to show convergence rates it is assumed that the function to be approximated is an element of the range of some operator, where this range depends mainly on the activation function Φ. Therefore, there is no network structure that fits for all problems, but the network must be designed in dependence of the function (or class of functions) to be approximated. For instance for image or speech recognition multilayer networks seem to be a good choice (see e. g. [Köh90, Chapter 6]).

A disadvantage of neural networks is that the occuring parameters do not have a physical meaning. This does not matter if one is not interested in the actual values of these parameters, but only in the output of the network, for fitting to given data ("data driven model"). Especially for high-dimensional functions neural networks have better approximation capabilities than linear approximation schemes as we shall see in the next two sections.

1.5 Limited Completeness

The first important question we have to ask about neural networks is if they have the property of *limited completeness*, i. e., if we are able to approximate arbitrary functions with neural networks. Constructions that have this property are also called *universal approximators*.

Definition 1.1. A sequence of sets (X_n) is called *limited complete* if the closure of the infinite union $\bigcup_{n=1}^{\infty} X_n$ is equal to set of continuous functions $C[U]$.

The property of limited completeness is also referred to as *Weierstrass-property* since it is related to the Weierstrass approximation theorem.

Many types of neural networks have this property. To see this for $n \in \mathbb{N}$ we define the set

$$\begin{aligned} X_n = \big\{ f \in C[U] \,|\, f(x) = \sum_{i=1}^{n} c_i \sigma(a_i^T x + b_i), \\ U \subset \mathbb{R}^d, a_i \in \mathbb{R}^d, b_i, c_i \in \mathbb{R} \big\}, \end{aligned} \tag{1.4}$$

which is the set of all continuous functions that can be represented by a neural network of ridge construction type with n nodes. Observe that the definition of X_n yields a *monotonically increasing* sequence of sets. It can be shown that this sequence is limited complete in $C[U]$ for many choices of σ (cf. [GP90], [HSW89], [Bar93]). For instance if σ is chosen from the class of sigmoidal functions (see equation (1.2)) then X_n is limited complete. Also if σ is such that its mean value is not zero and it is contained in each space L_p for $p < \infty$, i. e.,

$$\int_{\mathbb{R}} \sigma(x)\, dx \neq 0 \quad \text{and} \quad \|\sigma\|_{L^p} < \infty\,, \quad \forall p \in [1, \infty[\,,$$

the property of limited completeness can be proven. Hence, as long as σ is chosen properly and the number of nodes is large enough, any continuous function f can be approximated arbitrarily well with a neural network of ridge construction type.

Under weak assumptions on the activation function Ξ also radial basis function networks generate limited complete sets. Several activation functions that fulfill such assumptions are presented in [GP90] and [GJP95].

Note that all these results can be transferred from neural networks to standard function approximation schemes and vice versa. In standard schemes

the parameters t are fixed, whereas in neural networks the parameters t may vary. Nevertheless, in the limit, i. e., if the number of nodes tends to infinity, also in the case of standard schemes all values for t are obtained and therefore the same approximation properties are gained. A short example is given to clarify this:

> We define the set $X_1 = \{c\Phi(\cdot, t) \mid c \in \mathbb{R},\, t \in [0,1]\}$ which belongs to a neural network with one node. Associated to this set we define for $i \leq j$ the sequence of sets $Y_{1,i,j} = \{c\Phi(\cdot, t) \mid c \in \mathbb{R},\, t = \frac{i}{j}\}$ which belongs to a linear approximation scheme. The union of these sets tends to the set X_1. Analogously we can find a mapping between X_2 and another sequence $Y_{2,i,j}$ and so on. Of course the sequence of Y is much longer than the sequence of X. Nevertheless, if we compute the closure of the infinite union for both, we find the same set. Therefore the sequence $Y_{k,i,j}$ is limited complete if and only if the sequence X_k is.

The property of Limited Completeness is *not* an indicator for the quality of an approximation scheme, since it gives no information on the quality of approximations done with *finitely many* nodes. Furthermore, we have seen that both, neural networks and linear approximation schemes enjoy this property.

In the example above we have seen that we need more nodes in a standard scheme than in a neural network to gain the same approximation quality. In the next section we will see that especially if the input space is of high dimension, neural networks can outperform the standard schemes.

1.6 Function Approximation

As we have seen above, many approximation schemes have the property of limited completeness. Therefore, more interesting than the question *whether* a sequence of sets is limited complete, is the question *how fast* functions can be approximated. In this context the first theorem shows that neural networks are well suited for the approximation of high-dimensional functions. This result is a version of a lemma attributed to B. Maurey ([DS96]). For the case of ridge-constructions with sigmoidal activation function, it has been shown in [Bar93], another version of this theorem will be presented in Chapter 5. The theorem shows that we can obtain a *dimension independent* convergence rate if we approximate proper functions with neural networks.

We assume that the parameters t occuring in formula (1.3) are restricted to a compact set P. The function f to be approximated is defined on the domain $\Omega \subset \mathbb{R}^n$. This function has to fulfill an integral representation, depending on the activation function which is used in the network.

Theorem 1.2. *Let the set of parameters P be compact and Φ be an element of $C(P, H^s(\Omega))$. Moreover, let f fulfill the representation*

$$f = \int_P h(t)\Phi(\cdot, t)dt$$

for some function $h \in L^1(P)$. Then there exists a sequence of functions f_k of the form

$$f_k = \sum_{i=1}^{k} c_i^k \Phi(\cdot, t_i^k)$$

such that the approximation rate

$$\|f - f_k\|^2_{H^s(\Omega)} = O(k^{-1}).$$

holds.

Proof. The proof of the Theorem for this special setting is given in [BN03]. For a more general setting a proof of this theorem can be found in Chapter 5 in Theorem 5.3. How the results in there can be applied to neural networks is discussed later in Section 5.4. □

The bound we have gained in this Theorem is very interesting because the rate of convergence does not depend on the dimension n of the input space. From the theory of n-widths (cf. e. g. [NG99], [Pin85], [Lor66]) it is known that if we approximate a function of n variables and smoothness s in the usual linear way, then in the worst case the approximation error goes to zero as $O(k^{-\frac{s}{n}})$, i. e., depends on the dimension of the input space. For this reason in Theorems 2.3 and 2.4 below, which belong to the linear case, only a dimension dependent convergence rate can be proven.

For high-dimensional problems such as speech- or picture-recognition where n is very large (several thousands, i. e., the number of values in the discretized frequency spectrum or the number of pixels), such a rate is of course very poor and using neural networks will be a much better choice.

In the next section we take a different point of view, and interpret the training of a neural network as a nonlinear ill-posed parameter-identification problem.

Chapter 2

Training Neural Networks with Noisy Data as an Ill-Posed Problem

In this chapter we investigate two reasons for the ill-posedness of training a neural network. The first one applies for both, neural networks and linear approximation schemes, namely that with increasing amount of approximating functions (or nodes) the approximation becomes more sensitive to errors in the data. This type of error can be regularized if the network size is chosen properly in dependence of the noise level as we will explain in Section 2.1.3.

The second reason only applies for neural networks and originates in the nonlinearity of this approximation scheme. The parameters t_i can be chosen arbitrarily from some compact set P, in particular there is no restriction for the minimal distance between two parameters. Therefore it is possible that two nodes are situated at the same place, but with opposite sign and annihilate each other. As a consequence the set of functions that can be represented by a neural network is *not closed*, even for a finite, fixed number of nodes, and hence the training process is an ill-posed problem (see Section 2.2.1 or [BBEH02]).

This type of error can be controlled using special regularization techniques. We will briefly present several techniques ([GJP95, GP90, LS95, Bis93, Bis95, MHL92, BN03, BN01]) in Section 2.3. In Chapter 3 we will discuss special Tikhonov-type regularization methods. The main focus will be Chapter 4 where we investigate *iterative regularization methods*. We present several results for two important iterative regularization methods, namely Landweber iteration, which is closely related to the backpropagation algorithm, and

Newton's method, and apply these results in Chapter 5 to derive a greedy algorithm for the training of neural networks.

2.1 Increasing Number of Nodes

In this section we turn to the first reason for the ill-posedness of training a neural network – the approximations become more sensitive to errors in the data if the number of nodes increases. We investigate the influence of the number of nodes if the parameters t_i are *fixed*, therefore these results can also be transferred to the case where the function is not approximated with a neural network but with a linear approximation scheme (e. g. Fourier series). For this analysis we follow the lines of [BE00] (cf. also [Sar73], [Gro80]).

First of all it is shown that training neural networks is equivalent to least-squares collocation for a corresponding integral equation with mollified data, if the weights inside the nodes of the network are not optimized but chosen a-priori.

Convergence results for least-squares collocation (see e. g. [NW74, Eng83, EHN96]) can be used to prove a dimension dependent convergence rate for neural networks. This rate cannot be improved to a dimension independent one, since we are not looking at usual neural networks but at a linearized version.

Finally the influence of noise in the data is investigated. We will see that convergence and stability can be obtained if the number of nodes n is chosen properly in dependence of the noise level δ.

2.1.1 Connection to Integral Equations

We assume that $\Phi \in C^{2s}(\Omega \times P)$, where the domain $\Omega \subset \mathbb{R}^d$ and the set of parameters $P \subset \mathbb{R}^p$ are bounded domains ([BE00]). Let $L_s : H^s(\Omega) \to L^2(\Omega)$ denote the linear operator with

$$\|g\|_{H^s(\Omega)} = \|L_s g\|_{L^2(\Omega)}, \qquad \forall g \in H^s(\Omega).$$

Under the above smoothness assumptions we may define a continuous function $k \in C(\Omega \times P)$ by

$$k(x,t) := L_s^* L_s \Phi(x,t)$$

where $L_s^* : L^2(\Omega) \to H^s(\Omega)$ is the adjoint of L_s and the elliptic differential operator $L_s^* L_s$ is evaluated with respect to x (the operator is elliptic with respect to the L_2-scalar product and the H^s-norm). Furthermore we define an integral operator by

$$\begin{aligned} \mathcal{F}_k : H^s(\Omega) &\to L^2(P) \\ g &\mapsto \int_\Omega k(x,\cdot) g(x)\, dx. \end{aligned}$$

Note that all results in this section depend on the choice of the smoothness-parameter s. A different choice of s affects the operator L_s and consequently the integral opterator $\mathcal{F}_k$. For the choice $s = 0$ we obtain convergence results for approximation with respect to the L^2-norm.

Using this setting, the (linear) function approximation problem can be connected with solving an integral equation via least squares collocation, as can be seen in the next lemma ([BE00, Lemma 2.2]):

Lemma 2.1. *Let f_n be the unique minimizer of the approximation problem*

$$\|f - f_n\|_{H^s(\Omega)} = \min_{g_n \in \operatorname{span}\{\Phi(\cdot,t_j)\}} \|f - g_n\|_{H^s(\Omega)}$$

for fixed t_j, $j = 1, \ldots, n$. Then f_n is the minimum-norm solution of

$$(\mathcal{F}_k f_n)(t_j) = y_j, \quad j = 1, \ldots, n \tag{2.1}$$

with y_j defined as

$$y_j := \int_\Omega f(x) k(x, t_j)\, dx.$$

We can now interpret the neural network approximation procedure with fixed nodes as a two-step algorithm:

1. The mollified data function

   $$y := \mathcal{F}_k f \in C(P)$$

   is computed.

2. The first-kind integral equation

   $$y(t) = \int_\Omega k(x,t) g(x)\, dx, \quad t \in P \tag{2.2}$$

   is solved for g approximately via least-squares collocation in (2.1).

2.1.2 Convergence Results

Results about least-squares collocation can now be applied to neural networks. The parameters t_i are not optimized in this setting, therefore we will not obtain dimension independent convergence rates.

Results are known in dependence of the distance of the parameters which is measured via

$$\Delta_n := \sup_{t \in P} \inf_{i=1,\dots,n} \|t - t_i\| .$$

We start with some results for the noise-free case. The first theorem ([BE00, Theorem 3.1.(1)]) is dealing with convergence of the approximations f_n to f in $H^s(\Omega)$ for $\Delta_n \to 0$.

Theorem 2.2. *Let $f \in H^s(\Omega)$ such that there exists $h \in L^2(P)$ with*

$$f(x) = \int_P \Phi(x,t) h(t)\, dt \tag{2.3}$$

(i. e., $f \in \mathrm{R}(\mathcal{F}_k^*) \subset \mathrm{N}(\mathcal{F}_k)^\perp$), then $f_n \to f^\dagger$ for $\Delta_n \to 0$.

Here $f^\dagger$ denotes the minimum-norm solution of (2.2). If the activation function Φ fulfills a smoothness condition then convergence rates can be obtained ([BE00, Proposition 3.4]):

Theorem 2.3. *Let $P \subset \mathbb{R}^1$ and f as above, $\Phi \in C^{2s+2m}(\Omega \times P)$ then*

$$\|f_n - f\|_{H^s(\Omega)} \leq c \Delta_n^{s+m}.$$

Using Theorem 2.3 the approximation quality can be estimated in dependence of the network size n. In general, $n \sim h^{-p}$ points are required to cover a domain $P \subset \mathbb{R}^p$ such that $\Delta_n = \mathcal{O}(h)$. Hence, we obtain the relation

$$\Delta_n = \mathcal{O}\left(n^{-1/p}\right)$$

for the number of parameters and their distance. This results in a dimension dependent convergence rate, decreasing with increasing p ([BE00, Theorem 3.6]):

Theorem 2.4. *Let $P \subset \mathbb{R}^p$ be a rectangle, $\Phi \in C^{s+m}(\Omega \times P)$ and $f \in H^s(\Omega)$ be a function such that (2.3) holds for some $h \in L^2(P)$. If $\{(c_i, t_i)\}_{i=1,\dots,n}$ are such that*

$$\|f - f_n\|_{H^s(\Omega)} = \min_{\{(c_i,t_i)\}} \|f - \sum_{i=1}^{n} c_i \Phi(x, t_i)\|_{H^s(\Omega)} ,$$

then the asymptotic estimate

$$\|f - f_n\|_{H^s(\Omega)} = \mathcal{O}\left(n^{-m/p}\right)$$

holds.

In the next section we turn to the problem of stability with respect to data errors.

2.1.3 Stability

Now we investigate the behaviour of the approximation scheme for noisy data. All results are based on the following theorem ([BE00, Proposition 4.1]):
Theorem 2.5. *Let Ψ_n denote the matrix*

$$\Psi_n = \left(\langle \Phi(\cdot, t_i), \Phi(\cdot, t_j)\rangle_{H^s(\Omega)}\right)_{i,j=1,\dots,n}$$

and σ_n its minimal (nonzero) eigenvalue. Then the estimate

$$\left\|f_n - f_n^\delta\right\|_{H^s(\Omega)} \leq c\frac{\sqrt{n}\delta}{\sqrt{\sigma_n}} \tag{2.4}$$

holds. This estimate is sharp, i. e., there exists a noise-pattern such that equality holds.

For increasing n the matrix Ψ_n becomes more and more ill-conditioned. Therefore a stable behaviour of the approximations when noise is present can only be obtained if the network size is bounded properly.

Observe that a problem occurs if the parameters t_i are not fixed but variable, because the matrix Ψ_n becomes singular (ill-conditioned) if two values t_i and t_j coincide (or are close, respectively). Thus, for variable t_i instable behaviour may arise as soon as the network consists of two or more nodes. We will find this problem again later.

Using Theorem 2.5 a parameter-choice rule for n can be derived ([BE00, Theorem 4.2]):
Corollary 2.6. *Let $\Phi \in C^{2s}(\Omega \times P)$ and f fulfill (2.3). If $n = n(\delta)$ is chosen such that $n\delta^2/\sigma_n \to 0$ and if $\Delta_{n(\delta)} \to 0$ then*

$$f_{n(\delta)}^\delta \to f \text{ in } H^s(\Omega)$$

This theorem is a direct consequence of Theorems 2.2 and 2.5 and the triangle inequality.

Similarly the next result is obtained ([BE00, Theorem 4.3]).
Corollary 2.7. *Let* $\Phi \in C^{2s+2m}(\Omega \times P)$ *and* f *fulfill* (2.3). *Then the estimate*

$$\left\| f - f_n^\delta \right\|_{H^s(\Omega)} \leq c_1 \Delta_n^{s+m} + c_2 \frac{\sqrt{n}\delta}{\sqrt{\sigma_n}} \tag{2.5}$$

holds.

If $2s > d$ then all these estimates hold with $\|\cdot\|_{C(\bar{\Omega})}$ instead of $\|\cdot\|_{H^s(\Omega)}$, due to continuous embedding.

2.2 Flexible Choice of Weights

Now we turn to the second reason why training a neural network is an ill posed problem — the parameters t_i are not chosen a priori but they are varied during the optimization process. It is therefore possible that two parameters t_i and t_j coincide. If this happens all standard optimization algorithms like Landweber iteration and the corresponding backpropagation algorithm or Newton's method can encounter difficulties, as will be shown in Chapter 4. In Chapter 5 a greedy algorithm is presented, to overcome this problem.

First of all we look at an example, which is related to the optimization of fuzzy controllers to demonstrate this effect. Then we briefly introduce the concept of the *best approximation property* and present one Tikhonov-type regularization technique, which is due to Girosi, Jones and Poggio (cf. [GP90] and [GJP95]) and which ensures this property.

2.2.1 An Example of Ill-posedness

We consider the problem of training a neural network with finitely many data, this means for a given data set (x_i, y_i), $i = 1, \ldots, m$ we look for a solution of the least squares minimization problem (cf. [BBEH02])

$$\sum_{i=1}^{m} \left(y_i - \sum_{j=1}^{2} c_j \Phi_j(x_i, \mathbf{t}) \right)^2 \to \min, \quad (c_1, c_2, \mathbf{t}) \in \mathbb{R}^2 \times [0,1]^4.$$

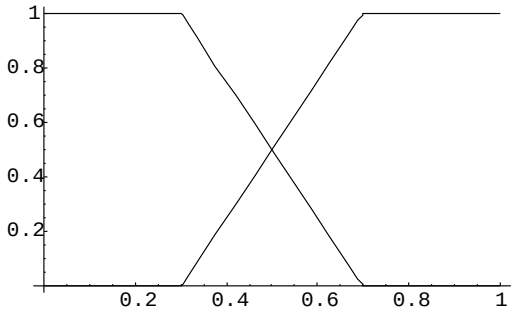


Figure 2.1: Activation functions from Example 2.2.1. The function decreasing from the left to the right is $\Phi_1(x,t)$.

This is a nonlinear parameter identification problem for the values (c_1, c_2) and $\mathbf{t} = (t_1, \ldots, t_4)$. As activation functions Φ_j we choose (see Figure 2.1)

$$\Phi_1(x,\mathbf{t}) = \begin{cases} 1 & x \le t_2 \\ \frac{t_3 - x}{t_3 - t_2} & t_2 < x < t_3 \\ 0 & t_3 \le x \end{cases} \qquad \text{and}$$

$$\Phi_2(x,\mathbf{t}) = 1 - \Phi_1(x,\mathbf{t})\,.$$

If the input is given as $(x_i, 0)$, $i = 1, \ldots, m$ then the solution for the weights c_1 and c_2 is given *uniquely* as $(c_1, c_2) = (0, 0)$. The parameters t_i might be chosen arbitrarily to solve the least squares problem, but if we look for a least squares *minimum norm* solution, then also these parameters are determined *uniquely* and are given as $\mathbf{t} = \mathbf{0}$.

Now we inspect the effect of noise on this parameter identification problem. Therefore we construct a sequence of parameters and choose $t_1^k = 0$, $t_2^k = k^{-3}$, $t_3^k = 2k^{-3}$ and $t_4^k = 1$. Further we set $c_1^k = k$ and $c_2^k = 0$. The steps 2 to 6 of this sequence are shown in Figure 2.2.

The sequence $f^k = c_1^k \Phi_1(x; t^k) + c_2^k \Phi_2(x; t^k)$ converges to zero in $L^2([0,1])$, but the sequence (c_1^k, c_2^k) has no bounded subsequence. Hence for k sufficiently large the function f^k is arbitrary close to 0 and can be interpreted as a noisy version of the 0-function, nevertheless at the same time c_1^k is arbitrarily far away from 0, which would be the corresponding solution.

There are various possibilities how this example can be transferred to the case of ridge constructions. The straightforward way would be to choose a sequence of parameters t such that the ridge leaves the domain Ω and increasing c at the same time. Nevertheless, such a behaviour could be

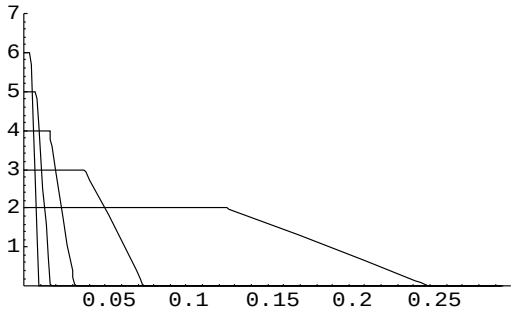


Figure 2.2: A sequence that converges to 0 in $L^2[0,1]$. The broadest function is f^2, the smallest one is f^6.

prohibited by restricting the parameters t to a sufficiently small set P. If σ is differentiable another possibility is to choose two nodes such, that their weights a and b are almost equal, and the factors c have same magnitude but opposite sign, i. e., they almost annihilate each other. We construct the sequence $f^k = c^k \left(\sigma(a^T x + b) - \sigma(a^T x + b + k^{-2})\right)$. For the choice $c^k = k$ we obtain $f^k \to 0$ and $c^k \to \infty$ for $k \to \infty$.

Thus, for variable parameters t_i the parameter-estimation problem for c_i and t_i is ill-posed, even for a network consisting of finitely many nodes.

2.2.2 Best-approximation Property

One motivation for the use of regularization techniques is the *best-approximation property*, which is usually not fulfilled for neural networks.
Definition 2.8. A set $X_n \subseteq X$ is said to have the best-approximation property if for any $f \in X$ there exists some f_0 in X_n such that $\|f - f_0\| = \inf_{f_n \in X_n} \|f - f_n\|$, i. e., the infimum is attained. ([GP90]).

Feed forward neural networks (as defined in (1.3)) do not have the best approximation property, because the set

$$X_n = \Big\{ \sum^{n} c_i \Phi(x, t_i) \mid c_i \in \mathbb{R},\, t_i \in P \Big\}$$

is not closed. Using a similar idea as in the section before, a sequence can be constructed that converges to the derivative of Φ, which is not an element of X_n (see [GP90]). Let us now first look at the following problem (cf. [GJP95]):

We minimize the modified error functional

$$\sum_{i=1}^{m} (f(x_i) - y_i)^2 + \lambda \Psi(f) \tag{2.6}$$

where Ψ is a smoothness functional in such a way, that lower values of the functional correspond to smoother functions. This can be realized by defining Ψ in the form

$$\Psi(f) := \int_{\mathbb{R}^d} \frac{|\hat{f}(s)|^2}{\hat{G}(s)}\, ds\,,$$

where $\hat{f}$ indicates the Fourier transform of f and $\hat{G}$ is some positive function that tends to zero as $\|s\| \to \infty$. In such a case $1/\hat{G}$ acts as a high-pass filter. If $\hat{G}$ is symmetric then the solution of (2.6) can be computed explicitly and has the form ([GJP95])

$$f(x) = \sum_{i=1}^{m} c_i G(x - x_i).$$

The nonlinearity seems to have vanished, since the solution f is represented by a linear combination of functions $G_i(x) = G(x - x_i)$, which only depend on the choice of the points x_i, but not on the function that shall be approximated. The function G is the Green's function of some (pseudo) differential operator (see [GP90]).

So the addition of the penalty term in (2.6) yields, that the solution of the modified minimization problem is in the set

$$T^m = \Big\{ \sum^{m} c_i G_i(x) \mid c_i \in \mathbb{R} \Big\}\,,$$

which is closed and therefore has the best approximation property ([GP90]).

In the next section a short overview of reasons for regularization is given and some methods are presented.

2.3 Regularization Methods

First we give several motivations for using regularization techniques in the training of neural networks. The first two are connected to the two types of ill-posedness we have presented above. We also present two other motivations which belong to the behaviour during the optimization process and the smoothness of the approximation.

Noise in the data (in x_i and y_i)

Best-approximation property: As we have mentioned in the section before, this property can be achieved by adding a penalty term to the error-functional.

Overtraining: The approximation can become worse if too many optimization steps ("training cycles") are performed (see [LS95]).

Generalization: Reliable estimation of functions in areas where only few data are given is desired.

Several regularization methods have been used and investigated in the training of neural networks, but not all of them in a mathematically sound way:

Output smoothing and weight decay: are two Tikhonov-type regularization methods. We will take a closer look at them in Chapter 3

Curvature driven smoothing: is a Tikhonov-type method using the curvature of the approximating function as penalty term ([Bis93]).

Training with noise: Here the network is not trained with the function f itself, but with several noisy variants of it. It has been shown by Bishop (cf. [Bis95]), that such a procedure has same effect as output smoothing with the H^1-norm.

Network-architecture: As we have seen above in Section 2.2.2, the choice of the activation-functions Φ_j can also be motivated by looking at the properties of the Fourier transform when conferred to as a highpass-filter. This has been done in [GP90] and [GJP95].

Early stopping: Means that the search for the minimum is stopped after a given number of iterations to avoid overtraining. This behaves similarly to a special iterative Tikhonov-regularization ([LS95]).

Restriction of network size n: This is the method which is the easiest to implement, but it has been only investigated for the linear case yet (cf. e. g. Theorems 2.2 and 2.3). In the nonlinear case various difficulties occur, for instance the *effective* number of nodes, i. e., the ones that contribute to the model, can be less than the total network size as was shown by Moody, Hanson and Lippmann in [MHL92].

Our main focus in the remaining parts will be on iterative regularization techniques. We will investigate Landweber's and Newton's method in detail in Chapter 4. Before, we present some results about Tikhonov regularization, which are due to Burger and Neubauer ([BN03], [BN01]).

Chapter 3

Tikhonov Regularization

In this chapter we apply the well-known method of Tikhonov regularization to the nonlinear problem of training a neural network. An introduction to Tikhonov regularization of general nonlinear problems can be found in [EHN96, Chapter 10]. We present some results due to Burger and Neubauer (see [BN03] and [BN01]).

First of all we need some assumptions on the activation function Φ and the function f to be approximated. The major assumption will be an integral representation for f, which is difficult to verify in general. We present some results about convergence and convergence rates for special Tikhonov-type regularization methods. Finally, we present a sufficient smoothness condition on f, guaranteeing the existence of an integral representation.

3.1 Introduction

In [BN03], two different Tikhonov-type regularization methods have been investigated, namely output smoothing and weight decay. In the method of output smoothing the penalty term measures the smoothness of the network function f_n, whereas in weight decay the norm of c_i is kept small. A combination of both has been studied too. The resulting minimization functionals are:

Output smoothing:

$$\min_{\{(c_i,t_i)\}\in(\mathbb{R}\times P)^n} \|\sum_{i=1}^{n} c_i\Phi(\cdot,t_i) - f^\delta\|^2_{L^2(\Omega)} + \alpha\|\sum_{i=1}^{n} c_i\Phi(\cdot,t_i) - f_*\|^2_{H^s(\Omega)}$$

Weight decay:

$$\min_{\{(c_i,t_i)\}\in(\mathbb{R}\times P)^n} \|\sum_{i=1}^{n} c_i\Phi(\cdot,t_i) - f^\delta\|^2_{L^2(\Omega)} + \beta n \sum_{i=1}^{n} c_i^2$$

Combination of output smoothing and weight decay:

$$\min_{\{(c_i,t_i)\}\in(\mathbb{R}\times P)^n} \|\sum_{i=1}^{n} c_i\Phi(\cdot,t_i) - f^\delta\|^2_{L^2(\Omega)} + \alpha\|\sum_{i=1}^{n} c_i\Phi(\cdot,t_i) - f_*\|^2_{H^s(\Omega)} + \beta n\sum_{i=1}^{n} c_i^2 \quad (3.1)$$

Since X_n is usually not weakly closed, the first problem might not have a solution. To overcome this problem, one does not look for an exact solution $f_{n,\alpha}$, but for an approximate solution $f^\mu_{n,\alpha}$ which fulfills the minimality condition up to an error of μ. Such an approximate solution always must exist by definition of the infimum.

For the analysis the following assumptions are needed:

(A1) The parameter set $P \subset \mathbb{R}^p$ is compact.

(A2) The activation function Φ is in the space $C(P; H^s(\Omega))$, $s \in \mathbb{N}$.

(A3) The function $f \in H^s(\Omega)$, $s \in \mathbb{N}$ has a representation

$$f = \int_P \Phi(\cdot,t)h(t)\,dt \quad \text{for some} \quad h \in L^1(P). \quad (3.2)$$

Some of the results can be improved using the slightly stronger assumptions:

(A2') The activation function Φ is Hölder continuous with respect to the second parameter with Hölder-exponent ρ, i. e.,

$$\left\|\Phi(\cdot,t) - \Phi(\cdot,\tilde{t})\right\|_{H^s(\Omega)} \le c\left\|t-\tilde{t}\right\|^\rho, \quad \rho \in (0,1].$$

(A3') The function h in (A2) is in $L^2(P)$.

For instance, under the stronger assumptions rates of approximation can be improved from $\mathcal{O}(n^{-\frac{1}{2}})$ to $\mathcal{O}(n^{-\frac{1}{2}-\frac{\rho}{p}})$. In the following two typical results using the weaker conditions (A1) - (A3) are presented, to give an idea of possible results. For the sake of brevity, we only review the results for the combined approach (3.1).

3.2 Combination of Output Smoothing and Weight Decay

The first result we present is dealing with convergence in the H^s-norm ([BN03, Theorem 4.1]):

Theorem 3.1. *Let $f^\delta \in L^2(\Omega)$ be such that $\|f - f^\delta\| \leq \delta$ and let assumptions (A1) - (A3) be fulfilled. Moreover, let $\alpha = \alpha(n, \delta)$ and $\beta = \beta(n, \delta)$ be chosen such that*

$$\alpha \to 0, \quad \delta^2 \alpha^{-1} \to 0, \quad n\alpha \to \infty, \quad \beta \to 0, \quad \beta\alpha^{-1} \to 0$$

as $n \to \infty$ and $\delta \to 0$. Then $f^\delta_{n,\alpha,\beta} \to f$ in $H^s(\Omega)$.

Here $f^\delta_{n,\alpha,\beta}$ denotes a solution of the minimization problem (3.1). To prove convergence rates in $H^s(\Omega)$ we have to impose an additional smoothness assumption on f, namely the source condition

$$f - f_* \in \mathcal{R}((E^*E)^\nu), \quad \text{for some } 0 < \nu \leq \frac{1}{2}. \tag{3.3}$$

In our case E denotes the embedding operator from $H^s(\Omega)$ into $L^2(\Omega)$. The need for a source condition is typical for ill-posed problems, but the selection of E may differ, for instance for general operator equations $F(x) = y$, one needs the operator $F'(x^\dagger)$ instead of E in condition (3.3) (cf. [EHN96, Theorem 10.7]). Under this additional assumption also convergence rates in $H^r(\Omega)$ for $r < s$ can be shown ([BN03, Theorem 4.2]):

Theorem 3.2. *Let f satisfy (3.3) and f^δ be such that $\|f - f^\delta\| \leq \delta$ holds. Moreover let assumptions (A1) - (A3) be fulfilled. If the regularization parameters are chosen such that*

$$\alpha \sim (n^{-\frac{1}{2}} + \delta)^{\frac{2}{1+2\nu}} \quad \text{and} \quad \beta \sim (n^{-1} + \delta^2),$$

then

$$\|f^\delta_{n,\alpha,\beta} - f\|_{H^r(\Omega)} = \mathcal{O}(n^{-\frac{1}{2}} + \delta)^{1 - \frac{r}{s(1+2\nu)}}.$$

Observe that, in order to obtain optimal convergence rates, it is necessary to choose the size of the network properly in dependence of the noise level.

3.3 Existence of an Integral Representation

In [BN01] a result guaranteeing the existence of the integral representation (A3) is given. The following theorem is valid for neural networks built by

ridge constructions, i. e.,

$$X_n = \Big\{ g = \sum_{j=1}^{n} c_j \sigma(a_j^T x + b_j) \mid a_j \in A \subset \mathbb{R}^d, b_j \in B \subset \mathbb{R} \Big\},$$

where σ is a piecewise continuous, monotonically increasing function of sigmoidal form. It connects the existence of an integral representation with decay properties of the Fourier transform of f (such an assumption has been used directly to prove approximation results in [Bar93]):

Theorem 3.3. *Let* $A := \overset{d}{\underset{i=1}{\times}} [-a_i, a_i]$ *and* $B := [-b, b]$ *such that*

$$\forall a \in A\,, \forall x \in \Omega : \; |a^T x| \leq b - 1.$$

Let f *be such that*

$$(1 + |\cdot|)\hat{f}(\cdot) \in L^1(\mathbb{R}^d) \quad \textit{and} \quad (1 + |\cdot|^{3+\alpha-1/p})\hat{f}(\cdot) \in L^p(\mathbb{R}^d)$$

for some $\alpha > 0$. *Then* f *has an integral representation of the form* (3.2) *for some* $h \in L^p(A \times B)$.

Using Theorem 3.3 the assumptions (A3) and (A3') can now be transferred into a more interpretable form. For the case $p = 1$, the condition $(1+|\cdot|)\hat{f}(\cdot) \in L^1(\mathbb{R}^d)$ is superfluous, since it is implied by condition $(1+|\cdot|^2)\hat{f}(\cdot) \in L^1(\mathbb{R}^d)$. This condition actually means that f has a C^2-extension into the exterior of Ω. For the case $p = 2$, the conditions in Theorem 3.3 mean that f has a C^1-extension into the exterior of Ω and that f may be extended to a function in $H^{\frac{5}{2}+\alpha}(\mathbb{R}^d)$ for some $\alpha > 0$.

Chapter 4

Iterative Regularization Methods

Now we turn our attention to iterative regularization methods. We derive results about Landweber's and Newton's method and provide some fundamental results which we will need for the development of the greedy algorithm in Chapter 5.

We first introduce Landweber's and Newton's method. For this sake we will have to formulate the problem of training a neural network as a nonlinear operator equation.

Next we show that the widely used backpropagation algorithm and Landweber's method are equivalent. The only difference is that backpropagation is designed for a finite number of data, whereas in our context Landweber's method is used for continuous data. This connection is very interesting since we can show that Landweber's as well as Newton's method encounter problems as soon as the network consists of more than one node, which is always the case in practice. As a direct consequence, also backpropagation and Newton-type modifications (as proposed e. g. in [LS95]) can fail. This problem is hidden in the fact that the initial values for the backpropagation algorithm are chosen as random values ([Pao89, p.127], [Köh90, p.84]) – usually if nothing is known about the function to be approximated one would expect the zero vector as initial value, but this is one of the choices that cause troubles for the backpropagation algorithm.

A possible method to overcome this problem is a greedy algorithm, which will be presented in detail in Chapter 5. As we shall see below, in this algorithm the size of the network is increased step by step by one neuron,

and hence, we only have to solve problems with single nodes. Consequently it is important that Landweber's and Newton's method converge as long as the network consists of only one node. The proof of this property is quite long and therefore divided into several parts. Section 4.4 contains preparations for this proof and consists itself of several parts. The main convergence result for Landweber's and Newton's method is given in Section 4.5 and 4.6 respectively.

4.1 Description of Landweber's and Newton's Method

The summation formula of a neural network (1.3) can be associated with a nonlinear operator F, mapping the parameter values $(c_1, t_1, \ldots, c_k, t_k) \in (\mathbb{R} \times P)^k$, with compact P, into an infinite dimensional function space like e. g. $H^1(\Omega)$. This operator is given by

$$\begin{array}{rll} F: & (\mathbb{R} \times P)^k & \to H^1(\Omega), \\ & (c_1, t_1, \ldots, c_k, t_k) & \mapsto \sum_{i=1}^k c_i \Phi(x, t_i), \end{array} \tag{4.1}$$

or in the case of a single node as

$$F(c, t) = c\Phi(x, t).$$

We will sometimes use the abbreviation $p = (c_1, t_1, \ldots, c_k, t_k)$. The derivative of F with respect to the parameters is given by

$$F'(c_1, t_1, \ldots, c_k, t_k) = \begin{pmatrix} \Phi(x, t_1) \\ c_1 \nabla_t \Phi(x, t_1) \\ \vdots \\ \Phi(x, t_k) \\ c_k \nabla_t \Phi(x, t_k) \end{pmatrix}. \tag{4.2}$$

If we interpret F as an operator from $(\mathbb{R} \times P)^k$ to $L^2(\Omega)$ we can compute the adjoint $F'(c_1, t_1, \ldots, c_k, t_k)^*$, $F'^* : L^2(\Omega) \to (\mathbb{R} \times P)^k$ as

$$F'(p)^* v = \int_\Omega F'(p) v(x) d\, x = \int_\Omega \begin{pmatrix} \Phi(x, t_1) \\ c_1 \nabla_t \Phi(x, t_1) \\ \vdots \\ \Phi(x, t_k) \\ c_k \nabla_t \Phi(x, t_k) \end{pmatrix} v(x) d\, x \tag{4.3}$$

where $v \in L^2(\Omega)$. In these two definitions we used an abbreviation for the derivative of Φ with respect to t. Since each parameter t_i may itself be a vector $t_{i1}, \ldots, t_{in}$ also this derivative is usually a vector and given by

$$\nabla_t \Phi(x,t) = \begin{pmatrix} \frac{\partial \Phi(x,t)}{\partial t_1} \\ \vdots \\ \frac{\partial \Phi(x,t)}{\partial t_n} \end{pmatrix}.$$

For deriving the results in Sections 4.5 and 4.6 we also need the second derivative, i. e., the Hessian matrix given by

$$\nabla_t\big(\nabla_t \Phi\big)(x,t) = \begin{pmatrix} \frac{\partial^2 \Phi(x,t)}{\partial t_1 \partial t_1} & \cdots & \frac{\partial^2 \Phi(x,t)}{\partial t_1 \partial t_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 \Phi(x,t)}{\partial t_n \partial t_1} & \cdots & \frac{\partial^2 \Phi(x,t)}{\partial t_n \partial t_n} \end{pmatrix}.$$

Landweber's method (cf. [EHN96, Chap. 11.1]) is a gradient method for minimizing the quadratic functional

$$f(p) := \frac{1}{2} \left\| y - F(p) \right\|^2 .$$

The iteration is defined via

$$p_{k+1} = p_k + F'(p_k)^*(y - F(p_k)), \quad k \in \mathbb{N}, \tag{4.4}$$

where p_0 is some initial guess for the solution $p^\dagger$. A criterion to ensure local convergence of Landweber's method is the following *nonlinearity condition*

$$\left\| F(\tilde{p}) - F(p) - F'(p)(\tilde{p} - p) \right\| \leq \eta \left\| F(\tilde{p}) - F(p) \right\|, \qquad \eta < \frac{1}{2}, \tag{4.5}$$

also called *tangential cone condition* (cf. [EHN96, Chap. 11]). This estimate has to hold for $\tilde{p} - p$ sufficiently small. If this condition is fulfilled it can be shown (cf. [HNS95]) that the distance of the iterates p_k and the solution p_* *decreases monotonically* if the starting point p_0 is chosen sufficiently close to p_*. Note, that $F(p)$ has to be equal to $F(\tilde{p})$ for all $\tilde{p}$ such that $\tilde{p} - p$ is in the nullspace of $F'(p)$, if (4.5) holds. For neural networks with two or more nodes this is not the case and we are able to construct examples where the distance to the optimal parameter, i. e., $p_k - p^\dagger$ increases when Landweber's method is applied (see Section 4.3).

Newton's method is motivated by linearizing the equation $F(p) = y$ and solving the resulting linearized problem. This leads to the equation

$$F'(p_k)(p_{k+1} - p_k) = y - F(p_k),$$

which is solved via

$$p_{k+1} = p_k + F'^{-1}(p_k)(y - F(p_k)). \tag{4.6}$$

Since this procedure is itself an ill-posed problem it has to be regularized (see e. g. [EHN96, Chap. 11.2] or [Kal97]). In the *Levenberg-Marquardt method* the iterations are computed as

$$p_{k+1} = p_k + (F'(p_k)^* F'(p_k) + \alpha_k I)^{-1} F'(p_k)^*(y - F(p_k)),$$

where the parameter $\alpha_k \to 0$ is a regularization parameter for Newton's method, and not for the (perhaps also ill-posed) problem $F(p) = y$.

Another regularized version of Newton's method is the *iteratively regularized Gauß-Newton method*, where the iterations are defined via

$$p_{k+1} = p_k + \big(F'(p_k)^* F'(p_k) + \alpha_k I\big)^{-1}\big(F'(p_k)^*(y - F(p_k)) + \alpha_k(p_0 - p_k)\big).$$

When we discuss convergence of Newton's method in the succeeding sections we will always mean this variant of it. A criterion to ensure local convergence of this Newton type method is condition (11.27) in [EHN96]:

$$F'(\tilde{p}) = R(\tilde{p}, p)F'(p) + Q(\tilde{p}, p) \tag{4.7a}$$

$$\|I - R(\tilde{p}, p)\| \le C_R \tag{4.7b}$$

$$\|Q(\tilde{p}, p)\| \le C_Q \left\|F'(p^\dagger)(\tilde{p} - p)\right\| \tag{4.7c}$$

In Theorem 4.19 we will show that this condition is fulfilled locally for a neural network consisting of a single neuron.

4.2 Backpropagation and Landweber Iteration

In this section we will show that the backpropagation algorithm (in the case of batch learning) tends to Landweber's method for the function approximation problem as the amount of data increases.

For the definition of backpropagation we refer to the monograph by Bishop [Bis96, Chapter 4.8]. Therefore we will change our notation for the moment and use the notation given there. The network discussed in [Bis96, Section 4.8.2] has the form

$$y_k = \sum_j w_{kj} g(a_j) \quad \text{where} \quad a_j = \sum_i w_{ji} x_i \,.$$

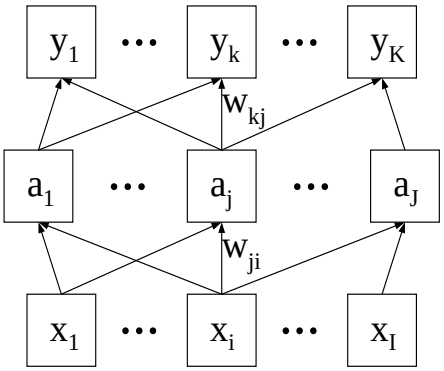


Figure 4.1: Neural network in the setting of Bishop [Bis96]. Some of the lines have been omitted to clarify the picture.

Here x is the input pattern, w_{ji} denotes the weights in the jth unit of the hidden layer and w_{kj} denotes the weights in the kth unit of the output layer (see Figure 4.1). This network has vectorial output, y_k is the kth component of an output vector y. Since we only discuss scalar neural networks here, we may write y and w_j instead of y_k and w_{kj} respectively.

In the backpropagation algorithm first an input pattern x is presented to the network. The network response yields an output y, which is compared with the corresponding target t. The difference $y - t$ is denoted as δ. This value δ is on the one hand used to adjust ("train") the weights w_j, on the other hand it is propagated backwards through the network to obtain values δ_j for the hidden units in the network. The values δ_j are given as (cf. (4.42) in [Bis96] for scalar output)

$$\delta_j = w_j \sigma'(a_j)\delta.$$

Having computed the values for δ_j and δ we can either update the corresponding values of w_{ji} and w_j directly after presentation of each pattern (on-line learning), or after summation over the derivatives for all the patterns in the training set (batch learning).

For the case of on-line learning we get two updates

$$\Delta w_j = -\eta\delta\sigma(a_j) \quad \text{and} \quad \Delta w_{ji} = -\eta\delta_j x_i = -\eta w_j \sigma'(a_j)\delta x_i\,.$$

In the case of batch learning the updates are summed up (we assume that

we have N data points), and the new updates are computed by

$$\Delta w_j = -\eta \sum_{n=1}^{N} \delta^n \sigma(a_j^n) \quad \text{and} \quad \Delta w_{ji} = -\eta \sum_{n=1}^{N} \delta_j^n x_i^n = -\eta \sum_{n=1}^{N} w_j \sigma'(a_j^n) \delta^n x_i^n \,.$$

where the factor η is chosen in dependence of N as $\eta = \tilde{\eta}/N$.

Now we rewrite these equations using the setup of Section 4.1. The relations are given by

$$w_j \leftrightarrow c_j, \quad w_{ji} \leftrightarrow t_{ji} \quad \sigma(a_j) \leftrightarrow \Phi(x, t_j) \quad \sigma'(a_j) x_i \leftrightarrow (\nabla_t \Phi)_i \,.$$

We collect the values t_{ji} into a vector t_j and obtain

$$\begin{aligned} \Delta c_j &= -\eta \sum_{n=1}^{N} \Phi(x^n, t_j) \delta^n, \\ \Delta t_j &= -\eta \sum_{n=1}^{N} c_j \nabla_t \Phi(x^n, t_j) \delta^n. \end{aligned}$$

If the input patterns x^n, which are chosen to train the network, are equally distributed in the bounded domain Ω, then the term on the right hand side will converge to an integral for N tending to infinity. Furthermore, we replace $-\delta$ with its definition $y(x) - F(p)(x)$ and thus obtain

$$\begin{aligned} \Delta c_j &= \tilde{\eta} \int_\Omega \Phi(x, t_j) \big(y(x) - F(p)(x)\big) \, dx \\ \Delta t_j &= \tilde{\eta} \int_\Omega c_j \nabla_t \Phi(x, t_j) \big(y(x) - F(p)(x)\big) \, dx \,. \end{aligned}$$

If we compare this result with equation (4.4) we observe that these updates are exactly the same as the ones we get for a step of Landweber's method. If the data are not equally distributed along the domain Ω, we find a modified Landweber method with a weighted integral. Hence, in this case backpropagation corresponds to Landweber's method in a weighted L_2-space.

So far, we have shown that backpropagation corresponds to Landweber's method in the sense that the updates computed with backpropagation converge to those computed with Landweber's method for increasing amounts of data. In the next section we show that Landweber's method can fail to converge. For some initial values the iterates do not converge to the solution, independently of the distance between solution and initial value.

4.3 Convergence Problems for Multiple Noded Networks

In this section we construct two examples showing that Landweber's as well as Newton's method may encounter problems as soon as the network consists of more than one node. The reason for this is the freedom to choose the parameters t in dependence of the function being approximated. During the optimization process it can happen that two nodes come very close. If this happens both Landweber's and Newton's method can fail to reduce the error. This effect could be avoided by imposing a constraint on the minimal distance between two nodes, but with such a restriction we are not aware of any result guaranteeing a dimension independent rate of approximation.

Both examples are based on the fact that the derivative F' of the neural-network-operator F, as introduced in Section 4.1, has non–trivial null space. There exist vectors, such that their difference is in this null space but the difference of the corresponding function values is not an element of the null space of the adjoint F'^* (the precise statement can be found in Lemma 4.1).

4.3.1 Preliminaries

For the sake of simplicity we use the notation $p = (c_1, t_1, \ldots, c_n, t_n)$ for the parameters of the neural network operator F as defined in (4.1). For the examples for Landweber's and Newton's method, presented in Sections 4.3.2 and 4.3.3, the following lemma is fundamental:

Lemma 4.1. *Let $\Phi \in L^2(\Omega \times P)$ be a function such that there exist parameters t and $\tilde{t}$ with $\Phi(x,t)$ and $\Phi(x,\tilde{t})$ being linearly independent. Moreover, let $n \geq 2$. Then there exists a combination of parameters p and q such that*

- *the difference of p and q is an element of the null space of the network operator, i. e., $q - p \in N(F'(p))$, and*
- *the difference of the corresponding output values of F is not an element of the null space of the operator $F'(p)^*$, i. e., $F(q) - F(p) \notin N(F'(p)^*)$.*

Proof. First we choose two parameters t and $\tilde{t}$ such that $\Phi(x,t)$ and $\Phi(x,\tilde{t})$ are linearly independent and $\|\Phi(x,t)\| \geq \left\|\Phi(x,\tilde{t})\right\|$. To construct the example we choose the value of the first vector as $p = (0,t,0,t)$ and the value of the second vector as $q = (c,t,-c,\tilde{t})$. If we take the scalar product of the

difference $q - p$ with the vector $F'(p)$ we obtain

$$(q-p)F'(p) = (c, 0, -c, \tilde{t} - t) \cdot \begin{pmatrix} \Phi(x,t) \\ 0 \\ \Phi(x,t) \\ 0 \end{pmatrix} = c\Phi(x,t) - c\Phi(x,t) = 0,$$

and hence, $q - p$ is an element of the null space of $F'(p)$.

Now we show that $F(q) - F(p)$ is not an element of the null space of $F'(p)^*$. For this sake we compute $F'(p)^*(F(q) - F(p))$ (observe that $F(p) = 0$) which is given as

$$F'(p)^*(F(q) - F(p)) = \int_\Omega \begin{pmatrix} \Phi(x,t) \\ 0 \\ \Phi(x,t) \\ 0 \end{pmatrix} (c\Phi(x,t) - c\Phi(x,\tilde{t}))\, dx$$

$$= c \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \left\langle \Phi(x,t), \Phi(x,t) - \Phi(x,\tilde{t}) \right\rangle .$$

To prove that the scalar product in the second line is different from zero we distinguish two cases

- $\left\langle \Phi(x,t), \Phi(x,\tilde{t}) \right\rangle \leq 0$: In this case we can estimate

  $$\begin{aligned} \left\langle \Phi(x,t), \Phi(x,t) - \Phi(x,\tilde{t}) \right\rangle &= \langle \Phi(x,t), \Phi(x,t) \rangle - \left\langle \Phi(x,t), \Phi(x,\tilde{t}) \right\rangle \\ &\geq \langle \Phi(x,t), \Phi(x,t) \rangle > 0 . \end{aligned}$$

  The last term is strictly greater than zero, since otherwise $\Phi(x,t)$ would vanish for all x.

- $\left\langle \Phi(x,t), \Phi(x,\tilde{t}) \right\rangle > 0$: In this case we use the Cauchy Schwarz inequality. Since $\Phi(x,t)$ and $\Phi(x,\tilde{t})$ are linearly independent, the inequality becomes strict and we can write

  $$\begin{aligned} \left\langle \Phi(x,t), \Phi(x,t) - \Phi(x,\tilde{t}) \right\rangle &= \langle \Phi(x,t), \Phi(x,t) \rangle - \left\langle \Phi(x,t), \Phi(x,\tilde{t}) \right\rangle \\ &> \langle \Phi(x,t), \Phi(x,t) \rangle - \langle \Phi(x,t), \Phi(x,t) \rangle^{\frac{1}{2}} \left\langle \Phi(x,\tilde{t}), \Phi(x,\tilde{t}) \right\rangle^{\frac{1}{2}} \\ &= \langle \Phi(x,t), \Phi(x,t) \rangle^{\frac{1}{2}} \left(\langle \Phi(x,t), \Phi(x,t) \rangle^{\frac{1}{2}} - \left\langle \Phi(x,\tilde{t}), \Phi(x,\tilde{t}) \right\rangle^{\frac{1}{2}} \right) \\ &\geq 0 . \end{aligned}$$

  Note that the term in the last line is nonnegative due to the choice of t and $\tilde{t}$.

Thus, in both cases we have shown that the scalar product does not vanish, and therefore that $F(q) - F(p)$ is not an element of the null space of $F'(p)^*$. □

4.3.2 Example for Landweber's Method

Using this lemma we can now show that Landweber's method and therefore also backpropagation may encounter convergence problems, independently of the distance between initial guesses and the optimal choices of the weights.

Theorem 4.2. *Let the assumptions of Lemma 4.1 be fulfilled. Then Landweber's method does not reduce the error monotonically, i. e., for all sufficiently small neighbourhoods there exists a combination of points, such that the distance to the optimal parameter* increases *when one step of Landweber's method is applied.*

Proof. According to Lemma 4.1 we can find parameters p_k and p_* such that $p_k - p_*$ is an element of the null space of $F'(p_k)$, but at the same time the difference of the function values $F(p_k) - F(p_*)$ is not an element of the null space of $F'(p_k)^*$. Let the parameter p_* be the optimal choice of the weights in the neural network, this means we choose $F(p_*)$ as target function.

We can now show that the approximation p_{k+1} which we get when we do one step of Landweber's method is further away from p_* than the kth approximation. If we use that $F'(p_k)(p_k - p_*)$ vanishes we can compute (see also [HNS95])

$$\begin{aligned} &\|p_* - p_{k+1}\|^2 - \|p_* - p_k\|^2 \\ &\quad = \|F'(p_k)^*(F(p_*) - F(p_k))\|^2 - 2\langle F(p_*) - F(p_k), F'(p_k)(p_* - p_k)\rangle \\ &\quad = \|F'(p_k)^*(F(p_*) - F(p_k))\|^2 . \end{aligned} \tag{4.8}$$

The last line does not vanish due to the choice of p_* and p_k. So we have shown that the $k+1$st iterate is a worse approximation to p_* than the kth approximate. □

Observe that this example can also be transferred to the case of damped Landweber methods, i. e.,

$$p_{k+1} = p_k + \eta F'(p_k)^*(y - F(p_k)), \quad k \in \mathbb{N},$$

where η is a scaling parameter.

The theorem above shows that the difference $\|p_* - p_k\|$ may increase within *one* step of Landweber's method, but it does not show that Landweber's method diverges — the error might decrease again in the next step. Although it seems likely, we were not able to prove that there exist choices where the error increases throughout the iteration. The reason for this is that even if $p_* - p_k$ was chosen as an element of the nullspace of $F'(p_k)$ the next iterate p_{k+1} will not necessarily have this property. Nevertheless, the vector $p_* - p_{k+1}$ can still be close to the nullspace, and therefore the sum of the two terms in equation (4.8) can still be positive. We present two numerical examples that support these arguments very well.

As activation function Φ we choose the Gaussian curve

$$\Phi(x,t) = e^{-25(x-t)^2}$$

on the one-dimensional domain $\Omega = [-1,1]$. In both examples the objective function is built of two neurons and given as

$$f(x) = \frac{1}{2}\big(\Phi(x,-0.1) - \Phi(x,0.1)\big).$$

This corresponds to the choice $p_* = (c,t,-c,\tilde{t})$ with $c = 0.5$, $t = -0.1$ and $\tilde{t} = 0.1$. We approximate this function with a neural network consisting of two nodes. For this task we use Landweber's method with two different initial values. It should be mentioned that the non-convergence in the following examples is not caused by insufficient damping, and that the damped version of Landweber's method behaves similarly in both cases. In the first example the nodes overlap, in the second one they are separated.

Example 4.3. (Overlapping nodes) As initial value for p we choose $p_0 = (0,t,0,t)$. With this choice $p_* - p_0$ is an element of the nullspace of $F'(p_0)$ and as predicted by Theorem 4.2 the distance to the optimal parameter increases within the first step of Landweber's method. The difference $p_* - p_1$ is *not* an element of the nullspace of $F'(p_1)$, therefore it would be possible that in the next step of the iteration this difference decreases again. This does not happen, because the second term in equation (4.8) is still small in comparison to the first one and hence, the error increases again. Figure 4.2 shows the qualitative behaviour of the iteration for several indices k. As can be seen in Figure 4.3 the distance to the optimal parameter increases monotonically throughout the iteration – Landweber's method fails.

The problems in the example above occured, because both nodes started at the same place. The next example shows that even if the nodes are well separated, convergence problems can occur and the distance to the optimal parameter can increase within the iteration.

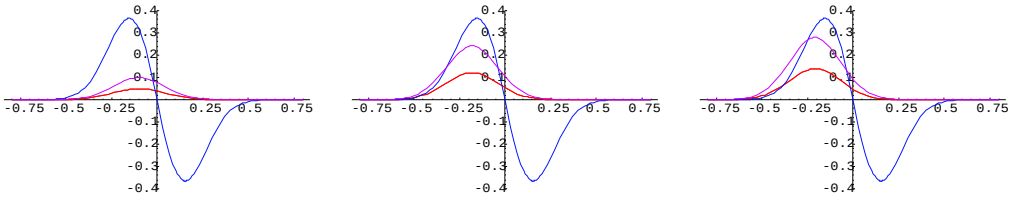


Figure 4.2: Approximation of a function (blue) with a neural network (violet), consisting of two nodes. The nodes (red) overlap, therefore only one line is visible. The plot shows the iterations for $k = 1$, 5 and 20 (from the left to the right).

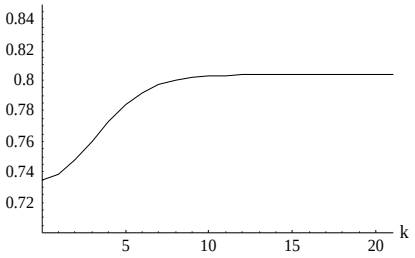


Figure 4.3: Evolution of $\|p_* - p_k\|$ during the Landweber iteration. The difference increases monotonically, and remains constant after about 10 steps.

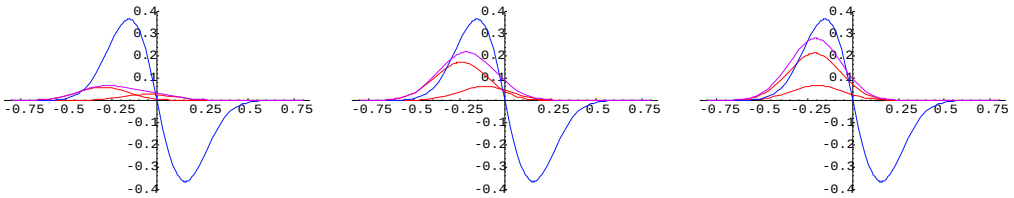


Figure 4.4: Approximation of a function (blue) with a neural network (violet), consisting of two nodes. The two nodes are drawn in red. The plot shows the iterations for $k = 1$, 5 and 20 (from the left to the right).

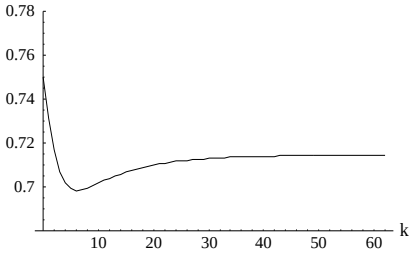


Figure 4.5: Evolution of $\|p_* - p_k\|$ during the Landweber iteration. The difference decreases until $k \approx 5$ and increases again afterwards.

Example 4.4. (Separated nodes) In this example we choose the initial value as $p_0 = (0, -0.3, 0, -0.05)$, so now the two nodes of the network are well separated, their distance is even larger than the distance of the nodes that where used to construct the objective function. Figure 4.4 shows three steps of Landweber's method. Both nodes are attracted by the left, positive part of the objective function and therefore the distance between them decreases. After 20 steps of the iteration the centers of the two nodes coincide. As can be seen in Figure 4.5 the distance to the optimal parameter decreases for a few steps, but when the nodes are too close to each other Landweber's method fails and the error starts to increase monotonically as in the example above.

4.3.3 Example for Newton Type Methods

Using similar ideas, we can also derive an example where Newton's method as defined in (4.6) does not decrease the error monotonically.

Theorem 4.5. *Let Φ fulfill the conditions of Lemma 4.1 and $n \geq 2$. Then Newton's method does not reduce the error, i. e., for all sufficiently small*

neighbourhoods there exists a combination of points, such that the distance to the optimal parameter increases *when one step of Newton's method is applied.*

Proof. Since $F'(p_k)$ has a non trivial null space for $n \geq 2$ we do not use the inverse of $F'(p_k)$ but we use the Moore-Penrose-Inverse $F'(p_k)^\dagger$ (see e. g. [EHN96]). Since $F'(p_k)$ is a linear operator from $\mathbb{R}^p$ to L^2 the Moore Penrose Inverse is an operator that goes from L^2 to the orthogonal complement of the null space of F', i. e.,

$$F'(p_k)^\dagger : L^2(\Omega) \mapsto N(F'(p_k))^\perp \subsetneq \mathbb{R}^p \,.$$

This means that for any vector x in the null space of $F'(p_k)$ and for any arbitrary function y in L^2 the scalar product $\left\langle x, F'(p_k)^\dagger(y)\right\rangle$ has to vanish.

According to Lemma 4.1 we are able to find a combination of parameters p_k and p_* such that $p_k - p_*$ is an element of the null space of $F'(p_k)$, but the corresponding difference $F(p_k) - F(p_*)$ is not an element of the nullspace of $F'(p_k)^*$. We choose p_* as our solution, this means we choose f as $f = F(p_*)$.

Now we look what happens to the distance to the optimal value, if we start the iteration with p_k. To this end we compute the difference

$$\begin{aligned}
&\|p_* - p_{k+1}\|^2 - \|p_* - p_k\|^2 \\
&\quad = \left\|p_* - p_k - F'(p_k)^{-1}(F(p_*) - F(p_k))\right\|^2 - \|p_* - p_k\|^2 \\
&\quad = \|p_* - p_k\|^2 - 2\left\langle p_* - p_k, F'(p_k)^{-1}(F(p_*) - F(p_k))\right\rangle \\
&\qquad + \left\|F'(p_k)^{-1}(F(p_*) - F(p_k))\right\| - \|p_* - p_k\|^2 \\
&\quad = \left\|F'(p_k)^{-1}(F(p_*) - F(p_k))\right\|^2 \qquad (4.9)
\end{aligned}$$

Since $F(p_k) - F(p_*)$ is not an element of the nullspace of $F'(p_k)^*$ the last term does not vanish. This means that the $k+1$st iterate is a worse approximation to p_* than the kth approximate. □

Now we demonstrate briefly how this result can be transferred to the iteratively regularized Gauß-Newton method. The iterates fulfill the equation

$$\left(F'(p_k)^* F'(p_k) + \alpha_k I\right)(p_{k+1} - p_k) = F'(p_k)^*(y - F(p_k)) + \alpha_k(p_0 - p_k)\,,$$

which can be rewritten as

$$F'(p_k)^*\left(F'(p_k)(p_{k+1} - p_k) - (y - F(p_k))\right) + \alpha_k(p_{k+1} - p_k) = \alpha_k(p_0 - p_k).$$

For $k = 0$ the right hand side vanishes. Since the first part of the left hand side is in $R(F'(p_0)^*)$ also $p_1 - p_0 \in R(F'(p_0)^*)$. For $k \geq 1$ we obtain by

induction that the right hand side as well as the first part of the left hand side are in $R(F'(p_k)^*)$ and consequently $p_{k+1} - p_k \in R(F'(p_k)^*)$. Since for any operator $R(F'^*) \subseteq N(F')^\perp$ holds, the result of Theorem 4.5 follows with an analogous proof for the iteratively regularized Gauß-Newton method.

4.4 Basic Properties for a Single Node

In this section we will provide some preparations needed later to prove convergence of Landweber's and Newton's method for networks with a single node. These results will be fundamental for deriving the greedy algorithm in Chapter 5, since in the context of this algorithm we need to solve approximation problems with single nodes.

First we make some (very natural) assumptions on the activation function Φ. These assumptions will depend on whether we look at ridge constructions or radial basis function networks. Next we give two simple Lemmas which apply generally for linearly independent vectors A and B, and which will be needed in the proofs of Theorems 4.15 and 4.19.

Since we want to utilize these results later, with the two vectors A and B being given as the activation function Φ and its derivative with respect to the parameters t (denoted as $\nabla_t \Phi$), we have to show that the assumptions are fulfilled for this special choice. For this reason we mainly have to show that these two functions are linearly independent. At first sight this seems clear because a function and its derivative are linearly dependent only for the case of exponential functions. Nevertheless this is not true for our case in general, since we are looking at a derivative with respect to parameters t, but we need linear independence with respect to the input variable x.

In Sections 4.4.3 and 4.4.4 we show that Φ and $\mathbf{e}\nabla_t \Phi$ are linearly independent for a fixed choice of a unit vector $\mathbf{e}$ for ridge constructions and radial basis functions, respectively. Finally in Section 4.4.5 we will expand this result to the case of arbitrary choice of $\mathbf{e}$.

Altogether, we will be able to conclude that for a function Φ, satisfying the conditions in the standard Assumptions 4.6, the function and its derivative are linearly independent and therefore the assumptions of the Lemmas in Section 4.4.2 are fulfilled.

4.4.1 Assumptions on the Activation Function

For the following discussion some assumptions on the activation function Φ have to be made. We will choose these assumptions such, that they are satisfied by the usual activation functions in neural networks.

Standard Assumptions 4.6. We say that Φ fulfills the *standard assumptions* if either

(i)
- $\Phi(x,t)$ is a ridge construction, i. e., a function of form

$$\Phi(x,t) = \sigma(a^T x + b) = \sigma(\zeta_1 + \cdots + \zeta_n), \tag{4.10}$$

  with the new variables $\zeta_1 = a_1 x_1 + b$ and $\zeta_i = a_i x_i$ for $i \neq 1$.

  The values a_i and b_i correspond to the parameters t_i and are only introduced to emphasize the different meanings of the various parameters.

- The factors a_i are chosen such that the vector $(a_1, \ldots, a_n)$ is an element of some compact set that does not contain 0.

- the function σ does not have the form

$$\sigma(\xi) = Ce^{\alpha\xi} \qquad \text{or} \qquad \sigma(\xi) = (\alpha\xi + \beta)^\gamma$$

  for any combination of real numbers α, β and γ.

- the function σ is twice continuously differentiable.

or

(ii)
- $\Phi(x,t)$ is a radial basis function, i. e., a function of form

$$\begin{aligned}\Phi(x,t) &= \Xi\left((a_1 x_1 + b_1)^2 + \cdots + (a_n x_n + b_n)^2\right) \\ &= \Xi\left(\zeta_1^2 + \cdots + \zeta_n^2\right),\end{aligned} \tag{4.11}$$

  with the new variables $\zeta_i = a_i x_i + b_i$

- the values of the parameters a_i are chosen from a compact set that does not contain 0.

- the function Ξ does not have the form

$$\Xi(\xi) = C\xi^\alpha$$

  for any combination of real numbers α and C

- the function Ξ is twice continuously differentiable.

Remark 4.7. Neither the activation function σ nor the function Ξ are allowed to be constant functions. In both cases this restriction is already contained in the forbidden special forms for the choice $\alpha = 0$.

For the case of ridge constructions we defined an artificial correspondence between b_1 and ζ_1. It is no matter to which of the ζ_i the variable b_1 is associated, the goal is just to have as many variables ζ_i as variables x_i. Therefore also the dimension of Ω and Ω_ζ (as defined in the proof of the theorem below) will be the same.

Our definition of radial basis functions is more general than the usual one, e. g. in [BE00] the "radius" of the function is fixed, in [GJP95] it may vary but the shape is always circular. In this setting also ellipsoidal shapes are allowed. If all the a_i are equal (or especially fixed and equal to one) then the original case is attained.

For the case of ridge constructions it is possible to choose some of the parameters a_i equal to zero (this happens when the ridge is orientated parallel to an axis), while for radial basis functions this can not happen, all the values a_i are nonzero.

In both cases the derivatives with respect to ζ_i look very simple. They are given by

$$\frac{\partial \Phi}{\partial \zeta_i} = \sigma'(\zeta_1 + \cdots + \zeta_n) \quad \text{and} \quad \frac{\partial \Phi}{\partial \zeta_i} = 2\zeta_i \Xi'(\zeta_1^2 + \cdots + \zeta_n^2)$$

for ridge constructions and radial basis functions respectively.

4.4.2 Two Lemmas about Linear Independence

In this section we derive two general results for linearly independent vectors. The first lemma will be needed in the proof of Theorem 4.15.

Lemma 4.8. *Let A and B be two vectors with $\|A\| \neq 0$ and $\|B\| \neq 0$, such that*

$$\left\| A \pm \frac{\|A\|}{\|B\|} B \right\| \geq \mu > 0 \tag{4.12}$$

holds. Then

$$\begin{aligned} &\forall c > 0\ \exists \alpha_0, \beta_0 > 0\ \forall \alpha \in [0, \alpha_0]\ \forall \beta \in [0, \beta_0] : \\ &\qquad c \max\{\alpha^2, \beta^2\} \leq \|\alpha A + \beta B\| \quad \textit{and} \qquad (4.13a) \\ &\qquad c \max\{\alpha^2, \beta^2\} \leq \|\alpha A - \beta B\| \ . \qquad (4.13b) \end{aligned}$$

Proof. The proof is only presented for the case (4.13a). Since both, the second triangle inequality and condition (4.12) are symmetric with respect to the sign of B, the proof does not change if it is carried out for the case (4.13b).

The second triangle inequality yields

$$\|\alpha A + \beta B\| \geq |\alpha \|A\| - \beta \|B\||.$$

We distinguish three cases:

- $\alpha \|A\| - \beta \|B\| > 0$: If $\alpha \|A\| - \beta \|B\| > 0$ then for fixed values of α and β some $\varphi > 0$ exists with $\alpha \|A\| - \beta \|B\| = \alpha\varphi$. With this φ the identity $\alpha(\|A\| - \varphi)/\|B\| = \beta$ holds and we obtain

$$\begin{aligned}\max\{\alpha^2, \beta^2\} &= \max\{\alpha^2, \left(\alpha \frac{\|A\| - \varphi}{\|B\|}\right)^2\} \\ &= \alpha^2 \max\{1, \left(\frac{\|A\| - \varphi}{\|B\|}\right)^2\}.\end{aligned}$$

  For the choice

$$\alpha \leq \frac{\varphi}{c \max\{1, \left(\frac{\|A\| - \varphi}{\|B\|}\right)^2\}}$$

  condition (4.13a) is fulfilled along the line $\beta = \alpha(\|A\| - \varphi)/\|B\|$. The parameter φ determines the angle that this line encloses with the line $\alpha \|A\| = \beta \|B\|$. If we take the union of all such lines for $\varphi \in [0, \|A\|]$, we find an area in which the estimate (4.13a) is fulfilled. This area of validity is shown schematically in Figure 4.6 and corresponds to zone I.

- $\alpha \|A\| - \beta \|B\| < 0$: The proof for this case can be carried out in an analogous way, the associated area of validity is zone II in Figure 4.6.

  Obviously, with this method we cannot gain an open neighbourhood of 0 where equation (4.13a) is fulfilled. In the first part we used that α was much larger than β, in the second part we used that β was much larger than α.

  Along the diagonal $\alpha \|A\| = \beta \|B\|$ (zone III in Figure 4.6) the second triangle inequality cannot lead to success and the additional assumption (4.12) is needed.

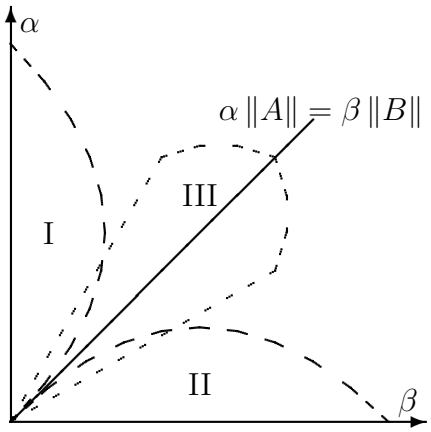


Figure 4.6: Representation of the 3 zones for the different cases in Lemma 4.8

- $\alpha \|A\| - \beta \|B\| \approx 0$: For fixed α and β again some (not necessarily positive) parameter φ exists so that

  $$\beta = \alpha \frac{\|A\|}{\|B\|}(1+\varphi).$$

  For sufficiently small φ we obtain the estimate

  $$\begin{aligned} \|\alpha A + \beta B\| &= \|\alpha A + \alpha \frac{\|A\|}{\|B\|} B + \alpha\varphi \frac{\|A\|}{\|B\|} B\| \\ &\geq |\|\alpha A + \alpha \frac{\|A\|}{\|B\|} B\| - \|\alpha\varphi \frac{\|A\|}{\|B\|} B\|| \\ &\geq \alpha \left(\mu - |\varphi| \, \|A\|\right), \end{aligned}$$

  and for the special choice

  $$\alpha \leq \frac{\mu - |\varphi| \, \|A\|}{c \max\{1, \left(\frac{\|A\|}{\|B\|}(1+\varphi)\right)^2\}}$$

  the estimate (4.13a) is fulfilled along the line $\beta = \alpha \frac{\|A\|}{\|B\|}(1+\varphi)$. Here again the parameter φ determines the angle that this line encloses with the line $\alpha \|A\| = \beta \|B\|$.

Since the three zones overlap for sufficiently small α and β, their union contains an open neighbourhood of 0 in which the estimate (4.13a) holds. □

For the verification of condition (4.7) for Newton's method in Theorem 4.19 a different estimate is needed.

Lemma 4.9. *Let A and B be two vectors with $\|A\| \neq 0$ and $\|B\| \neq 0$, such that the estimate* (4.12) *holds. Then*

$$\begin{aligned}
&\forall \kappa > 0\ \exists c > 0\ \forall \alpha, \beta \in \mathbb{R}: \\
&\quad \kappa \max\{\alpha, \beta\} \leq c \|\alpha A + \beta B\| \quad \textit{and} && (4.14a) \\
&\quad \kappa \max\{\alpha, \beta\} \leq c \|\alpha A - \beta B\| \,. && (4.14b)
\end{aligned}$$

Proof. First of all, we show that the two vectors A and B fulfill the estimates

$$\|A + \gamma B\| \geq \tilde{\mu} > 0 \quad \text{and} \quad \|\gamma A + B\| \geq \tilde{\mu} > 0 \qquad \forall \gamma \in \mathbb{R}. \tag{4.15}$$

Note, that for linearly independent vectors there can exist a value γ such that $\|A + \gamma B\|$ is less than $\|A + \frac{\|A\|}{\|B\|} B\|$, and hence, in general $\tilde{\mu}$ will be less than μ. At least if the absolute value of γ is greater than $\gamma_{\max} = \frac{\mu + \|A\|}{\|B\|}$ the value of the first expression in (4.15) is greater than μ. So to find the infimum of this expression we can restrict our search to values of γ being in the interval $[-\gamma_{\max}, \gamma_{\max}]$. Since this interval is compact the infimum is attained and we denote it with $\tilde{\mu}_1$. This infimum cannot be zero, otherwise also condition (4.12) could not be fulfilled.

If we interchange the roles of A and B in equation (4.12) we get the equation

$$\left\| \frac{\|B\|}{\|A\|} A \pm B \right\| \geq \frac{\|B\|}{\|A\|} \mu > 0 \,.$$

Starting from this equation we do the same procedure as above for the term $\|\gamma A + B\|$. Again we find some infimum $\tilde{\mu}_2$ which is also not zero. Finally we take the minimum of the two infima $\tilde{\mu}_1$ and $\tilde{\mu}_2$, and denote it by $\tilde{\mu}$. With this choice the estimates in (4.15) are fulfilled.

After this preparations we can now use (4.15) to prove the relation (4.14a). First we derive an estimate for α using the first relation in (4.15),

$$\|\alpha A + \beta B\| = |\alpha| \left\| A + \frac{\beta}{\alpha} B \right\| \geq |\alpha| \tilde{\mu},$$

and then the same for β but now using the second relation in (4.15),

$$\|\alpha A + \beta B\| = |\beta| \left\| \frac{\alpha}{\beta} A + B \right\| \geq |\beta| \tilde{\mu}.$$

If we combine these two results we obtain

$$\|\alpha A + \beta B\| \geq \max\{\alpha, \beta\} \tilde{\mu} \,,$$

and with the choice $c \geq \frac{\kappa}{\tilde{\mu}}$ this yields (4.14a).

The proof of equation (4.14b) follows similarly since the initial equation (4.12) is symmetric with respect to the sign. □

4.4.3 Ridge Constructions

In this section we show that condition (4.12) in Lemma 4.8 is fulfilled for the special choice

$$A = c\mathbf{e}(\tilde{t}, t)\nabla_t \Phi(x, t) \quad \text{and} \quad B = \Phi(x, t), \tag{4.16}$$

where $\mathbf{e}(\tilde{t}, t)$ is a unit vector and depends on $\tilde{t}$ and t. We would only have to verify if the two expressions $A + \frac{\|A\|}{\|B\|}B$ and $A - \frac{\|A\|}{\|B\|}B$ are different from zero, but unfortunately the factor $\frac{\|A\|}{\|B\|}$ is (if at all possible) difficult to compute analytically. Therefore instead we show that $A + \gamma B$ can not be zero for any fixed $\gamma \in \mathbb{R}$, and that $\frac{\|A\|}{\|B\|}$ is bounded and hence restricted to a compact set. From this we can conclude that there exists some μ such that condition (4.12) is fulfilled. Hence, we show that

$$\|A + \gamma B\| > 0 \tag{4.17}$$

is fulfilled for an arbitrary $\gamma \in \mathbb{R}$.

First we inspect condition (4.17) for a fixed value of γ, later we expand this result to arbitrary choices of γ, which will yield (4.12).

Theorem 4.10. *Let $c \neq 0$ and Φ fulfill the standard Assumptions 4.6(i). Then Φ fulfills condition* (4.17) *for arbitrary, but fixed values of $\tilde{t}, t$ and γ.*

Proof. Without loss of generality we assume c to be 1, otherwise we would simply divide by c (which is not 0) and continue the proof with some different value for γ. If condition (4.17) is *not* satisfied, then the identity

$$\int_\Omega \left(\mathbf{e}(t, \tilde{t})\nabla_t \Phi(x, t) + \gamma \Phi(x, t)\right)^2 \, dx_1 \cdots dx_n = 0 \tag{4.18}$$

holds for some fixed γ. Since all components of this equation are continuous we can conclude that the integrand has to vanish everywhere in the domain Ω, i. e.,

$$\mathbf{e}(t, \tilde{t})\nabla_t \Phi(x, t) + \gamma \Phi(x, t) = 0 \qquad \forall x \in \Omega.$$

Note that the derivatives in the first part are taken with respect to the parameters t and not with respect to x, so this equation is not a differential equation at the moment.

If we use the notation for the parameters t_i given in the standard Assumptions 4.6(i) we can rewrite this equation as

$$e_b \frac{\partial \Phi}{\partial b} + e_{a_1} \frac{\partial \Phi}{\partial a_1} + \cdots + e_{a_n} \frac{\partial \Phi}{\partial a_n} + \gamma \Phi(x,t) = 0, \qquad \forall x \in \Omega. \tag{4.19}$$

By use of equation (4.10) we are able to replace the derivatives of Φ with respect to the variables a_j and b_j by derivatives of Φ with respect to ζ_j. We use the relations

$$\frac{\partial \Phi}{\partial a_j} = x_j \frac{\partial \Phi}{\partial \zeta_j} \quad \text{and} \quad \frac{\partial \Phi}{\partial b_j} = \frac{\partial \Phi}{\partial \zeta_j}$$

to transform the equation above into a partial differential equation. If we assume that none of the a_j equals zero, we can further derive

$$x_1 = \frac{\zeta_1 - b}{a_1}, \qquad x_i = \frac{\zeta_i}{a_i}, \qquad i = 2, \ldots, n\,. \tag{4.20}$$

If we combine this result with (4.19) then we get the equation

$$\left(\frac{e_{a_1}}{a_1} \zeta_1 + e_b - \frac{e_{a_1} b}{a_1} \right) \frac{\partial \Phi}{\partial \zeta_1} + \frac{e_{a_2}}{a_2} \zeta_2 \frac{\partial \Phi}{\partial \zeta_2} + \cdots + \frac{e_{a_n}}{a_n} \zeta_n \frac{\partial \Phi}{\partial \zeta_n} + \gamma \Phi(\zeta) = 0, \qquad \forall \zeta \in \Omega_\zeta,$$

where Ω_ζ is the domain containing the values of ζ. The shape of this domain depends on the values of the a_i and b. If none of the a_i is zero then the dimension of Ω_ζ is equal to the dimension of the original domain Ω.

This equation can be interpreted as a partial differential equation for Φ. Now we seek a solution having a representation as in (4.10).

For the sake of simplicity we write α_j instead of $\frac{e a_j}{a_j}$ and β instead of $e_b - \frac{e_{a_1} b}{a_1}$. Furthermore, we conclude from (4.10) that all the derivatives are equal and rewrite the equation as

$$(\alpha_1 \zeta_1 + \cdots + \alpha_n \zeta_n + \beta)\, \sigma'(\zeta_1 + \cdots + \zeta_n) + \gamma \sigma(\zeta_1 + \cdots + \zeta_n) = 0, \qquad \forall \zeta \in \Omega_\zeta.$$

To prove that this equation cannot be fulfilled as long as Φ fulfills the standard Assumptions 4.6(i) we have to distinguish several cases:

- Not all of the α_i are equal: We will try to find values for the variables ζ_i such that the argument of σ remains constant, but the term in front of σ' varies. Then we can conclude that σ' is zero.

  First we choose an interior point in the domain Ω_ζ. We assume that α_1 and α_2 are not equal and choose a value for ε that is small enough to ensure that the vector $(\zeta_1 + \varepsilon, \zeta_2 - \varepsilon, \zeta_3, \dots, \zeta_n)$ is an element of Ω_ζ, too.

  Now we compute the relation above for this value and obtain

  $$\begin{aligned} 0 &= (\alpha_1(\zeta_1 + \varepsilon) + \alpha_2(\zeta_2 - \varepsilon) + \alpha_3\zeta_3 \cdots + \alpha_n\zeta_n + \beta)\,\sigma'(\cdot) \\ &\qquad + \gamma\sigma(\zeta_1 + \cdots + \zeta_n) \\ &= (\alpha_1\zeta_1 + \cdots + \alpha_n\zeta_n + \beta)\,\sigma'(\zeta_1 + \cdots + \zeta_n) \\ &\qquad + \gamma\sigma(\zeta_1 + \cdots + \zeta_n) + \varepsilon(\alpha_1 - \alpha_2)\sigma'(\zeta_1 + \cdots + \zeta_n) \\ &= \varepsilon(\alpha_1 - \alpha_2)\sigma'(\zeta_1 + \cdots + \zeta_n). \end{aligned}$$

  Since $\alpha_1 - \alpha_2$ is not zero, the function σ' has to vanish at the point $(\zeta_1, \dots, \zeta_n)$. This procedure can be done for any interior point, and hence σ' vanishes in the whole domain. This implies that σ would have to be a constant function, which is not compatible with the Assumptions 4.6(i).

- All α_i are equal and different from 0: We may rewrite the equation above as

  $$\begin{aligned} &(\alpha_1\,(\zeta_1 + \cdots + \zeta_n) + \beta)\,\sigma'(\zeta_1 + \cdots + \zeta_n) \\ &\qquad\qquad + \gamma\sigma(\zeta_1 + \cdots + \zeta_n) = 0, \qquad \forall \zeta \in \Omega_\zeta. \end{aligned}$$

  Now we compute the minimal and maximal values for the argument of the function σ as

  $$\xi_{\min} = \inf_{(\zeta_1,\dots,\zeta_n)\in\Omega_\zeta} \zeta_1 + \cdots + \zeta_n \quad \text{and} \quad \xi_{\max} = \sup_{(\zeta_1,\dots,\zeta_n)\in\Omega_\zeta} \zeta_1 + \cdots + \zeta_n$$

  and derive an ordinary differential equation for a new variable $\xi := \zeta_1 + \cdots + \zeta_n$, which is given as

  $$(\alpha_1\xi + \beta)\,\sigma'(\xi) + \gamma\sigma(\xi) = 0 \qquad \forall \xi \in [\xi_{\min}, \xi_{\max}]\,,$$

  with the general solution

  $$\sigma(\xi) = C(\alpha_1\xi + \beta)^{-\frac{\gamma}{\alpha_1}} \qquad C \in \mathbb{R}.$$

  Since we have not allowed this special structure for σ in the standard Assumptions 4.6(i), this equation cannot be fulfilled.

- all α_i are equal to 0: We proceed as above and arrive at the equation

  $$\beta\sigma'(\xi) + \gamma\sigma(\xi) = 0, \qquad \forall \xi \in [\xi_{\min}, \xi_{\max}],$$

  with the general solution

  $$\sigma(\xi) = Ce^{-\frac{\gamma}{\beta}\xi} \qquad C \in \mathbb{R}$$

  which we have also excluded.

Now we look what happens if one of the values a_i vanishes: Replacing the derivatives with respect to a_i and b in equation (4.19) with derivatives with respect to ζ_i we obtain the partial differential equation

$$e_b \frac{\partial \Phi}{\partial \zeta_1} + e_{a_1} x_1 \frac{\partial \Phi}{\partial \zeta_1} + \cdots + e_{a_n} x_n \frac{\partial \Phi}{\partial \zeta_n} + \gamma\Phi(x,t) = 0, \qquad \forall x \in \Omega.$$

If we assume that w. l. o. g. a_n is equal to 0, then Φ as well as $\frac{\partial \Phi}{\partial \zeta_i}$ do not depend on x_n (cf. (4.10)). This means that the equation has to hold independently of the value of x_n and we can therefore conclude that $e_{a_n} \frac{\partial \Phi}{\partial \zeta_n}$ must be zero. Since $\frac{\partial \Phi}{\partial \zeta_n}$ can only vanish if the function σ is constant, the value of e_{a_n} must vanish.

Remaining is the slightly different equation

$$e_b \frac{\partial \Phi}{\partial \zeta_1} + e_{a_1} x_1 \frac{\partial \Phi}{\partial \zeta_1} + \cdots + e_{a_{n-1}} x_{n-1} \frac{\partial \Phi}{\partial \zeta_{n-1}} + \gamma\Phi(x,t) = 0, \qquad \forall x \in \Omega,$$

where we can again replace the parameters x_i with the corresponding parameters ζ_i and continue the proof as above.

We proceed analogously when more than one a_j is zero. The only difference arises when all a_j except one happen to be zero. This case is treated exactly like the two cases above with all α_i being equal. Which of the two cases has to be chosen depends on whether the corresponding factor e_{a_j} is zero or not.

The case that all a_j are equal to zero can not arise because it has been explicitly forbidden in the standard Assumptions 4.6(i).

Hence, for the case of ridge constructions condition (4.17) is fulfilled for arbitrary, but fixed values of γ, $\tilde{t}$ and t. □

4.4.4 Radial Basis Functions

Now we show that radial basis function networks fulfill condition (4.17) for fixed values of $\tilde{t}, t$ and γ.

Theorem 4.11. *Let $c \neq 0$ and Φ fulfill the standard Assumptions 4.6(ii). Then Φ fulfills condition* (4.17) *for arbitrary, but fixed values of $\tilde{t}, t$ and γ.*

Proof. We start the proof as before. Instead of (4.19) we get the equation

$$e_{a_1}\frac{\partial\Phi}{\partial a_1} + e_{b_1}\frac{\partial\Phi}{\partial b_1} + \cdots + e_{a_n}\frac{\partial\Phi}{\partial a_n} + e_{b_n}\frac{\partial\Phi}{\partial b_n} + \gamma\Phi(x,t) = 0, \qquad \forall x \in \Omega.$$

In contrast to Theorem 4.10, we do not have to distinguish whether an a_j is zero or not, because we have excluded this possibility explicitly in the standard Assumptions 4.6(ii).

We proceed as in the proof above and replace derivatives with respect to a_i and b_i by derivatives with respect to ζ_i. Then we replace the variables x_i with the variable ζ_i and get the equation

$$\begin{aligned}\left(\frac{e_{a_1}}{a_1}\zeta_1 + e_{b_1} - \frac{e_{a_1}b_1}{a_1}\right)\frac{\partial\Phi}{\partial\zeta_1} + \cdots \\ \cdots + \left(\frac{e_{a_n}}{a_n}\zeta_n + e_{b_n} - \frac{e_{a_n}b_n}{a_n}\right)\frac{\partial\Phi}{\partial\zeta_n} + \gamma\Phi(\zeta) = 0, \qquad \forall\zeta \in \Omega_\zeta.\end{aligned} \tag{4.21}$$

Now we use the representation of Φ in (4.11) to compute the derivatives of Φ. For the sake of simplicity we again introduce abbreviations and write α_j instead of $\frac{ea_j}{a_j}$ and β_j instead of $e_{b_j} - \frac{e_{a_j}b_j}{a_j}$, yielding the equation

$$\begin{aligned}\big((\alpha_1\zeta_1 + \beta_1)\zeta_1 + \cdots + (\alpha_n\zeta_n + \beta_n)\zeta_n\big)\,\Xi'(\zeta_1^2 + \cdots + \zeta_n^2) \\ + \gamma\Xi(\zeta_1^2 + \cdots + \zeta_n^2) = 0 \qquad \forall\zeta \in \Omega_\zeta.\end{aligned} \tag{4.22}$$

To prove that this equation cannot be satisfied if the function Φ fulfills the standard Assumptions 4.6(ii), we have to distinguish between two cases:

- all α_i are equal, all β_i are equal to zero: We may rewrite the equation above as

  $$\alpha_1\big(\zeta_1^2 + \cdots + \zeta_n^2\big)\,\Xi'(\zeta_1^2 + \cdots + \zeta_n^2) + \gamma\Xi(\zeta_1^2 + \cdots + \zeta_n^2) = 0, \qquad \forall\zeta \in \Omega_\zeta.$$

  The value of α_1 cannot be zero, otherwise e_{a_1} would have to vanish and, since $\beta_1 = 0$ also e_{b_1} would be equal to zero. Since all a_i are equal, the same reasoning applies for e_{a_i} and e_{b_i}, and we can conclude that $\mathbf{e} = \mathbf{0}$, which is impossible, because $\mathbf{e}$ is a unit vector.

  Now we compute the minimal and maximal values for the argument of the function Ξ as

  $$\xi_{\min} = \inf_{(\zeta_1,\ldots,\zeta_n)\in\Omega_\zeta} \zeta_1^2 + \cdots + \zeta_n^2 \quad \text{and} \quad \xi_{\max} = \sup_{(\zeta_1,\ldots,\zeta_n)\in\Omega_\zeta} \zeta_1^2 + \cdots + \zeta_n^2$$

and obtain an ordinary differential equation for a new variable $\xi = \zeta_1^2 + \cdots + \zeta_n^2$,

$$\alpha_1 \xi \, \Xi'(\xi) + \gamma \Xi(\xi) = 0, \quad \forall \xi \in [\xi_{\min}, \xi_{\max}] \,,$$

with the solution

$$\Xi(\xi) = C \xi^{-\frac{\gamma}{\alpha_1}}, \quad C \in \mathbb{R}.$$

Since this special structure for Ξ has been excluded in the standard Assumptions 4.6(ii) this equation cannot be fulfilled.

- any other case: We vary the parameters ζ_i such, that the argument of Ξ' remains constant but the factor in front of Ξ' varies. Then we can conclude that Ξ' vanishes and that Ξ cannot fulfill the standard Assumptions 4.6(ii).

  Therefore we choose the variables ζ_i such that the vector $(\zeta_1, \ldots, \zeta_n)$ lies in the interior of the domain Ω_ζ and none of the ζ_i is zero (for the remaining points the conclusion follows by continuity of Ξ'). Then we choose two indices i and j such that not $\beta_i = \beta_j = \alpha_i - \alpha_j = 0$. Since not all α_i are equal or not each β_i is zero, we are able to find such a combination of indices, and we assume w. l. o. g. that the choice $i = 1$, $j = 2$ is valid.

  Now we change the values of ζ_1 and ζ_2 in such a way, that the argument of Ξ remains constant, i. e., $\zeta_1^2 + \cdots + \zeta_n^2 = const.$ and show that the factor in front of Ξ' does not remain constant.

  We define the radius r as $r = \zeta_1^2 + \zeta_2^2$ and move along the circle $\zeta_2 = \sqrt{r^2 - \zeta_1^2}$ (w. l. o. g. ζ_2 is originally positive, otherwise we choose the negative branch of the square root).

  Now we show that the part of the term in front of Ξ' that depends on ζ_1 and ζ_2 does not remain constant. We assume that it is equal to some constant C and obtain

$$\begin{aligned} C &= (\alpha_1 \zeta_1 + \beta_1)\zeta_1 + (\alpha_2 \zeta_2 + \beta_2)\zeta_2 \\ &= (\alpha_1 \zeta_1 + \beta_1)\zeta_1 + (\alpha_2 \sqrt{r^2 - \zeta_1^2} + \beta_2)\sqrt{r^2 - \zeta_1^2} \\ &= (\alpha_1 - \alpha_2)\zeta_1^2 + \beta_1 \zeta_1 + \beta_2 \sqrt{r^2 - \zeta_1^2} + \alpha_2 r^2 \,. \end{aligned}$$

  Since ζ_1, ζ_1^2 and $\sqrt{r^2 - \zeta_1^2}$ are linearly independent the only possibility for the last line to be constant is that β_1 and β_2 vanish and α_1 and α_2 are equal, which is a contradiction.

So for the case of ridge constructions condition (4.17) is fulfilled for arbitrary, but fixed values of γ, $\tilde{t}$ and t. □

Altogether we have shown in the two theorems above, that for arbitrary functions which fulfill the standard Assumptions 4.6, condition (4.17) is fulfilled for arbitrary but fixed γ and for fixed values of $\tilde{t}$ and t (or the corresponding unit vector $\mathbf{e}$).

4.4.5 The Linear Independence Property

Using the results from Theorems 4.10 and 4.11 we can prove that the function A defined in equation (4.16) is bounded away from zero.

Corollary 4.12. *Let Φ fulfill the standard Assumptions 4.6. Then there exist constants A_{min} and A_{max} such that*

$$0 < A_{min} \leq \|\mathbf{e}\nabla_t\Phi(x,t)\| \leq A_{max} \leq \|\nabla_t\Phi(x,t)\| \quad \textit{with} \quad \|\mathbf{e}\| = 1.$$

Proof. We choose $A_{\max}$ to be the least upper bound for the expression above and obtain

$$A_{\max} = \sup_{\mathbf{e}\in S(0,1)} \|\mathbf{e}\nabla_t\Phi(x,t)\| \leq \|\mathbf{e}\| \, \|\nabla_t\Phi(x,t)\| = \|\nabla_t\Phi(x,t)\| \, .$$

Analogously we define $A_{\min}$ to be the greatest lower bound

$$A_{\min} = \inf_{\mathbf{e}\in S(0,1)} \|\mathbf{e}\nabla_t\Phi(x,t)\| \, .$$

Because the set $S(0,1)$ is compact, this infimum is attained for some $\mathbf{e}_0$. If we set $\gamma = 0$ we can use Theorem 4.10 or Theorem 4.11 respectively and obtain that the infimum as well as $A_{\min}$ is strictly greater than 0. □

We can now prove the major result of this section – neural networks fulfill the necessary conditions for Lemma 4.8 and Lemma 4.9.

Theorem 4.13. *Let Φ fulfill the standard Assumptions 4.6 and $c \neq 0$. Then for*

$$A = c\,\mathbf{e}\,\nabla_t\Phi(x,t) \quad \textit{and} \quad B = \Phi(x,t)$$

the conditions of Lemma 4.8 and Lemma 4.9, i. e., condition (4.12)*, $\|A\| > 0$ as well as $\|B\| > 0$, are fulfilled.*

Proof. The statement $\|A\| > 0$ follows immediately from Corollary 4.12, the statement $\|B\| > 0$ follows from the standard Assumptions 4.6 (if $\|B\| = 0$ then all $\frac{\partial \Psi}{\partial \zeta_i}$ would vanish, too).

Let us now turn to the proof of condition (4.12). We divide equation (4.12) by c and use the abbreviation

$$\gamma(\mathbf{e}) = \|\mathbf{e}\nabla_t \Phi(x,t)\| \,/\, \|\Phi(x,t)\|\,, \quad \mathbf{e} \in S(0,1).$$

From Corollary 4.12 we conclude that, with the constants

$$\gamma_{\min} = \frac{A_{\min}}{\|\Phi(x,t)\|}, \quad \gamma_{\max} = \frac{A_{\max}}{\|\Phi(x,t)\|},$$

the estimate $\gamma_{\min} \leq \gamma(\mathbf{e}) \leq \gamma_{\max}$ holds, and hence

$$\inf_{\mathbf{e}\in S(0,1)} \|\mathbf{e}\nabla_t \Phi(x,t) - \gamma(\mathbf{e})\Phi(x,t)\| \geq \inf_{\substack{\mathbf{e}\in S(0,1)\\ \gamma\in[\gamma_{\min},\gamma_{\max}]}} \|\mathbf{e}\nabla_t \Phi(x,t) - \gamma\Phi(x,t)\|\,.$$

The set $S(0,1) \times [\gamma_{\min}, \gamma_{\max}]$ is compact and so the infimum on the right hand side is attained for a combination $\mathbf{e}_0$ and γ_0. If we use these values in Theorem 4.10 or in Theorem 4.11 respectively, we obtain that the infimum is strictly greater than 0. Thus, there exists some positive μ such that

$$\inf_{\substack{\mathbf{e}\in S(0,1)\\ \gamma\in[A_{\min},A_{\max}]}} \|\mathbf{e}\nabla_t \Phi(x,t) - \gamma\Phi(x,t)\| \geq \mu > 0$$

and condition (4.12) is fulfilled. □

4.5 Landweber's Method for a Single Node

In this section we show that the nonlinearity condition (4.5) is fulfilled for neural networks with a single node and present a consequential convergence result for Landweber's method. Finally we give a result for noisy data.

For the first proof we need the following *integral representation of error*:
Lemma 4.14. *Let X and Y be Banach-spaces, $D \subset X$, open and not empty. Let $\Psi : D \mapsto Y$ be continuously differentiable. Then the equation*

$$\Psi(y) - \Psi(x) = \int_0^1 \Psi'\big(x + (y-x)\xi\big)(y-x)\,d\xi \tag{4.23}$$

holds

Proof. See e. g. [KF75, Section 10.1.7] □

Observe that this is not a trivial result due to the fact that Ψ is not a function but an *operator* between Banach-spaces in this setting. For this reason also the integral is not the usual Riemann-integral but a Bochner-Cauchy-integral.

In the next theorem we show that for a neural network consisting of a single node the nonlinearity condition (4.5) is fulfilled.

Theorem 4.15. *For neural networks with a single node, condition* (4.5) *is fulfilled, as long as the activation function* Φ *fulfills the standard Assumptions 4.6.*

Proof. For the case of a neural network with a single node, equation (4.5) becomes

$$\begin{aligned}\left\|\tilde{c}\Phi(x,\tilde{t}) - c\Phi(x,t) - c\nabla_t\Phi(x,t)(\tilde{t}-t) - (\tilde{c}-c)\Phi(x,t)\right\| \\ \leq \eta\left\|c(\Phi(x,\tilde{t}) - \Phi(x,t)) + (\tilde{c}-c)\Phi(x,\tilde{t})\right\|.\end{aligned} \tag{4.24}$$

First we look for an upper bound of the left hand side of this equation using the triangle inequality

$$\left\|\tilde{c}\Phi(x,\tilde{t}) - c\Phi(x,t) - c\nabla_t\Phi(x,t)(\tilde{t}-t) - (\tilde{c}-c)\Phi(x,t)\right\| \tag{4.25a}$$

$$\begin{aligned}\leq &|c|\left\|\Phi(x,\tilde{t}) - \Phi(x,t) - \nabla_t\Phi(x,t)(\tilde{t}-t)\right\| \\ &+ |\tilde{c}-c|\left\|\Phi(x,\tilde{t}) - \Phi(x,t)\right\|.\end{aligned} \tag{4.25b}$$

To get further estimates of the first term in equation (4.25b) we use the integral error representation (4.23) twice to deduce

$$\begin{aligned}&|c|\left\|\Phi(x,\tilde{t}) - \Phi(x,t) - \nabla_t\Phi(x,t)(\tilde{t}-t)\right\| \\ &= |c|\left\|\int_0^1 \nabla_t\Phi(x,t+(\tilde{t}-t)\xi)(\tilde{t}-t)\,d\xi - \int_0^1 \nabla_t\Phi(x,t)(\tilde{t}-t)\,d\xi\right\| \\ &= |c|\left\|\int_0^1 \left(\int_0^1 \nabla_t\big(\nabla_t\Phi\big)(x,t+(\tilde{t}-t)\xi\zeta)(\tilde{t}-t)\xi\,d\zeta\right)(\tilde{t}-t)\,d\xi\right\| \\ &\leq |c|\left\|\tilde{t}-t\right\|^2\int_0^1\int_0^1 \left\|\nabla_t\big(\nabla_t\Phi\big)(x,t+(\tilde{t}-t)\xi\zeta)\right\|\xi\,d\zeta\,d\xi \\ &\leq \frac{|c|m_{\Phi''}(t,\tilde{t})}{2}\left\|\tilde{t}-t\right\|^2.\end{aligned}$$

Here $m_{\Phi''}(t,\tilde{t})$ is defined by

$$m_{\Phi''}(t,\tilde{t}) = \sup_{\eta\in[0,1]}\left\|\nabla_t\big(\nabla_t\Phi\big)(x,t+(\tilde{t}-t)\eta)\right\|.$$

To eliminate the dependence of the leading term on $\tilde{t}$, we have to distinguish two cases:

- $\left\|\nabla_t\big(\nabla_t\Phi\big)(x,t)\right\| \neq 0$: If $\tilde{t}$ is sufficiently close to t, by the continuity of $\nabla_t\big(\nabla_t\Phi\big)$ with respect to the second parameter and the continuity of the norm, the supremum can be estimated by $2\left\|\nabla_t\big(\nabla_t\Phi\big)(x,t)\right\|$.
- $\left\|\nabla_t\big(\nabla_t\Phi\big)(x,t)\right\| = 0$: The supremum has to converge to 0 for $\tilde{t} \to t$ and can therefore be estimated by an arbitrary constant, e. g. 1 for $\tilde{t}$ sufficiently close to t.

Altogether there exists a neighbourhood of t such that

$$m_{\Phi''}(t,\tilde{t}) \leq \max\{1, 2\left\|\nabla_t\big(\nabla_t\Phi\big)(x,t)\right\|\} =: M_{\Phi''}(t).$$

For the second part of equation (4.25b) we obtain analogously

$$|\tilde{c}-c|\left\|\Phi(x,\tilde{t})-\Phi(x,t)\right\| \leq |\tilde{c}-c|\left\|\tilde{t}-t\right\| m_{\Phi'}(t,\tilde{t}),$$

where $m_{\Phi'}(t,\tilde{t})$ denotes

$$m_{\Phi'}(t,\tilde{t}) = \sup_{\eta\in[0,1]}\left\|\nabla_t\Phi(x,t+(\tilde{t}-t)\eta)\right\|. \tag{4.26}$$

Again by investigation of several cases we obtain

$$m_{\Phi'}(t,\tilde{t}) \leq \max\{1, 2\left\|\nabla_t\Phi(x,t)\right\|\} =: M_{\Phi'}(t). \tag{4.27}$$

If we combine these two results with equation (4.25b) we obtain

$$\begin{aligned}
&\left\|\tilde{c}\Phi(x,\tilde{t}) - c\Phi(x,t) - c\nabla_t\Phi(x,t)(\tilde{t}-t) - (\tilde{c}-c)\Phi(x,t)\right\| \\
&\quad\leq \frac{|c|M_{\Phi''}(t)}{2}\left\|\tilde{t}-t\right\|^2 + |\tilde{c}-c|\left\|\tilde{t}-t\right\| M_{\Phi'}(t) &(4.28)\\
&\quad\leq 2\max\{\frac{|c|M_{\Phi''}(t)}{2}, M_{\Phi'}(t)\}\max\{(\tilde{c}-c)^2, \left\|\tilde{t}-t\right\|^2\} &(4.29)
\end{aligned}$$

as an estimate for the left hand side of condition (4.24).

If we now look at the right hand side of condition (4.24) we achieve by using the second triangle inequality

$$\begin{aligned}
&\eta\left\|c(\Phi(x,\tilde{t})-\Phi(x,t)) + (\tilde{c}-c)\Phi(x,\tilde{t})\right\| \\
&\geq \eta\left\|c\nabla_t\Phi(x,t)(\tilde{t}-t) + (\tilde{c}-c)\Phi(x,t)\right\| \\
&-\eta\left\|c(\Phi(x,\tilde{t})-\Phi(x,t)-\nabla_t\Phi(x,t)(\tilde{t}-t)) + (\tilde{c}-c)(\Phi(x,\tilde{t})-\Phi(x,t))\right\|
\end{aligned}$$

Note that the second part of this equation equals $-\eta$ times the expression in (4.25a). If we combine this result with estimate (4.29) it remains to show that

$$\begin{aligned}\frac{1+\eta}{\eta} 2\max\{\frac{|c|M_{\Phi''}(t)}{2}, M_{\Phi'}(t)\}\max\{(\tilde{c}-c)^2, \left\|\tilde{t}-t\right\|^2\} \\ \leq \left\|(\tilde{t}-t)c\nabla_t\Phi(x,t) + (\tilde{c}-c)\Phi(x,t)\right\| \\ = \left\| \left\|\tilde{t}-t\right\| \frac{\tilde{t}-t}{\left\|\tilde{t}-t\right\|} c\nabla_t\Phi(x,t) + \operatorname{sign}(\tilde{c}-c)|\tilde{c}-c|\Phi(x,t)\right\| \end{aligned} \tag{4.30}$$

holds for $\left\|\tilde{t}-t\right\|$ and $|\tilde{c}-c|$ sufficiently small.

For $c \neq 0$ this follows immediately from Lemma 4.8 with the aid of Theorem 4.13. For $c = 0$ we do not use (4.29), but (4.28) and instead of the estimate in (4.30) we obtain the relation

$$\frac{1+\eta}{\eta}|\tilde{c}-c|\left\|\tilde{t}-t\right\| M_{\Phi'}(t) \leq |\tilde{c}-c|\left\|\Phi(x,t)\right\|,$$

which is obviously fulfilled if $\left\|\tilde{t}-t\right\|$ is chosen sufficiently small. □

With this theorem we obtain, that for single nodes of a ridge construction or a radial basis function network (under the weak assumptions given in 4.6) the nonlinearity condition (4.5) is fulfilled. This implies local convergence and stability of the Landweber iteration as we will see in the next theorem (cf. [HNS95, Theorem 2.3]). This theorem investigates the nonlinear operator equation

$$F(p) = y \tag{4.31}$$

where $F : \mathcal{D}(F) \to Y$ with domain $\mathcal{D}(F) \subset X$, and Hilbert spaces X and Y.

Theorem 4.16. *Let the nonlinear Landweber iteration be defined via*

$$p_{k+1} = p_k + F'(p_k)^*(y - F(p_k)), \quad k = 0, 1, 2, \ldots,$$

where p_0 is an initial guess for a solution of (4.31). Let the nonlinearity condition (4.5) be fulfilled in the ball $B(p_0, \rho)$ for some radius ρ, and the operator F be scaled such that

$$\|F'(p)\| \leq 1, \quad p \in B(p_0, \rho). \tag{4.32}$$

Furthermore, let problem (4.31) be solvable within $B(p_0, \frac{\rho}{2})$. Then the iterates p_k converge to a solution $p_* \in B(p_0, \frac{\rho}{2})$ of (4.31).

If we combine the results from Theorems 4.15 and 4.16 we obtain the following result for neural networks.

Corollary 4.17. *Let the activation function Φ fulfill the standard Assumptions 4.6. Let the function f be representable by a neural network consisting of single node, i. e., $f = F(p_*)$, $p_* \in \mathbb{R} \times P$ and the damped Landweber iteration be defined via*

$$p_{k+1} = p_k + \eta^2 F'(p_k)^*(f - F(p_k)), \quad k = 0, 1, 2, \ldots,$$

where $\eta \leq \|F'(p)\|^{-1}$, $\forall p \in \mathbb{R} \times P$. Then the iterates p_k converge to a solution p_*, if the initial guess p_0 is chosen sufficiently close to p_*.

So Landweber's method converges when applied to a neural network consisting of a single node and attainable data. For the case of non-attainable data in [HNS95] a *generalized discrepancy principle* is proposed. Assume that problem (4.31) is solvable for y and let y^δ be such that $\left\|y - y^\delta\right\| \leq \delta$. We denote by k_* the iteration index such that

$$\left\|y^\delta - F(p_{k_*}^\delta)\right\| \leq \tau\delta < \left\|y^\delta - F(p_k^\delta)\right\|, \quad 0 \leq k < k_* \tag{4.33}$$

where

$$\tau > 2\frac{1+\eta}{1-2\eta} > 2 \tag{4.34}$$

So k_* is one of the first indices where the size of the residual has about the same order of magnitude as the error in the data. Using this stopping rule the following result can be shown ([HNS95, Proposition 2.2]):

Theorem 4.18. *Assume that p_* is a solution of (4.31) in $B(p_0, \frac{\rho}{2})$ and denote by k_* the termination index of the iteration according to the stoppping rule (4.33), (4.34) for the case of perturbed data y^δ. If the nonlinearity condition (4.5) is fulfilled and F is scaled such that (4.32) holds, then we have*

$$\left\|p_* - p_{k+1}^\delta\right\| \leq \left\|p_* - p_k^\delta\right\|, \quad 0 \leq k \leq k_*$$

For a discussion of these results see Remark 4.22 at the end of the next section.

4.6 Newton Type Methods for a Single Node

In this section we show that condition (4.7) is fulfilled for neural networks consisting of a single node and present a consequential convergence result for the iteratively regularized Gauß-Newton method. Finally we discuss the convergence results of this section and the preceding one.

Theorem 4.19. *Let the activation function Φ fulfill the standard Assumptions 4.6 and $p^\dagger = (c^\dagger, t^\dagger)$ be a vector such that $c^\dagger \neq 0$. Moreover, the neural network may consist only of a single node. Then the equations in* (4.7) *are fulfilled.*

Proof. We choose R as the identity operator I, therefore equation (4.7b) is fulfilled with $C_R = 0$ and Q is given by

$$Q(\tilde{p}, p) = F'(\tilde{p}) - F'(p).$$

For the case of a neural network with a single node this can be rewritten as

$$Q(\tilde{p}, p) = \begin{pmatrix} \Phi(x, \tilde{t}) - \Phi(x, t) \\ c(\nabla_t \Phi(x, \tilde{t}) - \nabla_t \Phi(x, t)) + (\tilde{c} - c)\nabla_t \Phi(x, \tilde{t}) \end{pmatrix}.$$

Here p denotes the vector of the parameters (c, t). Using the triangle inequality we can estimate the left hand side of equation (4.7c) as

$$\begin{aligned} \|Q(\tilde{p}, p)\| = \left\|Q(\tilde{c}, \tilde{t}, c, t)\right\| &\leq \left\|\Phi(x, \tilde{t}) - \Phi(x, t)\right\| \\ &\quad + |c| \left\|\nabla_t \Phi(x, \tilde{t}) - \nabla_t \Phi(x, t)\right\| + |\tilde{c} - c| \left\|\nabla_t \Phi(x, \tilde{t})\right\|. \end{aligned} \tag{4.35}$$

As in the proof of Theorem 4.15 we estimate the first and the second part of this relation using the integral representation of error (4.23). We demonstrate this method only for the second part, where we obtain

$$\begin{aligned} &|c| \left\|\nabla_t \Phi(x, \tilde{t}) - \nabla_t \Phi(x, t)\right\| \\ &\quad = |c| \left\| \int_0^1 \nabla_t \big(\nabla_t \Phi\big)(x, t + (\tilde{t} - t)\xi)(\tilde{t} - t)\, d\xi \right\| \\ &\quad \leq |c| \left\|\tilde{t} - t\right\| \int_0^1 \left\|\nabla_t \big(\nabla_t \Phi\big)(x, t + (\tilde{t} - t)\xi)\right\| d\xi \\ &\quad \leq |c| \left\|\tilde{t} - t\right\| m_{\Phi''}(t, \tilde{t}), \end{aligned}$$

with $m_{\Phi''}(t, \tilde{t})$ defined as in the proof of Theorem 4.15. To eliminate the dependence of our estimate on $\tilde{t}$ and t (now $t^\dagger$ is the central point) we distinguish the two cases where either $\nabla_t \big(\nabla_t \Phi\big)(x, t^\dagger) \neq 0$ or $\nabla_t \big(\nabla_t \Phi\big)(x, t^\dagger) = 0$ as we did in Theorem 4.15 and obtain the estimate

$$m_{\Phi''}(t, \tilde{t}) \leq \max\{1, 2 \left\|\nabla_t \big(\nabla_t \Phi\big)(x, t^\dagger)\right\|\} =: M_{\Phi''}(t^\dagger)$$

for t and $\tilde{t}$ sufficiently close to $t^\dagger$.

The last part in equation (4.35) can be estimated by $m_{\Phi'}(t, \tilde{t})$ and therefore further with $M_{\Phi'}(t^\dagger)$ as defined in Theorem 4.15 in equation (4.26) and (4.27).

Altogether there exists a neighbourhood of $t^\dagger$ such that

$$\begin{aligned}
\|Q(\tilde{p},p)\| &= \left\|Q(\tilde{c},\tilde{t},c,t)\right\| \\
&\le \left\|\tilde{t}-t\right\| m_{\Phi'}(t,\tilde{t}) + |c|\left\|\tilde{t}-t\right\| m_{\Phi''}(t,\tilde{t}) + |\tilde{c}-c| M_{\Phi'}(t^\dagger) \\
&\le \left\|\tilde{t}-t\right\| \left(M_{\Phi'}(t^\dagger) + |c| M_{\Phi''}(t^\dagger)\right) + |\tilde{c}-c| M_{\Phi'}(t^\dagger) \\
&\le \left(2M_{\Phi'}(t^\dagger) + |c| M_{\Phi''}(t^\dagger)\right) \max\{\left\|\tilde{t}-t\right\|, |\tilde{c}-c|\}.
\end{aligned}$$

Now we investigate the right hand side of equation (4.7c). It is given as

$$\begin{aligned}
\left\|F'(p^\dagger)(\tilde{p}-p)\right\| &= \left\|F'(c^\dagger,t^\dagger)(\tilde{c}-c,\tilde{t}-t)\right\| \\
&= \left\|(\tilde{c}-c)\Phi(x,t^\dagger) + c^\dagger(\tilde{t}-t)\nabla_t\Phi(x,t^\dagger)\right\| \\
&= \left\|(\tilde{c}-c)\Phi(x,t^\dagger) + \left\|\tilde{t}-t\right\| c^\dagger \frac{\tilde{t}-t}{\left\|\tilde{t}-t\right\|}\nabla_t\Phi(x,t^\dagger)\right\| \\
&= \left\|(\tilde{c}-c)\Phi(x,t^\dagger) + \left\|\tilde{t}-t\right\| c^\dagger \mathbf{e}(\tilde{t},t)\nabla_t\Phi(x,t^\dagger)\right\|.
\end{aligned}$$

We can now complete the proof, using Theorem 4.13 and Lemma 4.9. Therefore we set $A = c^\dagger \mathbf{e}(\tilde{t},t)\nabla_t\Phi(x,t^\dagger)$ and $B = \Phi(x,t^\dagger)$ and obtain from Theorem 4.13 that the conditions for Lemma 4.9 are fulfilled. With the setting $\alpha = \left\|\tilde{t}-t\right\|$, $\beta = |\tilde{c}-c|$ and $\kappa = 2M_{\Phi'}(t^\dagger) + |c|M_{\Phi''}(t^\dagger)$ we obtain from Lemma 4.9 that there exists a constant C_Q such that condition (4.7c) is fulfilled.

This estimate holds independently from the actual values of α and β as long as the underlying values for p and $\tilde{p}$ are close enough to $p^\dagger$ (so that the estimates for $M_{\Phi'}$ and $M_{\Phi''}$ hold).

Altogether we have shown that the convergence criterion (4.7) is fulfilled. □

In contrast to Theorem 4.15 we are not able to prove this result for the case $c^\dagger = 0$, because for the choice $\tilde{c} = c$ the right hand side of equation (4.7c) vanishes, whereas the left hand side will not vanish in general.

The next theorem (Theorem 2.5 in [Kal97] for the choice $\nu = 0$ and $\beta = 1$) shows that condition (4.7) implies local convergence of the iteratively regularized Gauß-Newton method.

Theorem 4.20. *Let the iteratively regularized Gauß-Newton method be defined via*

$$p_{k+1} = p_k - (F'(p_k)^*F'(p_k) + \alpha_k I)^{-1}\big(F'(p_k)^*(F(p_k)-y) + \alpha_k(p_k-p_0)\big),$$

where p_0 is an initial guess for a solution of (4.31) *and the regularization parameters α_k are chosen such that*

$$\alpha_k > 0, \quad 1 \le \frac{\alpha_k}{\alpha_{k+1}} \le r, \quad \lim_{k\to\infty}\alpha_k = 0$$

for some $r > 1$. Let condition (4.7) be fulfilled in the ball $B(p_0, \rho)$ for some sufficiently small radius ρ. Furthermore, let problem (4.31) be solvable within $B(p_0, \frac{\rho}{2})$. Then the iterates p_k converge to a solution $p_* \in B(p_0, \frac{\rho}{2})$ of (4.31).

If we combine the results from Theorems 4.19 and 4.20 we obtain the following result for neural networks.

Corollary 4.21. *Let the activation function Φ fulfill the standard Assumptions 4.6 and the function f be representable by a (nonzero) neural network consisting of single node, i. e., $f = F(p_*)$, $p_* \in \mathbb{R}\backslash\{0\} \times P$. Then the iterates p_k computed with the iteratively regularized Gauß-Newton method converge to a solution p_*, if the initial guess p_0 is chosen sufficiently close to p_*.*

So we have shown that on the one hand Landweber's and Newton's method may fail to reduce the error when applied to a network consisting of two or more nodes (Section 4.3). On the other hand we have shown that for both methods and attainable data the iterates converge locally to solutions of problem (4.31).

In Chapter 5 we will apply these methods to single nodes, but to problems that do not have a solution, i. e., where the right hand side is not attainable. In the setting that we have later we use these methods to minimize a quadratic functional, where we know that there exist elements p_* such that $\|F(p_*) - y\|^2 < \frac{M}{k}$ and seek for an element p which at least fulfills $\|F(p) - y\|^2 = \frac{M}{k}$. In the following we explain why we also may expect good convergence properties for this case.

Remark 4.22. For the nonlinear operator equation

$$F(p) = y, \quad p \in \mathcal{D}(F) \subset X,\, y \in Y$$

there exist several convergence results such as the theorems of Kantorovich and Ostrowski, ensuring convergence of the iterates if the initial value p_0 is sufficiently close to a solution p_* and F fulfills various assumptions. For the optimization problem

$$\|F(p) - y\|^2 \to \min_p \tag{4.36}$$

such results do not exist in general. Typical results in standard optimization theory (cf. e. g. [Wal75, Cea78, BO75]) show for instance that points to which the algorithm converges have to be stationary points. For another class of results it is assumed that the algorithm converges to some local minimum and convergence rates are shown. But unfortunately, there are in general *no* results of the form "if the starting value p_0 is sufficiently close to the optimum p_*, then the iterates p_k converge to p_*".

The reason for this is, that for the investigation of the first problem it is sufficient to look at properties of F, whereas for the second problem also information about the difference $y - F(p_*)$ is needed. Usually no such additional information is present, this means, the difference can look arbitrary bad and the minimization functional can be full of local minima. Typically an optimization algorithm follows some descent direction towards the nearest stationary point, and therefore there *cannot* be any general convergence result of the form stated above.

There are several reasons why this need not be a problem for us.

- Both methods we investigated are *descent methods*. It is a standard result in optimization theory that these converge to local minima.
- We do not need convergence to a minimum of (4.36), but only to a point where the functional $\|F(p) - y\|^2$ is sufficiently small.
- We have the additional knowledge that the difference $y - F(p_*)$ has to be an element of the set $\overline{\text{co}}(G_b)$ (see Chapter 5), which is generated by the activation function Φ and is a set of smooth functions (Section 5.4.4).

Especially the last point is of interest. In Theorem 4.18 we presented a result showing that at least for the first few steps the iterates have to move towards the optimal parameter choice. For our case this result might be improved, since Theorem 4.18 holds for any function y^δ with $\left\|y - y^\delta\right\| \leq \delta$. So it seems possible, that this additional information can be utilized to prove convergence results of the desired form. Nevertheless, deriving such results is beyond the scope of this thesis and will be postponed for future work.

Chapter 5

Greedy Approximation

We will now utilize the results from the previous section to derive an iterative algorithm for training a neural network, which preserves the good approximation properties (see Section 1.6) and can be implemented efficiently.

As we have seen in the Sections 4.3.2 and 4.3.3 Landweber's and Newton's method may fail if they are applied to a neural network that consists of two or more nodes. In contrast we have shown that both methods satisfy sufficient conditions for convergence if the neural network consists of only a single node.

The algorithm we shall present utilizes this property and increases the size of the network step by step by one neuron each. It will therefore enable us to use Landweber's and Newton's method in each such step. In each step of the iteration we will seek for a neuron that approximates the objective almost optimal (with respect to the accuracy that can be obtained with the current number of nodes), therefore we will call this procedure *greedy algorithm*. This is somehow similar to a multilevel scheme presented in [DES98], although there the iterations are done in finite dimensional subspaces, whereas here we seek for solutions in infinite dimensional manifolds.

Usually a greedy algorithm gives the advantage of easy implementation at the cost of not always finding the optimal solution. Nevertheless, in our case this does not happen, but the greedy algorithm maintains the *dimension independent* convergence rate, and behaves therefore optimally. This means that we are able to combine the advantages of both, the convergence properties of iterative training methods and the approximation capabilities of neural networks.

First of all some results about convex approximation are presented. This will be done in a very general and abstract setting. Then we present the general algorithm and investigate the influence of noise in the data on the greedy algorithm. Finally we show explicitly how this algorithm can be implemented to train a neural network and give results about convergence properties in stronger Sobolev norms.

Sections 5.1 and 5.2 concern the noise free case in an *abstract setting* and follow the lines of [DS96].

5.1 Preliminaries

In the following let G be a subset of an inner product space H with induced norm $\|\cdot\|$. Furthermore let the elements of G be bounded in the norm by some constant b, which may be abbreviated as $G \subset B(0;b)$, and let $\overline{\text{co}}(G)$ denote the closure of the convex hull of G.

We assume that the function f to be approximated is an element of $\overline{\text{co}}(G)$. For the further analysis we define a constant γ via

$$\gamma = \inf_{v \in H} \sup_{g \in G} \left(\|g - v\|^2 - \|f - v\|^2 \right). \tag{5.1}$$

This constant is in some sense a measure for the number of different elements of G that are needed to represent f. The value of γ can be bounded above by $b^2 - \|f\|^2$ since $0 \in H$. If the norm of f is close to the bound b and therefore f is close to the boundary of $\overline{\text{co}}(G)$ the value of γ will be very small. In this case f can be represented by few different elements of G.

The constant γ provides an estimate for the rate of convex approximation, as we will see in the following lemma [DS96, Lemma 2]:

Lemma 5.1. *Let $G \subset B(0;b)$, $f \in \overline{\text{co}}(G)$, $h \in \text{co}(G)$ and let γ be defined via* (5.1)*. Then the estimate*

$$\inf_{g \in G} \|f - \lambda h - (1-\lambda)g\|^2 \leq \lambda^2 \|f - h\|^2 + (1-\lambda)^2 \gamma$$

holds for $\lambda \in [0,1]$.

5.2 Greedy Approximation without Noise

Now we present an iterative algorithm for function approximation, which can be applied to neural networks easily, and for which convergence can be

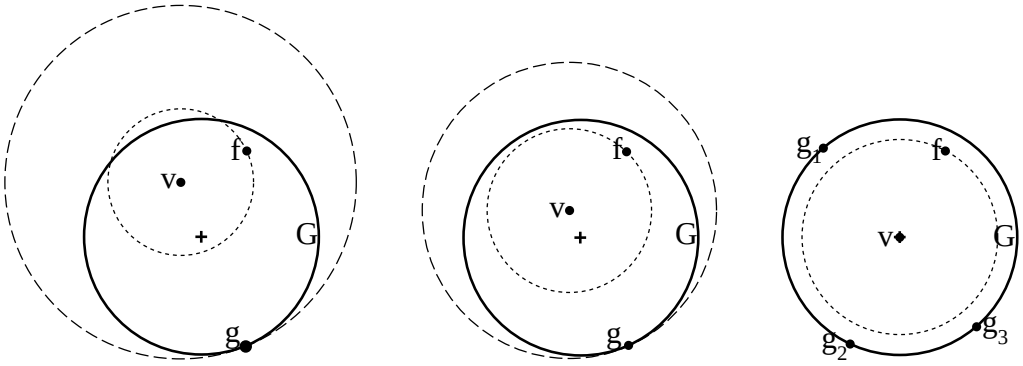


Figure 5.1: Interpretation of Definition (5.1). The objective function is the difference of the radii of the two dashed circles. In this symmetric case the infimum is attained for v lying in the center of G.

shown.

Algorithm 5.2. Greedy approximation in the noise free case

Initialization:

Choose a constant M, such that $M > \gamma$ (as defined in (5.1)).
Choose a positive sequence ε_k, tending to zero that fulfills

$$\varepsilon_k \leq \frac{M-\gamma}{k^2} \qquad \text{for} \quad k = 1, 2, \ldots$$

Find an initial approximation $g_1 \in G$ such that

$$\|f - g_1\|^2 \leq \inf_{g \in G} \|f - g\|^2 + \varepsilon_1$$

is fulfilled and define $f_1 = g_1$.

Iteration:

for $k := 2$ **to** maxit **do**

Find an element $g_k \in G$ such that

$$\left\| f - \frac{k-1}{k} f_{k-1} - \frac{1}{k} g_k \right\|^2 \leq \inf_{g \in G} \left\| f - \frac{k-1}{k} f_{k-1} - \frac{1}{k} g \right\|^2 + \varepsilon_k$$

is fulfilled and define f_k as

$$f_k = \frac{k-1}{k} f_{k-1} + \frac{1}{k} g_k \,.$$

end for

We will call Algorithm 5.2 *greedy algorithm*. Observe that in each step only *one* element of G is chosen, the other components of f_k are fixed. In the language of neural networks each step can be interpreted as approximation of a function (e. g. $f - \frac{k-1}{k} f_{k-1}$) with a neural network consisting of only a single node. Nevertheless with the greedy algorithm still the dimension independent convergence rate is obtained, as can be seen in the next theorem (cf. [DS96]):

Theorem 5.3. *Let the conditions of Lemma 5.1 be satisfied. Then the approximating functions f_k generated by Algorithm 5.2 fulfill the error estimate*

$$\|f - f_k\|^2 \leq \frac{M}{k}\,. \tag{5.2}$$

Proof. For the proof we will use an induction argument based on Lemma 5.1. Therefore we first look at the initialization step $k = 1$. According to the algorithm we have chosen f_1 such that the estimate

$$\|f - f_1\|^2 \leq \inf_{g \in G} \|f - g\|^2 + \varepsilon_1$$

holds. If we use $\varepsilon_1 \leq (M - \gamma)$ and choose $\lambda = 0$ in Lemma 5.1 we obtain

$$\begin{aligned}\|f - f_1\|^2 &\leq \inf_{g \in G} \|f - g\|^2 + (M - \gamma) \\ &\leq \gamma + (M - \gamma) = M,\end{aligned}$$

hence, the estimate (5.2) holds for $k = 1$.

Now we use induction to show that the estimate (5.2) is valid in the kth step if it was valid in the $k-1$st step. For this sake we suppose that the estimate (5.2) holds for $k - 1$, i. e.,

$$\|f - f_{k-1}\|^2 \leq M/(k-1)$$

is satisfied. According to Algorithm 5.2 f_k is chosen such that

$$\|f - f_k\|^2 \leq \inf_{g \in G} \left\| f - \frac{k-1}{k} f_{k-1} - \frac{1}{k} g \right\|^2 + \varepsilon_k$$

holds. Using the definition of ε_k and Lemma 5.1 with $\lambda = \frac{k-1}{k}$ we obtain

$$\begin{aligned}\|f - f_k\|^2 &\leq \left(\frac{k-1}{k}\right)^2 \|f - f_{k-1}\|^2 + \frac{1}{k^2}\gamma + \varepsilon_k \\ &\leq \frac{M(k-1) + \gamma}{k^2} + \varepsilon_k \\ &= \frac{M}{k} - \frac{M - \gamma}{k^2} + \varepsilon_k = \frac{M}{k},\end{aligned}$$

this means the estimate (5.2) is fulfilled in the kth step of the iteration. □

In Section 5.4.2 we will apply this abstract algorithm to neural networks and modify it such that it becomes *feasible*. In the next section the influence of noise on the greedy algorithm is investigated.

5.3 Greedy Approximation with Noise

Now we modify the greedy algorithm such that we can use it for noisy data. Since we cannot guarantee that f^δ also lies in the closed convex hull of G, we have to introduce the projection of f^δ onto $\overline{\text{co}}(G)$.

The modification that we will finally find for the greedy algorithm from the section before is a *stopping rule*, i. e., a rule that tells us after how many iterations k we have to terminate the algorithm. This is a typical result for iterative regularization methods (see [EHN96, Chap. 6]), here the iteration index plays the same role as the regularization parameter α in Tikhonov regularization, the stopping rule corresponds to the parameter selection method.

In general the stopping rule is a function of the noise level δ, but it may also depend on the noisy data f^δ. The rule we present is an *a-priori stopping rule* and is a function of the noise level δ and the projection Pf^δ.

5.3.1 Projection

The projection is defined as the element in $\overline{\text{co}}(G)$ which is closest to f^δ. Since $\overline{\text{co}}(G)$ is by definition convex, this element is *unique*. We denote the (nonlinear) operator that maps a function to its projection with P and have the equation

$$\left\|f^\delta - Pf^\delta\right\| = \inf_{h\in\overline{\text{co}}(G)} \left\|f^\delta - h\right\|.$$

For function approximation the projection has useful properties which will be needed later. It should be mentioned that we will need all these properties only inside the proofs but not in the algorithm. The projection itself never has to be computed, only the related value γ^{Pf^δ} (see below) must be estimated.

One property is that elements f_n^δ of $\overline{\text{co}}(G)$ that approximate f^δ are always even better approximations to Pf^δ as can be seen in the next lemma:

Lemma 5.4. *Let $f_n^\delta \in \overline{\text{co}}(G)$, $f^\delta \in H$ and Pf^δ denote the projection of f^δ onto $\overline{\text{co}}(G)$. Then the estimate*

$$\left\|Pf^\delta - f_n^\delta\right\| \leq \left\|f^\delta - f_n^\delta\right\|$$

is fulfilled.

Proof. For the characterization of a projection onto a convex set we have the relation

$$\begin{aligned} &\left\|f^\delta - Pf^\delta\right\| = \inf_{h\in\overline{\text{co}}(G)} \left\|f^\delta - h\right\| \\ \iff &\left\langle f^\delta - Pf^\delta, Pf^\delta - h\right\rangle \geq 0 \quad \forall h \in \overline{\text{co}}(G). \end{aligned}$$

The proof of this relation is simple and can be found e. g. in [Wer95, Chapter 5.3]. We use this to find an estimate for $\left\|Pf^\delta - f_n^\delta\right\|$. We start with an approximation to f^δ and find

$$\begin{aligned} \left\|f^\delta - f_n^\delta\right\|^2 &= \left\|f^\delta - Pf^\delta\right\|^2 + \left\|Pf^\delta - f_n^\delta\right\|^2 + 2\underbrace{\left\langle f^\delta - Pf^\delta, Pf^\delta - f_n^\delta\right\rangle}_{\geq 0} \\ &\geq \left\|f^\delta - Pf^\delta\right\|^2 + \left\|Pf^\delta - f_n^\delta\right\|^2 \\ &\geq \left\|Pf^\delta - f_n^\delta\right\|^2, \end{aligned} \tag{5.3}$$

so an approximation to f^δ is also an approximation to Pf^δ and the proof is completed. □

Analogously to the noise free case we now define

$$\gamma^{Pf^\delta} = \inf_{v\in H} \sup_{g\in G} \left(\|g - v\|^2 - \left\|Pf^\delta - v\right\|^2\right), \tag{5.4}$$

which is a measure for how many different elements of G have to be used to represent Pf^δ.

Lemma 5.1 together with an application of the triangle inequality yields:

Corollary 5.5. *Let G be as above, γ^{Pf^δ} defined by equation (5.4) and h be an element of the convex hull of G. Further let f^δ be an arbitrary element of the Hilbert space H and Pf^δ denote its projection onto $\overline{\text{co}}(G)$.*

Then the estimate

$$\begin{aligned} \inf_{g\in G} \left\|f^\delta - \lambda h - (1-\lambda)g\right\|^2 \leq \delta^2 + \lambda^2 \left\|Pf^\delta - h\right\|^2 + (1-\lambda)^2\gamma^{Pf^\delta} \\ + 2\delta\sqrt{\lambda^2 \left\|Pf^\delta - h\right\|^2 + (1-\lambda)^2\gamma^{Pf^\delta}} \end{aligned} \tag{5.5}$$

holds for all $\lambda \in [0,1]$.

Observe that the element f^δ does not necessarily belong to the closed convex hull of G but can be chosen arbitrarily in H.

5.3.2 Application to Noisy Data

Now we investigate the behaviour of the algorithm when applied to a function f^δ that is not necessarily in the closed convex hull of G. To this end we inspect the proof of Theorem 5.3.

Our aim is to derive a modified version of Algorithm 5.2. Again we try to prove the rate $\frac{M}{k}$ by induction

- For the step $k = 1$ we now obtain using Corollary 5.5

$$\begin{aligned}\left\|f^\delta - f_1^\delta\right\|^2 &\leq \inf_{g\in G}\left\|f^\delta - g\right\|^2 + \varepsilon_1\\ &\leq \delta^2 + \gamma^{Pf^\delta} + 2\delta\sqrt{\gamma^{Pf^\delta}} + \varepsilon_1\\ &\leq M\end{aligned}$$

  if ε_1 is chosen such that

$$\varepsilon_1 \leq M - \left(\delta + \sqrt{\gamma^{Pf^\delta}}\right)^2$$

  holds. Such a choice is possible as long as M is chosen larger than $(\delta + \sqrt{\gamma^{Pf^\delta}})^2$.

- Now we inspect the case $k > 1$. We assume that the convergence rate was preserved up to this step of the iteration, this means that the estimate $\left\|f^\delta - f_{k-1}^\delta\right\|^2 < \frac{M}{k-1}$ holds. From Lemma 5.4 we know that this estimate remains valid if we replace f^δ with Pf^δ.

$$\begin{aligned}\left\|f^\delta - f_k^\delta\right\|^2 &\leq \inf_{g\in G}\left\|f - \frac{k-1}{k}f_{k-1}^\delta - \frac{1}{k}g\right\|^2 + \varepsilon_k\\ &\leq \left(\delta + \sqrt{\left(\frac{k-1}{k}\right)^2\left\|Pf^\delta - f_{k-1}^\delta\right\|^2 + \frac{1}{k^2}\gamma^{Pf^\delta}}\right)^2 + \varepsilon_k\\ &\leq \left(\delta + \sqrt{\frac{k-1}{k^2}M + \frac{1}{k^2}\gamma^{Pf^\delta}}\right)^2 + \varepsilon_k\\ &\leq \frac{M}{k}\end{aligned}$$

  if ε_k is chosen such that

$$\varepsilon_k \leq \frac{M}{k} - \left(\delta + \sqrt{\frac{k-1}{k^2}M + \frac{1}{k^2}\gamma^{Pf^\delta}}\right)^2. \tag{5.6}$$

For $k \to \infty$ the right hand side tends to $-\delta^2$, it becomes negative. Since ε_k has to be chosen larger than zero, the algorithm will have to terminate after a number of steps, depending on the magnitude of M, δ and γ^{Pf^δ}. For a given function f^δ also the values of δ and γ^{Pf^δ} are fixed. Therefore the iteration index k for which ε_k becomes negative depends only on the magnitude of M.

5.3.3 Optimal Parameter Choices

If we denote the index where ε becomes negative by $k_{\max}$, then we can estimate the residual with

$$\left\| f^\delta - f^\delta_{k_{\max}} \right\|^2 \leq \frac{M}{k_{\max}}$$

so it is desired to find a combination of M and the corresponding $k_{\max}$ such that the right hand side of this equation becomes minimal.

To find this optimal combination (for given values of δ and γ^{Pf^δ}) we can for instance use the software system *Mathematica*. First we look at the zeroes of equation (5.6) i. e., we investigate the equation

$$\frac{M}{k} - \left(\delta + \sqrt{\frac{k-1}{k^2} M + \frac{1}{k^2} \gamma^{Pf^\delta}} \right)^2 = 0 \,.$$

Since this equation is of degree 6 with respect to k and only of degree 2 with respect to M we will first look for solutions for M and search for the minimum with respect to k afterwards. This yields two solutions $M_1 > M_2$, of which only M_1 is of interest, since M_2 is not a solution for $k \geq 1$. The remaining solution is given as

$$M_1 = g + (2k-1)\delta^2 k^2 + 2\delta k^{\frac{3}{2}} \sqrt{g + (k-1)k^2\delta^2} \,. \tag{5.7}$$

For the solution M_1 we try to find the optimal value for k, this means we try to solve the minimization problem

$$\frac{M_1(\gamma^{Pf^\delta}, \delta, k)}{k} \to \min_k \,. \tag{5.8}$$

To find stationary points we differentiate the equation with respect to k and search for zeroes. *Mathematica* finds 6 stationary points, from which one is negative and two others are not real numbers. The remaining 3 stationary

points depend only on the ratio of γ^{Pf^δ} and δ^2, for this reason we define the positive constant

$$\nu := \frac{\sqrt{\gamma^{Pf^\delta}}}{\delta} \tag{5.9}$$

to simplify the equations. For small values of ν (exactly for $0 \leq \nu < \frac{2}{3\sqrt{3}}$) two other solutions become complex numbers and so only a single solution remains. In general the solution k_3 seems to be a maximum of (5.8) and not a minimum, but we could not prove this.

To abbreviate the formulas for k_2 and k_3 we define

$$a = -2 + 27\nu^2 + 3\sqrt{3}\nu\sqrt{27\nu^2 - 4}$$

$$b = \frac{\nu}{\sqrt{6}} \sqrt{-4 - \frac{2\,2^{\frac{1}{3}}}{a^{\frac{1}{3}}} - 2^{\frac{2}{3}}\, a^{\frac{1}{3}} + 48\,\nu^2 + \frac{12\sqrt{6}\,\nu\,(8\,\nu^2 - 1)}{\sqrt{-2 + \frac{2\,2^{\frac{1}{3}}}{a^{\frac{1}{3}}} + 2^{\frac{2}{3}}\, a^{\frac{1}{3}} + 24\,\nu^2}}}.$$

With these auxiliary values we can express the three stationary points as

$$k_1 = \nu \tag{5.10a}$$

$$k_{2/3} = 2\,\nu^2 + \frac{\nu}{\sqrt{6}} \sqrt{-2 + \frac{2\,2^{\frac{1}{3}}}{a^{\frac{1}{3}}} + 2^{\frac{2}{3}}\, a^{\frac{1}{3}} + 24\,\nu^2} \mp b\,. \tag{5.10b}$$

As long as $\nu \geq \frac{2}{3\sqrt{3}}$ all these are positive and real valued. If ν decreases below this bound then only the solution k_1 remains. This stopping rule cannot be improved if we use the estimate (5.5), but for special architectures it might be possible to obtain a sharper estimate than (5.5) and consequently a better stopping criterion. Deriving such estimates is beyond the scope of this thesis and will be postponed to future work.

5.3.4 A Modified Algorithm for Noisy Data

Now we briefly collect the results from the preceding sections and formulate them as a modified version of Algorithm 5.2.

Algorithm 5.6. Greedy approximation in the case of noisy data

Initialization:

- Calculate $\nu(\delta, \gamma^{Pf^\delta})$ as defined in (5.9).
- Compute the 3 different solutions for k and the corresponding values for M_i via (5.10) and (5.7)

- Choose the index i for which $\frac{M(k_i)}{k_i}$ is minimal and set

$$M_{\text{opt}} := M_i, \qquad k_{\text{opt}} := k_i \,.$$

- Choose a positive sequence ε_k that fulfills

$$\varepsilon_k \leq \frac{M_{\text{opt}} - \gamma}{k^2} \qquad \text{for} \quad k = 1, \ldots, k_{\text{opt}} \,.$$

- Set $f_0^\delta := 0$.

Iteration:

for $k := 1$ **to Min**$(k_{\text{opt}}$, maxit$)$ **do**

Find an element $g_k^\delta \in G$ such that

$$\left\| f^\delta - \frac{k-1}{k} f_{k-1}^\delta - \frac{1}{k} g_k^\delta \right\| \leq \inf_{g \in G} \left\| f^\delta - \frac{k-1}{k} f_{k-1}^\delta - \frac{1}{k} g \right\| + \varepsilon_k$$

is fulfilled and define f_k as

$$f_k^\delta = \frac{k-1}{k} f_{k-1}^\delta + \frac{1}{k} g_k^\delta \,.$$

end for

Observe that the step for $k = 1$ is done the same way as in Algorithm 5.2 and is only written in a different way.

For decreasing noise level also $\frac{M_{\text{opt}}}{k_{\text{opt}}}$ tends to zero, where the rate is given (numerically) as

$$\frac{M_{\text{opt}}}{k_{\text{opt}}} = \mathcal{O}\left(\delta^{2/3}\right) \,.$$

Consequently the difference $\left\| f^\delta - f_k^\delta \right\|$ converges to zero as $\mathcal{O}\left(\delta^{1/3}\right)$ and, since $\left\| f - f^\delta \right\| \leq \delta$, the difference $\left\| f - f_k^\delta \right\|$ converges with the same rate (see also Figure 6.9 and the corresponding comments in Section 6.1.2).

5.4 Application to Neural Networks

Now we combine the results from Chapter 4 on iterative methods with the results from Section 5.2 on greedy approximation to derive an algorithm for the

training of neural networks. For this algorithm we are able to prove convergence of the output functions in the L^2-norm and, under weak assumptions, also in stronger norms.

First we introduce some notations in order to apply the results of the preceding sections to neural networks. Then we formulate the algorithm and discuss in Section 5.4.3 how the bounds for c and t can be implemented numerically. Finally we inspect the convergence behaviour of the greedy algorithm in stronger Sobolev norms.

5.4.1 Notations

In order to apply the results of the previous sections to neural networks we have to to concretize the abstract setting. The natural Hilbert-space H seems to be the Lebesgue-space $L^2(\Omega)$. As subset G we choose the set

$$G_b = \{c\Phi(\cdot, t) \mid |c| \leq b,\, t \in P\} \subset L^2(\Omega)\,,$$

where Φ is the activation function of the network and P is the compact set of parameters. The set G_b can be interpreted as the set of all possible nodes of the network. If the function Φ is scaled such that its L^2-norm is bounded above by 1 uniformly in t, i. e.,

$$\int_\Omega |\Phi(x,t)|^2 d\,x \leq 1 \qquad \forall t \in P\,, \tag{5.11}$$

then G_b is bounded and $G_b \subset B(0;b)$. For the sake of simplicity we assume (5.11), otherwise one can use the bound $\tilde{b} = b/\sup_{t\in P} \|\Phi(\cdot,t)\|_{L^2(\Omega)}$ for the factor c to bound the norm of the elements of G_b by b. Observe that $\tilde{b}$ can not be zero because the set P is compact. The value γ is now given as

$$\gamma = \inf_{v\in L^2} \sup_{|c|\leq b,\, t\in P} \|c\Phi(x,t) - v\|^2 - \|f - v\|^2\,. \tag{5.12}$$

The convex hull of the set G_b is defined as

$$\mathrm{co}(G_b) = \{f \in L^2(\Omega) \mid f = \sum_{i=1}^{n} c_i \Phi(\cdot, t_i),\, \sum |c_i| \leq b,\, t \in P, n \in \mathbb{N}\},$$

the sign of the parameters c_i does not matter, because the original set G_b is symmetric. Further the sum need not be equal to b but can also be smaller,

because the zero function is an element of G_b. If we compute the closure of this set the sums turn into integrals and we find

$$\overline{\text{co}}(G_b) = \{f \in L^2(\Omega) \mid f = \int_P \Phi(\cdot, t) d\mu(t),\ \mu \in \mathcal{M}_1,\ \|\mu\|_{\mathcal{M}_1} \leq b\}.$$

Here $\mathcal{M}_1$ denotes the set of all *Radon measures* (see e. g. [Lin83]). Note, that $\overline{\text{co}}(G_b)$ contains all functions having a representation of the form $\sum^n c_i \Phi(\cdot, t_i)$ and $\int_P c(t)\Phi(\cdot, t)\, dt$.

Using this setting we are able to reformulate Algorithm 5.2 for the case of neural networks. Since in each step only the weights of *one* neuron are changed, according to the results of Section 4 we can use a combination of Landweber iteration and Newton's method to find the optimal weights.

5.4.2 A Greedy Algorithm for Neural Networks

In order to apply Algorithm 5.2 to neural networks, it is necessary that the function f that shall be approximated is an element of the closed convex hull $\overline{\text{co}}(G_b)$. For a given function f this can often be assured by choosing b sufficiently large. Nevertheless, to obtain good convergence rates (i. e., a small value for M) it is preferable that the norm of f is close to b. This can only be ensured by a proper choice of the activation function Φ depending on the specific structure of f.

Algorithm 5.7. Greedy approximation with neural networks.

Initialization:

- Choose a constant M, such that $M > \gamma$ (as defined in (5.12)), for example choose $M = \frac{3}{2}(b^2 - \|f\|^2)$.
- set $f_0 = 0$.

Iteration:

for $k := 1$ **to** maxit **do**

if $\|f - f_{k-1}\|^2 \leq \frac{M}{k}$ **then** set $f_k := f_{k-1}$ and jump to the next step of the iteration.

Find parameters c and t such that

$$\left\| f - \frac{k-1}{k} f_{k-1} - \frac{1}{k} c\Phi(\cdot, t) \right\|^2 \leq \frac{M}{k} \qquad (5.13)$$

is fulfilled (see Remark 5.8) and define f_k as

$$f_k := \frac{k-1}{k} f_{k-1} + \frac{1}{k} c\Phi(\cdot, t)\,.$$

end for

This algorithm looks slightly different than the one presented in Section 5.2, the sequence ε_k and the seek for infima are gone. In practice we are usually not able to calculate the infimum formulated in the original algorithm and therefore we are also not able to check if the difference to this infimum is less than ε_k. Nevertheless Algorithm 5.2 in combination with Theorem 5.3 ensures that there *exists* an element, which can give us the rate $\frac{M}{k}$. This means that in each step of the iteration we are able to find an approximation and the algorithm above is feasible. Furthermore, with this modified algorithm we are able to check if the approximation we found is sufficiently good, we only have to look if the estimate (5.13) is fulfilled. Below we describe a possibility to actually find these elements.

Note that we do not have to change f_k in each step. If f_k is already a good approximation to f we increase the index k as much as possible. This can be done without affecting the convergence rate, since for the induction argument in the proof of Theorem 5.3 we only needed that the kth approximation fulfills the rate $\|f - f_k\|^2 \leq M/k$, but not that it consists of k nodes.

Remark 5.8. To find the approximating elements in Algorithm 5.7 we propose the following procedure:

- Set $c = 0$ and choose an initial value for t. First of all perform several Landweber iterations. For attainable data we have shown that this iteration will work if the starting point is sufficiently close to the optimal values for c and t, for non-attainable data this is a standard optimization method.
- Then[1] switch to Newton's method (in form of the iteratively regularized Gauß-Newton method). If Newton's method fails to converge (this may happen if the current approximation c and t is too far away from the optimum), switch back to Landweber iteration.
- Since Landweber's method has better *global* convergence properties and Newton's method has better *local* ones, usually the first iterations will be Landweber steps, the later ones will mainly be Newton steps.

[1]Since Corollary 4.21 about convergence of Newton's method only works for $c \neq 0$ we should not start directly with Newton's method.

If this procedure is repeated for several different initial values for t we are likely to choose a sufficiently good starting value and approach the infimum we are looking for.

Summarizing we find the following advantages of Algorithm 5.7 compared to Algorithm 5.2:

- In each step there exists an element g_k that fulfills (5.13), namely at least one of those that fulfill the infimum-condition in Algorithm 5.2.
- We are able to check if the approximation we found is sufficiently good, since everything in relation (5.13) can be computed, as opposed to the infimum-condition which can not be checked in practice.
- Since we are looking only for a single node in each step, we can utilize Landweber's and Newton's method to compute the element g_k. Both are efficient optimization methods and for attainable data we have shown local convergence in Corollaries 4.17 and 4.21.

Remark 5.9. Observe that by implementing a stopping rule as in Section 5.3.4, Algorithm 5.7 can be modified to be applicable to noisy data. To this end as in Algorithm 5.6 the values k_{opt} and M_{opt} are computed and the **for**-loops are changed from

for $k := 1$ **to** maxit to **for** $k := 1$ **to Min**$(k_{\text{opt}}, \text{maxit})$

The rest of the algorithm remains unchanged. As before, for this algorithm we obtain the convergence rate $\left\|f - f_k^\delta\right\| = \mathcal{O}(\delta^{1/3})$.

5.4.3 Modified Neural Network

According to the standard Assumptions 4.6 the parameters t must be restricted to compact sets. Within the greedy algorithm only elements $g \in G_b$ may be chosen, therefore also the parameter c has to be bounded. We now present a method for implementing these bounds in the greedy algorithm. Therefore we first inspect the case that due to the iteration the parameter t was increased too much and has therefore left the set P.

In the case of radial basis functions the parameter t describes the center of the network function. The parameter set P is therefore chosen approximately as the domain Ω. If t is chosen far outside from P then, for instance in the case of Gaussian functions, the norm of $\Phi(x,t)$ will be very small, since only the part inside Ω contributes to the norm. The corresponding functions Φ and its derivative $\nabla_t \Phi$ are very close to the zero function and if t is sufficiently far

outside the domain they are numerically equal to the zero function. Therefore also the operator F'^* will be zero and Landweber iteration as well as Newton's method will stop. Of course the corresponding approximation will be a bad one, but the parameters t stay inside some compact set, even if we do not implement any bound for them.

In the case of ridge constructions a similar effect may happen with respect to the parameter b. If b is too large then the function will be constant along the domain Ω, its derivative will therefore be almost zero and the updates for b vanish.

For the vector a we propose to use a decomposition of a into a unit vector a_0 and a scaling factor s, and to represent a_0 by use of angles in polar coordinates. The angles need not be restricted to a compact set and so the iteration can again be implemented without any additional cost. For the scaling factor s we can use the same procedure as described below for the weight c.

Above we argued that even if t does not stay inside P at least it lies in a sufficiently large compact set $\tilde{P} \supset P$. For the weight c we may not use such arguments, because the corresponding bound b may not be enlarged to a different bound $\tilde{b}$, otherwise we lose the convergence statement of Theorem 5.3. Nevertheless there is a simple possibility to bound the functions in the set G_b, namely to modify the network operator and define it as

$$F(c,t) = \kappa(c)\Phi(x,t), \tag{5.14}$$

where $\kappa(\cdot)$ is an invertible, differentiable mapping of $\mathbb{R}$ to the interval $[-b,b]$ with the properties

$$\lim_{c\to\infty} \kappa(c) = b, \quad \lim_{c\to-\infty} \kappa(c) = -b, \quad \text{and} \quad \kappa(0) = 0\,.$$

The theorems remain valid under these assumptions on $\kappa(\cdot)$. The value of c does not have to stay within a compact set, but the norm of the elements of the set G_b remains bounded by b.

5.4.4 Convergence in Stronger Norms

An interesting property of the greedy algorithm is that it leads to convergence in stronger Sobolev norms. This seems surprising at a first glance, since during the algorithm only the L^2-norm of the elements is observed, also when

performing the Landweber or Newton iterations all adjoints are computed with respect to the L^2-scalar product.

However, the set G_b is *compact* in stronger topologies, which allows us to show convergence and even convergence rates in stronger norms. In the following we assume that

$$\Phi(\cdot, t) \in H^s(\Omega) \quad \forall t \in P.$$

Since the set G_b is compact we can find an upper bound for the H^s-norm of the elements of G_b, which is given via

$$b_s = \sup_{|c| \leq b,\, t \in P} \|c\Phi(\cdot, t)\|_{H^s(\Omega)} < \infty \,. \tag{5.15}$$

Hence, G_b is *bounded* in the H^s-topology.

In the first theorem we show that convergence in the L^2-norm implies weak convergence in the H^s-norm if the activation function $\Phi(\cdot, t)$ belongs to $H^s(\Omega)$. From this we can easily deduce strong convergence in the H^r-norm for $r < s$. Finally we give rates for the convergence in spaces $H^r(\Omega)$.

Theorem 5.10. *Let $\Phi(\cdot, t) \in H^s(\Omega)$ for all values $t \in P$, and let (f_k) be the sequence generated according to Algorithm 5.7. Then the sequence is bounded in $H^s(\Omega)$ and converges weakly in $H^s(\Omega)$ with limit f, i. e., $f_k \rightharpoonup f$.*

Proof. Using equation (5.15) we can conclude that the H^s-norm of any f_k is bounded by

$$\begin{aligned} \|f_k\|_{H^s(\Omega)} &= \left\| \frac{1}{k} g_1 + \cdots + \frac{1}{k} g_k \right\|_{H^s(\Omega)} \\ &\leq \frac{1}{k} \left(\|g_1\|_{H^s(\Omega)} + \cdots + \|g_k\|_{H^s(\Omega)} \right) \\ &\leq \frac{1}{k} k \sup_{1 \leq i \leq k} \|g_i\|_{H^s(\Omega)} \leq b_s \,. \end{aligned} \tag{5.16}$$

Hence, the H^s-norm of the residual is also bounded and can be estimated via

$$\|f - f_k\|_{H^s(\Omega)} \leq \|f\|_{H^s(\Omega)} + \|f_k\|_{H^s(\Omega)} \leq \|f\|_{H^s(\Omega)} + b_s \,.$$

Since the Sobolev space $H^s(\Omega)$ is reflexive and the sequence f_k is bounded we can find a subsequence f_{k_l} which converges weakly to some function f^*. Since there exists a compact embedding operator E from $H^s(\Omega)$ to $L^2(\Omega)$ and the sequence f_k converges in $L^2(\Omega)$ the relation $Ef^* = f$ must be fulfilled. Furthermore $f \in H^s(\Omega)$, because $\overline{\text{co}}(G) \subset H^s(\Omega)$, and thus, $f^* = f$.

Hence, we found that the sequence (f_k) has a weakly converging subsequence f_{k_l} and that the limit of f_{k_l} is equal to f. Since analogous reasoning applies if we start with a subsequence of (f_k), we obtain that every subsequence of (f_k) contains a weakly converging subsequence whose limit is f. Consequently we can conclude that the original sequence (f_k) itself converges weakly to f and the proof is completed. □

We can use this result to prove strong convergence in spaces $H^r(\Omega)$ with $r < s$.

Corollary 5.11. *Let $\Phi(\cdot, t) \in H^s(\Omega)$ for all values $t \in P$, and let (f_k) be the sequence generated according to Algorithm 5.7. Then for $r < s$ the sequence also converges in $H^r(\Omega)$, i. e.,*

$$\|f - f_k\|^2_{H^r(\Omega)} \to 0$$

holds.

Proof. From Theorem 5.10 we know that (f_k) converges weakly to f in $H^s(\Omega)$. For $r < s$ there exists a compact embedding K from the Sobolev spaces $H^s(\Omega)$ to $H^r(\Omega)$. Compact operators transfer weakly converging sequences to norm converging sequences, and therefore f_k converges to f in $H^r(\Omega)$. □

The fact that the approximating functions f_k are bounded in $H^s(\Omega)$ can be used to prove convergence rates in spaces $H^r(\Omega)$ with $r < s$.

Corollary 5.12. *Let $\Phi \in H^s(\Omega)$ and $f \in \overline{\text{co}}(G)$. Then for $r < s$ the convergence rate*

$$\|f - f_k\|^2_{H^r(\Omega)} = \mathcal{O}\left(k^{\frac{r-s}{s}}\right)$$

holds.

Proof. In equation (5.16) in the proof of Theorem 5.10 we have seen that the H^s-norm of the approximating functions is bounded. Further we know that the convergence rate in the L^2-norm is given as $\mathcal{O}(k^{-1})$. If we combine these two results the proof follows immediately using the interpolation inequality (see [EHN96, (2.49)] or [LM72, (2.43)]). □

A special situation arises if the activation function is of class C^∞. In this case we find the same rate of convergence in the H^r-norm as in the L^2-norm, namely $\mathcal{O}(k^{-1})$. Nevertheless, in practice this behaviour will not be

visible, since the constants in the convergence rates can be large, and therefore all these results only hold for k sufficiently large. If we interpret Φ as an element of $H^2(\Omega)$ we obtain the weaker convergence rate $\mathcal{O}(k^{-\frac{1}{2}})$, but the constants will be less. So if we observe the H^1-norm of the residual we will find that it decreases, but not from the start with a high convergence rate, but remaining almost constant at the beginning and then converging with gradually increasing speed. A similar behaviour was observed also with Tikhonov regularization in [BN03].

Chapter 6

Numerical Examples

In this section we verify the theoretical results from Chapter 5 by numerical examples. First of all, we investigate ridge constructions where we inspect the qualitative and quantitative behaviour of the approximations in the L_2- and the H^1-norm during the iteration. Next we examine the influence of noise in the data and compare the numerical results with the prediction, provided by the stopping rule in Section 5.3.3. Finally we investigate the qualitative behavior of the algorithm when applied to radial basis function networks with Gaussian activation functions.

All examples were computed using the software system *Mathematica* on an SGI Origin 3800.

6.1 Ridge Constructions

In the first example we consider a neural network based on a ridge construction for an approximation problem on the 2-dimensional domain $\Omega = [-1, 1] \times [-2, 2]$. As activation function we choose

$$\Phi(x, t) = \Phi(x_1, x_2, t_1, t_2) = \sigma\big((\sin(t_1), \cos(t_1))\,(x_1, x_2)^{\mathrm{T}} + t_2\big)\,,$$

where σ is given as

$$\sigma(\xi) = \frac{1}{1 + \exp(-50\xi)}\,.$$

As proposed in Section 5.4.3 we implement the bound for the parameter c using an auxiliary function κ. The neural network-operator is therefore

defined via

$$F(c, t_1, t_2) = \kappa(c)\Phi(\cdot, t_1, t_2)\,,$$

where the function κ is given as

$$\kappa(c) = \frac{2}{1 + e^{-c}} - 1\,.$$

To ensure that the function to be approximated is an element of $\overline{\text{co}}(G_b)$ we define it explicitly as a convex combination of three elements of G_b namely as

$$f = \frac{F(5, 1, 0.6) + F(-2, 3, 0.3) + F(5, 5, 0.4)}{3}\,. \tag{6.1}$$

A plot of this function can be seen in the upper right corner of Figure 6.1. Using this choice, the function f is an element of $\overline{\text{co}}(G_b)$ for $b = 2.45$. Since $\|f\| = 1.24$ it suffices to choose $M = 5$ according to Algorithm 5.7.

To find parameters satisfying equation (5.13) we set $c^0 = 0$ and take (t_1^0, t_2^0) randomly. Then we perform several iterations of Landweber's and Newton's method until a convergence criterion is fulfilled or the maximal number of iterations is exceeded. To ensure that the approximation we compute is an element of $\overline{\text{co}}(G_b)$ we implement the algorithm such that, as soon as the norm of $\Phi(\cdot, t_1^j, t_2^j)$ is greater than[1] the value $b = 2.45$, the search is terminated and restarted with a different initial value (t_1^0, t_2^0).

6.1.1 Behaviour during the Algorithm

Figure 6.1 illustrates the qualitative behaviour of Algorithm 5.7 in dependence of the iteration index k. Since the number of nodes need not be increased in order to satisfy the estimate (5.13) in each step, the network size (denoted as k_{eff}) has to be less than or equal to k. In our example k is always much larger than k_{eff}. For instance in the third column the network consists of 17 nodes but the error estimate (5.13) is fulfilled for $k = 40$. This means that the *if*-clause in Algorithm 5.7 was true 23 times. This behaviour can be seen in more detail in Figure 6.2. The ratio $\frac{k_{\text{eff}}}{k}$ remains approximately constant during the iteration.

In Figure 6.3 the evolution of the residual, i. e., $\|f - f_k\|_{L_2(\Omega)}$, is shown. The

[1] This is done only because we want to verify our theoretical results, otherwise it does not matter if the iterates are elements of the set G_b, or of a set $G_{\tilde{b}} \supset G_b$.

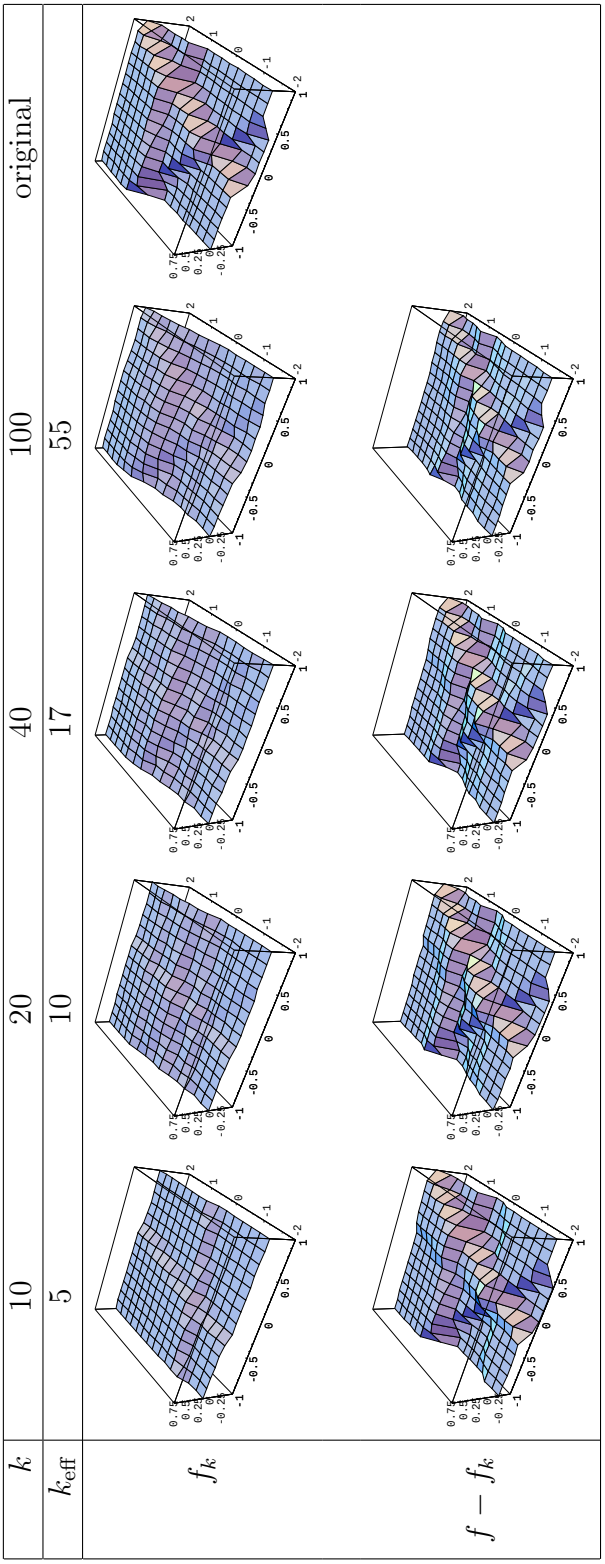


Figure 6.1: Evolution of the approximation f_k and the error $f_k - f$ during the greedy algorithm.

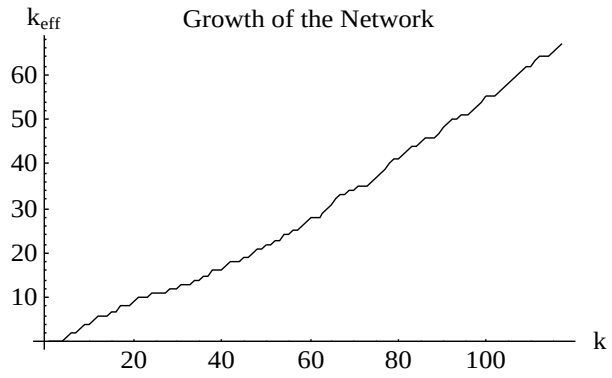


Figure 6.2: Evolution of the network size during the greedy algorithm.

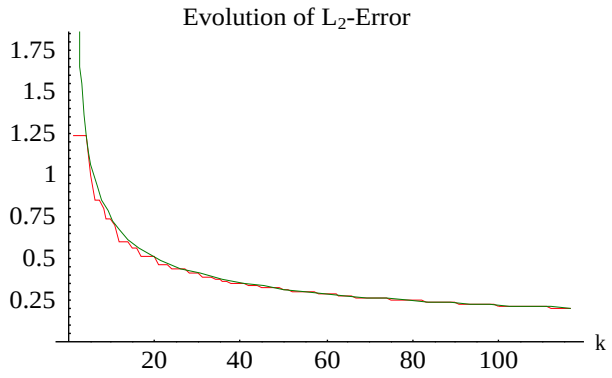


Figure 6.3: Evolution of the L_2-norm of the error $f - f_k$.

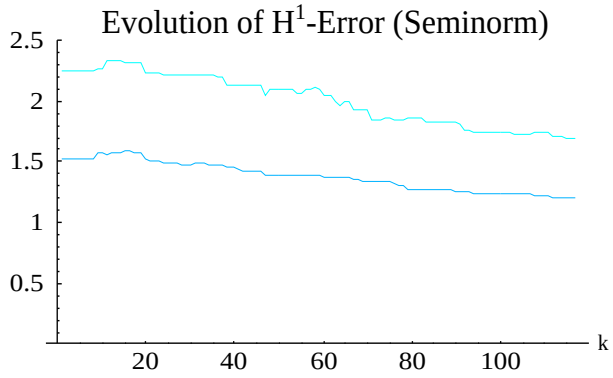


Figure 6.4: Evolution of the H^1-seminorm of the difference $f - f_k$. The upper line corresponds to $\|\partial_{x_1}(f - f_k)\|_{L_2(\Omega)}$, the lower one to $\|\partial_{x_2}(f - f_k)\|_{L_2(\Omega)}$.

green (smooth) line represents the error estimate

$$\|f - f_k\| \leq \sqrt{\frac{M}{k}},$$

the red one represents the residual. Observe that every time the approximation is improved and the red line moves downwards, the iteration index k is increased (the red line is horizontal) as much as possible, such that the green line is not hit.

Theorem 5.11 ensures convergence in stronger Sobolev-norms, if the activation function Φ is smooth. In our case the activation function is of class C^∞, hence we might expect the same convergence rate in the H^1-norm as in the L_2-norm. Nevertheless, the norm of the derivatives of Φ grows fast, and so the observed rate (i. e., the slope of the curve) for finite k will be less. Figure 6.4 shows the behaviour of the norm of the derivatives of $f - f_k$. Both derivatives are decreasing almost monotonically. In Figure 6.5 the behaviour of the full H^1-norm (blue line) and the L_2-norm (red line) of the error is plotted in a logarithmic scale. If only the values $k \geq 50$ are taken into account, then the slope of the blue line is approximately -0.25, i. e., we find numerically the rate $\|f - f_k\|_{H^1(\Omega)} = \mathcal{O}(k^{-1/4})$. According to Theorem 5.12 such a rate can be gained if $\Phi(\cdot, t) \in H^2(\Omega)$ for all values $t \in P$.

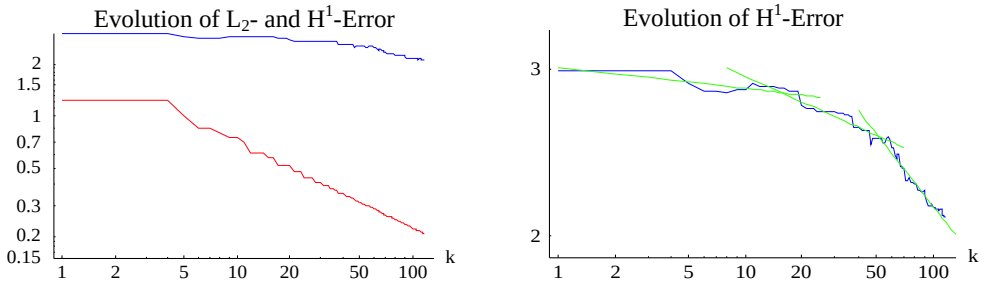


Figure 6.5: Logarithmic plot of the evolution of the L_2-norm (red) and the H^1-norm (blue) of the difference $f - f_k$. The right graph is a magnification of the left one and shows interpolations of the H^1-error (green). The slope is gradually increasing.

6.1.2 Influence of Noise

Now we investigate the influence of noise on the algorithm. Therefore we add high frequency deterministic noise with variable amplitude to the data. From Section 5.3.2 we know that the algorithm will fail to find new updates g_k if noise is present and k is too large.

We implement the algorithm as above, which means that in the kth step of the iteration we choose a random value for the parameters (t_1, t_2), set $c = 0$ and perform several Landweber and Newton steps. If the computed approximation is not sufficiently good (i. e., equation (5.13) is not fulfilled), we repeat the same procedure for a different starting value for (t_1, t_2). In the noise-free case this procedure works well and we find a new update after around 2–4 steps. If noise is present the algorithm encounters problems and fails to find a new update if k is too large. For this reason, we terminate the iteration if no valid update is found for 20 different initial values. So we did not implement the stopping rule from Section 5.3.3, but looked for the point where the algorithm naturally terminates.

Figure 6.6 illustrates the qualitative behaviour of this procedure. For instance in column 3 the amplitude of the perturbation was set to 0.57 which results in a noise level of 46%. The algorithm stopped after 12 iterations, at this time the network consisted of 8 nodes. The first line shows the noisy function f^δ, the second one the approximation f_k^δ found by the algorithm, line three shows the difference $f^\delta - f_k^\delta$ and the last one the difference between the approximation and the original, undisturbed function f. One observes that the iterates f_k^δ are not sensitive to the noise, they are always smooth functions

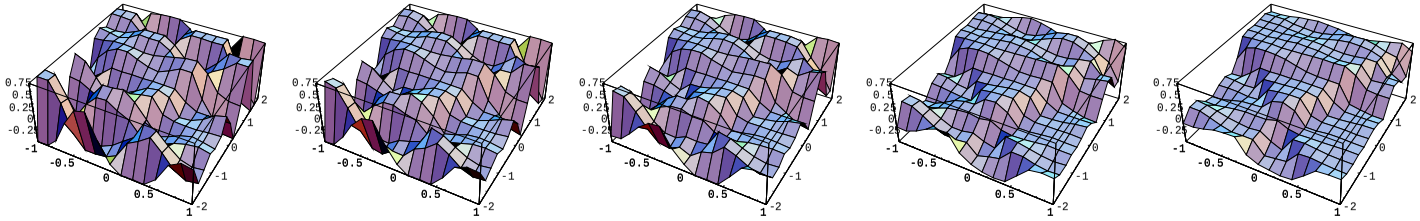
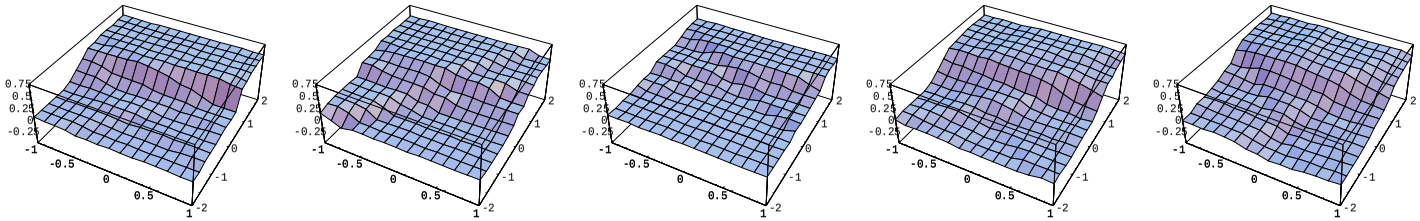
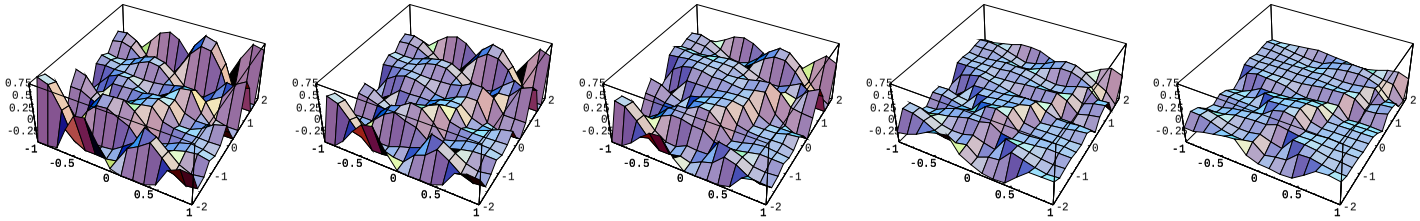
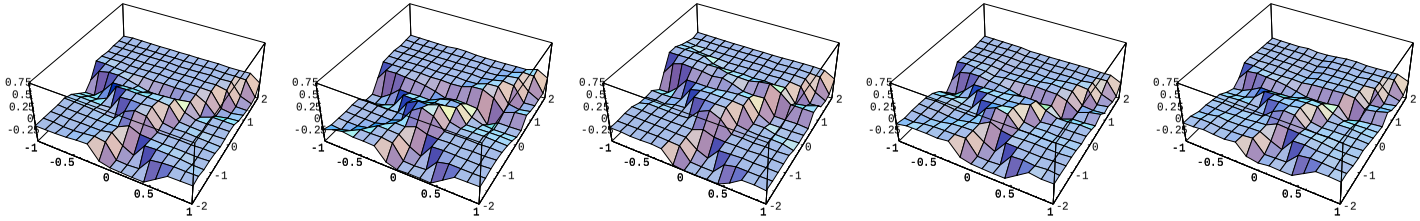
Noise	0.8 (65%)	0.57 (46%)	0.4 (32%)	0.2 (16%)	0.1 (8%)
k	24(7)	12(8)	18(5)	48(23)	72(41)
f^δ					
f_k^δ					
$f^\delta - f_k^\delta$					
$f - f_k^\delta$					

Figure 6.6: Qualitative behaviour of the algorithm for noise level tending to zero.

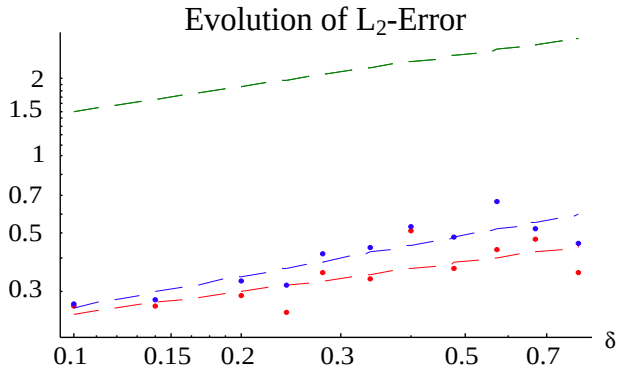


Figure 6.7: Evolution of the L_2-norm of the difference $f^\delta - f_k^\delta$ (blue) and $f - f_k^\delta$ (red) for noise level tending to zero. The green line indicates the rate which is obtained for the stopping rule.

and better approximations to f than to f^δ. The reason for this is that the search for elements f_k^δ is restricted to the set $\overline{\text{co}}(G)$, which is a set of smooth functions (see also Section 5.4.4).

In Figure 6.7 this behaviour is analyzed quantitatively. The blue line indicates the norm of $f^\delta - f_k^\delta$, the red line corresponds to $\left\|f - f_k^\delta\right\|$. Although the blue line is slightly steeper than the red one, the values for the blue line are always above the red ones. The green line corresponds to the theoretical prediction from the stopping rule from Section 5.3.3 for the error in dependence of δ. The predicted values are always far above the measured ones, but the slope and therefore the rate of convergence is approximately equal to the experimental one. This indicates that the rate expected according to the stopping rule above is obtained also numerically, but that the constants are possibly too large and might be improved, in particular by using sharper estimates instead of (5.5).

Figure 6.8 shows the behaviour of the H^1-norm of the error. Clearly, if the noise level is high, it tends to zero much faster for $f^\delta - f_k^\delta$ than for $f - f_k^\delta$. This is due to the fact that the H^1-norm of f^δ is much larger than the H^1-norm of f. Since f_k^δ is a smooth function, also the corresponding difference $f^\delta - f_k^\delta$ is larger than $f - f_k^\delta$ in the H^1-norm. As the noise level decreases also this effect vanishes and the slope of the blue line decreases. Note that the blue line always lies above the red one, i. e., f_k^δ always fits better to f than to f^δ.

Figure 6.9 illustrates the behaviour of the three different solutions given

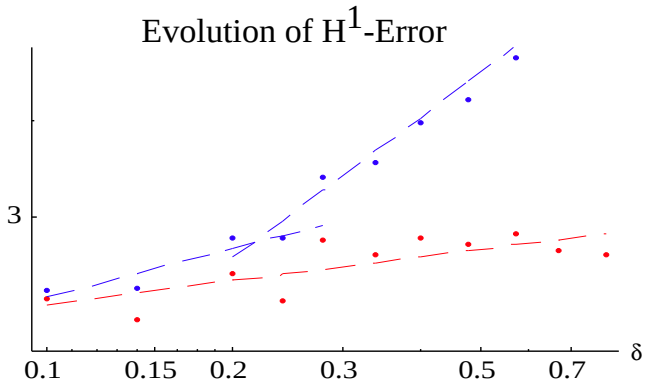


Figure 6.8: Evolution of the H^1-norm of the difference $f^\delta - f_k^\delta$ (blue) and $f - f_k^\delta$ (red) for noise level tending to zero.

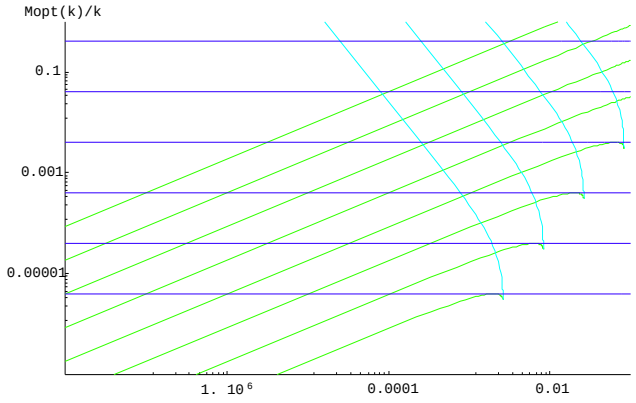


Figure 6.9: Behaviour of $\frac{M(k_i)}{k_i}$ corresponding to the three different choices for k in Section 5.3.3. The lower curves correspond to the choice $\gamma = 10^{-6}$, the top ones to $\gamma = 10^{-1}$.

in Section 5.3.3. For k_1 the ratio $\frac{M(k_1)}{k_1}$ becomes constant for δ tending to zero. For k_3 the ratio increases, as mentioned before this solution seems to correspond to a local maximum. Only for k_2 the ratio $\frac{M(k_3)}{k_3}$ tends to zero, the slope of the corresponding line is 0.66, this means the convergence rate for noise level tending to zero is approximately $\delta^{2/3}$. Since $\frac{M}{k}$ measures the squared norm we obtain

$$\left\| f^\delta - f_k^\delta \right\| = \mathcal{O}(\delta^{\frac{1}{3}}) \,.$$

As we have seen above, we find approximately the same rate in Figure 6.7.

6.2 Radial Basis Functions

In the second example we investigate the qualitative behaviour of the algorithm when applied to *radial basis function* networks. Again we choose the 2-dimensional domain $\Omega = [-1, 1] \times [-2, 2]$. As activation function we use

$$\Phi(x, t) = \Phi(x_1, x_2, t_1, t_2) = \Xi\left(\sqrt{(x_1 - t_1)^2 + (x_2 - t_2)^2}\right),$$

where Ξ is given as a Gaussian function, namely

$$\Xi(\xi) = 5\exp(-10\xi^2).$$

Using this choice the norm of Φ is bounded by the value 2 uniformly in t. The bound for c is implemented as above via the function $\kappa(c)$.

To ensure that the function to be approximated is an element of $\overline{\text{co}}(G_b)$ we again define it explicitly as a convex combination of elements of G_b, but now via an integral, namely

$$f = \int_P \mathbf{1}_{[-\frac{1}{2},\frac{1}{2}]\times[-1,1]}(t_1, t_2)\Phi(\cdot, t_1, t_2)\, dt_1\, dt_2 \,, \tag{6.2}$$

where $\mathbf{1}_{[\cdot]\times[\cdot]}$ is the characteristic function. A plot of this function can be seen in the right upper corner of Figure 6.10. Using this choice, the function f is an element of $\overline{\text{co}}(G_b)$ for $b = 2$. We choose $M = 3.83$ which is equal to $1.2(b^2 - \|f\|^2)$.

In Figure 6.10 the qualitative behaviour of Algorithm 5.7 is shown in dependence of the iteration index k. Again the network size k_{eff} is far below the iteration index k. Note that the ratio $\frac{k_{\text{eff}}}{k}$ is even smaller than in the example of Section 6.1. The network we computed for $k = 150$ effectively uses only

$k_{\text{eff}} = 55$ nodes, but yields a very good approximation. A possible reason for the (optically) better performance in the second example is, that the function being approximated fulfills an integral representation (as in equation (3.2)) for a much smoother function h than in the first example. For the function defined via equation (6.1) h is a distribution, whereas for the one defined via equation (6.2) $h \in L^\infty(P)$.

Figure 6.11 shows the influence of noise on the algorithm for two different noise-levels. Again the algorithm was terminated if no valid approximation was found for 20 different initial values for (t_1, t_2). As in the case of ridge-construction the approximations are smooth functions. They are better approximations to f than to f^δ.

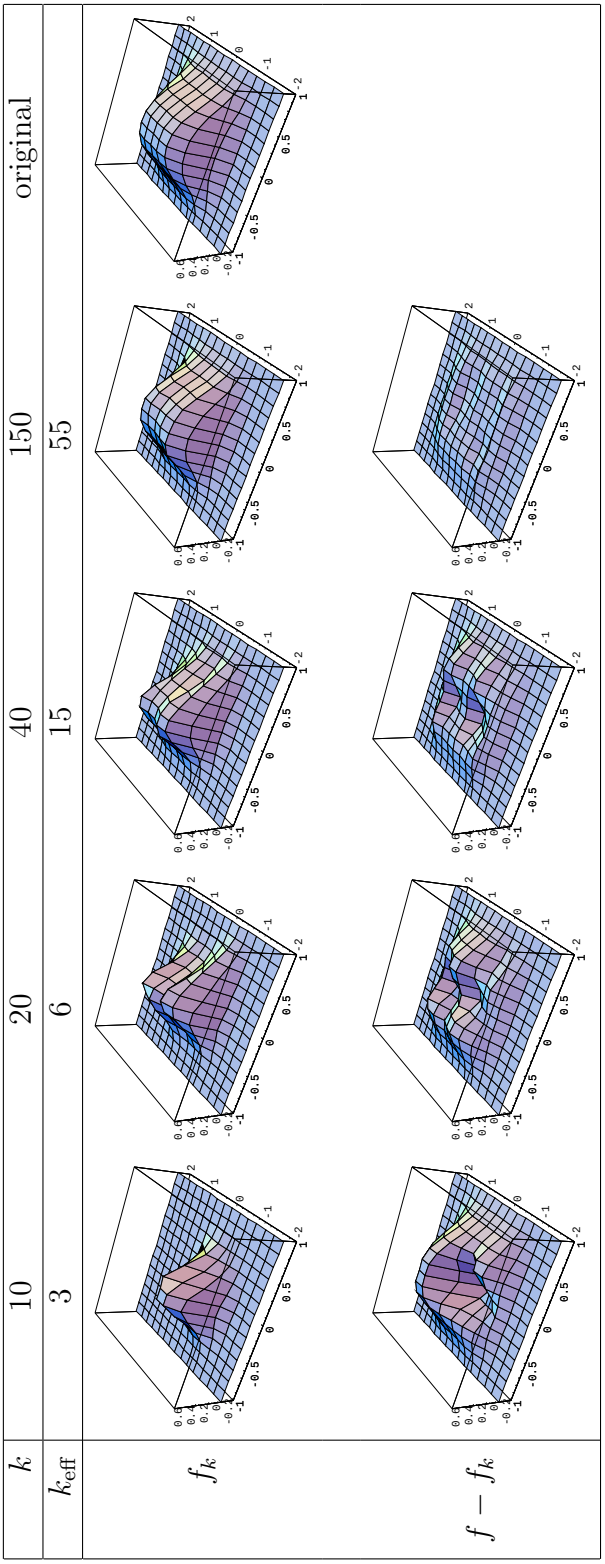


Figure 6.10: Evolution of the approximation f_k and the difference $f_k - f$ during the greedy algorithm.

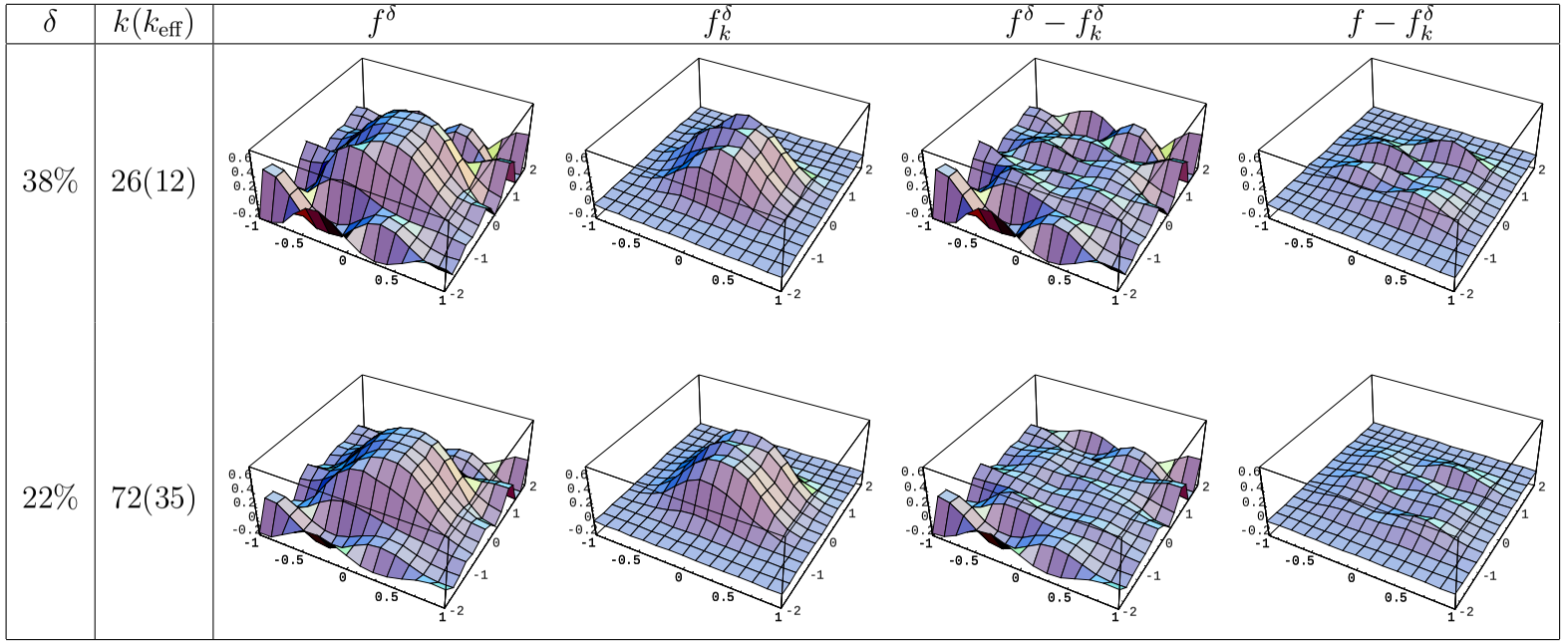


Figure 6.11: Behaviour of the algorithm for noisy data and two different noise levels.

Bibliography

[Bar93] A. R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, 1993.

[BBEH02] U. Bodenhofer, M. Burger, H. W. Engl, and J. Haslinger. Regularized data-driven construction of fuzzy controller. *Journal of Inverse and Ill-posed Problems*, 10:319–344, 2002.

[BE92] L. E. Bourne and B. R. Ekstrand. *Einführung in die Psychologie*, volume 1. Klotz, Eschborn, 1992.

[BE00] M. Burger and H. W. Engl. Training neural networks with noisy data as an ill-posed problem. *Advances in Computational Mathematics*, 13:335–354, 2000.

[Bis93] C. M. Bishop. Curvature–driven smoothing: a learning algorithm for feedforward networks. *IEEE Transactions on Neural Networks*, 4(1):882–884, 1993.

[Bis95] C. M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural Computation*, 7:108–116, 1995.

[Bis96] C. M. Bishop. *Neural Networks for Pattern Recognition*. University Press, Oxford, 1996.

[BN01] M. Burger and A. Neubauer. Error bounds for approximation with neural networks. *Journal of Approximation Theory*, 112:235–250, 2001.

[BN03] M. Burger and A. Neubauer. Analysis of tikhonov regularization for function approximation by neural networks. *Neural Networks*, to appear, 2003.

[BO75] E. Blum and W. Oettli. *Mathematische Optimierung*. Springer, Berlin, Heidelberg, 1975.

[Cea78] J. Cea. *Optimization. Theory and Algorithms*. Springer, Berlin, Heidelberg, 1978.

[DES98] P. Deuflhard, H. W. Engl, and O. Scherzer. A convergence analysis of iterative methods for the solution of nonlinear ill-posed problems under affinely invariant conditions. *Inverse Problems*, 14(5):1081–1106, 1998.

[DS96] A. T. Dingankar and I. W. Sandberg. A note on error bounds for approximation in inner product spaces. *Circuits, Systems and Signal Processing*, 15(4):519–522, 1996.

[EHN96] H. W. Engl, M. Hanke, and A. Neubauer. *Regularization of Inverse Problems*. Kluwer, Dordrecht, 1996.

[Eng83] H. W. Engl. Regularization by least-squares collocation. In P. Deuflhard and E. Hairer, editors, *Numerical Treatment of Inverse Problems in Differential and Integral Equations*, pages 345–354. Birkhäuser, Boston, 1983.

[GJP95] F. Girosi, M. Jones, and T. Poggio. Regularization theory and neural networks architectures. *Neural Computation*, 7:219–269, 1995.

[GP90] F. Girosi and T. Poggio. Networks and the best approximation property. *Biological Cybernetics*, 63:169–176, 1990.

[Gro80] C. W. Groetsch. Generalized inverses and generalized splines. *Numerical Functional Analysis and Optimization*, 2:93–97, 1980.

[HNS95] M. Hanke, A. Neubauer, and O. Scherzer. A convergence analysis of the landweber iteration for nonlinear ill–posed problems. *Numerische Mathematik*, 72:21–37, 1995.

[HSW89] K. Hornik, M. Stinchcombe, and H. White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2:359–366, 1989.

[Kal97] B. Kaltenbacher. Some newton type methods for the regularization of nonlinear ill-posed problems. *Inverse Problems*, 13:729–753, 1997.

[KF75] A. N. Kolmogorov and S. V. Fomin. *Reelle Funktionen und Funktionalanalysis*. Deutscher Verlag der Wissenschaften, 1975.

[KKN96] F. Klawonn, R. Kruse, and D. Nauck. *Neuronale Netze und Fuzzy–Systeme*. Vieweg, Braunschweig, second edition, 1996.

[Köh90] M. Köhle. *Neurale Netze*. Springer-Verlag, Wien, New York, 1990.

[Lin83] W. Linde. *Infinitely Divisible and Stable Measures on Banach Spaces*. Teubner, Leipzig, 1983.

[LM72] J. L. Lions and E. Magenes. *Non-Homogeneous Boundary Value Problems and Applications*, volume 1. Springer, Berlin, Heidelberg, 1972.

[Lor66] G. G. Lorentz. *Approximation of Functions*. Holt, Rinehalt and Winston, New-York, 1966.

[LS95] L. Ljung and J. Sjöberg. Overtraining, regularization, and searching for minimum in neural networks. *International Journal of Control*, 62:1391–1407, 1995.

[MHL92] J. E. Moody, S. J. Hanson, and R. P. Lippmann. The effective number of parameters: an analysis of generalization and regularization in nonlinear learning systems. *Advances in Neural Information Processing Systems*, 4, 1992.

[MRS91] T. Martinetz, H. Ritter, and K. Schulten. *Neuronale Netze. Eine Einführung in die Neuroinformatik selbstorganisierender Netzwerke*. Addison–Wesley publishing company, Bonn, München, second edition, 1991.

[NG99] P. Niyogi and F. Girosi. Generalization bounds for function approximation from scattered noisy data. *Advances in Computational Mathematics*, 10:51–80, 1999.

[NW74] M. Z. Nashed and G. Wahba. Convergence rates of approximate least squares solutions to linear integral and operator equations of the first kind. *Mathematics of Computation*, 28:69–80, 1974.

[Pao89] Yoh-Han Pao. *Adaptive Pattern Recognition and Neural Networks*. Addison–Wesley publishing company, Massachusetts, 1989.

[Pin85] A. Pinkus. *n-Widths in Approximation Theory*. Springer, Berlin, Heidelberg, 1985.

[Sar73] A. Sard. Approximations based on nonscalar observation. *Journal of Approximation Theory*, 8:315–334, 1973.

[Wal75] G. R. Walsh. *Methods of Optimization*. John Wiley & Sons, 1975.

[Wer95] D. Werner. *Funktionalanalysis*. Springer, Berlin, Heidelberg, 1995.

Lebenslauf

Name: Andreas Hofinger

Geburtsdatum: 6. August 1978

Geburtsort: Grieskirchen

Staatsbürgerschaft: Österreich

Wohnort: 4712 Michaelnbach, Nr. 58

Eltern: Anton Hofinger; Gemeindevertragsbediensteter
Rosemarie Hofinger, geb. Reiter; Schulköchin
wohnhaft in 4712 Michaelnbach, Nr. 58

Geschwister: Markus und Christian Hofinger
Karin Sestak-Hofinger

Schulbildung:

1984 – 1988	4 Klassen Volksschule Michaelnbach
1988 – 1992	4 Klassen im Ordensgymnasium Dachsberg
1992 – 1997	5 Klassen an der Chemie-HTL, Wels
1998 – 2003	Studium der Technischen Mathematik an der Johannes Kepler Universität Linz

Sonstiges:

1997 – 1998	Zivildienst beim Roten Kreuz, Grieskirchen
seit 1998	Ehrenamtlicher Mitarbeiter beim Roten Kreuz
August 2002	Teilnahme an der 16. ECMI Modelling Week

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Diplomarbeit selbstständig und ohne fremde Hilfe verfasst habe. Ich habe dazu keine weiteren als die angeführten Hilfsmittel benutzt und die aus anderen Quellen wörtlich oder sinngemäß entnommenen Stellen sind als solche gekennzeichnet.

Andreas Hofinger