

Chapter 9

Neural networks

Chapter summary

- Neural networks are flexible models for non-linear predictions. They can be studied in terms of the three errors usually related to empirical risk minimization: optimization, estimation, and approximation errors. In this chapter, we focus primarily on single hidden layer neural networks, which are linear combinations of simple affine functions with additional non-linearities.
- Optimization error: as the prediction functions are non-linearly dependent on their parameters, we obtain non-convex optimization problems with only guaranteed convergence to stationary points.
- Estimation error: the number of parameters is not the driver of the estimation error, as the norms of the various weights play an important role, with explicit rates in $O(1/\sqrt{n})$ obtained from Rademacher complexity tools.
- Approximation error: for the “ReLU” activation function, the universal approximation properties can be characterized and are superior to kernel methods because they are adaptive to linear latent variables.

9.1 Introduction

In supervised learning, the main focus has been put on methods to learn from n observations $(x_i, y_i), i = 1, \dots, n$, with $x_i \in \mathcal{X}$ (input space) and $y_i \in \mathcal{Y}$ (output/label space). As presented in Chapter 4, a large class of methods relies on minimizing a regularized empirical risk with respect to a function $f : \mathcal{X} \rightarrow \mathbb{R}$ where the following cost function is minimized:

$$\frac{1}{n} \sum_{i=1}^n \ell(y_i, f(x_i)) + \Omega(f),$$

where $\ell : \mathcal{Y} \times \mathbb{R} \rightarrow \mathbb{R}$ is a loss function, and $\Omega(f)$ is a regularization term. Typical examples were:

- **Regression:** $\mathcal{Y} = \mathbb{R}$ and $\ell(y_i, f(x_i)) = \frac{1}{2}(y_i - f(x_i))^2$.
- **Classification:** $\mathcal{Y} = \{-1, 1\}$ and $\ell(y_i, f(x_i)) = \Phi(y_i f(x_i))$ where Φ is convex, e.g., $\Phi(u) = \max\{1 - u, 0\}$ (hinge loss leading to the support vector machine) or $\Phi(u) = \log(1 + \exp(-u))$ (leading to logistic regression). See more examples in Section 4.1.1.

The class of prediction functions we have considered so far were (with their “pros” and “cons”):

- **Linear functions in some explicit features:** given a feature map $\varphi : \mathcal{X} \rightarrow \mathbb{R}^d$, we consider $f(x) = \theta^\top \varphi(x)$, with parameters $\theta \in \mathbb{R}^d$, as analyzed in Chapter 3 (for least-squares) and Chapter 4 (for Lipschitz-continuous losses).
 - *Pros:* Simple to implement, as this leads to convex optimization with gradient descent algorithms, with running time complexity in $O(nd)$, as shown in Chapter 5, and theoretical guarantees which are not necessary scaling badly with dimension d if regularizers are used (ℓ_2 or ℓ_1).
 - *Cons:* Only applies to linear functions on explicit (and fixed feature spaces), so they can underfit the data.
- **Linear functions in some implicit features through kernel methods:** the feature map can have arbitrarily large dimension, that is, $\varphi(x) \in \mathcal{H}$ where \mathcal{H} is a Hilbert space, accessed through the kernel function $k(x, x') = \langle \varphi(x), \varphi(x') \rangle_{\mathcal{H}}$, as presented in Chapter 7.
 - *Pros:* Non-linear flexible predictions, simple to implement, can be used as convex optimization algorithms with strong guarantees. Provides adaptivity to the regularity of the target function, allowing higher-dimensional applications than local averaging methods from Chapter 6.
 - *Cons:* Running-time complexity up to $O(n^2)$ with algorithms from Section 7.4 (but this scaling can be improved with appropriate techniques also discussed in the same section, such as column sampling or random features). The method may still suffer from the curse of dimensionality for target functions that are not smooth enough.

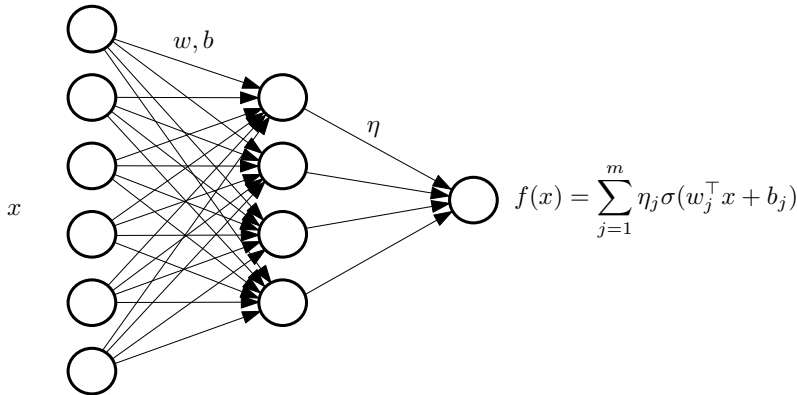
This chapter aims to explore another class of functions for non-linear predictions, namely neural networks, that come with additional benefits, such as more “adaptivity to linear latent variables”, but comes with some potential drawbacks, such as a harder optimization problem.

9.2 Single hidden layer neural network

We consider $\mathcal{X} = \mathbb{R}^d$ and the set of prediction functions that can be written as

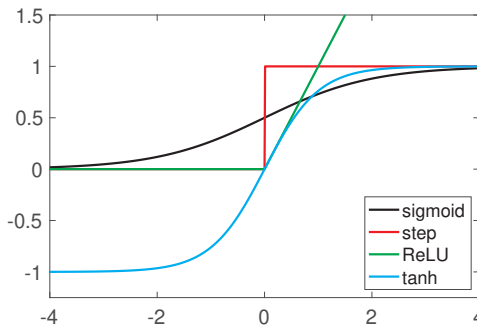
$$f(x) = \sum_{j=1}^m \eta_j \sigma(w_j^\top x + b_j), \quad (9.1)$$

where $w_j \in \mathbb{R}^d$, $b_j \in \mathbb{R}$, $j = 1, \dots, m$, are the “input weights”, $\eta_j \in \mathbb{R}$, $j = 1, \dots, m$, are the “output weights”, and σ is an “activation function”. This is often represented as a graph (see below). The same architecture can also be considered with $\eta_j \in \mathbb{R}^k$, for $k > 1$ to deal with multi-category classification (see Section 13.1).



The activation function is typically chosen from one of the following examples (see plot below):

- sigmoid $\sigma(u) = \frac{1}{1+e^{-u}}$,
- step function $\sigma(u) = 1_{u>0}$,
- “rectified linear unit” (ReLU) $\sigma(u) = (u)_+ = \max\{u, 0\}$, which will be the main focus of this chapter.
- hyperbolic tangent $\sigma(u) = \tanh(u) = \frac{e^u - e^{-u}}{e^u + e^{-u}}$.



The function f is defined as the linear combination of m functions $x \mapsto \sigma(w_j^\top x + b_j)$,

which are the “hidden neurons”.¹



The constant terms b_j are sometimes referred to as “biases”, which is unfortunate in a statistical context, as it already has a precise meaning within the bias/variance trade-off (see Chapter 3).



Do not get confused by the name “neural network” and its biological inspiration. This inspiration is not a proper justification for its behavior on machine learning problems.

Cross-entropy loss and sigmoid activation function for the last layer. Following standard practice, we are not adding a non-linearity to the last layer; note that if we were to use an additional sigmoid activation and consider the cross-entropy loss for binary classification, we would exactly be using the logistic loss on the output without an extra activation function.

Indeed, if we consider $g(x) = \frac{1}{1+\exp(-f(x))} \in [0, 1]$, and given an output variable $y \in \{-1, 1\}$, the so-called “cross-entropy loss”, an instance of maximum likelihood (see more details in Chapter 14), is equal to $-\frac{1+y}{2} \log g(x) - \frac{1-y}{2} \log(1-g(x))$. It can be rewritten as $\log(1 + \exp(-yf(x)))$, which is exactly the logistic loss defined in Section 4.1.1, applied to $f(x)$.


Theoretical analysis of neural networks. As with any method based on empirical risk minimization, we have to study the three classical aspects: (1) optimization (convergence properties of algorithms for minimizing the risk), (2) estimation error (effect of having a finite amount of data on the prediction performance), and (3) approximation error (effect of having a finite number of parameters or a constraint on the norm of these parameters).

9.2.1 Optimization

To find parameters $\theta = \{(\eta_j), (w_j), (b_j)\} \in \mathbb{R}^{m(d+2)}$, empirical risk minimization can be applied, and the following optimization problem has to be solved:

$$\min_{\theta \in \mathbb{R}^{m(d+2)}} \frac{1}{n} \sum_{i=1}^n \ell\left(y_i, \sum_{j=1}^m \eta_j \sigma(w_j^\top x_i + b_j)\right),$$

with potentially additional regularization (often squared ℓ_2 -norm of all weights).

 Note that (as discussed in Chapter 5) the true objective is to perform well on unseen data, and the optimization problem above is just a means to an end.

This is a non-convex optimization problem where the gradient descent algorithms from Chapter 5 can be applied without a strong guarantee beyond obtaining a vector with a

¹See <https://playground.tensorflow.org/> for a nice interactive illustration.

small gradient norm (Section 5.2.6). See below for recent results on providing qualitative global convergence guarantees when m is large.

While stochastic gradient descent remains an algorithm of choice (with also a good generalization behavior as discussed in Section 5.4), several algorithmic improvements have been observed to lead to better stability and performance: specific step-size decay schedules, preconditioning like presented in Section 5.4.2 (Duchi et al., 2011), momentum (Kingma and Ba, 2014), batch-normalization (Ioffe and Szegedy, 2015) or layer-normalization (Ba et al., 2016) to make the optimization better behaved, but overall, the objective function is non-convex, and it remains challenging to understand precisely why gradient-based methods perform well in practice, particularly for deeper networks (some elements are presented below and in Chapter 12). See also boosting procedures in Section 10.3 and Chapter 12, which learn neuron weights incrementally.

Global convergence of gradient descent for infinite widths (♦). It turns out that global convergence can be shown for this non-convex optimization problem (Chizat and Bach, 2018; Bach and Chizat, 2022), with tools that go beyond the scope of this book and which are partially described in Chapter 12.²

We simply show some experimental evidence below for a simple one-dimensional setup, where we compare several runs of stochastic gradient descent (SGD) where observations are only seen once (so no overfitting is possible) and with random initializations, on a regression problem with deterministic outputs, thus with the optimal testing error (the Bayes rate) being equal to zero. We show in Figure 9.1 the estimated predictors and the corresponding testing errors with 20 different initializations. We see that small errors are never achieved when $m = 5$ (which is sufficient to attain zero testing errors). With $m = 20$ neurons, SGD finds the optimal predictor for most restarts. When $m = 100$, all restarts have the desired behaviors, highlighting the benefits of over-parameterization.

9.2.2 Rectified linear units and homogeneity

From now on, we will mostly focus on the rectified linear unit $\sigma(u) = u_+$. The main property we will leverage is its “positive homogeneity”, that is, for $\alpha > 0$, $(\alpha u)_+ = \alpha u_+$. This implies that in the definition of the prediction function as the sum of terms $\eta_j(w_j^\top x + b_j)_+$, we can freely multiply $\eta_j \in \mathbb{R}$ by a positive scalar α_j and divide $(w_j, b_j) \in \mathbb{R}^{d+1}$ by the same α_j , without changing the prediction function.

This has a particular effect when using a squared ℓ_2 -regularizer on all weights, which is standard, either explicitly (by adding a penalty to the cost function) or implicitly (see Section 12.1). Indeed, we consider penalizing $\eta_j^2 + \|w_j\|_2^2 + b_j^2/R^2$ for each $j \in \{1, \dots, m\}$, where we have added the factor R^2 on the constant term for homogeneity reasons between the slope w_j and the constant term b_j (R will be a bound on the ℓ_2 -norm of input data). Dealing with unit homogeneity between η_j and $(w_j, b_j/R)$ does not matter, because of the invariance by rescaling described below.

²See also <https://francisbach.com/gradient-descent-neural-networks-global-convergence/> for more details.

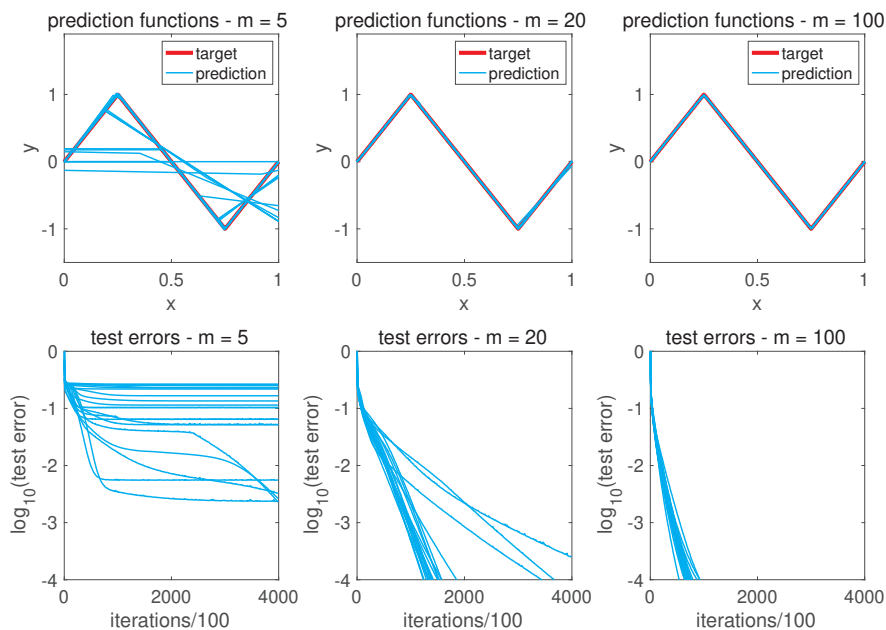


Figure 9.1: Comparison of optimization behavior for different numbers m of neurons, for ReLU activations (left: $m = 5$, middle: $m = 20$, and right: $m = 100$). The neural network used to generate the data (without noise) has 4 hidden neurons. Top: examples of final prediction functions at convergence, bottom: plot of test errors vs. number of iterations.

Optimizing with respect to a scaling factor α_j above (which impacts only the regularizer), we have to minimize $\alpha_j^2 \eta_j^2 + (\|w_j\|_2^2 + b_j^2/R^2)/\alpha_j^2$, with $\alpha_j^2 = [(\|w_j\|_2^2 + b_j^2/R^2)^{1/2}]/|\eta_j|$ as a minimizer, and with the optimal value of the penalty equal to $2|\eta_j|(\|w_j\|_2^2 + b_j^2/R^2)^{1/2}$.

For the theoretical analysis, we can thus choose to normalize each (w_j, b_j) to have unit norm $\|w_j\|_2^2 + b_j^2/R^2 = 1$, and use the penalty $|\eta_j|$ for each $j \in \{1, \dots, m\}$, and thus use an overall ℓ_1 -norm penalty on η , that is, $\|\eta\|_1$ (we will consider other normalizations for the input weights below, either to ease the exposition, or to induce another behavior, e.g., by using ℓ_1 -norms on the w_j 's). We now focus on this choice of regularization in the following sections.



In this chapter, R denotes an almost upper-bound on x directly, and not on a feature map $\varphi(x)$ (as done in earlier chapters).

9.2.3 Estimation error

To study the estimation error, we will consider that the parameters of the network are constrained, that is, $\|w_j\|_2^2 + b_j^2/R^2 \leq 1$ for each $j \in \{1, \dots, m\}$, and $\|\eta\|_1 \leq D$. This defines a set Θ of allowed parameters. Note that we use $\|w_j\|_2^2 + b_j^2/R^2 \leq 1$ instead of $\|w_j\|_2^2 + b_j^2/R^2 = 1$ (as suggested above) as it does not impact the bound on estimation error (and uniform convergence results on a bigger set of functions apply to a smaller set).

We can then compute the Rademacher complexity of the associated class \mathcal{F} of functions we just defined, using tools from Chapter 4 (Section 4.5). We assume that almost surely, $\|x\|_2 \leq R$, that is, the input data are bounded in ℓ_2 -norm by R .

Following the developments of Section 4.5 on Rademacher averages, we denote by $\mathcal{G} = \{(x, y) \mapsto \ell(y, f(x)), f \in \mathcal{F}\}$, the set of loss functions for a prediction function $f \in \mathcal{F}$ (which is here the set of neural network models f_θ with parameters θ such that $\theta \in \Theta$). Note that following Section 4.5.3, we consider a constraint on $\|\eta\|_1$, but we could also penalize, which is closer to practice and can be tackled with tools from Section 4.5.5.

We have, by definition of the Rademacher complexity $R_n(\mathcal{G})$ of \mathcal{G} , and taking expectations with respect to the data (x_i, y_i) , $i = 1, \dots, n$ (which are assumed i.i.d.) and the independent Rademacher random variables $\varepsilon_i \in \{-1, 1\}$, $i = 1, \dots, n$:

$$R_n(\mathcal{G}) = \mathbb{E} \left[\sup_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^n \varepsilon_i \ell(y_i, f_\theta(x_i)) \right].$$

This quantity is known to provide an upper-bound on the expected risk (e.g., testing error) $\mathcal{R}(\hat{f})$ of the minimizer $\hat{f} \in \mathcal{F}$ of the empirical risk, through the estimation error, as (using symmetrization from Prop. 4.2 and Eq. (4.8) from Section 4.4):

$$\mathbb{E} \left[\mathcal{R}(\hat{f}) - \inf_{f \in \mathcal{F}} \mathcal{R}(f) \right] \leq 4R_n(\mathcal{G}).$$

We can now use properties of Rademacher complexities presented in Section 4.5, particularly their nice handling of non-linearities. Assuming the loss is almost surely G -Lipschitz-

continuous with respect to the second variable, using Proposition 4.3 from Chapter 4 that allows getting rid of the loss, we get the bound:

$$R_n(\mathcal{G}) \leq G \cdot \mathbb{E} \left[\sup_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^n \varepsilon_i f_{\theta}(x_i) \right] = G \cdot \mathbb{E} \left[\sup_{\theta \in \Theta} \frac{1}{n} \sum_{i=1}^n \sum_{j=1}^m \eta_j \varepsilon_i \sigma(w_j^{\top} x_i + b_j) \right].$$

Using the ℓ_1 -constraint on η and using $\sup_{\|\eta\|_1 \leq D} z^{\top} \eta = D \|z\|_{\infty}$, we can directly maximize with respect to $\eta \in \mathbb{R}^m$, leading to (note that another ℓ_p -constraint on η , with $p \neq 1$, would be harder to deal with):

$$R_n(\mathcal{G}) \leq G \cdot \mathbb{E} \left[\sup_{j \in \{1, \dots, m\}} \sup_{\|w_j\|_2^2 + b_j^2 / R^2 \leq 1} D \left| \frac{1}{n} \sum_{i=1}^n \varepsilon_i \sigma(w_j^{\top} x_i + b_j) \right| \right].$$

Since the ReLU activation function σ is 1-Lipschitz continuous and satisfies $\sigma(0) = 0$, we get, this time using the extension of Proposition 4.3 from Chapter 4 to Rademacher complexities defined with an absolute value (that is, Prop. 4.4), which adds an extra factor of 2:

$$R_n(\mathcal{G}) \leq 2GD \cdot \mathbb{E} \left[\sup_{j \in \{1, \dots, m\}} \sup_{\|w_j\|_2^2 + b_j^2 / R^2 \leq 1} \left| w_j^{\top} \left(\frac{1}{n} \sum_{i=1}^n \varepsilon_i x_i \right) + b_j \left(\frac{1}{n} \sum_{i=1}^n \varepsilon_i \right) \right| \right].$$

We can now perform the optimization with respect to (w_j, b_j) in closed form (which can be done using Cauchy-Schwarz inequality), with the same value for all $j \in \{1, \dots, m\}$, leading to:

$$R_n(\mathcal{G}) \leq 2GD \cdot \mathbb{E} \left[\left(\left\| \frac{1}{n} \sum_{i=1}^n \varepsilon_i x_i \right\|_2^2 + R^2 \left(\frac{1}{n} \sum_{i=1}^n \varepsilon_i \right)^2 \right)^{1/2} \right].$$

We thus get, using Jensen's inequality (here of the form $\mathbb{E}[Z] \leq \sqrt{\mathbb{E}[Z^2]}$), as well as the independence, zero mean, and unit variance of $\varepsilon_1, \dots, \varepsilon_n$:

$$\begin{aligned} R_n(\mathcal{G}) &\leq 2GD \left(\mathbb{E} \left[\left\| \frac{1}{n} \sum_{i=1}^n \varepsilon_i x_i \right\|_2^2 + R^2 \left(\frac{1}{n} \sum_{i=1}^n \varepsilon_i \right)^2 \right] \right)^{1/2} \\ &= 2GD \left(\frac{1}{n} \mathbb{E}[\|x\|_2^2] + \frac{R^2}{n} \right)^{1/2} \leq \frac{2GDR\sqrt{2}}{\sqrt{n}}. \end{aligned} \tag{9.2}$$

Thus, we get the following proposition, with a bound proportional to $1/\sqrt{n}$ with no explicit dependence in the number of parameters.

Proposition 9.1 *Let \mathcal{G} be the class of functions $(y, x) \mapsto \ell(y, f(x))$ where f is a neural network defined in Eq. (9.1), with the constraint that $\|\eta\|_1 \leq D$, $\|w_j\|_2^2 + b_j^2 / R^2 \leq 1$ for all $j \in \{1, \dots, m\}$. If the loss function is G -Lipschitz-continuous and the activation function σ is the ReLU, the Rademacher complexity is upper bounded as*

$$R_n(\mathcal{G}) \leq \frac{4GDR}{\sqrt{n}}.$$

The proposition above allows for the estimation error to be bounded for neural networks, as the maximal deviation between expected risk and empirical risk over all potential networks with bounded parameters is bounded in expectation by four times the Rademacher complexity above.

This will be combined with a study of the approximation properties in Section 9.3, with a summary in Section 9.4.



For the estimation error, the number of parameters is irrelevant!
What counts is the overall norm of the weights.

We will see in Chapter 12 some recent results showing how optimization algorithms add an implicit regularization that leads to provable generalization in over-parameterized neural networks (that is, networks with many hidden units).

Exercise 9.1 (♦) *Provide a bound similar to Prop. 9.1 for the constraint $\|w_j\|_1 + |b_j|/R \leq 1$, where R denotes the supremum of $\|x\|_\infty$ over all x in the support of its distribution.*

Before moving on to approximation properties of neural networks, we note that the reasoning above to compute the Rademacher complexity can be extended by recursion to deeper networks, as the following exercise shows (see, e.g., Neyshabur et al., 2015, for further results).

Exercise 9.2 (♦) *We consider a 1-Lipschitz-continuous activation function σ such that $\sigma(0) = 0$, and the classes of functions defined recursively as $\mathcal{F}_0 = \{x \mapsto \theta^\top x, \|\theta\|_2 \leq D_0\}$, and, for $i = 1, \dots, M$, $\mathcal{F}_i = \{x \mapsto \sum_{j=1}^{m_i} \theta_j \sigma(f_j(x)), f_j \in \mathcal{F}_{i-1}, \|\theta\|_1 \leq D_i\}$, corresponding to a neural network with M layers. Assuming that $\|x\|_2 \leq R$ almost surely, show by recursion that the Rademacher complexity satisfies $R_n(\mathcal{F}_M) \leq 2^M \frac{R}{\sqrt{n}} \prod_{i=0}^M D_i$.*

9.3 Approximation properties

As seen above, the estimation error for constrained output weights grows as $\|\eta\|_1/\sqrt{n}$, where η is the vector of output weights and is independent of the number m of neurons. Three important questions will be tackled in the following sections:

- **Universality:** Can we approximate any prediction function with a sufficiently large number of neurons?
- **Bound on approximation error:** What is the associated approximation error so that we can derive generalization bounds? How can we use the control of the ℓ_1 -norm $\|\eta\|_1$, particularly when the number of neurons m is allowed to tend to infinity?
- **Finite number of neurons:** What is the number of neurons required to reach such a behavior?

For this, we need to understand the space of functions that neural networks span and how they relate to the smoothness properties of the function (like we did for kernel methods in Chapter 7).

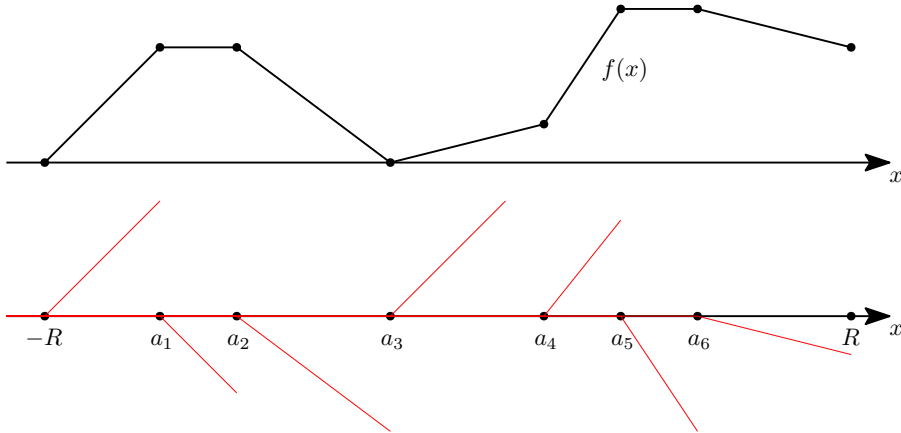
In this section, like in the previous section, we focus primarily on the ReLU activation function, noting that universal approximation results exist as soon as σ is not a polynomial (Leshno et al., 1993). We start with a simple non-quantitative argument to show universality in one dimension (and then in all dimensions) before formalizing the function space obtained by letting the number of neurons go to infinity.

9.3.1 Universal approximation property in one dimension

We start with simple non-quantitative arguments.

Approximation of piecewise affine functions. Since each individual function $x \mapsto \eta_j(w_j x + b_j)_+$ is piecewise affine, the output of a neural network has to be piecewise affine. It turns out that all piecewise affine functions with $m - 2$ kinks in the open interval $(-R, R)$ can be represented by m neurons on $[-R, R]$.

Indeed, as illustrated below with $m = 8$, if we assume that the function f is such that $f(-R) = 0$, with kinks $a_1 < \dots < a_{m-2}$ on $(-R, R)$, we can approximate it on $[-R, a_1]$ by the function $v_1(x + R)_+$ where v_1 is the slope of f on $[-R, a_1]$. The approximation is tight on $[-R, a_1]$. To have a tight approximation on $[a_1, a_2]$ without perturbing the approximation on $[-R, a_1]$, we can add to the approximation $v_2(x - a_1)_+$ where v_2 is exactly what is needed to compensate the change in slope of f . By pursuing the reasoning, we can represent the function on $[-R, R]$ exactly with $m - 1$ neurons.



To remove the constraint that $f(-R) = 0$, we can simply notice that $\frac{1}{2R}(x + R)_+ + \frac{1}{2R}(-x + R)_+$ is equal to 1 on $[-R, R]$. Thus, with one additional neuron (only one since $(x + R)_+$ has already been used), we can represent any piecewise-affine function with $m - 2$ kinks with m neurons.

Universal approximation properties. Now that we can represent precisely all piecewise affine functions on $[-R, R]$, we can use classical approximation theorems for functions on $[-R, R]$. They come in different flavors depending on the norm we use to characterize the approximation. For example, continuous functions can be approximated by piecewise affine functions with arbitrary precision in L_∞ -norm (defined as the maximal value of $|f(x)|$ for $x \in [-R, R]$) by simply taking the piecewise interpolant from a grid (see quantitative arguments in Section 9.3.3). With a weaker criterion such as the L_2 -norm (with respect to the Lebesgue measure), we can approximate any function in L_2 (see, e.g., Rudin, 1987). This can be extended to any dimension d by using the Fourier transform representation as $f(x) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \hat{f}(\omega) e^{i\omega^\top x} d\omega$ and approximating the one-dimensional functions sine and cosine as linear superpositions of ReLU's. See a more formal quantitative argument in Section 9.3.4.

To obtain precise bounds in all dimensions, in terms of the number of kinks or the ℓ_1 -norm of output weights, we first need to define the limit when the number of neurons can be unbounded.

9.3.2 Infinitely many neurons and variation norm

In this section, we consider neural networks of the form $f(x) = \sum_{j=1}^m \eta_j \sigma(w_j^\top x + b_j)$, where the input weights are constrained, that is, $(w_j, b_j/R) \in K$, for K a compact subset of \mathbb{R}^{d+1} , such as the unit ℓ_2 -sphere (but we will consider a slightly different set at the end of this section). In subsequent sections, we will primarily consider the ReLU activation σ , but this is not needed in this section (where boundedness or Lipschitz-continuity are sufficient).

In this section, for a function $f : \mathcal{X} \rightarrow \mathbb{R}$, where \mathcal{X} is the ℓ_2 -ball of radius R and center 0 in \mathbb{R}^d , we want to study the limit when $m \rightarrow \infty$, of the smallest ℓ_1 -norm of η for a function f representable with m neurons and output weights η , that is,

$$\gamma_1^{(m)}(f) = \inf_{\eta_j \in \mathbb{R}, (w_j, b_j) \in K, \forall j \in \{1, \dots, m\}} \|\eta\|_1 \quad \text{such that} \quad \forall x \in \mathcal{X}, f(x) = \sum_{j=1}^m \eta_j \sigma(w_j^\top x + b_j).$$

The index 1 in $\gamma_1^{(m)}$ will become natural when we compare with kernels in Section 9.5. When f is not representable by m neurons, we let $\gamma_1^{(m)}(f) = +\infty$. By construction the sequence $\gamma_1^{(m)}(f)$ is non-increasing in m , and non-negative. Thus, it has a limit when m tends to ∞ , which we denote $\gamma_1(f)$, which is infinite when f cannot be approximated by a neural network with finitely many neurons and output weights bounded in ℓ_1 -norm. The function γ_1 is positively homogeneous, that is $\gamma_1(\lambda f) = \lambda \gamma_1(f)$ when $\lambda > 0$ and sub-additive, that is, $\gamma_1(f + g) \leq \gamma_1(f) + \gamma_1(g)$. Moreover, one can show that $\gamma_1(f) = 0$ implies that $f = 0$ (proofs left as an exercise). Thus, γ_1 is a norm on the set of functions $f : \mathcal{X} \rightarrow \mathbb{R}$ such that $\gamma_1(f) < +\infty$. It is possible to “complete” this space by adding limits of functions such that their γ_1 -norm remains bounded. We then obtain a Banach space \mathcal{F}_1 of functions, with a norm γ_1 , often referred to as the “variation norm” (Kurková and Sanguineti, 2001). This characterizes the set of functions that can be represented

by neural networks with bounded ℓ_1 -norm of output weights, regardless of the number of neurons.

Formulation through measures. We can write $f(x) = \sum_{j=1}^m \eta_j \sigma(w_j^\top x + b_j)$, as $f(x) = \int_K (w^\top x + b)_+ d\nu(w, b)$, for ν the measure $\nu = \sum_{j=1}^m \eta_j \delta_{(w_j, b_j)}$ where $\delta_{(w_j, b_j)}$ is the Dirac measure at (w_j, b_j) . Then, the penalty can be written as $\|\eta\|_1 = \int_K |d\nu(w, b)|$, which is the “total variation” of ν .³ We can therefore see $\gamma_1(f)$ as the infimum of all $\int_K |d\nu(w, b)|$ such that $\forall x \in \mathcal{X}$, $f(x) = \int_K (w^\top x + b)_+ d\nu(w, b)$, and ν is supported on a countable set. By a density argument (every measure is the “weak” limit of empirical measures), we can remove the constraint on the support and get that

$$\gamma_1(f) = \inf_{\nu \in \mathcal{M}(K)} \int_K |d\nu(w, b)| \text{ such that } \forall x \in \mathcal{X}, f(x) = \int_K \sigma(w^\top x + b) d\nu(w, b), \quad (9.3)$$

where $\mathcal{M}(K)$ is the set of measures on K with finite total variation (see [Bach, 2017](#), and references therein for more details and formal definitions). This formulation is typically easier to deal with for sufficiently smooth functions, as the optimal measure ν is typically not a finite sum of Diracs (see examples below, in particular in one dimension). Note that when K is a compact convex set, then the norm γ_1 is the same when using K in its definition or replacing it by its boundary (that is, we can choose the unit ℓ_2 -sphere or the unit ℓ_2 -ball).

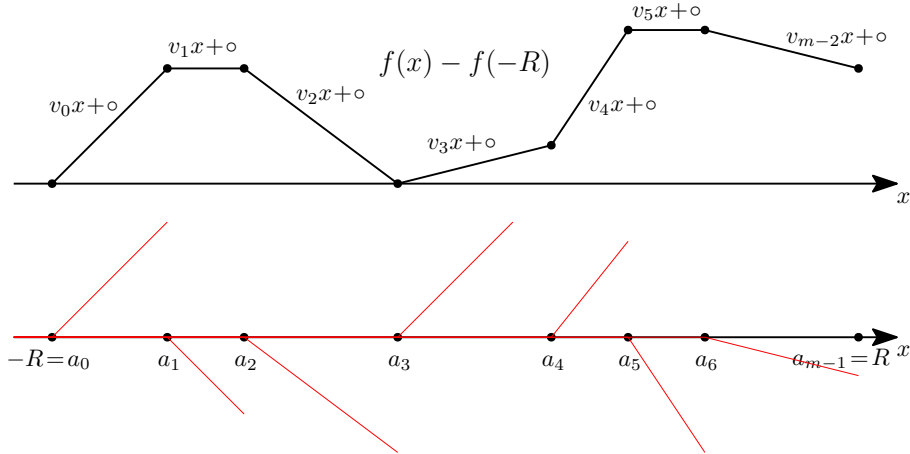
Studying the approximation properties of \mathcal{F}_1 . Now that we have defined the function space through Eq. (9.3), we need to describe the set of functions with finite norm and relate this norm to classical smoothness properties (like done for kernel methods in Chapter 7). To do so, we consider a smaller set than the unit ℓ_2 -ball, that is, the set of $(w, b/R)$ such that $\|w\|_2 = 1/\sqrt{2}$ and $|b| \leq R/\sqrt{2}$, which is enough to obtain upper-bounds on the approximation errors. For simplicity, and losing a factor $\sqrt{2}$, we consider the normalization $K = \{(w, b) \in \mathbb{R}^{d+1}, \|w\|_2 = 1, |b| \leq R\}$, and consider the norm γ_1 defined in Eq. (9.3) with this set K . Note that for $d = 1$, we have $K = \{(w, b) \in \mathbb{R}^2, w \in \{-1, 1\}, |b| \leq R\}$. We could stick to the ℓ_2 -sphere, but our particular choice of K leads to simpler formulas.

9.3.3 Variation norm in one dimension

The ReLU activation function is specific and leads to simple approximation properties in the interval $[-R, R]$. As already qualitatively described in Section 9.3.1, we start with piecewise affine functions, which, given the shape of the ReLU activation, should be easy to approximate (and immediately lead to universal approximation results as all “reasonable” functions can be approximated by piecewise affine functions). See more details by [Breiman \(1993\)](#); [Barron and Klusowski \(2018\)](#).

³When ν has a density $d\nu/d\tau$ with respect to a base measure τ , then the total variation is defined as the integral $\int_K |d\nu/d\tau(w, b)| d\tau(w, b)$ and is independent of the choice of τ . See https://en.wikipedia.org/wiki/Total_variation for more details.

Piecewise affine functions. We can make the reasoning in Section 9.3.1 quantitative. We consider a continuous piecewise affine function on $[-R, R]$ with specific knots at each $R = a_0 < a_1 < \dots < a_{m-2} < a_{m-1} = R$, so that on $[a_j, a_{j+1}]$, f is affine with slope v_j , for $j \in \{0, \dots, m-2\}$.



We can first start to fit the function $x \mapsto f(x) - f(-R)$ (which is equal to 0 at $x = R$) on $[a_0, a_1] = [-R, a_1]$, as $g_0(x) = v_0(x - a_0)_+$. For $x > a_0$, this approximation has slope v_0 . In order for the approximation to be exact on $[a_1, a_2]$ (while not modifying the function on $[a_0, a_1]$), we consider $g_1(x) = g_0(x) + (v_1 - v_0)(x - a_1)_+$, which is now exact on $[a_0, a_2]$; we can pursue recursively by considering, for $j \in \{1, \dots, m-2\}$

$$g_j(x) = g_{j-1}(x) + (v_j - v_{j-1})(x - a_j)_+,$$

which is equal to $f(x) - f(-R)$ for $x \in [a_0, a_{j+1}]$. We can thus represent $f(x) - f(-R)$ on $[a_0, a_{m-1}] = [0, R]$ exactly with $g_{m-2}(x)$. We have:

$$g_{m-2}(x) = v_0(x - a_0)_+ + \sum_{j=1}^{m-2} (v_j - v_{j-1})(x - a_j)_+.$$

In other words, we can represent any piecewise affine function as (using that on $[-R, R]$, $(x - a_0)_+ = (x + R)_+ = x + R$):

$$f(x) = f(-R) + v_0(x + R) + \sum_{j=1}^{m-2} (v_j - v_{j-1})(x - a_j)_+. \quad (9.4)$$

To obtain a representation that is invariant by a sign change, we also consider the same representation starting from the right (which can, for example, be obtained by applying the one above to $x \mapsto f(-x)$):

$$f(x) = f(R) + v_{m-2}(R - x) + \sum_{j=1}^{m-2} (v_{j-1} - v_j)(a_j - x)_+. \quad (9.5)$$

Note that this also shows that such representations are not unique. By averaging Eq. (9.4) and Eq. (9.5), and using that $\frac{1}{2R}(x+R)_+ + \frac{1}{2R}(-x+R)_+$ is equal to 1 on $[-R, R]$, we get:

$$\begin{aligned} f(x) &= \frac{1}{2}[f(R) + f(-R)] \left[\frac{1}{2R}(x+R)_+ + \frac{1}{2R}(-x+R)_+ \right] \\ &\quad + \frac{1}{2}v_0(x+R)_+ - \frac{1}{2}v_{m-2}(-x+R)_+ + \frac{1}{2} \sum_{j=1}^{m-2} (v_j - v_{j-1}) [(x - a_j)_+ + (a_j - x)_+], \end{aligned}$$

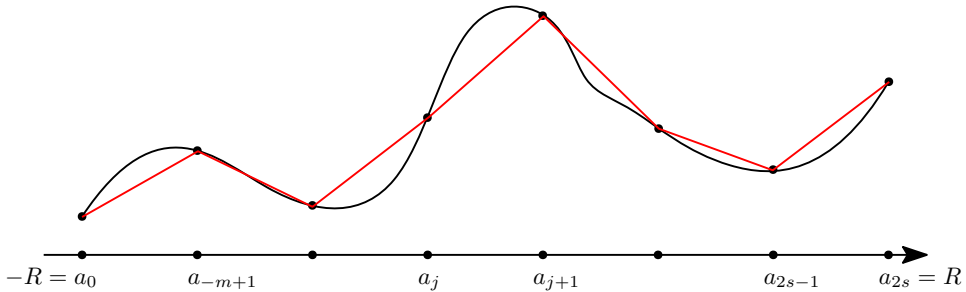
and thus, by construction of the norm γ_1 , we have

$$\gamma_1(f) \leq \frac{1}{2} \left| \frac{1}{2R}[f(-R) + f(R)] + v_0 \right| + \frac{1}{2} \left| \frac{1}{2R}[f(-R) + f(R)] - v_{m-2} \right| + \sum_{j=1}^{m-2} |v_j - v_{j-1}|.$$

The norm is thus upper-bounded by the values of f and its derivatives at the boundaries of the interval and the sums of the changes in slope.

Twice continuously differentiable functions. We consider a twice continuously differentiable function f on $[-R, R]$, and we would like to express it as a continuous linear combination of functions $x \mapsto (\pm x + b)_+$. We will consider two arguments: one through approximation by piecewise affine functions and one through the Taylor formula with integral remainder.

Piecewise-affine approximation. We consider equally-spaced knots $a_j = -R + \frac{j}{s}R$, for $j \in \{0, \dots, 2s\}$, and the piecewise affine interpolation from values $a_j, f(a_j)$, with $j \in \{0, \dots, 2s\}$, for s that will tend to infinity (see illustration below, where we have $m-1 = 2s$).



For the approximant \hat{f} , the value v_0 is equal to $s[f(-R + 1/s) - f(-R)] \sim f'(-R)$, and $v_{2s-1} = s[f(R) - f(R - 1/s)] \sim f'(R)$ when s tends to infinity, while the differences in slopes $|v_j - v_{j-1}|$ are equal to

$$\left| \frac{s}{R} (f(\frac{j+1}{s}R) - f(\frac{j}{s}R)) - \frac{s}{R} (f(\frac{j}{s}R) - f(\frac{j-1}{s}R)) \right| = \frac{s}{R} \left| f(\frac{j+1}{s}R) - 2f(\frac{j}{s}R) + f(\frac{j-1}{s}R) \right|,$$

which is equivalent to $\frac{R}{s} |f''(\frac{j}{s}R)|$ when $s \rightarrow +\infty$ (using a second-order Taylor expansion).

Thus, the approximant \hat{f} has γ_1 -norm bounded asymptotically as

$$\gamma_1(\hat{f}) \leq \frac{1}{2} \left| \frac{1}{2R} [f(-R) + f(R)] + f'(-R) \right| + \frac{1}{2} \left| \frac{1}{2R} [f(-R) + f(R)] - f'(R) \right| + \frac{R}{s} \sum_{j=1}^{2s-1} \left| f''\left(\frac{j}{s}R\right) \right|.$$

The last term $\frac{R}{s} \sum_{j=1}^{2s-1} \left| f''\left(\frac{j}{s}R\right) \right|$ tends to $\int_{-R}^R |f''(x)| dx$. Thus, letting s tend to infinity, we get (informally here, as the reasoning below will make it more formal):

$$\gamma_1(f) \leq \frac{1}{2} \left| \frac{1}{2R} [f(-R) + f(R)] + f'(-R) \right| + \frac{1}{2} \left| \frac{1}{2R} [f(-R) + f(R)] - f'(R) \right| + \int_{-R}^R |f''(x)| dx. \quad (9.6)$$

This notably shows that if the number of neurons is allowed to grow, then the ℓ_1 -norm of the weights remain bounded by the quantity above to represent the function f exactly.

Direct proof through Taylor formula. Eq. (9.6) above can be extended to continuous functions, which are only twice differentiable almost everywhere with integrable second-order derivatives. In this section, we assume that the function f is twice continuously differentiable and can extend to only integrable second-derivatives by a density argument (see, e.g., [Rudin, 1987](#)). For such a function, using the Taylor formula with integral remainder, we have, for $x \in [-R, R]$, using the fact that $(x - b)_+ = 0$ as soon as $b \geq x$:

$$\begin{aligned} f(x) &= f(-R) + f'(-R)(x + R) + \int_{-R}^x f''(b)(x - b) db \\ &= f(-R) + f'(-R)(x + R) + \int_{-R}^{\textcolor{red}{x}} f''(b)(x - b) \textcolor{red}{+} db. \end{aligned}$$

We also have the symmetric version (obtained by applying the one above to $x \mapsto f(-x)$, replacing x by $-x$, and by a change of variable $b \rightarrow -b$ in the integral):

$$f(x) = f(R) - f'(R)(R - x) + \int_{-R}^R f''(b)(-x - b)_+ db.$$

By averaging the two equalities, we get:

$$\begin{aligned} f(x) &= \frac{1}{2} \left[\frac{f(-R) + f(R)}{2R} + f'(-R) \right] (x + R) + \frac{1}{2} \left[\frac{f(-R) + f(R)}{2R} - f'(R) \right] (R - x) \\ &\quad + \frac{1}{2} \int_{-R}^R f''(b)(x - b)_+ db - \frac{1}{2} \int_{-R}^R f''(b)(-x - b)_+ db. \end{aligned}$$

This leads to the exact same upper-bound on $\gamma_1(f)$ as obtained from piecewise affine interpolation:

$$\gamma_1(f) \leq \frac{1}{2} \left| \frac{1}{2R} [f(-R) + f(R)] + f'(-R) \right| + \frac{1}{2} \left| \frac{1}{2R} [f(-R) + f(R)] - f'(R) \right| + \int_{-R}^R |f''(x)| dx. \quad (9.7)$$

One can check that the upper bound is indeed a norm (left as an exercise). Moreover, the upper bound happens to be tight (see the exercise below).

We will also use a simpler upper-bound, obtained from the triangle inequality:

$$\gamma_1(f) \leq \frac{1}{2R} |f(-R) + f(R)| + \frac{1}{2} |f'(R) + f'(-R)| + \int_{-R}^R |f''(x)| dx. \quad (9.8)$$

Exercise 9.3 (♦) *Show that the upper-bound in Eq. (9.7) is in fact an equality.*

Exercise 9.4 (♦) *Show that the minimum norm interpolant from ordered observations $-R < x_1 < \dots < x_n < R$, $y_1, \dots, y_n \in \mathbb{R}$, is equal to the piecewise-affine interpolant on $[x_1, x_n]$.*

9.3.4 Variation norm in arbitrary dimension

If we assume that f is continuous on the ball of center zero and radius R , then the Fourier transform $\hat{f}(\omega) = \int_{\mathbb{R}^d} f(x) e^{-i\omega^\top x} dx$ is defined everywhere, and we can write f as the inverse Fourier transform of \hat{f} , that is,

$$f(x) = \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} \hat{f}(\omega) e^{i\omega^\top x} d\omega. \quad (9.9)$$

To compute an upper-bound on $\gamma_1(f)$, it suffices to upper-bound for each $\omega \in \mathbb{R}^d$, $\gamma_1(x \mapsto e^{i\omega^\top x})$ (using complex-valued functions, for which the developments of the previous section still apply, or using sines and cosines), which is possible because we have the representation from Section 9.3.3 and Eq. (9.8) applied to $g : u \mapsto e^{iu\|\omega\|_2}$, for $u \in [-R, R]$,

$$e^{iu\|\omega\|_2} = \int_{-R}^R \eta_+(b, \|\omega\|_2) (u - b)_+ db + \int_{-R}^R \eta_-(b, \|\omega\|_2) (-u - b)_+ db,$$

$$\text{with } \int_{-R}^R |\eta_+(b, \|\omega\|_2)| db + \int_{-R}^R |\eta_-(b, \|\omega\|_2)| db \leq \frac{1}{R} + \|\omega\|_2 + 2R\|\omega\|_2^2 \leq \frac{2}{R}(1 + 2R^2\|\omega\|_2^2).$$

We can, therefore, decompose the function defined on the ball of center 0 and radius R :

$$\begin{aligned} e^{i\omega^\top x} &= e^{i(x^\top \omega / \|\omega\|_2) \|\omega\|_2} \\ &= \int_{-R}^R \eta_+(b, \|\omega\|_2) (x^\top (\omega / \|\omega\|_2) - b)_+ db + \int_{-R}^R \eta_-(b, \|\omega\|_2) (x^\top (-\omega / \|\omega\|_2) - b)_+ db, \end{aligned}$$

with weights being in the correct constraint set (unit norm for the slopes $\omega / \|\omega\|_2$ and constant terms $|b| \leq R$), leading to

$$\gamma_1(x \mapsto e^{i\omega^\top x}) \leq \frac{2}{R}(1 + 2R^2\|\omega\|_2^2).$$

Thus, we obtain, from Eq. (9.9) and the triangular inequality for the norm γ_1 :

$$\gamma_1(f) \leq \frac{1}{(2\pi)^d} \int_{\mathbb{R}^d} |\hat{f}(\omega)| \gamma_1(x \mapsto e^{i\omega^\top x}) d\omega \leq \frac{1}{(2\pi)^d} \frac{2}{R} \int_{\mathbb{R}^d} |\hat{f}(\omega)| (1 + 2R^2\|\omega\|_2^2) d\omega. \quad (9.10)$$

Given a function $g : \mathbb{R}^d \rightarrow \mathbb{R}$, $\int_{\mathbb{R}^d} |\hat{g}(\omega)| d\omega$ is a measure of smoothness of g , and so $\gamma_1(f)$ being finite imposes that f and all second-order derivatives of f have this form of smoothness. The right-hand side of Eq. (9.10) is often referred to as the “Barron norm”, named after Barron (1993, 1994). See Klusowski and Barron (2018) for more details.

To relate the norm γ_1 to other function spaces such as Sobolev spaces, we will consider further upper bounds (and relate them to another norm γ_2 in Section 9.5).

9.3.5 Precise approximation properties

Precise rates of approximation. In this section, we will relate the space \mathcal{F}_1 to Sobolev spaces, bounding, using Cauchy-Schwarz inequality, the norm γ_1 as:

$$\begin{aligned} \gamma_1(f) &\leq \frac{1}{(2\pi)^d} \frac{2}{R} \int_{\mathbb{R}^d} |\hat{f}(\omega)| (1 + 2R^2 \|\omega\|_2^2) d\omega \quad \text{from Eq. (9.10),} \\ &= \frac{1}{(2\pi)^d} \frac{2}{R} \int_{\mathbb{R}^d} |\hat{f}(\omega)| (1 + 2R^2 \|\omega\|_2^2)^{d/4+5/4} \frac{d\omega}{(1 + 2R^2 \|\omega\|_2^2)^{d/4+1/4}} \\ &\leq \frac{1}{(2\pi)^d} \frac{2}{R} \sqrt{\int_{\mathbb{R}^d} |\hat{f}(\omega)|^2 (1 + 2R^2 \|\omega\|_2^2)^{d/2+5/2} d\omega} \sqrt{\int_{\mathbb{R}^d} \frac{d\omega}{(1 + 2R^2 \|\omega\|_2^2)^{d/2+1/2}}}, \quad (9.11) \end{aligned}$$

which is a constant times $\sqrt{\int_{\mathbb{R}^d} |\hat{f}(\omega)|^2 (1 + 2R^2 \|\omega\|_2^2)^s d\omega}$, which is exactly the Sobolev norm from Chapter 7, with $s = \frac{d}{2} + \frac{5}{2}$ derivatives, which is a reproducing kernel Hilbert space (RKHS) since $s > d/2$.

Thus, all approximation properties from Chapter 7 apply (see there for precise rates, and their application to generalization bounds in Section 9.4). Note, however, that, *using this reasoning*, if we start from a Lipschitz-continuous function, then to approximate it up to $L_2(\mathbb{R}^d)$ -norm ε requires a γ_1 -norm growing as $\varepsilon^{-(s-1)} \geq \varepsilon^{-(d/2+3/2)}$ (as obtained at the end of Section 7.5.2 of Chapter 7). Thus, in the generic situation where no particular directions are preferred, using \mathcal{F}_1 (neural networks) is not really more advantageous than using kernel methods (see also more details in Section 9.4 and Section 9.5). This changes drastically when such linear structures are present, as shown below.

Adaptivity to linear latent variables. We consider a target function f^* that depends only on a r -dimensional projection of the data, that is, f^* is of the form $f^*(x) = g(V^\top x)$, where $V \in \mathbb{R}^{d \times r}$ is full rank and has all singular values less than 1, and $g : \mathbb{R}^r \rightarrow \mathbb{R}$. Without loss of generality, we can assume that V has orthonormal columns. Then if $\gamma_1(g)$ is finite (for the function g defined on \mathbb{R}^r), it can be written as

$$g(z) = \int_{\mathbb{R}^{r+1}} (w^\top z + b)_+ d\mu(w, b),$$

with μ supported on $\{(w, b) \in \mathbb{R}^{r+1}, \|w\|_2 \leq 1, |b| \leq R\}$, and $\gamma_1(g) = \int_{\mathbb{R}^{r+1}} |d\mu(w, b)|$. We can then use this representation of g to obtain a representation of f^* as:

$$f^*(x) = g(V^\top x) = \int_{\mathbb{R}^{r+1}} ((Vw)^\top x + b)_+ d\mu(w, b).$$

Since $\|Vw\|_2 \leq 1$ as soon as $\|w\|_2 \leq 1$, and V have orthonormal columns, the measure μ on (w, b) defines a measure for (Vw, b) on $\{(w', b) \in \mathbb{R}^{d+1}, \|w'\|_2 \leq 1, |b| \leq R\}$ with a total variation which is less than the one of μ . Thus $\gamma_1(f^*) \leq \int_{\mathbb{R}^{r+1}} |d\mu(w, b)| = \gamma_1(g)$. In other words, the approximation properties of g translate to f^* , and thus, we pay only the price of these r dimensions and not all d variables, *without* the need to know V in advance. For example, (a) if g has more than $r/2 + 5/2$ squared integrable derivatives, then $\gamma_1(g)$ and thus $\gamma_1(f^*)$ is finite, or (b) if g is Lipschitz-continuous, then both g and f can be approached in $L_2(\mathbb{R}^d)$ with error ε with a function with γ_1 -norm of order $\varepsilon^{-(r/2+5/2)}$, thus escaping the curse of dimensionality. See [Bach \(2017\)](#) for more details and precise learning rates in Section 9.4.



Kernel methods do not have such adaptivity. In other words, as shown in Section 9.5, using the ℓ_2 -norm instead of the ℓ_1 -norm on the output weights leads to worse performance.

We will combine these approximation results with the estimation error results in Section 9.4.

9.3.6 From the variation norm to a finite number of neurons (♦)

Given a measure μ on \mathbb{R}^d , and a function $g : \mathbb{R}^d \rightarrow \mathbb{R}$ such that $\gamma_1(g)$ is finite, we would like to find a set of m neurons $(w_j, b_j) \in K \subset \mathbb{R}^{d+1}$ (which is the compact support of all measures that we consider), such that the associated function defined through

$$f(x) = \sum_{j=1}^m \eta_j \sigma(w_j^\top x + b_j)$$

is close to g for the norm $L_2(\mu)$.

Since input weights are fixed in K , the bound on $\gamma_1(g)$ should translate to a bound on the ℓ_1 -norm of η : $\|\eta\|_1 \leq \gamma_1(g)$. The set of functions f such that $\gamma_1(f) \leq \gamma_1(g)$ is the convex hull of functions $s\gamma_1(g)\sigma(w^\top x + b)$, for $s \in \{-1, 1\}$, and $\|w\|_2 = 1, |b| \leq R$. Thus, we are faced with the problem of approximating elements of a convex hull as an explicit linear combination of extreme points, if possible, with as few extreme points as possible.

In finite dimension, Carathéodory's theorem says that the number of such extreme points can be taken equal to the dimension to get an exact representation. In our case of infinite dimensions, we need an approximate version of Carathéodory's theorem. It turns out that we can create a “fake” optimization problem of minimizing the squared L_2 -norm (for the input data distribution p) $\|f - g\|_{L_2(p)}^2$ such that $\gamma_1(f) \leq \gamma_1(g)$, whose solution is $f = g$, with an algorithm that constructs an approximate solution from extreme points. This will be achieved by the Frank-Wolfe algorithm (a.k.a. conditional gradient algorithm). This algorithm is applicable more generally; for more details, see [Jaggi \(2013\)](#); [Bach \(2015\)](#).

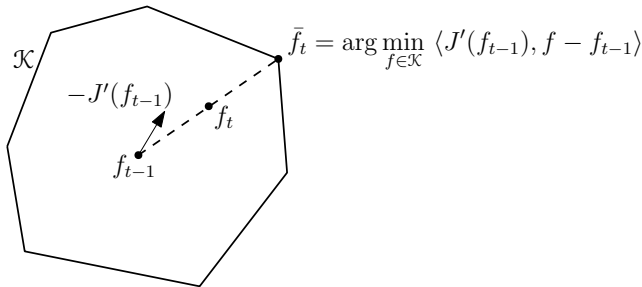
Frank-Wolfe algorithm. We thus make a detour by considering an algorithm defined in a Hilbert space \mathcal{H} , such that \mathcal{K} is a bounded convex set and J a convex, smooth function from \mathcal{H} to \mathbb{R} , that is such that there exists a gradient function $J' : \mathcal{H} \rightarrow \mathcal{H}$ such that for all elements f, g of \mathcal{H} (which is the traditional smoothness condition from Section 5.2.3):

$$J(g) + \langle J'(g), f - g \rangle_{\mathcal{H}} \leq J(f) \leq J(g) + \langle J'(g), f - g \rangle_{\mathcal{H}} + \frac{L}{2} \|f - g\|_{\mathcal{H}}^2.$$

The goal is to minimize J on the bounded convex set \mathcal{K} , with an algorithm that only requires access to the set \mathcal{K} through a “linear minimization” oracle (i.e., through minimizing linear functions), as opposed to the projection oracle that we required in Section 5.2.5.

We consider the following recursive algorithm, started from a vector $f_0 \in \mathcal{K}$:

$$\begin{aligned} \bar{f}_t &\in \arg \min_{f \in \mathcal{K}} \langle J'(f_{t-1}), f - f_{t-1} \rangle_{\mathcal{H}}, \\ f_t &= \frac{t-1}{t+1} f_{t-1} + \frac{2}{t+1} \bar{f}_t = f_{t-1} + \frac{2}{t+1} (\bar{f}_t - f_{t-1}). \end{aligned} \quad (9.12)$$



Because \bar{f}_t is obtained by minimizing a linear function on a bounded convex set, we can restrict the minimizer \bar{f}_t to be extreme points of \mathcal{K} , so that, f_t is the convex combination of t such extreme points $\bar{f}_1, \dots, \bar{f}_t$ (note that the first point f_0 disappears from the convex combination). We now show that

$$J(f_t) - \inf_{f \in \mathcal{K}} J(f) \leq \frac{2L}{t+1} \text{diam}_{\mathcal{H}}(\mathcal{K})^2.$$

Proof of convergence rate (♦). This is obtained by using smoothness:

$$\begin{aligned} J(f_t) &\leq J(f_{t-1}) + \langle J'(f_{t-1}), f_t - f_{t-1} \rangle_{\mathcal{H}} + \frac{L}{2} \|f_t - f_{t-1}\|_{\mathcal{H}}^2 \\ &= J(f_{t-1}) + \frac{2}{t+1} \langle J'(f_{t-1}), \bar{f}_t - f_{t-1} \rangle_{\mathcal{H}} + \frac{4}{(t+1)^2} \frac{L}{2} \|\bar{f}_t - f_{t-1}\|_{\mathcal{H}}^2 \\ &\leq J(f_{t-1}) + \frac{2}{t+1} \min_{f \in \mathcal{K}} \langle J'(f_{t-1}), f - f_{t-1} \rangle_{\mathcal{H}} + \frac{4}{(t+1)^2} \frac{L}{2} \text{diam}_{\mathcal{H}}(\mathcal{K})^2. \end{aligned}$$

By convexity of J , we have for all $f \in \mathcal{K}$, $J(f) \geq J(f_{t-1}) + \langle J'(f_{t-1}), f - f_{t-1} \rangle_{\mathcal{H}}$, leading to $\inf_{f \in \mathcal{K}} J(f) \geq J(f_{t-1}) + \inf_{f \in \mathcal{K}} \langle J'(f_{t-1}), f - f_{t-1} \rangle_{\mathcal{H}}$. Thus, we get

$$J(f_t) - \inf_{f \in \mathcal{K}} J(f) \leq [J(f_{t-1}) - \inf_{f \in \mathcal{K}} J(f)] \frac{t-1}{t+1} + \frac{4}{(t+1)^2} \frac{L}{2} \text{diam}_{\mathcal{H}}(\mathcal{K})^2$$

leading to

$$\begin{aligned} t(t+1)[J(f_t) - \inf_{f \in \mathcal{K}} J(f)] &\leq (t-1)t[J(f_{t-1}) - \inf_{f \in \mathcal{K}} J(f)] + 2L \text{diam}_{\mathcal{H}}(\mathcal{K})^2 \\ &\leq 2Lt \text{diam}_{\mathcal{H}}(\mathcal{K})^2 \text{ by using a telescoping sum,} \end{aligned}$$

and thus $J(f_t) - \inf_{f \in \mathcal{K}} J(f) \leq \frac{2L}{t+1} \text{diam}_{\mathcal{H}}(\mathcal{K})^2$, as claimed earlier.

Exercise 9.5 Show that if we replace Eq. (9.12) by $f_t = \frac{t-1}{t} f_{t-1} + \frac{1}{t} \bar{f}_t$, f_t is the uniform convex combination of $\bar{f}_1, \dots, \bar{f}_t$, and that, $J(f_t) - \inf_{f \in \mathcal{K}} J(f) \leq \frac{2L \log(t+1)}{t+1} \text{diam}_{\mathcal{H}}(\mathcal{K})^2$.

Application to approximate representations with a finite number of neurons.

We can apply this to $\mathcal{H} = L_2(\mathbb{R}^d)$ and $J(f) = \|f - g\|_{L_2(p)}^2$, leading to $L = 2$, with $\mathcal{K} = \{f \in L_2(\mathbb{R}^d), \gamma_1(f) \leq \gamma_1(g)\}$ for which the set of extreme points are exactly single neurons $\sigma(w^\top \cdot + b)$ scaled by $\gamma_1(g)$ and with an extra sign $s \in \{-1, 1\}$.

We thus obtain after t steps a representation of f with t neurons for which

$$\|f - g\|_{L_2(p)}^2 \leq \frac{4\gamma_1(g)^2}{t+1} \sup_{(w,b) \in \mathcal{K}} \|\sigma(w^\top \cdot + b)\|_{L_2(p)}^2.$$

Thus, it is sufficient to have t of order $O(\gamma_1(g)^2/\varepsilon^2)$ to achieve $\|f - g\|_{L_2(\mu)} \leq \varepsilon$. Therefore, the norm $\gamma_1(g)$ directly controls the approximability of the function g by a finite number of neurons and tells us how many neurons should be used for a given target function. For the ReLU activation, the bound above becomes: $\|f - g\|_{L_2(p)}^2 \leq \frac{16R^2\gamma_1(g)^2}{t+1}$; note that the dependence of the number of neurons in ε as ε^{-2} is not optimal, as it can be improved to $\varepsilon^{-2d/(d+3)}$ (see [Bach, 2017](#), and references therein).

Application to neural network fitting. The Frank-Wolfe algorithm can be used to fit a neural network from data by minimizing the empirical risk of a function f , which is constrained to have a norm γ_1 bounded by a fixed constant D . After t iterations, the general convergence result above leads to an approximate minimizer with an explicit provable convergence guarantee in $O(1/t)$.

However, as above, the corresponding set of extreme points are single neurons of the form $\sigma(w^\top \cdot + b)$ scaled by D and with an extra sign $s \in \{-1, 1\}$. Therefore, to implement the linear minimization oracle, given the derivative α_i of the loss function associated with the i -th observation, for $i = 1, \dots, n$, we need to minimize with respect to s, w, b the quantity $\sum_{i=1}^n s \alpha_i \sigma(w^\top x_i + b)$, for input observations $x_i \in \mathbb{R}^d$, $i = 1, \dots, n$,

for which there is no known polynomial-time algorithms. Thus, we do not obtain through the Frank-Wolfe algorithm a polynomial-time algorithm (see more details by [Bach, 2017](#)).

This incremental approach to estimating a neural network is related to boosting procedures that we present in [Section 10.3](#).

9.4 Generalization performance for neural networks

We can now consider putting both estimation and approximation errors together, using tools from [Section 7.5.1](#), that give a rate for constrained optimization (this is done for simplicity, as using tools from [Section 4.5.5](#), we could get similar results for penalized problems).

We thus minimized the empirical risk for a G -Lipschitz-continuous loss subject to $\gamma_1(f) \leq D$. From [Section 9.2.3](#), we get an estimation error less than $\frac{16GDR}{\sqrt{n}}$, on which we need to add $G \inf_{\gamma_1(f) \leq D} \|f - f^*\|_{L_2(p)}$, where f^* is the target function, minimizer of the expected risk. Following the same reasoning as in [Section 7.5.1](#), optimizing over D leads to an upper-bound of the form (where the constant is 256 rather than 16 in [Eq. \(7.8\)](#) because the extra factor of 4 in the estimation error):

$$\varepsilon_n = 2G \sqrt{\inf_{f \in \mathcal{F}_1} \left\{ \|f - f^*\|_{L_2(p)}^2 + \frac{256R^2}{n} \gamma_1(f)^2 \right\}}. \quad (9.13)$$

As shown in [Section 7.5](#), given this bound, we can recover the bound D as $\frac{\sqrt{n}}{16RG} \varepsilon_n$, and thus, using [Section 9.3.6](#) which shows how to approximate a function in \mathcal{F}_1 by finitely many neurons, we will lose an additional factor ε_n with a number of neurons greater than $m \geq \frac{16D^2 R^2 G^2}{\varepsilon_n^2}$ which is exactly equal to a constant times n , that is, with this analysis, there is no need to have a number of neurons that greatly exceeds the number of observations.

We can now look at a series of structural assumptions on the target function f^* , for which we will see that neural networks provide adaptivity if the regularization parameter is well-chosen:

- **No assumption:** If we assume that f_* is Lipschitz-continuous on the ball of center 0 and radius R , then, as shown at the end of [Section 7.5.2](#), f_* can be extended to a function in the Sobolev space of order 1. Using the comparison of γ_1 with the Sobolev norm of order $s = \frac{d}{2} + \frac{5}{2}$ in [Eq. \(9.11\)](#), we can reuse the results from kernel methods in [Section 7.5.2](#), and obtain a rate of $O(1/n^{1/(2s)}) = 1/n^{1/(d+5)}$, which exhibits the curse of dimensionality, which cannot anyway be much improved, as the optimal performance has to be larger than $1/n^{1/(d+2)}$ (see [Chapter 15](#)).
- **Linear latent-variable:** If we now assume that f_* depends on an r -dimensional *unknown* subspace, then we can reuse the same reasoning on the projected subspace, compare the norm γ_1 projected to the subspace (like done in [Section 9.3.5](#)) to the Sobolev norm on the same projected subspace, thus of order $s = r/2 + 5/2$ (instead of $d/2 + 5/2$). This leads to an estimation rate for the excess risk proportional

$1/n^{1/(r+5)}$ (with constants independent from d). This is where neural networks have a strong advantage over kernel methods and sparse methods: they can perform variable selection with non-linear predictions.

- “Teacher network”: if we assume that f^* is the linear combination of k hidden neurons, then we obtain a convergence rate proportional to k/\sqrt{n} , as the norm $\gamma_1(f^*)$ is proportional to k .

Exercise 9.6 Consider target functions of the form $f^*(x) = \sum_{j=1}^k f_j(w_j^\top x)$ for one-dimensional Lipschitz-continuous functions. Provide an upper bound on excess risk proportional to $k/n^{1/6}$.

Note that these rates are not as good as [Bach \(2017\)](#), since the exponent $s = \frac{d}{2} + \frac{5}{2}$ is not optimal, and in fact, a more careful analysis, as outlined in Section 9.5 would lead $s = \frac{d}{2} + \frac{3}{2}$, with a similar dependence on dimension.

Non-linear variable selection (♦). In this chapter, we focus primarily on ℓ_2 -norm constraints or penalty on the weight vectors $w_1, \dots, w_m \in \mathbb{R}^d$ of a neural network, but all developments can be carried out with the ℓ_1 -norm, leading to the high-dimensional behavior detailed in Section 8.3.3, but this time selecting variable with a *non-linear* prediction on top of them. For the rest of this section, we assume that $\|x\|_\infty \leq R$ almost surely.

The analysis has to be adapted for both the estimation error and the approximation error. For the estimation error, in the derivations of Section 9.2.3, we simply need to replace Eq. (9.2) by

$$\begin{aligned} R_n(\mathcal{G}) &\leq 2GD \left(\mathbb{E} \left[\left\| \frac{1}{n} \sum_{i=1}^n \varepsilon_i x_i \right\|_\infty^2 + R^2 \left(\frac{1}{n} \sum_{i=1}^n \varepsilon_i \right)^2 \right] \right)^{1/2} \\ &\leq 2GD \left(\mathbb{E} \left[\left\| \frac{1}{n} \sum_{i=1}^n \varepsilon_i x_i \right\|_\infty^2 \right] \right)^{1/2} + 2GDR \left(\mathbb{E} \left[\left(\frac{1}{n} \sum_{i=1}^n \varepsilon_i \right)^2 \right] \right)^{1/2} \\ &\leq 2GDR \frac{\sqrt{2 \log(2d)}}{\sqrt{n}} + \frac{2GDR}{\sqrt{n}} \leq 4GRD \sqrt{\frac{\log(4d)}{n}}, \end{aligned} \tag{9.14}$$

using expectations of maxima from Section 1.2.4.

Thus, in estimation rates, we need to consider instead of Eq. (9.13)

$$\varepsilon_n = 2G \sqrt{\inf_{f \in \mathcal{F}_1} \left\{ \|f - f^*\|_{L_2(p)}^2 + \frac{256 R^2 \log(4d)}{n} \gamma_1(f)^2 \right\}}$$

(note the extra factor $\log(4d)$ and the definition of R as an ℓ_∞ -bound). Regarding approximation error, we simply use the bound $\|w\|_1 \leq \sqrt{k} \|w\|_2$ if w has only k non-zero elements. Thus, if the target function f^* is a Lipschitz-continuous of only k (unknown) variables, we can use the approximation result for ℓ_2 -norm constraints, with an extra

dependence on k (which we already had). Thus, overall, the estimation rate of the excess risk is proportional to a constant depending on k , times $(\frac{\log(4d)}{n})^{1/(k+3)}$, thus with a high-dimensional estimation rate, where d only appears logarithmically.

9.5 Relationship with kernel methods (◆)

In this section, we relate our function space \mathcal{F}_1 to a simpler function space \mathcal{F}_2 that will, in the overparameterized regime when m tends to $+\infty$, correspond to only optimizing the output layer.

9.5.1 From a Banach space \mathcal{F}_1 to a Hilbert space \mathcal{F}_2 (◆)

Following the notations of Section 9.3.2, given a fixed probability measure τ on $K \subset \mathbb{R}^{d+1}$, we can define another norm as

$$\gamma_2^2(f) = \inf_{\nu \in \mathcal{M}(K)} \int_K \left| \frac{d\nu(w, b)}{d\tau(w, b)} \right|^2 d\tau(w, b) \text{ such that } \forall x \in \mathcal{X}, f(x) = \int_K \sigma(w^\top x + b) d\nu(w, b). \quad (9.15)$$

By construction (and by Jensen's inequality), $\gamma_1(f) \leq \gamma_2(f)$, so the space \mathcal{F}_2 of functions f such that $\gamma_2(f) < +\infty$ is included in \mathcal{F}_1 (moreover, γ_2 depends on the choice of the base measure τ , while γ_1 does not).

Moreover, as shown in the proposition below, the space \mathcal{F}_2 is a reproducing kernel Hilbert space on $\mathcal{X} = \{x \in \mathbb{R}^d, \|x\|_2 \leq R\}$, as defined in Chapter 7.

Proposition 9.2 *The space \mathcal{F}_2 is the reproducing kernel Hilbert space associated with the positive definite kernel function*

$$k(x, x') = \int_K \sigma(w^\top x + b) \sigma(w^\top x' + b) d\tau(w, b). \quad (9.16)$$

Proof For a formal proof for all compact sets K , see Bach (2017, Appendix A). We only provide a proof for finite K and τ the uniform probability measure on K , we then have, $\gamma_2^2(f) = \inf_{\nu \in \mathbb{R}^K} \frac{1}{|K|} \sum_{(w, b) \in K} \nu_k^2$ such that $f(x) = \frac{1}{|K|} \sum_{(w, b) \in K} \nu_k \sigma(w^\top x + b)$, which corresponds to penalizing the ℓ_2 -norm of $\theta = \frac{1}{\sqrt{|K|}} \nu \in \mathbb{R}^K$ for $f(x) = \theta^\top \varphi(x)$, and $\varphi(x)_{(w, b)} = \frac{1}{|K|^{1/2}} \sigma(w^\top x + b)$. We thus exactly get the desired kernel $k(x, x') = \frac{1}{|K|} \sum_{(w, b) \in K} \sigma(w^\top x + b) \sigma(w^\top x' + b)$. ■

Interpretation in terms of random features. As already mentioned in Section 7.4, the kernel defined in Eq. (9.16) can be approximated by sampling uniformly at random from τ , m points (w_j, b_j) , $j = 1, \dots, m$, and approximating $k(x, x')$ by

$$\hat{k}(x, x') = \frac{1}{m} \sum_{j=1}^m \sigma(w_j^\top x + b_j) \sigma(w_j^\top x' + b_j).$$

This corresponds to using $f(x) = \sum_{j=1}^m \eta_j \sigma(w_j^\top x + b_j)$, with a penalty proportional to $m\|\eta\|_2^2$. Thus random features correspond to only optimizing with respect to the output weights while keeping the input weights fixed (while for γ_1 , we optimize over all weights).

Therefore, infinite width networks where input weights are random and only output weights are learned are, in fact, kernel methods in disguise (Neal, 1995; Rahimi and Recht, 2008).

This kernel can be computed in closed form for simple activations and distributions of weights; see Section 9.5.2 and Cho and Saul (2009); Bach (2017). Thus, the same regularization properties may be achieved with algorithms from Chapter 7 (which are based on convex optimization and therefore come with guarantees). Note that, as shown in Section 7.4, a common strategy for kernels defined as expectations is to use the *random feature* approximation $\hat{k}(x, x')$, that is, here, use the neural network representation explicitly.



The kernel approximation corresponds to input weights w_j, b_j sampled randomly and *held fixed*. Only the output weights η_j are optimized.



Because Dirac measures are not square integrable, the prediction function $x \mapsto \sigma(w^\top x + b)$, that is, a single neuron, is typically not in the RKHS, which is typically composed of smooth functions. See the examples below.


Link between the two norms. To relate the two norms more precisely, we rewrite γ_1 using the fixed probability measure τ , as

$$\gamma_1(f) = \inf_{\eta: K \rightarrow \mathbb{R}} \int_K |\eta(w, b)| d\tau(w, b) \text{ such that } \forall x \in \mathcal{X}, f(x) = \int_K \sigma(w^\top x + b) \eta(w, b) d\tau(w, b).$$

The only difference with the squared RKHS norm above is that we consider the L_1 -norm instead of the squared L_2 -norm of η (with respect to the probability measure τ). The minimum achievable norm is exactly $\gamma_1(f)$.

Note that typically, the infimum over all η is not achieved as the optimal measure in Eq. (9.3) may not have a density with respect to τ . Because we use an L_1 -norm penalty, the measures $\mu(w, b) = \eta(w, b)\tau(w, b)$ can span in the limit all measures $\mu(w, b)$ with finite total variation $\int_{\mathbb{R}^{d+1}} |d\mu(\eta, b)| = \int_{\mathbb{R}^{d+1}} |\eta(w, b)| d\tau(w, b)$.

Overall, we have the following properties (see Table 9.1 for a summary):

- Because of Jensen's inequality, we have $\gamma_1(f) \leq \gamma_2(f)$, and thus $\mathcal{F}_2 \subset \mathcal{F}_1$, that is the space \mathcal{F}_1 contains many more functions.
-  A single neuron is in \mathcal{F}_1 with γ_1 -norm less than one, as the mass of a Dirac is equal to one.

| \mathcal{F}_2 | \mathcal{F}_1 |
|---|---|
| Hilbert space | Banach space |
| $\gamma_2(f)^2 = \inf \int_{\mathbb{R}^{d+1}} \eta(w, b) ^2 d\tau(w, b)$ s. t. $f(x) = \int_{\mathbb{R}^{d+1}} \eta(w, b) \sigma(w^\top x + b) d\tau(w, b)$ | $\gamma_1(f) = \inf \int_{\mathbb{R}^{d+1}} \eta(w, b) d\tau(w, b)$ s. t. $f(x) = \int_{\mathbb{R}^{d+1}} \eta(w, b) \sigma(w^\top x + b) d\tau(w, b)$ |
| Smooth functions Single neurons $\notin \mathcal{F}_2$ | Potentially non-smooth functions Single neurons $\in \mathcal{F}_1$ |

Table 9.1: Summary of properties of the norms γ_1 and γ_2 .

9.5.2 Kernel function (♦♦)

We can compute in closed form the kernel function, which is only useful computationally if the number of random features m is larger than the number of observations (when using the kernel trick is advantageous, as outlined in Section 7.4).

In one dimension, with w uniform on the unit sphere, that is, $w \in \{-1, 1\}$, and with b uniform on $[-R, R]$, we have the following kernel

$$k(x, x') = \frac{1}{4R} \int_{-R}^R \left((x-b)_+(x'-b)_+ + (-x-b)_+(-x'-b)_+ \right) db.$$

After a short calculation left as an exercise (see also [Bach, 2023b](#)), we can compute it in closed form as:

$$k(x, x') = \frac{R^2}{6} + \frac{xx'}{2} + \frac{1}{24R} |x - x'|^3.$$

In higher dimension, we have:

$$k(x, x') = \int_{\|w\|_2=1} \frac{1}{2R} \int_{-R}^R (w^\top x + b)_+(w^\top x' + b)_+ db d\tau(w),$$

where τ is the uniform distribution on the sphere. After a longer calculation, also left as an exercise (see [Bach, 2023b](#)), we get:

$$k(x, x') = \frac{R^2}{6} + \frac{1}{2d} x^\top x' + \frac{1}{24R} \frac{\Gamma(2)\Gamma(\frac{d}{2})}{\Gamma(\frac{1}{2})\Gamma(\frac{d}{2} + \frac{3}{2})} \|x - x'\|_2^3. \quad (9.17)$$

See Figure 9.2 for comparing the RKHS (corresponding to $m = +\infty$ neurons) and the approximation with finite m .

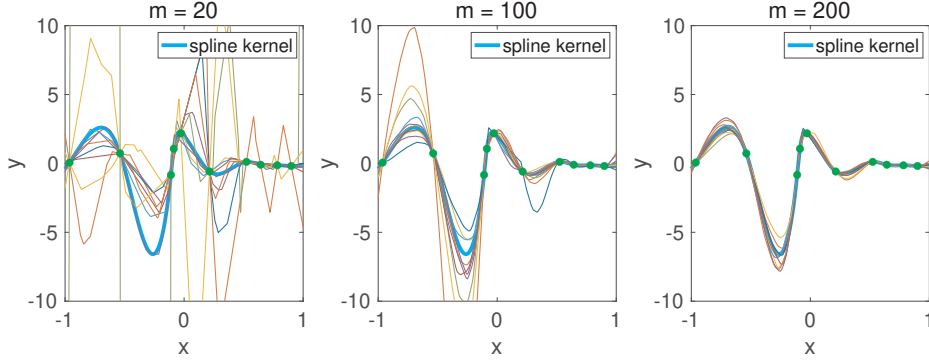


Figure 9.2: Examples of functions in the reproducing kernel Hilbert space \mathcal{F}_2 and its approximation based on random features, with $m = 20, 100$, and 200 . All functions are the minimum norm interpolators of the green points. This is to be contrasted with the Banach space \mathcal{F}_1 , where the minimum norm interpolator is the piecewise affine interpolator (see Exercise 9.4), and can be achieved with $m = n$ neurons, where n is the number of observed points.

9.5.3 Upper-bound on RKHS norm (◆◆)

We can now find upper bounds on the norm γ_2 . We can either use the kernel function from Eq. (9.17) or the random feature interpretation from Eq. (9.15). We first use the random feature interpretation in one dimension.

Upper-bound on RKHS norm γ_2 in one dimension. Using the same reasoning as the end of Section 9.5, we can get an upper-bound on $\gamma_2(f)$ by decomposing f as

$$f(x) = \int_{-R}^R \eta_+(b)(x-b)_+ \frac{db}{4R} + \int_{-R}^R \eta_-(b)(-x-b)_+ \frac{db}{4R},$$

$$\text{with now } \gamma_2(f)^2 \leq \int_{-R}^R \eta_+(b)^2 \frac{db}{4R} + \int_{-R}^R \eta_-(b)^2 \frac{db}{4R}.$$

By using as in Section 9.3.3 the Taylor expansion with integral remainder, we get, for any twice differentiable function f on $[-R, R]$:

$$\begin{aligned} f(x) &= \frac{1}{2}f(-R) + \frac{1}{2}f(R) + \frac{1}{2}f'(-R)(x+R) - \frac{1}{2}f'(R)(-x+R) \\ &\quad + \frac{1}{2} \int_{-R}^R f''(b)(x-b)_+ db - \frac{1}{2} \int_{-R}^R f''(b)(-x-b)_+ db \\ &= \frac{1}{2}[f'(R) + f'(-R)] + \frac{1}{2}[R(f'(-R) - f'(R)) + f(-R) + f(R)] \\ &\quad + \frac{1}{2} \int_{-R}^R f''(b)(x-b)_+ db - \frac{1}{2} \int_{-R}^R f''(b)(-x-b)_+ db. \end{aligned}$$

We can now use explicit representations of constants and linear functions, *without* Diracs, as we need finite L_2 -norms, as:

$$\begin{aligned} x &= \int_{-R}^R \frac{(x-b)_+ - (-x-b)_+}{2R} db = \int_{-R}^R \frac{x}{2R} db \\ -\frac{R^2}{6} &= \int_{-R}^R b(x-b)_+ \frac{db}{4R} + \int_{-R}^R b(-x-b)_+ \frac{db}{4R}. \end{aligned}$$

After a short calculation left as an exercise, this leads to

$$\gamma_2(f)^2 \leq 2R \int_{-R}^R f''(x)^2 dx + [f'(R) + f'(-R)]^2 + 3[R(f'(R) - f'(-R)) - f(-R) - f(R)]^2, \quad (9.18)$$

which happens to be an equality (which can be shown by showing that this defines a dot-product, for which $\langle f, k(\cdot, x) \rangle = f(x)$, see [Bach, 2023b](#)).

Exercise 9.7 Show that the upper bound on γ_2 from Eq. (9.18) is larger than the bound on γ_1 from Eq. (9.7).

The main difference with γ_1 is that the second-derivative is penalized by an L_2 -norm and not by an L_1 -norm, and that this L_2 -norm can be infinite when the L_1 -norm is finite, the classic example being for the hidden neuron functions $(x-b)_+$.

⚠ The RKHS is combining infinitely many hidden neuron functions $(x-b)_+$, none of them are inside the RKHS,

⚠ This smoothness penalty does not allow the ReLU to be part of the RKHS. However, this is still a universal penalty (as the set of functions with squared integrable second derivative is dense in L_2).

Upper-bound on RKHS norm γ_2 in all dimensions. We can first find a bound directly from the one on γ_1 in Eq. (9.10), which is exactly Eq. (9.11), ending up with the restriction on the ball of center 0 and radius R of the Sobolev space corresponding to square integrable $s = \frac{d}{2} + \frac{5}{2}$ derivatives on \mathbb{R}^d . It turns out this provides a bound on γ_2 (as can be shown by reproducing the reasoning from Section 9.3.4).

However, this bound is not optimal, which can already be seen in dimension $d = 1$, where we obtain $s = 3$ instead of $s = 2$. It turns out that, in general, it is possible to show γ_2 is less than a Sobolev norm with index $s = \frac{d}{2} + \frac{3}{2}$. This can be done by drawing links with multivariate splines ([Wahba, 1990](#); [Bach, 2023b](#)).

9.6 Experiments

We consider the same experimental set-up as Section 7.7, that is, one-dimensional problems to highlight the adaptivity of neural network methods to the regularity of the target function, with smooth targets and non-smooth targets. We consider several values for the

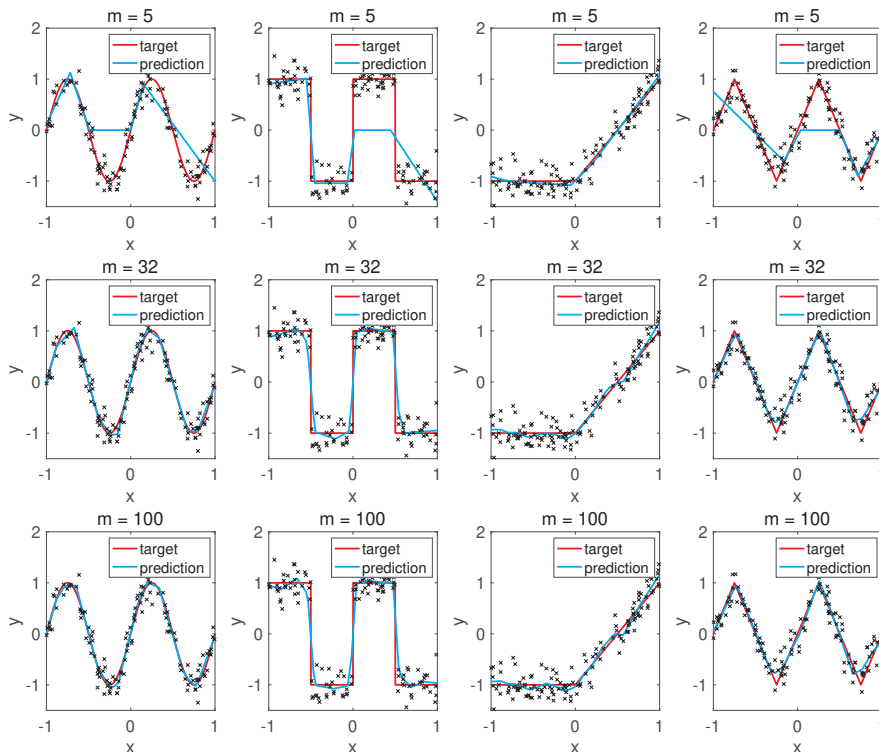


Figure 9.3: Fitting one-dimensional functions with various numbers of neurons m and no additional regularization (top: $m = 5$, middle: $m = 32$, bottom: $m = 100$), with four different prediction problems (one per column).

number m of hidden neurons, and we consider a neural network with ReLU activation functions and an additional global constant term. Training is done by stochastic gradient descent with a small constant step-size and random initialization.

Note that for small m , while a neural network with the same number of hidden neurons could fit the data better, optimization is unsuccessful (SGD gets trapped in a bad local minimum). Moreover, between $m = 32$ and $m = 100$, we do not see any overfitting, highlighting the potential under-fitting behavior of neural networks. See also <https://francisbach.com/quest-for-adaptivity/>.

9.7 Extensions

Fully-connected single-hidden layer neural networks are far from what is used in practice, particularly in computer vision and natural language processing. Indeed, state-of-the-art performance is typically achieved with the following extensions:

- **Going deep with multiple layers:** The most simple form of deep neural network

is a multilayer fully connected neural network. Ignoring the constant terms for simplicity, it is of the form $f(x^{(0)}) = y^{(L)}$ with input $x^{(0)}$ and output $y^{(L)}$ given by:

$$\begin{aligned} y^{(k)} &= (W^{(k)})^\top x^{(k-1)} \\ x^{(k)} &= \sigma(y^{(k)}), \end{aligned}$$

where $W^{(\ell)}$ is the matrix of weights for layer ℓ . For these models, obtaining simple and powerful theoretical results is still an active area of research, in terms of approximation, estimation, or optimization errors. See, e.g., [Lu et al. \(2020\)](#); [Ma et al. \(2020\)](#); [Yang and Hu \(2021\)](#). Among these results, the “neural tangent kernel” provides another link between neural networks and kernel methods beyond the one described in Section 9.5 and that applies more generally (see, e.g., [Jacot et al., 2018](#); [Chizat et al., 2019](#)).

- **Residual networks:** An alternative to stacking layers one after the other like above is to introduce a different architecture of the form:

$$\begin{aligned} y^{(k)} &= (W^{(k)})^\top x^{(k-1)} \\ x^{(k)} &= x^{(k-1)} + \sigma(y^{(k)}). \end{aligned}$$

The direct modeling of $x^{(k)} - x^{(k-1)}$ instead of $x^{(k)}$ through an extra non-linearity, originating from [He et al. \(2016\)](#), can be seen as a discretization of an ordinary differential equation ([Chen et al., 2018](#)).

- **Convolutional neural networks:** To tackle large data and improve performances, it is important to leverage prior knowledge about the typical data structure to process. For instance, for signals, images, or videos, it is important to take into account the translation invariance (up to boundary issues) of the domain. This is done by constraining the linear operators involved in the linear part of the neural networks to respect some form of translation invariance and, thus, to use convolutions. See [Goodfellow et al. \(2016\)](#) for details. This can be extended beyond grids to topologies expressed in terms of graphs, leading to graph neural networks (see, e.g., [Bronstein et al., 2021](#)).

9.8 Conclusion

In this chapter, we have focused primarily on neural networks with one-hidden layers and provided guarantees on the approximation and estimation errors, which show that this class of models, if empirical risk minimization can be performed, leads to a predictive performance that improves on kernel methods from Chapter 7, by being adaptive to linear latent variables (e.g., dependence on an unknown linear projection of the data). In particular, we highlight that having a number of neurons in the order of the number of observations is not detrimental to good generalization performance, as long as the norm of the weights is controlled.

We pursue the study of over-paramaterized models in Chapter 12, where we show how optimization algorithms can both globally converge and leads to implicit biases.