Joey de Mol

Analysis of Deep Learning Models

Approximation properties and improper learning

Master's Thesis, 20-7-2016

Supervisors:
Dr. T. van Erven
Dr. J. Schmidt-Hieber

Mathematical Institute
Leiden University

Abstract

Approximation properties and improper learning of deep learning models with the rectifier activation function are studied. Explicit approximation bounds are given for single-layer networks and a class of target functions with a smoothness property related to the Fourier transform. An alternative kernel is proposed for the kernel method [Zhang, Y., Lee, J. D. and Jordan, M. I. (2015). ℓ_1-regularized neural networks are improperly learnable in polynomial time.] that enables improper learning of single-layer neural networks with one-dimensional input and bounded weights.

Contents

1 Introduction **4**

2 Deep Learning **5**

2.1 Definition and interpretation 5

2.2 Historical overview 8

2.3 Activation functions 9

3 Expressive Power of Neural Networks **11**

3.1 Technical summary 11

3.2 Sketch of proof of Theorem 3.1 13

3.3 Refinement for bounded parameters 14

3.4 Extension to rectifier activation function 15

4 The Kernel Method for Improper Learning of Neural Networks **18**

4.1 Reproducing kernel Hilbert spaces 18

4.2 The recursive kernel method 20

4.3 Alternative kernel for rectifiers 24

4.4 Extension to multidimensional input 30

5 Discussion **32**

1 Introduction

In statistics it is often essential to represent the data in a way that simplifies the task at hand. Deep learning is a technique that learns functions from data by learning appropriate representations of the data. A hierarchical model of representations is learned where complex representations are expressed in terms of simpler representations, which distinguishes deep learning from other representation learning methods. In this thesis we study the mathematical models used in deep learning, known as feedforward artificial neural networks. We present approximation and learnability results for neural networks that use the rectifier activation function commonly used in practice.

Deep learning is performing exceptionally well in practice for several learning tasks such as image recognition and speech recognition, and is looking promising for several other learning tasks. However, deep learning is mostly based on heuristic arguments. The theoretical support for deep learning has fallen behind with the practical successes. We believe that further theoretical development in deep learning will help to develop better methods and algorithms.

The performance of statistical learning models depends on what kind of functions the model can approximate and how difficult it is to train the model. We show that functions with a certain smoothness property related to the Fourier transform can be approximated by single-layer neural networks with the rectifier activation function and we give an explicit bound on the approximation error in terms of the number of nodes n with rate of convergence $O(n^{-1/2})$. Furthermore, polynomial time learnability of single-layer neural networks with the rectifier activation function, one-dimensional input and bounded weights is proven under the framework of improper learning, where a predictor is learned that is not necessarily a neural network but that performs with high probability almost as well as the best-fitting neural network in the hypothesis class.

We give an introduction to deep learning in Chapter 2. First a mathematical definition of the deep learning model and an interpretation of the model in terms of learning representations of the data are given. Next we give a historical overview of deep learning and we discuss several activation functions. In Chapter 3 our approximation results for deep learning models are presented. Work by Barron [1, 2] on neural networks with sigmoidal activation functions is discussed before we extend the results to networks composed of rectifier units. Learnability of neural networks is studied in Chapter 4, where we present the kernel method developed by Zhang et al. [25] for improper learning of multilayer neural networks with polynomially expendable activation functions and bounded weights. We propose a different kernel based on a wavelet transform that enables the kernel method to be applied in the case of networks with the rectifier activation function and one-dimensional input.

2 Deep Learning

In statistical learning theory, a *feature* is a transformed representation of the raw data. In many cases the first step in a statistical prediction task is to attempt to extract the right features, in the sense that they explain the data well and are easy to work with. For example, in optical character recognition tasks the width and length of the character, and their ratio, are simple features extracted from the raw pixel data. It is often too difficult and time-consuming to select the features by hand, but there exist algorithms in machine learning that attempt to learn the right features themselves.

Deep learning is a branch of statistical learning that aims to learn a hierarchical model of features, where abstract, high-level features are formed by the composition of simpler, low-level features. The model is a *feedforward artificial neural network*, a layered network of nonlinear processing units where each layer uses the output from the previous layer as input. The nonlinear process is determined by the choice of a nonlinear, univariate function called the *activation function*.

In this chapter we first give the precise definition and interpretation of the network model in deep learning. We then present an overview of the history and applications of deep learning. Finally we consider the choice of the activation function.

For comprehensive works on deep learning we refer to the review paper by LeCun, Bengio and Hinton [16], the textbook in preparation by Bengio et al. [4], and an introductory paper by Bengio [3].

2.1 Definition and interpretation

Deep learning is a statistical learning method, which means that it is concerned with learning a predictive function based on data. The defining aspect of deep learning is that it uses successive layers of processing units that perform nonlinear transformations.

A deep learning model can be represented by a network consisting of multiple layers of nodes in a directed, weighted graph where each layer is fully connected to the next one. Figure 2.1 gives an illustration, for which the mathematical notation will be explained below. The input of the model is the bottom layer where the number of nodes is equal to the dimension of the data, and the output of the model is given by the top layer. Layers in between are called *hidden layers*, where each node is a processing unit that applies a nonlinear, univariate function σ called the *activation function* to the weighted combination of outputs in the previous layer. The activation function is fixed and the same for each unit. The weights are the parameters of the model that must be learned by training with data through optimization methods of which the gradient-based method called stochastic gradient descent is the one used most in practice [16]. The network is called a *feedforward artificial neural network* because it is a model of the biological nervous system, with the nodes representing neurons and the weights representing the strength of the connections between them.

Mathematically the *feedforward artifical neural network*, or neural network for short, is a function f that maps some input $x \in \mathbb{R}^d$ to an output $f(x)$. We

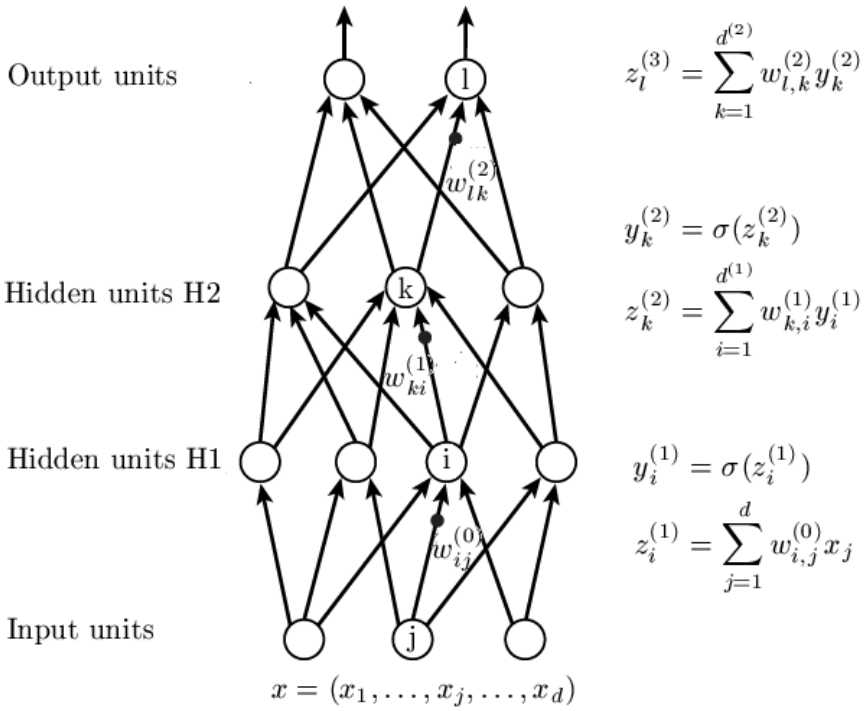


Figure 2.1: An artificial neural network (adapted from [16]).

assume that the top layer consists of one neuron, so that the network returns a one-dimensional real number $f(x)$. The bottom layer of the network is the input x and the top layer is the output $f(x)$, and they are connected through k *hidden layers*. Let $d^{(p)}$ denote the number of nodes or neurons in the p-th layer, and let $y_i^{(p)}$ represent the output of the i-th node in the p-th layer. We define the zero-th layer to be the input vector so that $d^{(0)} = d$ and $y^{(0)} = x$. Moreover $f(x) = y_1^{(k+1)}$. We also define $w_{i,j}^{(p-1)}$ to be the weight of the edge that connects the neuron j on the $(p-1)$-th layer to the neuron i on the p-th layer. The output of each neuron in hidden layer p is now given by

$$y_i^{(p)} = \sigma(z_i^{(p)})$$
$$z_i^{(p)} := \langle w_i^{(p-1)}, y^{(p-1)} \rangle = \sum_{j=1}^{d^{(p-1)}} w_{i,j}^{(p-1)} y_j^{(p-1)},$$

where the *activation function* $\sigma : \mathbb{R} \to \mathbb{R}$ is a nonlinear, univariate function. We also define the top layer output $f(x)$ to be the weighted combination of the previous layer, that is, $f(x) = y_1^{(k+1)} = z_1^{(k+1)} := \langle w_1^{(k)}, y^{(k)} \rangle$. Sometimes a neuron dependent additive constant $b_i^{(p)}$ is added to the weighted combinations, with

$$z_i^{(p)} := \sum_{j=1}^{d^{(p-1)}} w_{i,j}^{(p-1)} y_j^{(p-1)} + b_i^{(p)}.$$

Note that the additive constants can be written as weights if we add a neuron with a fixed value to all input and hidden layers. Finally, for the special case of neural networks with a single hidden layer with $n = d^{(1)}$ neurons we simply write

$$f(x) = \sum_{j=1}^{n} c_j \sigma(\langle a_j, x \rangle + b_j) + c_0$$

for the output of the network, where $a_j = w_j^{(0)} \in \mathbb{R}^d$, $b_j = b_j^{(1)}$ for $k = 1, 2 \ldots, n$, $(c_1, \ldots c_n) = w_1^{(1)} \in \mathbb{R}$ and $c_0 = b_1^{(2)} \in \mathbb{R}$.

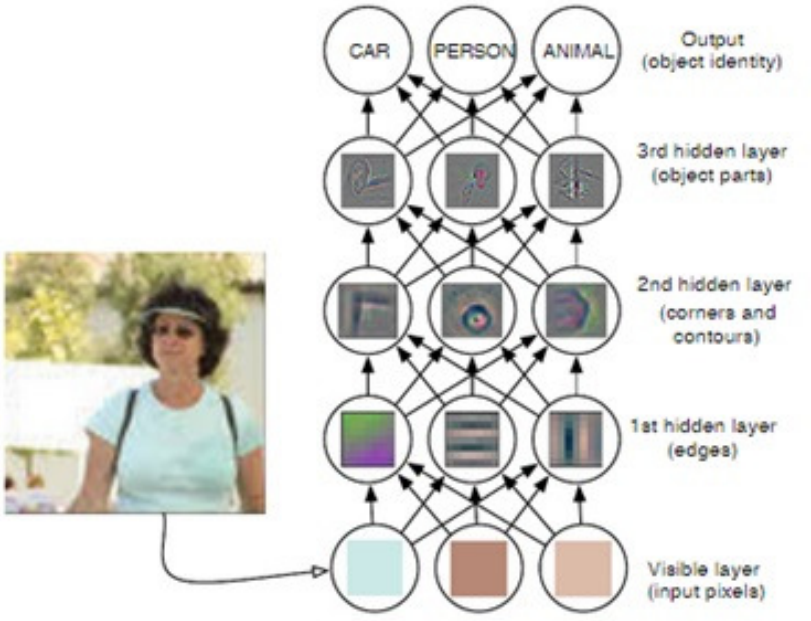


Figure 2.2: Figure by LeCun et al. [4] illustrating the hierarchical model of features with multiple levels of abstraction.

Deep learning methods are said to learn representations of data with multiple levels of abstraction [4, 16, 3]. Simple features in a low-level layer are combined and transformed into more abstract features in the next layer. For example, in an image recognition problem with pixel image data as in Figure 2.2 the neurons in the first hidden layer could represent the presence of edges, while second level neurons could represent particular arrangements of edges such as corners and contours. High-level neurons could represent even more complex visual shapes that make up the object in the picture so that the object can be classified. It would be difficult to extract the features by hand because it is often not clear what the right features are or what they represent. Moreover, the same object could look very different in two pictures depending on lighting, position, angle and more. The idea is that the model learns the right features by learning the weights between neurons from the training data.

The performance of deep learning methods will depend on what kind of functions a neural network can learn, and how difficult it is to train. The nonlinearity of the activation function is required because otherwise the network reduces to a linear function itself. In the final section of this chapter we discuss the choice of this activation function. One theoretically founded ([3], section 2) advantage of having multiple layers is that a function that can be approximated by a neural network with k hidden layers may require exponentially more nodes when approximated by a neural network with $k-1$ hidden layers. In the next chapter we discuss the expressive power of neural networks. In Chapter 4 we discuss the training time involved with learning neural networks.

2.2 Historical overview

In recent years deep learning has become an increasingly popular research area with practical successes following one another in quick succession, but these recent advancements are due to multiple breakthroughs in the last sixty years. In this section we provide the historical context to the recent developments in deep learning.

The history of deep learning can be roughly divided in three waves of developments; 1940s - 1960s, 1980s - 1990s and 2006 - present [4, Chapter 1]. The first wave started in 1943 when McCulloch and Pitts proposed the first mathematical model for biological learning in the brain [18]. The linear model was a binary classifier that represented a single neuron with weights that had to be selected by hand. Selection of the weights was a problem until 1958 when Rosenblatt introduced the *perceptron* model [19], which made it possible to learn the weights from data for single units. The breakthrough that launched the second wave of developments was the *backpropagation* algorithm introduced by Rumelhart et al. in 1986 [20]. With the backpropagation algorithm networks with one or two hidden layers of neurons could be trained, enabling the approximation of functions through networks. The field came to be known as *artificial neural networks* because of the analogy with biology. Training deep networks with three layers still proved too difficult. New machine learning methods such as the support vector machine outperformed artificial neural networks and that led to neural networks going out of fashion in the 1990s. Neural networks gained popularity again when in 2006 Hinton et al. [10] developed a fast, greedy algorithm that could learn multilayer networks one layer at a time. Computational advances and an increase in available data sets made training deep networks feasible, in particular because it requires little engineering by hand. It turned out that deep networks had superior performance to state-of-the-art machine learning models for certain tasks. The field is now named deep learning to emphasize the use of architectures which consist of many layers.

Today deep learning has exciting applications in all kinds of learning tasks, with unmatched performances in image recognition and speech recognition and encouraging results in natural language processing. For the statements of these results, see the 2015 Nature review paper [16]. However, many questions regarding deep learning are still unanswered. Deep learning models represent functions which are difficult to analyze, and theoretical results have fallen behind with the practical successes.

2.3 Activation functions

In the definition of a neural network there is the choice of the nonlinear *activation function* $\sigma : \mathbb{R} \to \mathbb{R}$ which maps the input of each node to its output. The activation function models the firing rate of a biological neuron. We discuss some common activation functions, plotted in Figure 2.3.

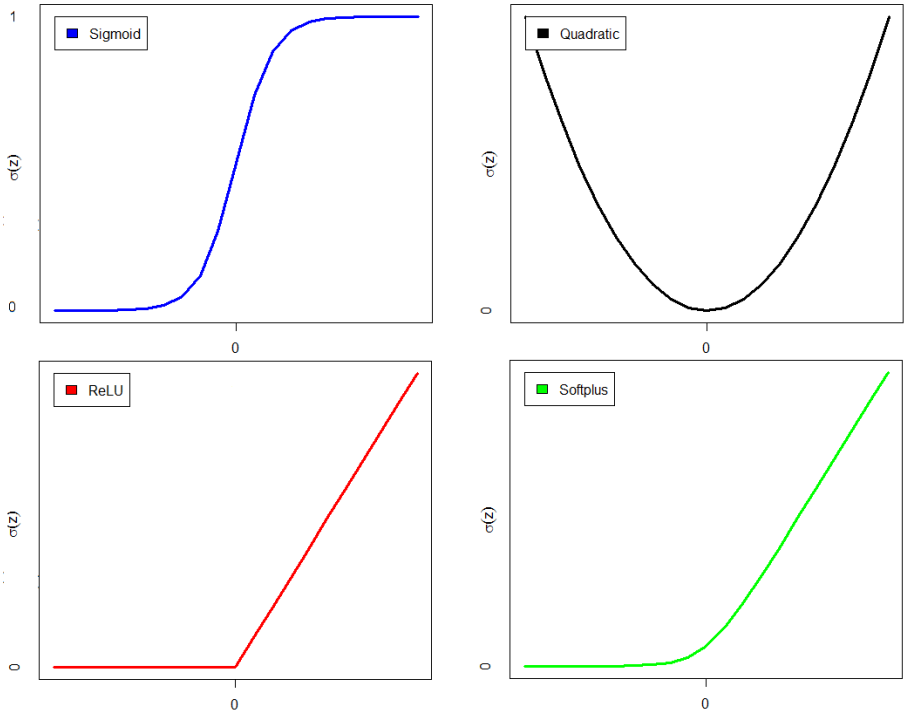


Figure 2.3: Plotted sigmoid, quadratic, rectifier and softplus activation functions.

A historically popular class of activation functions is the family of *sigmoidal functions*, functions σ_s that are bounded and differentiable and that satisfy $\sigma_s(z) \to 1$ as $z \to \infty$ and $\sigma_s(z) \to 0$ as $z \to -\infty$. The canonical sigmoidal function is the logistic sigmoid given by

$$\sigma_{s,\beta}(z) = \frac{1}{1 + e^{-\beta z}}, \qquad \beta > 0.$$

For future reference we also mention that the shifted error function defined by

$$\sigma_{\text{erf}}(z) = \frac{1}{2}(1 + \text{erf}(\sqrt{\pi} z))$$

is a sigmoidal function.

Sigmoidal activation functions were traditionally used because they are smooth and can approximate step functions. Universal approximation properties of neural networks were first shown for networks with sigmoidal activation [7, 11].

Sigmoidal activation functions are no longer used in practice as networks with particular other activation functions proved to be easier to train.

Because of its practical successes, the most used activation function currently is the *rectified linear unit* σ_r [16], rectifier or ReLU for short, defined by

$$\sigma_r(z) = \max(0, z).$$

Also a smoothed version called the softplus function σ_{sp} has been studied,

$$\sigma_{sp}(z) = \log(1 + e^z).$$

Experimental study shows that despite the potential difficulties of unboundedness and non-differentiability at zero that the rectifier unit is the best performing activation function so far for networks with multiple hidden layers [8]. In particular the rectifier performs better than the softplus function so the specific behavior of rectifiers around zero is beneficial. In networks composed of rectifier units there are only a few units having non-zero output at a time, as shown by the same study. The induced sparsity is argued to explain in some part the practical successes of rectifier units, but there are no hard theoretical results so far that show why rectifier units work as well as they do.

Finally we consider the quadratic activation function σ_q,

$$\sigma_q(z) = z^2.$$

The quadratic function is of theoretical importance because networks composed of quadratic units represent polynomial functions and are therefore mathematically easier to analyze than networks with the previously mentioned activation functions. Networks with quadratic activation perform well in the case of a single hidden layer, but they are hard to train when multiple hidden layers are used [17, 5].

3 Expressive Power of Neural Networks

Deep learning methods reduce to learning a neural network that approximates a certain target function. The overall accuracy of the network as an estimate is determined by a trade-off between the *approximation error*, that is the distance between the target function and the closest neural network, and the *estimation error*, the distance between the ideal network and the network that is learned based on the data. Networks with multiple layers or many nodes have more adjustable parameters and are therefore better at representing classes of target functions, but they are harder to train in terms of the estimation error and computation time. In this chapter we examine the expressive power of neural networks in terms of the combined effect of the approximation error and the estimation error.

Any continuous function on a compact subset of $\mathbb{R}^d$ can be approximated by neural networks with a single hidden layer if the number of nodes tends to infinity. In 1989 Cybenko [7] and Hornik et al. [11] showed independently that the approximation error between the target function and the best neural network can be made arbitrarily small by increasing the number of nodes in the hidden layer, under some assumptions on the activation function. An explicit bound on this approximation error in terms of the number of nodes was first proven in 1993 by Barron [1] for neural networks with the sigmoid activation function and for a class of target functions with some smoothness property formulated in terms of the Fourier transform. In a follow-up paper Barron [2] extended this to a bound on the mean integrated squared error by providing also a bound on the estimation error. We summarize the results by Barron [1, 2] and show how they can be adapted to allow networks with rectifier activation functions.

3.1 Technical summary

We consider neural networks with a single hidden layer on $[-1,1]^d$, that is functions of the form

$$f_{n,\theta}(x) = \sum_{k=1}^{n} c_k \sigma(a_k^T \cdot x + b_k) + c_0$$

with n the number of nodes and $\theta = (a_1, \ldots, a_k, b_1, \ldots, b_k, c_0, c_1, \ldots, c_k)$ a parameter vector with $a_k \in \mathbb{R}^d$, $b_k, c_k \in \mathbb{R}$ for $k = 1, 2 \ldots, n$ and $c_0 \in \mathbb{R}$.

We assume that the target function f^* has Fourier representation $f^*(x) = \int_{\mathbb{R}^d} e^{i\langle \omega, x\rangle} \tilde{F}(d\omega)$ and $C_{f^*} := \int |\omega|_1 |\tilde{F}(d\omega)|$ is finite, where $|\omega|_1 = \sum_{j=1}^d |\omega_j|$. The constant C_{f^*} is measure of the extent to which the function oscillates. A sufficient condition for C_{f^*} to be finite is that all partial derivatives of f^* of order less than or equal to $s = \lceil d/2 \rceil + 1$ be square-integrable on $\mathbb{R}^d$ [1, Section IX].

The distance between the target function f^* and a neural network $f_{n,\theta}$ is measured by the integrated squared error

$$\|f^* - f_{n,\theta}\|^2 = \int_{[-1,1]^d} |f^*(x) - f_{n,\theta}(x)|^2 \mu(dx)$$

for an arbitrary probability measure μ on $[-1,1]^d$. The norm $\|f^* - f_{n,\theta}\|$ measures the performance of the neural network on new data drawn with distribution μ.

We are now able to state the main result by Barron on approximation bounds for neural networks with sigmoidal activation.

Theorem 3.1. *[1, Theorem 1] Given an sigmoidal activation function σ_s and a target function f^* with C_{f^*} finite, there exists for every $n \geq 1$ a single-layer neural network f_{n,θ^*} such that*

$$\|f^* - f_{n,\theta^*}\| \leq \frac{2C_{f^*}}{\sqrt{n}}.$$

The parameter vector θ^* may be restricted to satisfy $|b_k| \leq |a_k|_1 = \sum_{j=1}^n |a_{k,j}|$ for all k, $\sum_{k=1}^n |c_k| \leq 2C_{f^*}$, and $c_0 = f^*(0)$.

We sketch the proof of Theorem 3.1 in Section 3.2. Theorem 3.1 gives a bound on the approximation error $\|f^* - f_{n,\theta^*}\|$, the distance between the target function and the closest single-layer neural network with n nodes. The approximation error tends to zero as $n \to \infty$ with rate of convergence $O(n^{-1/2})$. The constant C_{f^*} can be exponentially large in d since it involves a d-dimensional integral.

We denote by $f_{n,\hat{\theta},N}$ the network that minimizes the empirical risk $\frac{1}{N}\sum_{i=1}^N (f^*(X_i) - f_{n,\theta}(X_i))^2$ based on the training data $X_1, \ldots, X_N$. By the triangle inequality we have that

$$\|f^* - f_{n,\hat{\theta},N}\| \leq \|f^* - f_{n,\theta^*}\| + \|f_{n,\theta^*} - f_{n,\hat{\theta},N}\|,$$

and thus,

$$E\|f^* - f_{n,\hat{\theta},N}\| \leq \|f^* - f_{n,\theta^*}\| + E\|f_{n,\theta^*} - f_{n,\hat{\theta},N}\|,$$

where $\|f^* - f_{n,\theta^*}\|$ is the approximation error, or bias term, and $E\|f_{n,\theta^*} - f_{n,\hat{\theta},N}\|$ is the estimation error, or variance term. The idea is to control the two errors.

Some more conditions are required to be able to give a bound on the estimation error. The range of the target function f^* is assumed to be in a bounded interval. We further assume that the sigmoidal activation function σ_s is Lipschitz continuous and approaches its limit polynomially fast in the sense that respectively $\sigma_s(z)/|z|^p$ and $(1 - \sigma_s(z))/|z|^p$ remain bounded as $z \to -\infty$ and $z \to \infty$, for some $p > 0$. Now the estimation error is controlled by restricting the allowed parameters to a discretized, finite set $\Theta_{n,\tau_n,C_{f^*}}$ such that for each original parameter vector θ with $|a_k|_1 \leq \tau_n = O(n^{(p+1)/2p})$ for all k there is a $\theta' \in \Theta_{n,\tau_n,C_{f^*}}$ such that $|f_n(x,\theta) - f_n(x,\theta')|$ is small uniformly in x [2, Lemma 1]. Then the approximation bounds given in Theorem 3.1 still hold if we restrict the parameter vectors to the set $\Theta_{n,\tau_n,C_{f^*}}$. The estimation bound now follows by complexity regularization methods. We state the corollary.

Theorem 3.2. *[2, Theorem 3] For an Lipschitz continuous sigmoidal activation function σ_s that approaches its limit polynomially fast, and a target function f^* with C_{f^*} finite and range in a bounded interval, the least squares estimator $f_{n,\hat{\theta},N}$, with parameters restricted to some finite set $\Theta_{n,\tau_n,C_{f^*}}$, satisfies*

$$E\|f^* - f_{n,\hat{\theta},N}\|^2 \leq O\left(\frac{C_{f^*}^2}{n}\right) + O\left(\frac{nd}{N}\log N\right),$$

which is of order $O(C_{f^*}((d/N)\log N)^{1/2}$ *for* $n \sim C_{f^*}(N/(d\log N))^{1/2}$.

The first term in the upper bound for the mean integrated squared error is the contribution by the approximation error and the second term is the contribution from the estimation error. Selecting the optimal n requires C_{f^*} to be known, but the optimal bound $O(C_{f^*}((d/N)\log N)^{1/2}$ described in Theorem 3.2 is also obtained when n is optimized through complexity regularization methods [2, Theorem 4].

3.2 Sketch of proof of Theorem 3.1

The proof of Theorem 3.1 makes use of the following lemma on convex combinations in an arbitrary Hilbert space.

Lemma 3.3. *([1, Lemma 1]) If* f *is in the closure of the convex hull of a set* G *in a Hilbert space, with* $\|g\| \leq b'$ *for each* $g \in G$*, then for every* $n \geq 1$*, and every* $c > b'^2 - \|f\|^2$*, there is an* f_n *in the convex hull of* n *points in* G *such that*

$$\|f - f_n\| \leq \sqrt{\frac{c}{n}}.$$

For a proof, see [1, Lemma 1] or [13].

We consider a sigmoidal activation function σ_s and target function f^* with bounded C_{f^*}. Let

$$G_{\sigma_s} = \{\gamma\sigma_s(\langle a, x\rangle + b) : |\gamma| \leq 2C_{f^*}, a \in \mathbb{R}^d, b \in \mathbb{R}\},$$

then Theorem 3.1 follows from Lemma 3.3 when we show that the function $f^* - f^*(0)$ is contained in the closure of the convex hull of G_{σ_s} in $L_2(\mu, [-1,1]^d)$.

Using the Fourier representation of f^*, we obtain

$$\begin{aligned}
f^*(x) - f^*(0) &= \int_{\mathbb{R}^d} (e^{i\langle\omega,x\rangle} - 1)\tilde{F}(d\omega) \\
&= \mathrm{Re}\int_{\mathbb{R}^d} (e^{i\langle\omega,x\rangle} - 1)\tilde{F}(d\omega) \\
&= \mathrm{Re}\int_{\mathbb{R}^d} (e^{i\langle\omega,x\rangle} - 1)e^{i\theta(\omega)}F(d\omega) \\
&= \int_{\mathbb{R}^d} (\cos(\langle w, x\rangle + \theta(\omega)) - \cos(\theta(\omega)))F(d\omega) \\
&= \int_{\mathbb{R}^d} \frac{C_{f^*}}{|\omega|_1}(\cos(\langle w, x\rangle + \theta(\omega)) - \cos(\theta(\omega)))\Lambda(d\omega)
\end{aligned}$$

where $F = |\tilde{F}|$ and $\Lambda(d\omega) = \frac{|\omega|_1}{C_{f^*}}F(d\omega)$ is a probability distribution. The integral represents $f^* - f^*(0)$ as an infinite convex combination of functions in the set

$$G_{\cos} = \{\gamma/|\omega|_1(\cos(\langle\omega, x\rangle + b) - \cos(b)) : \omega \neq 0, |\gamma| \leq C_{f^*}, b \in \mathbb{R}\}.$$

We claim that $f^* - f^*(0)$ is therefore contained in the closure of the convex hull of $G_{\cos}$, but we omit the technical proof. The idea is to now show that functions in $G_{\cos}$ are contained in the closure of the convex hull of G_{σ_s}. For this, we first approximate functions in $G_{\cos}$ by step functions.

Functions in $G_{\cos}$ are univariate functions $g(z) = \gamma/|\omega|_1(\cos(|\omega|_1 z + b) - \cos(b))$ evaluated at $z = \frac{1}{|\omega|_1}\langle \omega, x\rangle \in [-1, 1]$. Hence it suffices to study the sinusoidal function $g(z)$ on $[-1, 1]$ which has derivative bounded by $|\gamma| \leq C_{f^*}$. Because $g(z)$ is continuous on a bounded interval it is uniformly continuous and therefore well approximated by piecewise constant functions. Piecewise constant functions can be written as a linear combination of unit step functions. It can therefore be shown that functions $g(z)$ are in the closure of the convex hull of the set of step functions $\gamma h(z - t)$ and $\gamma h(-z - t)$ where $h(z) = 1_{\{z \geq 0\}}$ and $|\gamma| \leq 2C$ and $|t| \leq 1$. Let

$$G_{\text{step}} = \left\{ \gamma h\left(\frac{1}{|\omega|_1}\langle \omega, x\rangle - t \right) : |\gamma| \leq 2C_{f^*}, \omega \neq 0, |t| \leq 1 \right\},$$

then $G_{\cos}$ is contained in the closure of the convex hull of G_{step}. Let G^{μ}_{step} be defined as G_{step} with t restricted to the continuity points of the distribution $z(x) = \frac{1}{|\omega|_1}\langle \omega, x\rangle$ induced by μ, that is t is restricted so that $\frac{1}{|\omega|_1}\langle \omega, x\rangle - t = 0$ has μ measure zero. Then $G_{\cos}$ is still contained in the closure of the convex hull of the set G^{μ}_{step}.

Finally, functions in G^{μ}_{step} are in the closure of G_{σ_s}. For this we note that the sequence $\sigma_s(\alpha(\frac{1}{|\omega|_1}\langle \omega, x\rangle - t))$ has pointwise limit equal to $h(\frac{1}{|\omega|_1}\langle \omega, x\rangle - t)$ as $\alpha \to \infty$, because of the restriction on t. The same limit holds in $L_2(\mu, [-1, 1])$ because of the dominated convergence theorem. We have that $\sigma_s(\alpha(\frac{1}{|\omega|_1}\langle \omega, x\rangle - t))$ is of the form $\sigma_s(\langle a, x\rangle + b)$ with $a = (\alpha/|\omega_1|)\omega$ and $b = -\alpha t$.

To summarize, we have argued that

$$f^* - f^*(0) \in \overline{c}oG_{cos} \subset \overline{c}oG_{step} \subset \overline{c}oG_{\sigma_s},$$

where $\overline{c}oG$ denotes the closure of the convex hull of G. Theorem 3.1 now follows by applying Lemma 3.3 and showing further that the constant c' can be taken to equal $(2C)^2$. We note that the parameters b_k can be restricted to satisfy $|b_k| \leq |a_k|_1$ since we are considering functions on $[-1, 1]$. Key points of the proof are that the smoothness property is used to express f^* in terms of sinusoidal functions, sinusoidal functions are approximated by step functions, and the step functions are in turn approximated by sigmoidal functions.

3.3 Refinement for bounded parameters

The approximation of step functions by sigmoidal functions in Section 3.2 requires that the components of a may become arbitrarily large. Barron [1, Section VI] considers a refinement where the scale parameter vectors a_k for $k = 1, \ldots, n$ are assumed to be bounded. We obtain an additional term in the approximation error since step functions can not be fully approximated by sigmoidal functions with bounded scale parameters.

Theorem 3.4. *[1, Theorem 3] Given an sigmoidal activation function $0 \leq \sigma_s(x) \leq 1$ and a target function f^* with C_{f^*} finite, there exists for every $n \geq 1$ a single-layer neural network f_{n,θ^*} with parameter vector θ^* satisfying $|a_k|_1 \leq \tau$ for all k, such that*

$$\|f^* - f_{n,\theta^*}\| \leq 2C_{f^*}(n^{-1/2} + \delta_\tau),$$

where δ_τ is a measure of the distance between $\sigma_s(\tau z)$ and $1_{\{z \geq 0\}}$ defined as

$$\delta_\tau = \inf_{0<\epsilon\leq 1/2}\{2\epsilon + \sup_{|z|\geq\epsilon} |\sigma_s(\tau z) - 1_{\{z\geq 0\}}|\}$$

Moreover, θ^* may be restricted to also satisfy $|b_k| \leq |a_k|_1$ for all k, $\sum_{k=1}^n |c_k| \leq 2C_{f^*}$, and $c_0 = f^*(0)$,

For example, for the logistic sigmoid $\sigma_s(z) = (1 + e^{-z})^{-1}$ a quick computation shows that $|\sigma_s(\tau z) - 1_{\{z\geq 0\}}| \leq e^{-\tau z}$ for $|z| \geq \epsilon$. By setting $\epsilon = (\ln \tau)/\tau$ we obtain a simple bound $\delta_\tau \leq (1 + 2\ln\tau)/\tau$.

In Theorem 3.4 the approximation error does not tend to zero. The $O(n^{-1/2})$ approximation bound of Theorem 3.1 can still be achieved by letting $\tau = \tau_n$ grow in n depending on how fast σ_s approaches it limits. That is, the parameters a_k are bounded in ℓ_1 for fixed n, but the bound tends to zero as $n \to \infty$. For the logistic sigmoid we may plug in $\tau_n = \sqrt{n}\ln n$. The refinement for bounded parameters is used in the proof of the bound on the estimation error, as we recall from Section 3.1 that parameters are restricted to satisfy $|a_k|_1 \leq \tau_n$ for some τ_n for all k.

3.4 Extension to rectifier activation function

Section 3.2 shows that Theorem 3.1 also holds with sinusoidal functions, step functions or other bounded functions as the activation function. Sigmoidal activation functions are of particular interest simply because they were commonly used in practice, and not because they have unique approximation properties. The proof of Theorem 3.1 as sketched in section 3.2 can not be directly adjusted to work for the rectifier activation function σ_r because application of Lemma 3.3 requires that the set

$$G_\sigma = \{\gamma\sigma(\langle a, x\rangle + b) : |\gamma| \leq 2C, a \in \mathbb{R}^d, b \in \mathbb{R}\},$$

is bounded in $L_2(\mu, [-1, 1])$. That norm will blow up for the rectifier activation function as a becomes large. The refinement for bounded parameters of Section 3.3 is also not suited for unbounded activation functions.

Despite the fact that the proofs break down when directly applied to rectifier units, we can still deduce from Theorem 3.1 a similar result. We note that two rectifier units can be linearly combined to create a univariate function

$$\begin{aligned}\sigma_{b_1,b_2}(z) :&= \frac{1}{|b_2 - b_1|}\Big(\sigma_r(z - b_1) - \sigma_r(z - b_2)\Big)\\ &= \frac{1}{|b_2 - b_1|}\Big(\max(0, z - b_1) - \max(0, z - b_2)\Big)\end{aligned}$$

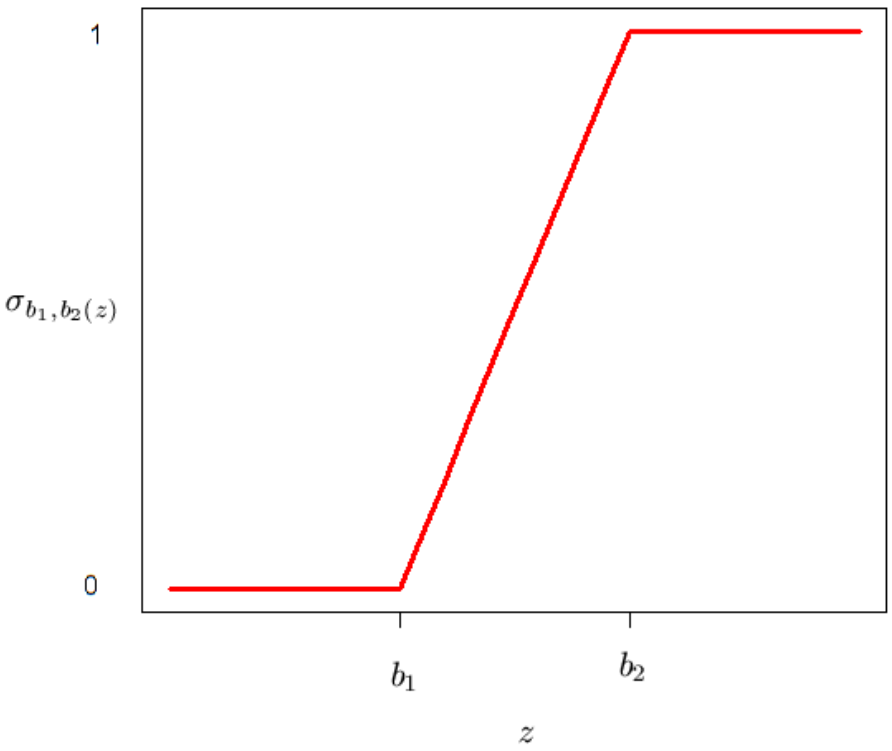


Figure 3.1: Plot of σ_{b_1,b_2} with $b_1 \leq b_2$.

See Figure 3.1 for a plot. For $b_1 \leq b_2$ we have that $\sigma_{b_1,b_2}(z) = 0$ if $z \leq b_1$ and $\sigma_{b_1,b_2}(z) = 1$ if $z \geq b_2$, and on $[b_1, b_2]$ the function σ_{b_1,b_2} is linear with slope $|b_2 - b_1|^{-1}$. Hence $\sigma_{b_1,b_2}(z)$ is a non-differentiable sigmoidal function.

Since the proof of Theorem 3.1 does not require the sigmoidal function σ_s to be differentiable but only to be bounded and satisfy $\sigma_s(z) \to 0$ as $z \to -\infty$ and $\sigma_s(z) \to 1$ as $z \to \infty$, we conclude from section 3.2 that the function $f^* - f^*(0)$ is contained in the closure of the convex hull of the set

$$\{\gamma\sigma_{b_1,b_2}(\langle a, x\rangle) : |\gamma| \leq 2C_{f^*}, a \in \mathbb{R}^d, b_1, b_2 \in \mathbb{R}\},$$

and because σ_{b_1,b_2} is the combination of two rectifier units we conclude that a similar result as Theorem 3.1 holds for single-layer networks with rectifier units with twice as many modes. That is,

Theorem 3.5. *Given the rectifier activation function σ_r and a target function f^* with C_{f^*} finite, there exists for every $n \geq 1$ a single-layer neural network f_{2n,θ^*} with $2n$ nodes such that*

$$\|f^* - f_{2n,\theta^*}\| \leq \frac{2C_{f^*}}{\sqrt{n}}.$$

Moreover, because the functions σ_{b_1,b_2} are Lipschitz continuous and approach their limits polynomially fast, we have the following analogy of theorem 3.2

Theorem 3.6. *For the rectifier activation function σ_r and a target function f^* with C_{f^*} finite and range in a bounded interval, the least squares estimator network $f_{2n,\hat{\theta},N}$ with $2n$ nodes, with parameters restricted to some finite set $\Theta_{n,\tau_n,C_{f^*}}$, satisfies*

$$E\|f^* - f_{2n,\hat{\theta},N}\|^2 \leq O\left(\frac{C_{f^*}^2}{n}\right) + O\left(\frac{nd}{N}\log N\right),$$

which is of order $O(C_{f^*}((d/N)\log N)^{1/2}$ for $n \sim C_{f^*}(N/(d\log N))^{1/2}$.

Theorem 3.5 provides a bound on the approximation error for single-layer networks composed of rectifier units and for a class of target functions satisfying some smoothness condition. We conclude that single-layer networks with the rectifier activation function also satisfy the universal approximation property. The bound on the approximation error in Theorem 3.5 should be improvable as we do not make the most efficient use of the units, but the proofs in [1, 2] break down for rectifier units so a different approach should be considered.

4 The Kernel Method for Improper Learning of Neural Networks

Neural networks are computationally hard to train because they represent highly nonconvex functions with many local minima, many parameters, and many hyperparameters such as the number of layers, number of neurons, as well as a parameter called the learning rate which determines how much the weights are adjusted in each step of the algorithm. In the previous chapter we have seen that a network with a single hidden layer can approximate functions satisfying some smoothness assumption. For any training algorithm for these networks that runs for an amount of time polynomial in the input dimension d there exist sets of training data for which the algorithm produces suboptimal weights, unless P = NP [6, 14]. Despite these negative results, many networks are succesfully learned in practice through use of optimization methods such as stochastic gradient descent [16] and a variety of tricks such as regularization, over-specification and choice of a computationally feasible activation function [17]. There is however little theoretical support for these successes in the nonconvex setting of deep learning.

Recently Zhang et al. [25] obtained polynomial-time improper learnability results for networks with polynomially expandable activation functions and ℓ_1-bounded weight vectors. They propose a kernel based method for learning neural networks under the framework of *improper learning*, also called "representation independent learning", where the output of the learning algorithm is not required to be in the hypothesis class of neural networks but in a larger class and where the output must perform with high probability almost as well as the best function in the hypothesis class. The idea is that the training is done in a different, larger space where the optimization problem is convex. The kernel method provides a practical algorithm and also contributes to the theoretical understanding of learnability of neural networks. However, the kernel used in [25] cannot be used to analyze networks that use the rectifier activation function. Our contribution is an analysis of a different kernel that enables improper learning of networks with rectifier units.

In this chapter we start by introducing kernels and reproducing kernel Hilbert spaces, or RKHS for short, based on the textbook by Schölkopf and Smola [22]. We then discuss the results by Zhang et al. [25] Finally, we propose an alternative kernel for the improper learning of networks with the rectifier activation function.

4.1 Reproducing kernel Hilbert spaces

In classification and other tasks in statistical learning it is often convenient to map the features from the feature space $\mathcal{X}$ to a higher-dimensional inner product space $\mathcal{H}$ where the classes can be easily separated. A kernel is a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that directly computes the inner product in this higher-dimensional space, without computing the new coordinates or requiring the explicit feature map $\psi : \mathcal{X} \to \mathcal{H}$. Therefore kernels can be thought of as similarity measures that arise from a particular representation of the features,

with large K indicating that the two features are similar. We formally define a *kernel* as a function $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ that for some map $\psi : \mathcal{X} \to \mathcal{H}$ satisfies

$$K(x, y) = \langle \psi(x), \psi(y) \rangle.$$

By this definition a kernel is symmetric, and positive definite, that is

$$\sum_{i=1}^{m} \sum_{j=1}^{m} c_i c_j K(x_i, x_j) \geq 0,$$

for all $m \in \mathbb{R}$, $c_1, \ldots, c_m \in \mathbb{R}$ and $x_1, \ldots, x_m \in \mathcal{X}$.

Conversely, for any symmetric and positive definite function K an inner product space $\mathcal{H}$ and a feature map ψ can be constructed for which K is the corresponding kernel. We sketch the construction because it introduces the concept of reproducing kernel Hilbert spaces.

Suppose that $K : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a symmetric and positive definite function, and consider the vector space $\mathcal{H}$ defined as

$$\mathcal{H} = \left\{ f \in \mathbb{R}^{\mathcal{X}} : \exists m \in \mathbb{N}, \alpha_1, \ldots, \alpha_m \in \mathbb{R}, x_1, \ldots, x_m \in \mathcal{X}; f(\cdot) = \sum_{i=1}^{m} \alpha_i K(\cdot, x_i) \right\}.$$

For $f, g \in \mathcal{H}$, given by

$$f(\cdot) = \sum_{i=1}^{m} \alpha_i K(\cdot, x_i), \qquad g(\cdot) = \sum_{j=1}^{m'} \beta_j K(\cdot, x'_j),$$

let

$$\langle f, g \rangle := \sum_{i=1}^{m} \sum_{j=1}^{m'} \alpha_i \beta_j K(x_i, x'_j),$$

which is a well-defined inner product on $\mathcal{H}$ by positive definiteness of K. Note that $f(x) = \langle K(\cdot, x), f \rangle$ for all $f \in \mathcal{H}$ and $x \in \mathcal{X}$. In particular we have that $K(x, y) = \langle K(\cdot, x), K(\cdot, y) \rangle$ for all $x, y \in \mathcal{H}$. We define the feature map ψ as

$$\begin{aligned} \psi : \mathcal{X} &\to \mathcal{H} \\ x &\mapsto K(\cdot, x). \end{aligned}$$

Then

$$K(x, y) = \langle K(\cdot, x), K(\cdot, y) \rangle = \langle \psi(x), \psi(y) \rangle,$$

for all $x, y \in \mathcal{X}$. We conclude that K is a kernel.

The *reproducing kernel Hilbert space* (RKHS) $(\mathcal{F}, \|\cdot\|)$ is the Hilbert space of functions on $\mathcal{X}$ obtained by completing $\mathcal{H}$ in the norm induced by the inner product. Let $\mathcal{F}_B = \{f \in \mathcal{F} : \|f\| \leq B\}$. By the Representer Theorem [25, 21], for training examples $\{(x_i, y_i)\}_{i=1}^{N}$ and any convex loss function ℓ, the empirical risk minimizer

$$\hat{f}_N := \arg\min_{f \in \mathcal{F}_B} \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i), y_i),$$

admits a representation of the form

$$\hat{f}_N(x) := \sum_{i=1}^{N} \alpha_i K(x_i, x), \qquad \text{where } \sum_{i,j=1}^{N} \alpha_i \alpha_j K(x_i, x_j) \leq B^2,$$

so that finding $\hat{f}_N$ is a convex optimization problem of computing the vector α.

4.2 The recursive kernel method

We consider an improper learning algorithm for networks with ℓ_1-bounded weight vectors, based on the kernel method. An *improper learning algorithm* for networks is an algorithm that learns a predictor that is not necessarily a network, but that performs with high probability almost as well as the best-fitting network. By constructing a kernel whose induced RKHS contains the class of networks with ℓ_1-bounded weights, Zhang et al. [25] obtain an improper learning algorithm for those networks by virtue of the Representer Theorem.

Recall the notation for networks introduced in Section 2.1. Let $\mathcal{N}_{k,\sigma,d,L}$ be the set of artificial neural networks with k hidden layers and activation function σ, where the ℓ_2-norm of the input $x \in [-1,1]^d$ is bounded by 1, where for all neurons in the first hidden layer the ℓ_2-norm of the vector of incoming weights is bounded by L, and where for all neurons in higher layers the ℓ_1-norm of the vector of incoming weights is bounded by L. These conditions imply that we only consider networks that are sufficiently sparse. Our goal is to construct a kernel whose induced RKHS contains $\mathcal{N}_{k,\sigma,d,L}$. In this section we present the case where the activation function has a polynomial expansion $\sigma(z) = \sum_{j=0}^{\infty} \beta_j z^j$, as developed by [25].

We first change the feature space by embedding the input $x = (x_1, \ldots, x_d) \in \mathbb{R}^d$ with $\|x\|_2 \leq 1$ into $\mathbb{R}^{\mathbb{N}}$ by setting $x_i = 0$ for all $i > d$. Let the function $K : \mathbb{R}^{\mathbb{N}} \times \mathbb{R}^{\mathbb{N}} \to \mathbb{R}$ be given by

$$K(x, y) := \frac{1}{2 - \langle x, y \rangle}.$$

Note that $\langle x, y \rangle \leq 1$, so K is bounded by 1. We show that K is a kernel. To define the feature map $\psi : \mathbb{R}^{\mathbb{N}} \to \mathbb{R}^{\mathbb{N}}$ we use a different enumeration of $\mathbb{N}$. The $(k_1, \ldots, k_j)$-th coordinate of $\psi(x)$ is given by $2^{-\frac{j+1}{2}} x_{k_1} \ldots x_{k_j}$ for all $j \in \mathbb{N}$ and $k_1, \ldots, k_j \in \mathbb{N}$. It follows that

$$\begin{aligned}
\langle \psi(x), \psi(y) \rangle &= \sum_{j=0}^{\infty} 2^{-(j+1)} \sum_{(k_1,\ldots,k_j) \in \mathbb{N}^j} x_{k_1} \ldots x_{k_j} y_{k_1} \ldots y_{k_j} \\
&= \sum_{j=0}^{\infty} 2^{-(j+1)} (\langle x, y \rangle)^j \\
&= \frac{1}{2 - \langle x, y \rangle}.
\end{aligned}$$

Therefore K is indeed a kernel.

Moreover, we have that $\psi(x) \in \mathbb{R}^{\mathbb{N}}$ with $\|\psi(x)\|_2^2 = K(x,x) \leq 1$ because $\|x\|_2 \leq 1$. Hence we can iterate the map ψ and obtain a sequence of feature maps

$$\psi^{(k)}(x) = \psi(\psi^{(k-1)}(x)) \text{ and } \psi^{(0)}(x) = x,$$

that corresponds to the sequence of kernels

$$K^{(k)}(x,y) = \frac{1}{2 - K^{(k-1)}(x,y)} \text{ and } K^{(0)}(x,y) = \langle x, y \rangle,$$

which by induction are all bounded by 1. The kernels induce a sequence of RKHS's $\mathcal{F}_k$, hence the name *recursive kernel method*.

For fixed k, with an appropriate assumption on the polynomial expansion of the activation function σ, there exists a constant $H^{(k)}(L)$ such that the set $\mathcal{F}_{k,H^{(k)}(L)} := \{f \in \mathcal{F}_k : \|f\| \leq H^{(k)}(L)\}$ contains the class of networks $\mathcal{N}_{k,\sigma,d,L}$. We show this in the following technical lemma.

Lemma 4.1. *Let $\sigma(z) = \sum_{j=0}^{\infty} \beta_j z^j$ and recursively define $H^{(k)}(\lambda) = H(H^{(k-1)}(\lambda))$ for $k \geq 1$ where $H(\lambda) := L\sqrt{\sum_{j=0}^{\infty} 2^{j+1}\beta_j^2 \lambda^{2j}}$ and $H^{(0)}(\lambda) = \lambda$. If $H^{(k)}(L)$ is finite, then*

$$\mathcal{N}_{k,\sigma,d,L} \subset \mathcal{F}_{k,H^{(k)}(L)}.$$

Proof. (Adapted from [25, Appendix A])

Let $f \in \mathcal{N}_{k,\sigma,d,L}$, and recall the notation for networks introduced in Section 2.1. To prove the lemma, we need to show that $z_1^{(k+1)} = \sum_{j=1}^{d^{(k)}} w_{1,j}^{(k)} y_j^{(k)} \in \mathcal{F}_{k,H^{(k)}(L)}$. We show by induction that $z_i^{(p+1)} = \sum_{j=1}^{d^{(p)}} w_{i,j}^{(p)} y_j^{(p)} \in \mathcal{F}_{p,H^{(p)}(L)}$ for all $i = 1, \ldots, d^{(p+1)}$ and all $p = 0, 1, \ldots, k$. Note that $z_i^{(p+1)} = z_i^{(p+1)}(x)$ is a function of the input x, and that it represents the input of neuron i on layer $p+1$.

For the input layer, $p = 0$, note that for all $i = 1, \ldots, d^{(1)}$

$$z_i^{(1)}(x) = \sum_{j=1}^{d} w_{i,j}^{(0)} x_j = \langle w_i^{(0)}, x \rangle = \langle w_i^{(0)}, \psi^{(0)}(x) \rangle \in \mathcal{F}_0.$$

We also have that $\|z_i^{(1)}\|_{\mathcal{F}_0} = \|w_i^{(0)}\|_2 \leq L = H^{(0)}(L)$, so we conclude that $z_i^{(1)} \in \mathcal{F}_{0,H^{(0)}(L)}$.

Now assume that $z_i^{(p)} \in \mathcal{F}_{k,H^{(p-1)}(L)}$ for all neurons i on layer p, then we have that for all neurons on layer $p+1$

$$\begin{aligned} z_i^{(p+1)} &= \sum_{j=1}^{d^{(p)}} w_{i,j}^{(p)} y_j^{(p)} \\ &= \sum_{j=1}^{d^{(p)}} w_{i,j}^{(p)} \sigma(z_j^{(p)}(x)) \\ &= \sum_{j=1}^{d^{(p)}} w_{i,j}^{(p)} \sigma(\langle v_j, \psi^{(p-1)}(x) \rangle), \end{aligned}$$

where $v_j \in \mathbb{R}^{\mathbb{N}}$ with $\|v_j\|_2 \leq H^{(p-1)}(L)$ by the inductive hypothesis.

By the polynomial expansion of the activation function $\sigma(z) = \sum_{t=0}^{\infty} \beta_t z^t$, we obtain

$$\begin{aligned}
\sigma(\langle v_j, \psi^{(p-1)}(x)\rangle) &= \sum_{t=0}^{\infty} \beta_t (\langle v_j, \psi^{(p-1)}(x)\rangle)^t \\
&= \sum_{t=0}^{\infty} \beta_t \sum_{(k_1,\ldots,k_t)\in\mathbb{N}^t} v_{j,k_1} \ldots v_{j,k_t} \psi^{(p-1)}(x)_{k_1} \ldots \psi^{(p-1)}(x)_{k_t} \\
&= \langle u_j, \psi(\psi^{(p-1)}(x))\rangle,
\end{aligned}$$

where in the last step we used the definition of ψ and where $u_j \in \mathbb{R}^{\mathbb{N}}$ is a vector defined as having $(k_1, \ldots, k_t)$-th coordinate equal to $2^{\frac{t+1}{2}} \beta_t v_{j,k_1} \ldots v_{j,k_t}$, for all $t \in \mathbb{N}$ and $k_1, \ldots, k_t \in \mathbb{N}^+$.

It follows that

$$\begin{aligned}
z_i^{(p+1)} &= \sum_{j=1}^{d^{(p)}} w_{i,j}^{(p)} \sigma(\langle v_j, \psi^{(p-1)}(x)\rangle) \\
&= \sum_{j=1}^{d^{(p)}} w_{i,j}^{(p)} \langle u_j, \psi(\psi^{(p-1)}(x))\rangle \\
&= \langle \sum_{j=1}^{d^{(p)}} w_{i,j}^{(p)} u_j, \psi(\psi^{(p-1)}(x))\rangle
\end{aligned}$$

Hence $z_i^{(p+1)} \in \mathcal{F}_p$. It remains to show that $\|z_i^{(p+1)}\|_{\mathcal{F}_p} = \|\sum_{j=1}^{d^{(p)}} w_{i,j}^{(p)} u_j\|_2 \leq H^{(p)}(L)$. By the regularization of the network we have that

$$\|\sum_{j=1}^{d^{(p)}} w_{i,j}^{(p)} u_j\|_2 \leq \sum_{j=1}^{d^{(p)}} |w_{i,j}^{(p)}| \cdot \|u_j\|_2 \leq L \cdot \max_{j \in d^{(p)}} \|u_j\|_2$$

and from the inductive hypothesis it follows for all j that

$$\begin{aligned}
\|u_j\|_2 &= \sum_{t=0}^{\infty} 2^{t+1} \beta_t^2 \sum_{(k_1,\ldots,k_t)\in\mathbb{N}^t} v_{j,k_1}^2 \ldots v_{j,k_t}^2 \\
&= \sum_{t=0}^{\infty} 2^{t+1} \beta_t^2 \|v_j\|_2^{2t} \\
&\leq \sum_{t=0}^{\infty} 2^{t+1} \beta_t^2 (H^{(p-1)}(L))^{2t},
\end{aligned}$$

So that indeed $\|\sum_{j=1}^{d^{(p)}} w_{i,j}^{(p)} u_j\|_2 \leq H^{(p)}(L)$ and $z_i^{(p+1)} \in \mathcal{F}_{p,H^{(p)}(L)}$. □

The proof of the lemma is quite technical, but it is based on the fact that networks with polynomial activation functions represent polynomial functions. The kernel and feature map are chosen to complement the computations involved with polynomial functions of inner products.

The Representer Theorem now readily provides us with an improper learning algorithm for neural networks. Since $K^{(k)}$ is bounded by 1, the Rademacher complexity of $\mathcal{F}_{k,B} = \{f \in \mathcal{F}_k : \|f\| \leq B\}$ is bounded by $\sqrt{B^2/N}$ (cf. [15, Theorem 3.1 and Example 3.1.1]). Therefore by Shalev-Shwartz et al. [23, Theorem 2.2], we have that the empirical risk minimizer $\hat{f}_N$ in $\mathcal{F}_{k,B}$ is solvable in polynomial time and with probability at least $1-\delta$ it achieves

$$\mathbb{E}[\ell(\hat{f}_n(x), y)] \leq \arg\min_{f \in \mathcal{F}_{k,B}} \mathbb{E}[\ell(f(x), y)] + \epsilon,$$

when the sample size is of order $N = \Omega(B^2 \log(1/\delta)/\epsilon^2)$. The run time complexity is bounded by $\text{poly}(d, 1/\epsilon, \log(1/\delta), B)$.

We obtain the main theorem of the recursive kernel method by combining this result with Lemma 4.1.

Theorem 4.2. *If $H^{(k)}(L)$ as defined in Lemma 4.1 is finite then the predictor $\hat{f}_N$ is solvable in polynomial time and with probability at least $1-\delta$ it achieves*

$$\mathbb{E}[\ell(\hat{f}_n(x), y)] \leq \arg\min_{\mathcal{N} \in \mathcal{N}_{k,\sigma,d,L}} \mathbb{E}[\ell(f(x), y)] + \epsilon,$$

when the sample size is of order $N = \Omega([H^{(k)}(L)]^2 \log(1/\delta)/\epsilon^2)$. The run time complexity is bounded by poly$(d, 1/\epsilon, \log(1/\delta), H^{(k)}(L))$.

Hence the algorithm learns in polynomial time a predictor, which is not required to be a neural network, that with high probability is not much worse than the best-fitting neural network in the hypothesis class. Recall that the hypothesis class $\mathcal{N}_{k,\sigma,d,L}$ consists of networks with a polynomially expandable activation function and with ℓ_2-bounded weight-vectors for the first layer and ℓ_1-bounded weight-vectors for higher layers. We note that the restriction on the ℓ_1-norm of the weight-vectors is a strong assumption on the sparsity of the network. Many connections in the network are not used since a vector that is optimized under ℓ_1-constraints typically contains only a small number of non-zero components This also explains why the result does not depend on the number of nodes in each layer, as those numbers are effectively bounded by the ℓ_1-constraint. Theorem 4.2 implies that any sufficiently sparse network with a polynomially expandable activation function is learnable in polynomial time.

The sample complexity and run time complexity are polynomial in $H^{(k)}(L)$. The sample complexity is the order of training samples required for the algorithm to succesfully learn a predictor. We compute $H^{(k)}(L)$ for a quadratic activation function and show that it is finite.

Example 4.3. Suppose $\sigma(z) = z^2$, then $H(\lambda) = L \cdot \sqrt{2^3 \lambda^4}$. We show by induction that $H^{(k)}(L) = 2^{\frac{3}{2}(2^k-1)} L^{2^{k+1}-1}$. Clearly $H^{(1)}(L) = \sqrt{2^3 L^6} = 2^{\frac{3}{2}} L^3$, and if $H^{(k)}(L) = 2^{\frac{3}{2}(2^k-1)} L^{2^{k+1}-1}$ then

$$\begin{aligned}
H^{(k+1)}(L) &= L \cdot \sqrt{2^3 [H^{(k)}(L)]^4} \\
&= 2^{\frac{3}{2}} L \cdot [H^{(k)}(L)]^2 \\
&= 2^{\frac{3}{2}} L \cdot 2^{3(2^k-1)} L^{2^{k+2}-2} \\
&= 2^{3(2^k-\frac{1}{2})} L^{2^{k+2}-1} \\
&= 2^{\frac{3}{2}(2^{k+1}-1)} L^{2^{k+2}-1}.
\end{aligned}$$

□

The kernel used in this section is chosen for its convenience when dealing with activation functions that have a polynomial expansion. We showed that the theoretically important quadratic activation function satisfies the conditions of Theorem 4.2. Also the shifted error function discussed in Section 2.3 is a sigmoid function with a polynomial expansion such that $H^{(k)}(L)$ is finite [23, Appendix C]. Likewise a smoothed version of the rectifier activation function such as the softplus function can be considered, but they are known to have worse performance than the rectifier [8]. The rectifier units do not allow a polynomial expansion, so the kernel cannot be used for analyzing networks with rectifier activation. We propose an alternative kernel for that purpose.

4.3 Alternative kernel for rectifiers

In this section we propose an alternative kernel for improper learning of neural networks with the rectifier activation function. The kernel is based on a wavelet transform. We derive learnability results for networks with a single hidden layer and one-dimensional input. In Section 4.4 we propose an extension for multidimensional input.

Let $\mathcal{N}_L$ be the set of networks with a single hidden layer of n nodes, rectifier activation function $\sigma_r(z) = \max(0, z)$, one-dimensonal input $x \in [0, 1]$, weights from input to the hidden layer bounded by L and the ℓ_1-norm of the weight vector from hidden layer to the output bounded by L. We allow additive constants in the neuron input and write for the output of neuron i

$$\phi_{a_i,b_i}(x) := \sigma_r(a_i x - b_i) = \max(0, a_i x - b_i),$$

and

$$f(x) = \sum_{i=1}^{n} c_i \phi_{a_i,b_i}(x)$$

for the output of a network. By assumption we have that $|a_i| \leq L$ for all i and $\sum_{i=1}^{n} |c_i| \leq L$. Without loss of generality we may assume that $0 \leq b_i/a_i \leq 1$. Our goal is again to find a kernel with a RKHS that contains $\mathcal{N}_L$. The idea is to use a wavelet expansion of the rectifier instead of a polynomial expansion.

For an introduction to multiresolution analysis and wavelets, see for example the lecture notes by Johnstone [12, Chapter 7]. In our approach we will only use the so-called *Haar wavelet* which is the set of functions $\psi_{j,k}$ on $[0, 1]$, indexed by $j \in \mathbb{N}_0, k = 0, 1, \ldots, 2^j - 1$, with

$$\psi_{j,k}(x) = 2^{j/2}\psi(2^j x - k), \qquad \text{and } \psi(x) = \mathbf{1}_{[0,\frac{1}{2}]} - \mathbf{1}_{[\frac{1}{2},1]}.$$

The functions $(\psi_{j,k})_{j,k}$ form an orthonormal basis for $L^2[0, 1]$. The support of $\psi_{j,k}$ is $[k2^{-j}, (k+1)2^{-j}]$. See Figure 4.1 for an illustration of the unscaled Haar wavelets.

We expand the node output $\phi_{a,b}(x)$ in Haar wavelets,

$$\phi_{a,b}(x) = \sum_{j=0}^{\infty} \sum_{k=0}^{2^j-1} \langle \phi_{a,b}, \psi_{j,k} \rangle \psi_{j,k}(x),$$

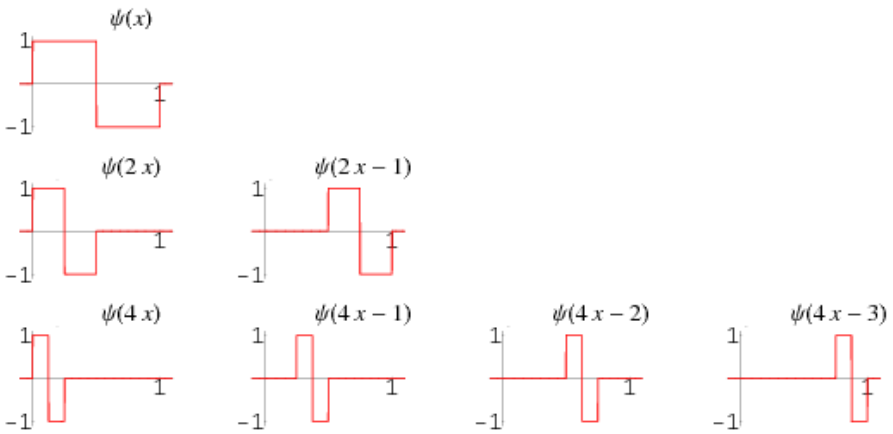


Figure 4.1: Illustration of the unscaled wavelets $\psi(2^j x - k)$, adapted from Wolfram Mathworld [24].

where the wavelet coefficients are given by

$$\langle \phi_{a,b}, \psi_{j,k} \rangle = \int_0^1 \phi_{a,b}(x)\psi_{j,k}(x)\,dx, \qquad j \in \mathbb{N}_0, k = 0, 1, \ldots, 2^j - 1.$$

If a function is approximately constant on the support of $\psi_{j,k}$ then the corresponding wavelet coefficient will be small. The wavelet coefficients are large only at low frequencies, that is for small j, or for wavelets with support on which the function changes in value relatively much. Therefore wavelets are well-suited for approximating discontinuous and non-differentiable functions.

We compute the wavelet coefficients for the rectifier activation function. We may assume that $a > 0$, as the case $a < 0$ follows similarly. If the support of $\psi_{j,k}$ contains only the constant part of the rectifier, that is if $(k+1)2^{-j} \leq \frac{b}{a}$, then $\langle \phi_{a,b}, \psi_{j,k} \rangle = 0$. Similarly if the support of $\psi_{j,k}$ contains only the non-zero linear part of the rectifier, that is if $k2^{-j} \geq \frac{b}{a}$, then

$$\begin{aligned}
\langle \phi_{a,b}, \psi_{j,k} \rangle &= \int_{k2^{-j}}^{(k+1)2^{-j}} (ax - b)\psi_{j,k}(x)\,dx \\
&= 2^{j/2}\Big[\int_{k2^{-j}}^{(k+1/2)2^{-j}} (ax - b)\,dx - \int_{(k+1/2)2^{-j}}^{(k+1)2^{-j}} (ax - b)\,dx \Big] \\
&= -\frac{a}{4} 2^{-3j/2}.
\end{aligned}$$

Finally, for the wavelet that overlaps the point of non-differentiability of $\phi_{a,b}$, that is $k2^{-j} < \frac{b}{a} < (k+1)2^{-j}$, we have that $|\langle \phi_{a,b}, \psi_{j,k} \rangle| \leq \frac{a}{4} 2^{-3j/2}$. If $a < 0$ then the signs of the coefficients are reversed.

Corollary 4.4. *Let $\phi_{a,b} = \max(0, ax - b)$, then for all j, k,*

$$|\langle \phi_{a,b}, \psi_{j,k} \rangle| \leq \frac{a}{4} 2^{-3j/2}.$$

We define

$$\Theta_{a,b} := (\langle \phi_{a,b}, \psi_{j,k} \rangle)_{j,k} \text{ and } \Psi(x) := (\psi_{j,k}(x))_{j,k},$$

so that

$$\phi_{a,b}(x) = \langle \Theta_{a,b}, \Psi(x) \rangle.$$

The output of the entire single-layer network f then becomes

$$f(x) = \sum_{i=1}^{n} c_i \phi_{a_i,b_i}(x) = \left\langle \sum_{i=1}^{n} c_i \Theta_{a_i,b_i}, \Psi(x) \right\rangle.$$

It would be naive to conclude that the network f is an element of a RKHS with feature map Ψ, since the inner product $\langle \Psi(x), \Psi(y) \rangle = \sum_{j=0}^{\infty} \sum_{k=0}^{2^j-1} \psi_{j,k}(x)\psi_{j,k}(y)$ is not well-defined if $x = y$ and not a dyadic rational. A dyadic rational is a rational number of the form $k2^{-j}$ for $j \in \mathbb{N}_0$ and $k = 0, 1, \ldots, 2^j - 1$. The distinction between dyadic rationals and other numbers is required because only for a dyadic rational x it holds that $\psi_{j,k}(x) = 0$ for all j large enough.

The idea is to obtain a well-defined kernel by shifting mass from $\Psi(x)$ to $\Theta_{a,b}$ by introduction of a reweighting factor. For $\frac{1}{2} < \gamma < 1$, we define

$$\Theta^{\gamma}_{a,b} := (2^{\gamma j} \langle \phi_{a,b}, \psi_{j,k} \rangle)_{j,k} \text{ and } \Psi^{\gamma}(x) := (2^{-\gamma j} \psi_{j,k}(x))_{j,k},$$

Then

$$\phi_{a,b} = \langle \Theta^{\gamma}_{a,b}, \Psi^{\gamma}(x) \rangle.$$

Moreover, we show that

$$K(x, y) := \langle \Psi^{\gamma}(x), \Psi^{\gamma}(y) \rangle$$

is a well-defined kernel.

Let $x, y \in [0, 1]$. We give an explicit formula for the kernel $K(x, y)$. Note that for fixed x and j there is at most one k such that $\psi_{j,k}(x) \neq 0$. We have to determine for all j and k whether $\psi_{j,k}(x)\psi_{j,k}(y)$ is positive, negative or zero. Consider the binary expansions

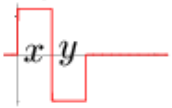
$$x = \sum_{i=1}^{\infty} \alpha_i 2^{-i} \text{ and } y = \sum_{i=1}^{\infty} \beta_i 2^{-i}, \qquad \alpha_i, \beta_i \in \{0, 1\}.$$

Also define

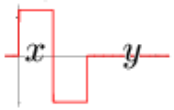
$$S(x, y) := \max\{i \geq 0 : a_j = b_j, \forall\ 0 < j \leq i\}.$$

If $j < S(x, y)$, then for exactly one k, $\psi_{j,k}(x)\psi_{j,k}(y) = 2^j$. If $j = S(x, y)$ and $x, y \neq (k' + 1/2)2^{-j}$ for some k', then for exactly one k, $\psi_{j,k}(x)\psi_{j,k}(y) = -2^j$. In all other cases $\psi_{j,k}(x)\psi_{j,k}(y) = 0$. See also Figure 4.2. It follows that

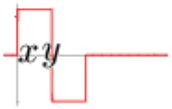
$\psi_{j,k}(x)\psi_{j,k}(y) = 2^j$



$\psi_{j,k}(x)\psi_{j,k}(y) = -2^j$



$\psi_{j,k}(x)\psi_{j,k}(y) = 0$



$\psi_{j,k}(x)\psi_{j,k}(y) = 0$

Figure 4.2: Illustration of possible outcomes of $\psi_{j,k}(x)\psi_{j,k}(y)$. Top left: x, y both in positive part of wavelet. Top right: x in positive part and y in negative part. Bottom left: y in zero part. Bottom right: y in zero part in the center of the wavelet.

$$
\begin{aligned}
K(x,y) &= \langle \Psi^\gamma(x), \Psi^\gamma(y) \rangle \qquad (1)\\
&= \sum_{j=0}^{\infty} \sum_{k=0}^{2^j-1} 2^{-2\gamma j} \psi_{j,k}(x)\psi_{j,k}(y) \\
&= \sum_{j=0}^{\infty} \sum_{k=0}^{2^j-1} 2^{-j(2\gamma-1)} \psi(2^j x - k)\psi(2^j y - k) \\
&= \sum_{j=0}^{S(x,y)-1} 2^{-j(2\gamma-1)} - 2^{-S(x,y)(2\gamma-1)} \mathbf{1}\{\nexists k' : x, y = (k'+1/2)2^{-j}\} \\
&= \frac{1 - 2^{-S(x,y)(2\gamma-1)}}{1 - 2^{-(2\gamma-1)}} - 2^{-S(x,y)(2\gamma-1)} \mathbf{1}\{\nexists k' : x, y = (k'+1/2)2^{-j}\}.
\end{aligned}
$$

In particular $S(x,x) = \infty$ and $K(x,x) = \frac{2^{(2\gamma-1)}}{2^{(2\gamma-1)}-1}$ when x is not a dyadic rational.

Corollary 4.5. *$K(x,y) = \langle \Psi^\gamma(x), \Psi^\gamma(y) \rangle \leq \frac{2^{(2\gamma-1)}}{2^{(2\gamma-1)}-1}$ for all x, y.*

Hence K is a well-defined kernel. We write $\mathcal{F}$ for the RKHS induced by K. Note that the explicit formula for K is needed to implement the improper learning algorithm.

For the kernel method we also require the norm of $\Theta^\gamma_{a,b}$ in the RKHS to be bounded. We already mentioned that wavelet coefficients typically have fast

decay if the expanded function does not have sharp fluctuations. We prove a lemma on the decay of wavelet coefficients for Hölder continuous functions.

Lemma 4.6. *(Adapted from lecture notes [9, week 8])*
Suppose that f is a square-integrable functon on $\mathbb{R}$ that is Hölder continuous with exponent $0 < \alpha \leq 1$ and constant β, that is $|f(x) - f(y)| \leq \beta|x - y|^\alpha$ for all x, y. Let ψ such that $\int \psi(x)\,dx = 0$ and $c_\psi(\alpha) := \int_0^1 |x|^\alpha |\psi(x)|\,dx < \infty$, then $|\langle f, \psi_{j,k}\rangle| \leq \beta c_\psi(\alpha) 2^{-j(\alpha+1/2)}$.

Proof. Since $\int \psi(x)\,dx = 0$,

$$\langle f, \psi_{j,k}\rangle = 2^{j/2} \int f(x)\psi(2^j x - k)\,dx = 2^{j/2} \int (f(x) - f(k2^{-j}))\psi(2^j x - k)\,dx.$$

Taking absolute values inside, and applying the Lipschitz condition,

$$\begin{aligned}
|\langle f, \psi_{j,k}\rangle| &\leq 2^{j/2}\beta \int |x - k2^{-j}|^\alpha |\psi(2^j x - k)|\,dx \\
&= 2^{j/2 - j\alpha}\beta \int |2^j x - k|^\alpha |\psi(2^j x - k)|\,dx \\
&= 2^{-j(\alpha+1/2)}\beta \int |x|^\alpha |\psi(x)|\,dx \\
&= 2^{-j(\alpha+1/2)}\beta c_\psi(\alpha)
\end{aligned}$$

□

Applying the general result to our case, we obtain the following:

Corollary 4.7. *Let $\phi_{a,b} = \max(0, ax - b)$, then*

$$|\langle \phi_{a,b}, \psi_{j,k}\rangle| \leq \frac{a}{2} 2^{-3j/2}$$

Proof. $\phi_{a,b} = \max(0, ax - b)$ is Lipschitz with exponent $\alpha = 1$ and constant a, since $|\phi_{a,b}(x) - \phi_{a,b}(y)| \leq a|x - y|$ for all $x, y \in [0, 1]$. The result follows by noting that $c_\psi(\alpha) := \int_0^1 |x|^\alpha |\psi(x)|\,dx = \int_0^1 |x|^\alpha\,dx = \frac{1}{\alpha+1}$. □

Now we are ready to bound the norm of $\Theta^\gamma_{a,b}$ and conclude that $\mathcal{N}_L$ is contained in the reproducing kernel Hilbert space $\mathcal{F}$ for the kernel $K(x, y) = \langle \Psi^\gamma(x), \Psi^\gamma(y)\rangle$.

Lemma 4.8. *For $\frac{1}{2} < \gamma < 1$ we have that*

$$\phi_{a,b}(x) = \sum_{j=0}^{\infty} \sum_{k=0}^{2^j - 1} \langle \phi_{a,b}, \psi_{j,k}\rangle \psi_{j,k}(x) = \langle \Theta^\gamma_{a,b}, \Psi^\gamma(x)\rangle,$$

with

$$\|\Theta^\gamma_{a,b}\|_2^2 \leq a^2 \frac{2^{-2\gamma}}{2^{2(1-\gamma)} - 1}.$$

Proof. By Corollary 4.7,

$$\begin{aligned}
\|\Theta^{\gamma}_{a,b}\|_2^2 &= \sum_{j=0}^{\infty}\sum_{k=0}^{2^j-1} 2^{2\gamma j}|\langle\phi_{a,b},\psi_{j,k}\rangle|^2 \\
&\leq \frac{a^2}{4}\sum_{j=0}^{\infty}\sum_{k=0}^{2^j-1} 2^{2\gamma j-3j} \\
&= \frac{a^2}{4}\sum_{j=0}^{\infty}(2^{-2(1-\gamma)})^j \\
&= \frac{a^2}{4}\frac{2^{2(1-\gamma)}}{2^{2(1-\gamma)}-1} \\
&= a^2\frac{2^{-2\gamma}}{2^{2(1-\gamma)}-1}.
\end{aligned}$$

□

Corollary 4.9. *Let* $B^{\gamma}(L) = L^2 \cdot \frac{1}{2}(\frac{2^{2(1-\gamma)}}{2^{2(1-\gamma)}-1})^{1/2}$*, then*

$$\mathcal{N}_L \subset \mathcal{F}_{B^{\gamma}(L)} = \{f \in \mathcal{F} : \|f\| \leq B^{\gamma}(L)\},$$

where $\mathcal{F}$ *is the RKHS induced by the kernel* K *given by Equation* (1)*.*

Proof. Let $f \in \mathcal{N}_L$, then

$$f(x) = \sum_{i=1}^{n} c_i\phi_{a_i,b_i}(x) = \left\langle \sum_{i=1}^{n} c_i\Theta^{\gamma}_{a_i,b_i}, \Psi(x)^{\gamma} \right\rangle,$$

with

$$\begin{aligned}
\|\sum_{i=1}^{n} c_i\Theta^{\gamma}_{a_i,b_i}\|_2 \leq \sum_{i=1}^{n}|c_i|\cdot\|\Theta^{\gamma}_{a_i,b_i}\|_2 &\leq \sum_{i=1}^{n}|c_i|\cdot\max_{1\leq i\leq n}\|\Theta^{\gamma}_{a_i,b_i}\|_2 \\
&= \sum_{i=1}^{n}|c_i|\cdot\max_{1\leq i\leq n} a_i\cdot\frac{1}{2}(\frac{2^{2(1-\gamma)}}{2^{2(1-\gamma)}-1})^{1/2} \\
&\leq L^2\cdot\frac{1}{2}(\frac{2^{2(1-\gamma)}}{2^{2(1-\gamma)}-1})^{1/2}.
\end{aligned}$$

□

Next we compute a bound on the Rademacher complexity. We have that K is bounded by $C^{\gamma} = \frac{2^{(2\gamma-1)}}{2^{(2\gamma-1)}-1}$ according to Corollary 4.5. The Rademacher complexity of $\mathcal{F}_{k,B^{\gamma}(L)} = \{f \in \mathcal{F}_k : \|f\| \leq B^{\gamma}(L)\}$ is bounded by $\sqrt{C^{\gamma}[B^{\gamma}(L)]^2/N}$ (cf. [15, Theorem 3.1 and Example 3.1.1]). We have that

$$\begin{aligned}
C^{\gamma}[B^{\gamma}(L)]^2 &= L^4\frac{1}{4}\frac{2^{(2\gamma-1)}}{2^{(2\gamma-1)}-1}\cdot\frac{2^{2(1-\gamma)}}{2^{2(1-\gamma)}-1} \\
&= L^4\frac{1}{2}\cdot\frac{1}{3-2^{2\gamma-1}-2^{2(1-\gamma)}}
\end{aligned}$$

where the minimum $C^\gamma[B^\gamma(L)]^2 = L^2(\frac{3}{2}+\sqrt{2})$ is obtained for $\gamma = \frac{3}{4}$. Note that $C^\gamma[B^\gamma(L)]^2$ becomes arbitrarily large as γ approaches $\frac{1}{2}$ or 1.

Therefore by Shalev-Shwartz et al. [23, Theorem 2.2], we have that the empirical risk minimizer $\hat{f}_N$ in $\mathcal{F}_{k,B^{(3/4)}(L)}$ is solvable in polynomial time and with probability at least $1-\delta$ it achieves

$$\mathbb{E}[\ell(\hat{f}_n(x), y)] \leq \arg\min_{f \in \mathcal{F}_{k,B^{(3/4)}(L)}} \mathbb{E}[\ell(f(x), y)] + \epsilon,$$

when the sample size is of order $N = \Omega(L^4 \log(1/\delta)/\epsilon^2)$. The run time complexity is bounded by $\text{poly}(d, 1/\epsilon, \log(1/\delta), L^2)$.

Combining this result with Corollary 4.9 we obtain the main theorem on improper learning of networks with rectifier units.

Theorem 4.10. *The predictor $\hat{f}_N$ is solvable in polynomial time and with probability at least $1-\delta$ it achieves*

$$\mathbb{E}[\ell(\hat{f}_n(x), y)] \leq \arg\min_{\mathcal{N} \in \mathcal{N}_L} \mathbb{E}[\ell(f(x), y)] + \epsilon,$$

when the sample size is of order $N = \Omega(L^4 \log(1/\delta)/\epsilon^2)$. The run time complexity is bounded by poly$(d, 1/\epsilon, \log(1/\delta), L^2)$.

Theorem 4.10 implies that we have a practical algorithm that learns in polynomial time a predictor, which is not required to be a neural network, that with high probability is not much worse than the best-fitting neural network in the hypothesis class. Recall that the hypothesis class consists of networks with a single hidden layer where the ℓ_1-norm of the vector of weights from the hidden layer to the output is bounded. The ℓ_1-norm restriction induces sparsity which for a single-layer networks considered here means that the number of nodes is effectively bounded. Theorem 4.10 is the equivalent of Theorem 4.2 for single-layer networks composed of rectifier units.

We have only considered networks with a single hidden layer and with one-dimensional input in this section. In the next section we propose as future work an extension of our methods to enable multidimensional input. An extension of our results which would enable improper learning of networks with more layers is not as straightforward since our kernel does not have the recursive property that is natural for polynomially expandable activation functions.

4.4 Extension to multidimensional input

In this section we propose an extension of the results of Section 4.3 to enable multidimensional input. To illustrate the idea, we assume two-dimensional input $x = (x_1, x_2) \in [0,1]^2$. The output of a node in the hidden layer is then given by

$$\phi_{a,b}(x) = \sigma_r(\langle a, x \rangle - b) = \max(0, a_1 x_1 + a_2 x_2 - b).$$

The method that we propose is to first consider $\phi_{a,b}$ as a function of x_1, with x_2 held constant. We apply the wavelet transform to obtain

$$\phi_{a,b}(x_1, x_2) = \sum_{j_1=0}^{\infty} \sum_{k_1=0}^{2^{j_1}-1} d_{j_1,k_1}(x_2) \psi_{j_1,k_1}(x_1),$$

where

$$d_{j_1,k_1}(x_2) = \int_0^1 \phi_{a,b}(x_1, x_2)\psi_{j_1,k_1}(x_1)\, dx_1$$

is a wavelet coefficient. Because these wavelet coefficients are a function of x_2 we can apply a wavelet transform to them to obtain

$$d_{j_1,k_1}(x_2) = \sum_{j_2=0}^{\infty} \sum_{k_2=0}^{2^{j_2}-1} d_{j_1,k_1,j_2,k_2}\psi_{j_2,k_2}(x_2)$$

with

$$\begin{aligned} d_{j_1,k_1,j_2,k_2} &= \int_0^1 d_{j_1,k_1}(x_2)\psi_{j_2,k_2}(x_2)\, dx_2 \\ &= \int_0^1 \int_0^1 \phi_{a,b}(x_1, x_2)\psi_{j_1,k_1}(x_1)\psi_{j_2,k_2}(x_2)\, dx_1 dx_2 \end{aligned}$$

Note that $d_{j_1,k_1,j_2,k_2} = 0$ if $[k_1 2^{-j_1}, (k_1+1)2^{-j_1}] \cap [k_2 2^{-j_2}, (k_2+1)2^{-j_2}] = \emptyset$. We substitute the expansion of the wavelet coefficients to obtain

$$\begin{aligned} \phi_{a,b}(x_1, x_2) &= \sum_{j_1=0}^{\infty} \sum_{k_1=0}^{2^{j_1}-1} \sum_{j_2=0}^{\infty} \sum_{k_2=0}^{2^{j_2}-1} d_{j_1,k_1,j_2,k_2}\psi_{j_2,k_2}(x_2)\psi_{j_1,k_1}(x_1) \\ &= \sum_{j_1=0}^{\infty} \sum_{k_1=0}^{2^{j_1}-1} \sum_{j_2=j_1}^{\infty} 2 \sum_{k_2=k_1 2^{(j_2-j_1)}}^{(k_1+1)2^{(j_2-j_1)}-1} d_{j_1,k_1,j_2,k_2}\psi_{j_2,k_2}(x_2)\psi_{j_1,k_1}(x_1) \\ &= \langle \Theta_{a,b}, \Psi(x_1, x_2) \rangle \end{aligned}$$

where

$$\Theta_{a,b} := (d_{j_1,k_1,j_2,k_2})_{j_1,k_1,j_2,k_2} \text{ and } \Psi(x_1, x_2) := (\psi_{j_1,k_1}(x_1)\psi_{j_2,k_2}(x_2))_{j_1,k_1,j_2,k_2}.$$

It remains to be investigated whether with a similar shift of mass as in Section 4.3 it would be possible to derive a well-defined kernel from $\Psi(x_1, x_2)$ while the norm of the other vector remains bounded.

5 Discussion

Explicit bounds on the approximation error have been given for single-layer neural networks with rectifier activation function and a class of functions satisfying some smoothness property. The rate of convergence was shown to be of order $O(n^{-1/2})$ in the number of nodes n. We expect that the bound can be improved as we have not made the most efficient use of the nodes.

Regarding the kernel method for improper learning of neural networks we have proposed and analyzed an alternative kernel based on a wavelet expansion because the original kernel does not always suffice, for example for studying networks with the rectifier activation function. Specifically, our results give a practical algorithm for the improper learning of single-layer neural networks with the rectifier activation function and bounded weights. The bound on the weights is effectively a bound on the number of nodes for our single-layer networks. We believe that our results can be extended to allow multidimensional inputs and we have proposed a method to investigate this. It is an open question whether networks with multiple hidden layers can be analyzed using our kernel or a different kernel. Also the analysis of neural networks using wavelets may be the subject of further studies as we have not made use of the full information provided by the wavelet coefficients.

The importance of our work is that it provides a stepping stone towards a better theoretical understanding of deep learning methods.

References

[1] Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory* **39.3** 930-945.

[2] Barron, A. R. (1994). Approximation and estimation bounds for artificial neural networks. *Machine Learning* **14.1** 115-133.

[3] Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning* **2.1** 1-127.

[4] Bengio, Y., Goodfellow, I. J. and Courville, A. (2016). Deep Learning. An MIT Press book in preparation. Draft chapters available at deeplearning-book.org.

[5] Bergstra, J., Desjardins, G., Lamblin, P. and Bengio, Y. (2009). Quadratic polynomials learn better image features. Technical Report 1337, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal.

[6] Blum, A. L. and Rivest, R. L. (1992). Training a 3-node neural network is NP-complete. *Neural Networks* **5.1** 117-127.

[7] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals and Systems* **2.4** 303-314.

[8] Glorot, X., Bordes, A. and Bengio, Y. (2011). Deep sparse rectifier neural networks. *Journal of Machine Learning Research: Workshop and Conference Proceedings* **15** 315-323.

[9] Güntürk, S. and Chou, E. (2010). Wavelets, Approximation Theory, and Signal Processing. New York University, Lecture notes.

[10] Hinton, G. E., Osindero, S. and Teh, Y. (2006). A fast learning algorithm for deep belief nets. *Neural Computation* **18.7** 1527-1554.

[11] Hornik, K., Stinchcombe, M. and White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural Networks* **2.5** 359-366.

[12] Johnstone, I. M. (2015). Gaussian estimation: Sequence and wavelet models. Stanford University, Lecture notes.

[13] Jones, L. K. (1992). A simple lemma on greedy approximation in Hilbert space and convergence rates for projection pursuit regression and neural network training. *The Annals of Statistics* **20.1** 608-613.

[14] Jones, L. K. (1997). The computational intractability of training sigmoidal neural networks. *IEEE Transactions on Information Theory* **43.1** 167-173.

[15] Kakade, S. M., Sridharan, K. and Tewari, A. (2009). On the complexity of linear prediction: Risk bounds, margin bounds, and regularization. *Advances in Neural Information Processing Systems* **22** 793-800.

[16] LeCun, Y., Bengio, Y. and Hinton, G. E. (2015). Deep learning. *Nature* **521** 436-444.

[17] Livni, R., Shalev-Shwartz, S. and Shamir, O. (2014). On the computational efficiency of training neural networks. *Advances in Neural Information Processing Systems* **27** 855-863.

[18] McCulloch, W. S. and Pitts, W. (1943) A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics* **5.4** 115-133.

[19] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review* **65.6** 386-408.

[20] Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature* **323** 533-536.

[21] Schölkopf, B., Herbrich, R. and Smola, A. J. (2001). A generalized representer theorem. *Lecture Notes in Artificial Intelligence* **2111** 416-426.

[22] Schölkopf, B. and Smola, A. J. (2001). Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT press, Cambridge, MA.

[23] Shalev-Shwartz, S., Shamir, O. and Sridharan, K. (2011). Learning kernel-based halfspaces with the 0-1 loss. *SIAM Journal on Computing* **40.6** 1623-1646.

[24] Weisstein, E. W. (2016). Haar Function. Available at MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/HaarFunction.html.

[25] Zhang, Y., Lee, J. D. and Jordan, M. I. (2015). ℓ_1-regularized neural networks are improperly learnable in polynomial time. Available at arXiv:1510.03528.