An unorthodox shift in the variance-bias tradeoff in Neural Networks: the double descent phenomenon and the ease of training in the overparametrized regime

by

Joan Velja

Bachelor's Thesis

of the requirements for the degree of

Bachelor of Science

Department of Computational Sciences

Università Bocconi

July, 2023

Dr. Enrico Maria Malatesta

Dedication

I am grateful to the many people who helped me during my three years at university which culminated in this Thesis.

To my family, the fundamental pillar to my life, who made all this possible. Words cannot express the utter recognition I have for you, for your efforts and the struggles you went through to ease this journey. I hope I made you proud.

My friends, the new ones and the old ones, who always supported me, through the ups and downs, the rants and what not. I hope this work justifies a tiny bit all of that.

To My advisor, Dr. Malatesta, who has always been an inspiring, patient and encouraging mentor, who pushed me to go always the extra mile. I hope I was a good student to work with.

La vita è bella.

Contents

List of Figures

List of Tables

1 Introduction

The primary objective of machine learning models is to discover statistical patterns in data that can be effectively generalized to new, previously unseen samples. The performance of these machines in achieving this goal is dependent on several key factors, including the size and characteristics of the training data set, the complexity of the specific learning task at hand, and the underlying inductive bias of the learning machine being utilized. Despite a significant amount of theoretical work in this area, accurately identifying the contributions of these factors to the generalization performance of learning machines remains a significant challenge. In statistical literature, the baseline for modern Machine Learning implementations, it is commonly advised to avoid utilizing models with excessive number of parameters, as doing so can result in *overfitting* and a reduced ability to generalize. Empirically, when evaluating the performance of models against test data, it is often observed that the test error as a function of model complexity exhibits a U-shaped curve. This phenomenon is explained by the fact that the test error initially decreases due to decreasing bias of the model, but eventually increases due to a rise in variance [12].

Recent advances in deep learning have highlighted the need for a new theoretical framework for generalization [17]. The empirical evidence suggests that larger models tend to yield better performance [14, 19, 21], leading to the trend of training progressively larger networks with cutting-edge architectures that can feature hundreds of billions of parameters [14]. Such networks operate in an *overparameterized* regime [1, 6, 21, 25] , where the number of parameters significantly exceeds the number of training samples, and can be highly expressive to the extent of fitting random noise [25]. Nonetheless, they demonstrate a remarkable ability to generalize, contradicting the traditional understanding from classical statistical learning theory. It is, therefore, evident that overparameterized networks possess inductive biases that suit the particular learning task, emphasizing the importance of a theoretical framework for generalization that can elucidate such biases.

The double descent phenomena refers to the unexpected behavior of the test error curve as a function of model size. It exhibits a U-shaped curve where the test error initially increases with model size, reaches a peak, and then decreases again as the model size continues to increase. This counterintuitive behavior challenges our understanding of the bias-variance tradeoff and has significant implications for determining the optimal complexity of deep learning models.
Understanding the double descent phenomena and the ease of training in overparametrized models has become a crucial area of research in deep learning. Exploring the underlying theoretical principles that explain the observed behavior and investigating its implications for model complexity can provide valuable insights into the learning dynamics of deep neural networks. Fur-

thermore, such understanding can inform the development of more efficient and effective training algorithms, guiding the design of future deep learning architectures.
In this dissertation, we aim to comprehensively analyze the double descent phenomena and the ease of training in overparametrized deep learning models. We will first assess the traditional bias-variance tradeoff and its limitations in explaining deep learning phenomena. We will then investigate the behavior of training in overparametrized models, reviewing the intutions that led to our final experiments on the interpolated loss paths for different parameter spaces, shedding light on the learning dynamics in this scenario.

By elucidating the theoretical principles behind the double descent phenomena and examining its manifestation in experimental settings, our aim is to contribute to a deeper understanding of the learning behavior of deep neural networks and provide insights into the optimal design of models for achieving superior generalization performance in limited-data regimes.

2 Theoretical background - Double Descent

2.1 Traditional view

The bias-variance tradeoff is a fundamental theoretical concept in supervised machine learning that addresses the problem of balancing the errors caused by two sources of prediction error: bias and variance.

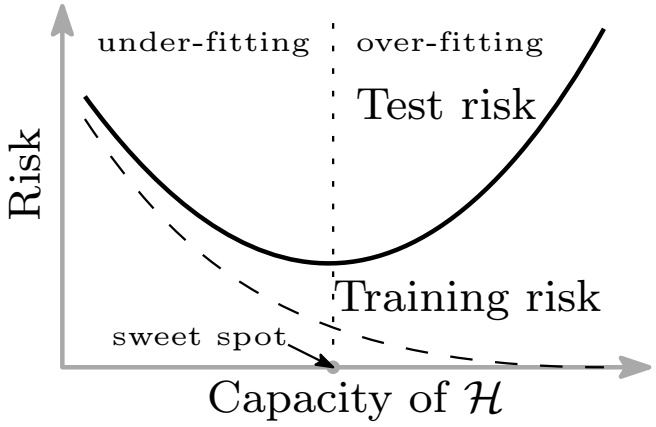


Figure 2.1: Traditional bias-variance, reprinted from Belkin et al. [1].

In a prediction setting, we wish to learn a function f that can predict y given $\boldsymbol{x}$ well enough (with respect to some loss function). Let us assume we are given a dataset $D = \{(\boldsymbol{x_1}, y_1), ..., (\boldsymbol{x_n}, y_n)\}$, drawn i.i.d. from some distribution $P(X, Y)$. Let us further assume a regression setting, i.e. $y \in \mathbb{R}$. When talking about fitting a model, we discuss the problem of whether to fit a "simple" model such as the linear $y = \theta_0 + \theta_1 x$ or a more complex non-linear polynomial such as $y = \theta_0 + \theta_1 x + \cdots + \theta_5 x^5$. We make the assumption that $y = f(x) + \epsilon$, where ϵ satisfies $\mathbb{E}[\epsilon] = 0$ and $var(\epsilon) = \sigma^2$. Our task now is to construct a hypothesis $\hat{f}_n$ given a fixed size training set S that mimics f well on all future unseen examples. In other words, we want $\hat{f}_n$ to generalize well, i.e. to have good *generalization error*.
Suppose $\hat{f}_n$ is obtained with some unspecified training process over S. As before, note that $\hat{f}_n$ is random due to the randomness of the error term, embedded in the training set examples. We

consider a new unseen example pair $(y_*, \boldsymbol{x}_*)$ and the corresponding generalization error, where the expectation is over the randomness in ϵ in the test example, and in $\hat{f}_n$:

$$
\begin{aligned}
MSE(\hat{f}_n) &= \mathbb{E}[(y_* - \hat{f}_n(\boldsymbol{x}_*))^2] \\
&= \mathbb{E}[(\epsilon + f(\boldsymbol{x}_*) - \hat{f}_n(\boldsymbol{x}_*))^2] \\
&= \mathbb{E}[\epsilon^2] + \mathbb{E}[(f(\boldsymbol{x}_*) - \hat{f}_n(\boldsymbol{x}_*))^2] - \mathbb{E}[2\epsilon(f(\boldsymbol{x}_*) - \hat{f}_n(\boldsymbol{x}_*))] \\
&= \sigma^2 + \mathbb{E}[(f(\boldsymbol{x}_*) - \hat{f}_n(\boldsymbol{x}_*))^2] - \underbrace{\mathbb{E}[\epsilon]}_{=0}\, \mathbb{E}[2(f(\boldsymbol{x}_*) - \hat{f}_n(\boldsymbol{x}_*))] \quad \text{by i.i.d. of } \epsilon \\
&= \sigma^2 + \mathbb{E}[(f(\boldsymbol{x}_*) - \hat{f}_n(\boldsymbol{x}_*))^2] \\
&= \sigma^2 + \mathbb{E}[f(\boldsymbol{x}_*) - \hat{f}_n(\boldsymbol{x}_*)]^2 + Var(f(\boldsymbol{x}_*) - \hat{f}_n(\boldsymbol{x}_*)) \quad \text{by } \mathbb{E}[X^2] = E[X]^2 + Var(X) \\
&= \underbrace{\sigma^2}_{\text{irreducible error}} + \underbrace{\mathbb{E}[f(\boldsymbol{x}_*) - \hat{f}_n(\boldsymbol{x}_*)]^2}_{\text{Bias}^2} + \underbrace{Var(\hat{f}_n(\boldsymbol{x}_*))}_{\text{Variance}} \quad \text{by } Var(a - X) = Var(X)
\end{aligned}
\tag{2.1}
$$

This leads us to a clear breakdown of our generalization error into three distinct parts:

- the **Irreducible error**, i.e. the data intrinsic noise, embedded into the distribution of the data itself that cannot be altered;
- the **Bias**, i.e. the inherent error that we obtain from the classifier even with infinite training data, inherent to the model;
- the **Variance**, i.e. the degree of overspecialization towards a given training set.

In order to provide a visualization for the phenomena, we can come up with a simple example. Let us assume some x in a range, in this case from [0,2] and y being $y = cos(\pi x) + \varepsilon$, where ε is some gaussian noise with $\mu = 0$ and $\sigma = 1$. We syntetically generate this data and plot it on a cartesian graph, creating two splits, the *train* and *test* splits (80/20 ratio).

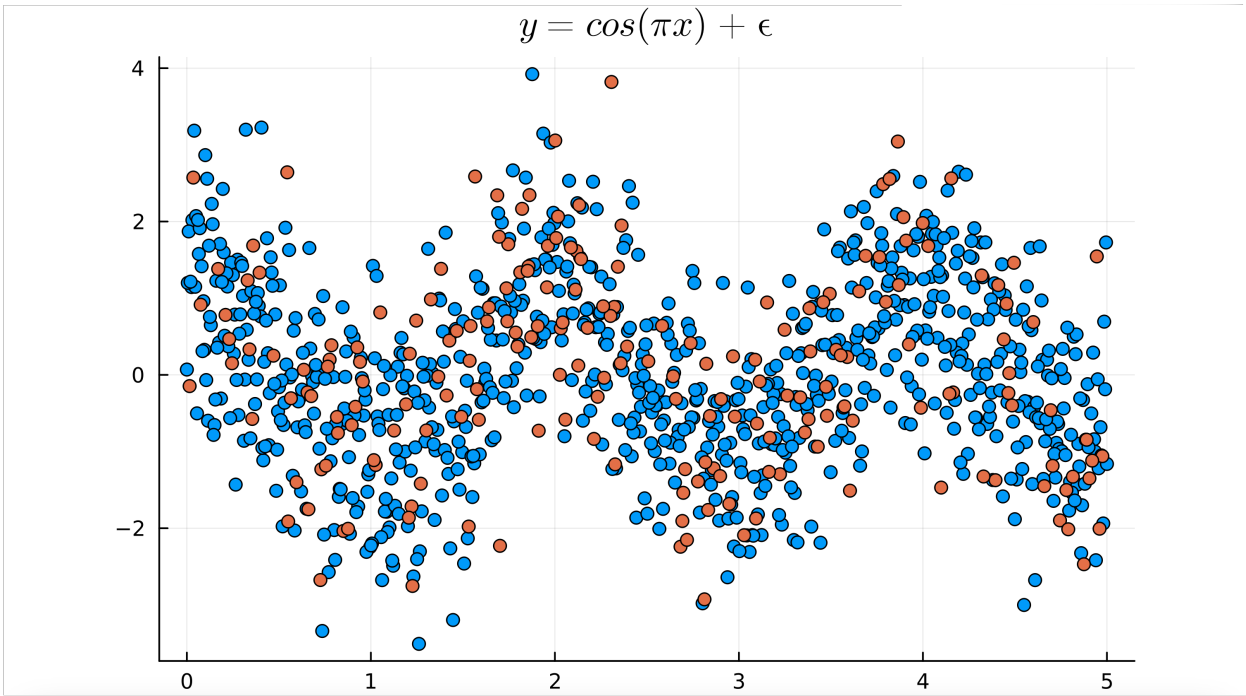


Figure 2.2: Scatter plot of points with distribution $cos(\pi x)$ plus some gaussian noise ϵ

From here, we will try to fit polynomials of different degrees to the training set, and evaluate the MSE of the fit in the test set. The polynomials will range from degree 1 to degree 100.

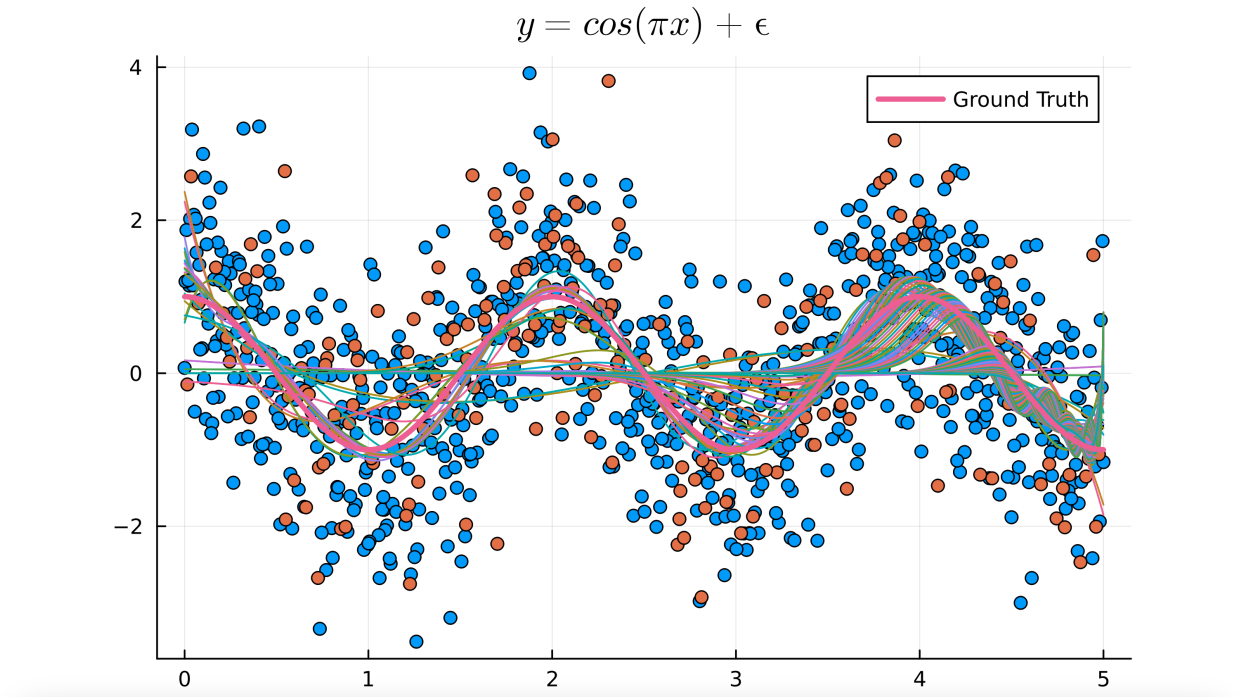


Figure 2.3: Fitted polynomials to the dataset

As the degree of polynomials increases, the fitted function struggles to fit the noise in the dataset and thus its MSE will increase:

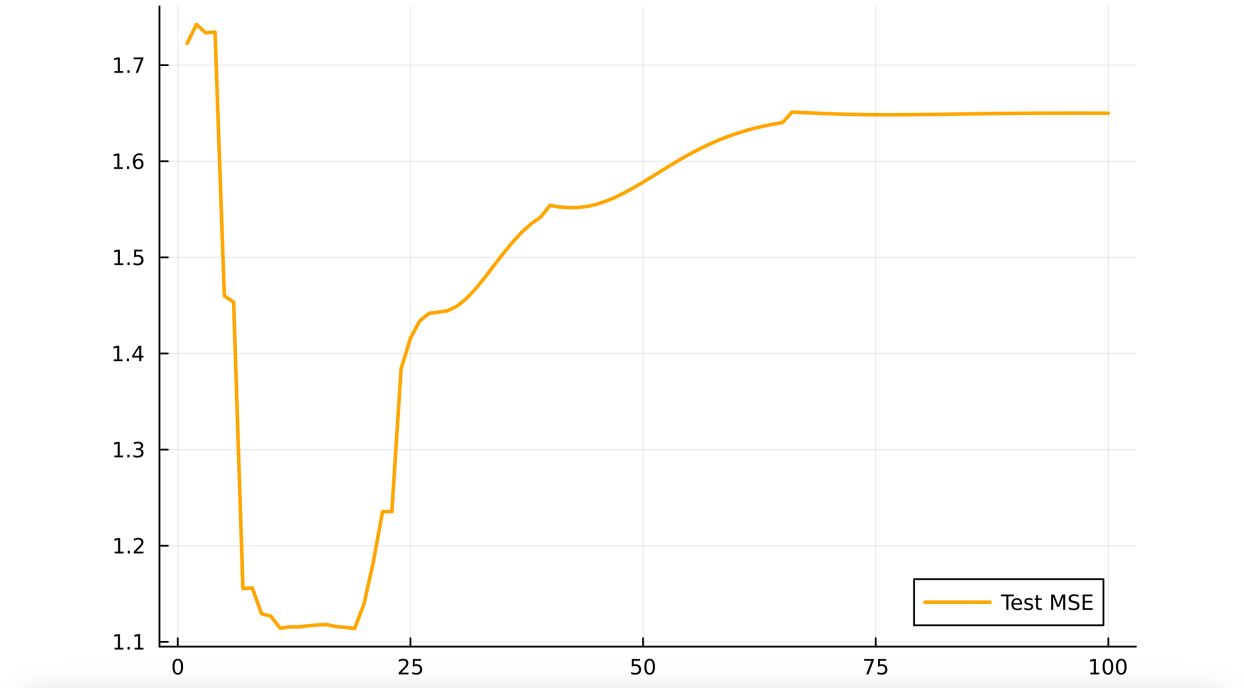


Figure 2.4: Test MSE as a function of the degree of the polynomial

Consistently with the aforementioned theory.

2.2 Double Descent

From the short experiment ran in the above section, we would expect the best model (in any scenario) to be obtained through some balance between bias (underfitting) and variance (overfitting). In applications instead, we observe remarkable test performance from very complex models. But why is so?

2.2.1 A mathematical explanation

Throughout this discussion, we will use a simple example dataset to demonstrate our thought process. This is the collection of points:

$$\boldsymbol{x} = (0, 1, 2, 3, 4, 5)$$

$$\boldsymbol{y} = 4\left(\boldsymbol{x} - \frac{5}{2}\right)^2 = (25, 9, 1, 1, 9, 25)$$

This dataset has no noise and it's simply a quadratic function sampled at 6 equally spaced points.

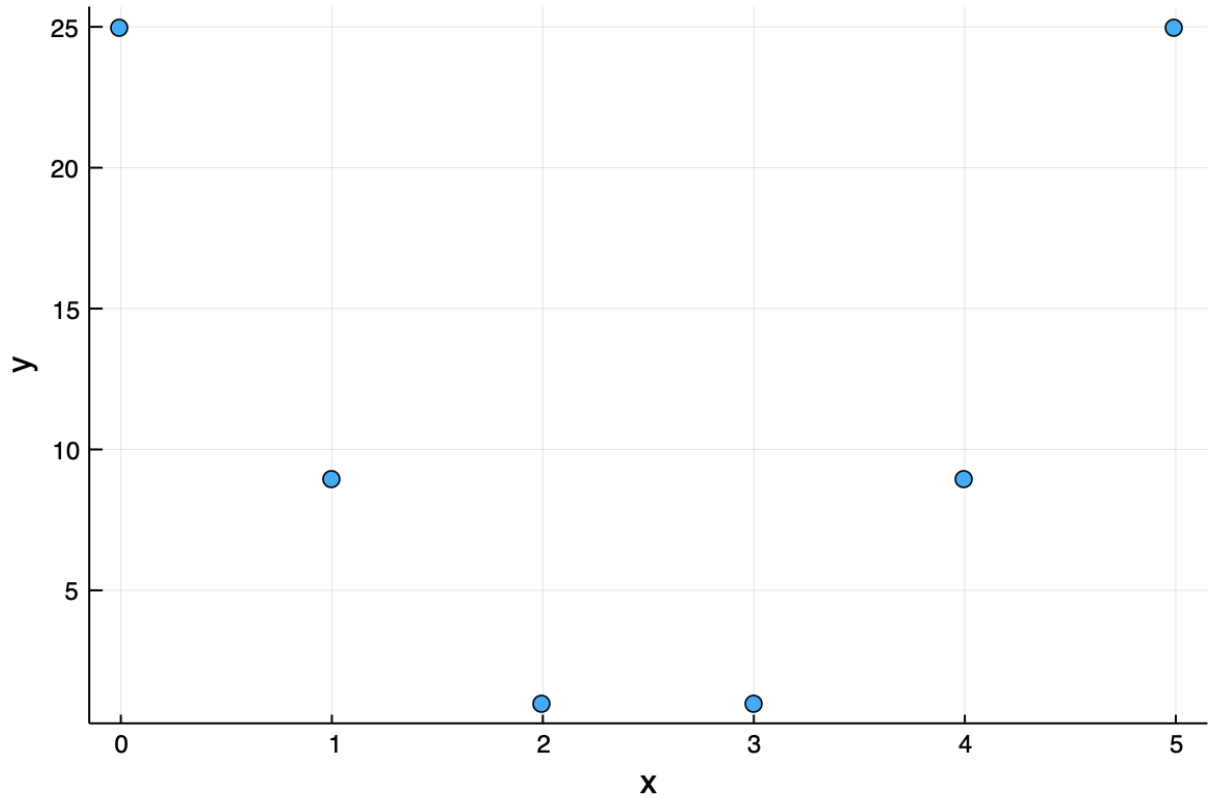


Figure 2.5: Scatter plot of points with distribution $4(x - \frac{5}{2})^2$

The input data is a vector $\boldsymbol{x} = (x_i)_{i=1}^N$ and the target is a vector $\boldsymbol{y} = (y_i)_{i=1}^N$. The model will attempt to fit a piecewise linear function to the dataset, from now on defined as $\mathcal{D} =< \boldsymbol{x}, \boldsymbol{y} >$, and the way this will be done is to pick $\boldsymbol{t} = (t_k)_{k=0}^K$ *knot points*, where the linear functions will be allowed to bend. The bending will be represented by use of *radial basis functions*. In particular we will define:

$$\phi_k(x) = \frac{1}{2}|x - t_k| \tag{2.2}$$

and say that our model will be written as

$$\alpha + \beta x + \sum_{k=0}^{K} \gamma_k \phi_k(x) \tag{2.3}$$

where α, β and $(\gamma_k)_{k=0}^K$ are learnable parameters. It can be verified that, since the absolute value is linear aside from the singularity point at the origin, that this is a parametrization of piecewise linear functions with breakpoints only allowed at the t_k points. We will consider the knot points t_k to be fixed, and picked as independent and identically distributed random points from a continuous distribution with density $p(x)$. Typically we take $p(x)$ to be uniform on the range of $\boldsymbol{x}$.

We will fit this model by Ordinary Least Squares regression on our learnable parameters. As is standard, we will seek the minimum norm solution when the setting is under-determined (i.e, when we have more parameters than data points, also known as the *overparametrized regime*). To simplify the analysis, we will only assume we are only looking to minimize $||\gamma||^2$, thus excluding α and β from the minimization for brevity reasons. It is nonetheless to be considered general, as the minimization process would just be more complex but not different.

2.2.1.1 Below the interpolation threshold

When we are well below the interpolation threshold the following happens: let us assume the very lowest case, no knot points at all ($K = 0$), which is OLS regression.

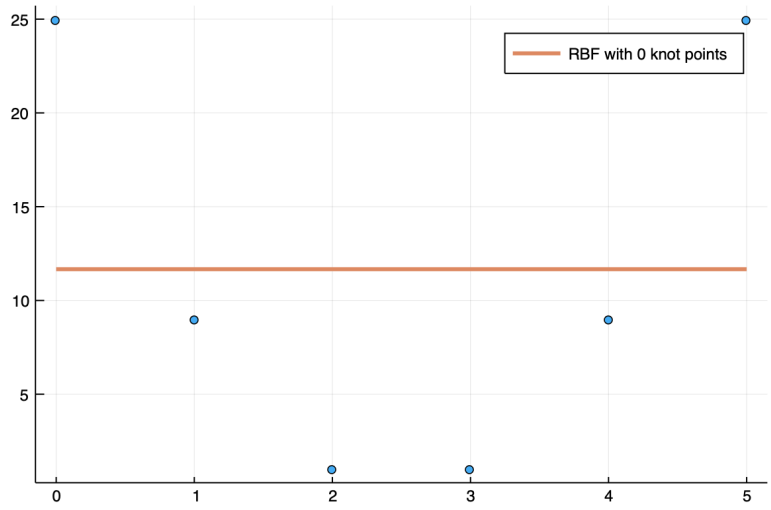


Figure 2.6: OLS regression on $\mathcal{D}$, 0 knot points

As our dataset is not linear, this fit is not particularly good, as it attains $MSE = 99.56$. We can improve our fit by adding a knot point.

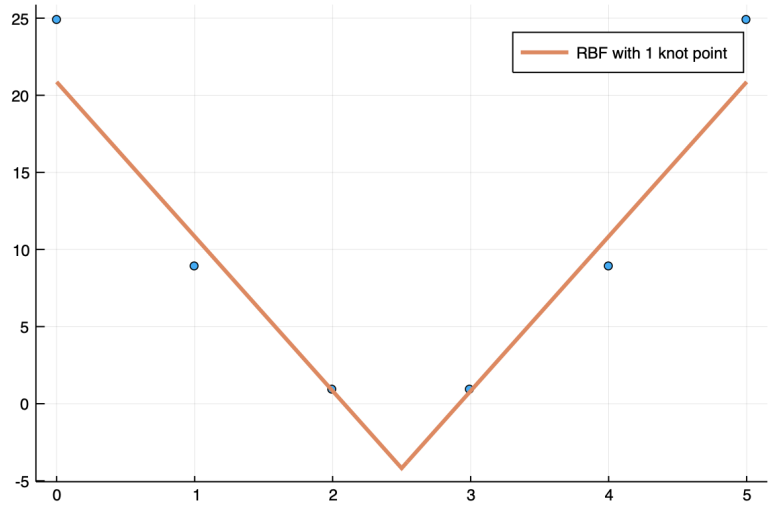


Figure 2.7: OLS regression on $\mathcal{D}$, 1 knot point

How much this improves performance depends on the dataset, the ground truth and the choice of the random knot. Both numerical testing and exact computation show that on average the performance is improved w.r.t. the linear solution for MSE metrics. Below we can find the MSE as a function of the location of t_0 in $[0, 5]$.

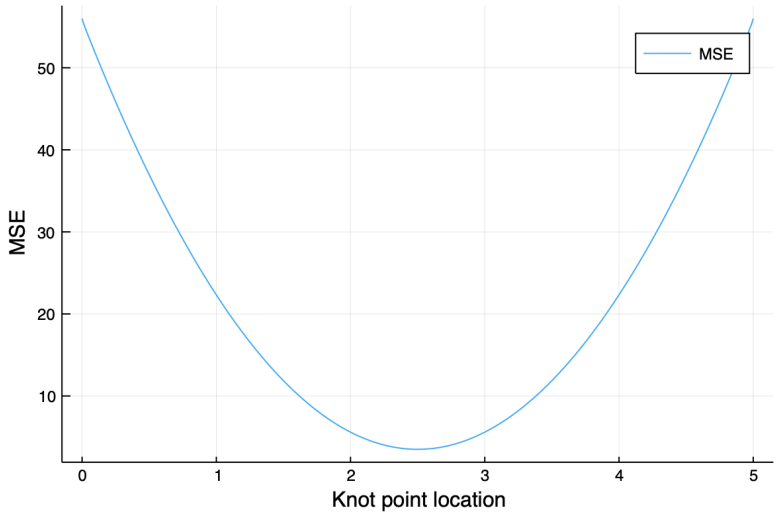


Figure 2.8: $MSE(t_0)$, where $t_0 \in [0, 5]$

In our case with our starting dataset, we attain a MSE of 6.84, a stark improvement from the previous instance, for a model with 1 knot positioned at $t = 2.5$.

So far, this phenomena matches traditional statistical intuition. Our simple linear fit is very poor since the ground truth is non linear. Expanding our model to allow for non linearity should improve the fit, as long as there is sufficient data.

2.2.1.2 At the *Interpolation Threshold*

We now turn our attention to the behavior *at* the interpolation threshold. Our goal here will be to illustrate why this will perform poorly by showing that, for this model, the average error (averaged over random knot points) is infinite.

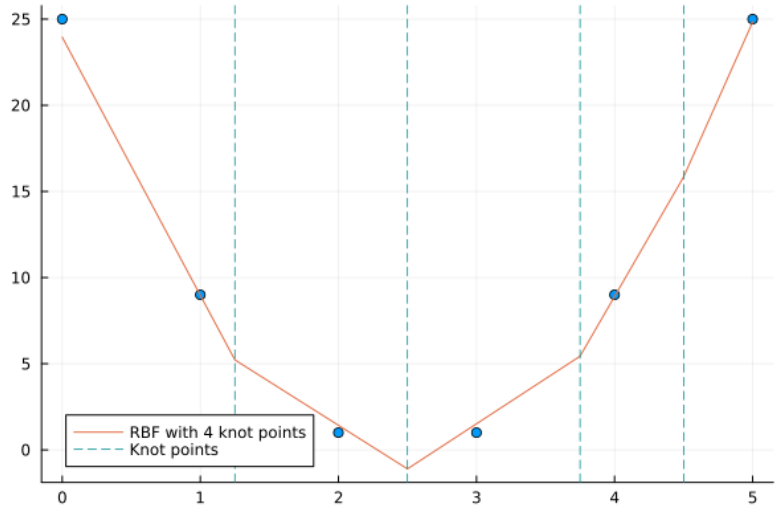


Figure 2.9: OLS regression on $\mathcal{D}$, 4 knot points

In this case, we see that we have 4 knots placed between the five right-most data points, and no knot point between the two left-most ones. The main takeaway is that, for this choice of points, there is no freedom in how the interpolating model is selected once the knot points are fixed: there is a unique line which passes through the two left points and if we want to exactly interpolate the data, we must start with that line.

This line intersects the knot point at $x = 1.25$ at a point which is uniquely determined by the intersection of the two lines, and then it must bend to pass through the data point at $(2, 1)$. From there it hits the following knot at $x = 2.5$, at which point again it must bend to hit the point at $(3, 1)$, and so on. There is never any choice since it needs to be an interpolating function, so the entire solution is (usually unique and) fixed.

This is the main lesson of what happens at the interpolation threshold: there is *rigidity* in what model we fit. Indeed, if the case is such that one unique interpolating function only exists, we have no control whether the model is good or not.

This phenomena can be made so pronounced that our **average error** can be made infinite for all L_p-norms, and so also for the *MSE* based norm, L_2. What we are doing is setting four random knot points, three of which are to the right of $x = 2.5$ and one occurs to the left of $x = 2$. To be explicit, we make that the first point falls in the interval $[2 - \epsilon, \frac{2-\epsilon}{2}]$. The event of the first three points being to the right of $x = 2.5$ and the last being just to the left of $x = 2$, occurs with probability

$$\left(\frac{1}{2}\right)^3 \left(\frac{\epsilon}{10}\right) \sim \epsilon \tag{2.4}$$

When this occurs, the left-most segment will hit the left-most knot point somewhere below $(2, 1)$ (since it must follow the dataset's trend, an upward curving function around 2). Let's say that we ensure this by picking ϵ sufficiently small that it is no larger than 0.9. Then, at that knot point, the next linear segment will need to go from a point no better than $(2 - \epsilon, 0.9)$, which means that it needs to have a slope of at least $\frac{0.1}{\epsilon} \sim \frac{1}{\epsilon}$ to pass through the point $(2, 1)$.

With this large slope, we have our first, key oscillation. The function has started behaving in such way that it is not allowing for good generalization of the ground truth, and we simply need to show that this oscillation is large enough to make the average error ininte under any L_p norm. Notice that we have no other knot points in the region from $[2.25, 2.5]$. On that region, our interpolating function takes a value no smaller than $1 + s\frac{1}{4}$, where $s \sim \frac{1}{\epsilon}$ is the slope, and thus the value and the error are both of order $\frac{1}{\epsilon}$ on that interval of width $\frac{1}{4}$.

By considering the sequence of $\epsilon = \frac{1}{2^m}$ we can create a sequence of disjoint events which can be used to lower-bound our expected error. In particular, letting $\hat{f}$ be the piecewise interpolating function and f be the ground-truth quadratic, we see that we may estimate (discarding constant factors)

$$\mathbb{E}\left[\|\hat{f} - f\|_q^q\right] \gtrsim \sum_{m=1}^{\infty} \frac{1}{2^m}(2^m)^q = \sum_{m=1}^{\infty} 2^{m(q-1)} \tag{2.5}$$

Where the expected value of the norm is defined as the integral over the range $[0, 5]$. Thus the average error, as measured for any q-norm greater than 1, is infinite. Given this fact, the following proposition holds:

Proposition 1. *For any dataset* $\mathcal{D} =< \boldsymbol{x}, \boldsymbol{y} >$ *with at least* $\boldsymbol{N \geq 3}$ *points, not all collinear, any bounded ground truth function* f *and any* $\boldsymbol{q \geq 1}$*, then the best fit model of this type* $\hat{f}$ *with* $\boldsymbol{N - 2}$ *knot points has* $\mathbb{E}\left[\|\hat{f} - f\|_q^q\right] = \infty$

What the proposition states is that these models are so rigid, that bad behavior is guaranteed independent of the dataset or ground truth. Simply the fact of having a unique interpolating function that can only bend at random points forces the fit to be poor on average when at the interpolation threshold.

2.2.1.3 After the threshold

The above conclusion already provides us some hints about larger models might be better at this task: they provide many choices about which interpolating model to pick.

To see why these models, which from now on we will define as *overparametrized*, perform well, we will start by assuming a twice continuously differentiable (at least) and we select a large number of random knot points $(t_k)_{k=1}^K$ from our density $p(x)$. As we did so far, we select the model so to minimize $\|\boldsymbol{\gamma}\|_2$, which is the norm of the coefficients of the basis function defined at 2.3. Assume the following scenario:

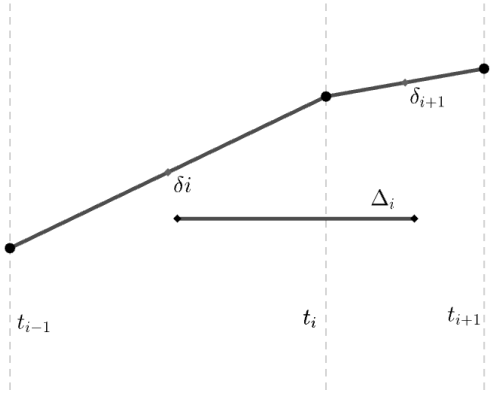


Figure 2.10: Setting

The figure 2.11 shows three consecutive knot points t_{i-1}, t_i, t_{i+1}. δ_i and δ_{i+1} instead represent the slopes of the two segments. Finally, Δ_i denotes the distance between the midpoints of the two segments.
Now:

1. Since δ_i and δ_{i+1} are two consecutive intervals in the piecewise linear function, we can express the difference in slopes in terms of the coefficient of the model. In particular, we

note that $1, x$ and $\phi_k(x)$ for all $k = i$ are all linear across both intervals. This means that they all contribute the same constant expression to the slope on both sides, thus $\delta_{i+1} - \delta_i$ is simply the difference in slopes between the two sides of $\gamma_i \phi_i(x) = \frac{\gamma_i}{2}|x - t_i|$. This difference is represented by γ_i. Thus:

$$\delta_{i+1} - \delta_i = \gamma_i \tag{2.6}$$

2. Since the points on our piecewise linear interpolation are assumed to be close together (being in the *overparametrized* regime), and all lie on $\hat{f}$, we can say that both δ_i and δ_{i-1} are approximately equal to the derivative of $\hat{f}$ at the midpoints of the intervals. Thus, their difference quotient is approximately the second derivative at t_i:

$$\hat{f}''(t_i) \approx \frac{\delta_{i=1} - \delta_i}{\Delta_i} = \frac{\gamma_i}{\Delta_i} \qquad \text{by 2.6} \tag{2.7}$$

3. Finally, let us collect two approximations for the probability that a randomly sampled point from $p(x)$ lands in the interval connecting the two midpoints. On the one hand, the definition of density tells us this probability is obtained by integrating $p(x)$ over the interval, which (due to us being in the overparametrized regime and so the integrals being small) is approximately equal to the rectangular approximation (Newton and Leibniz, 17th century) $\Delta_i p(t_i)$. On the other hand, sampling K points from the distribution and requiring one to land in the interval can be approximated as $\frac{1}{K}$. Thus:

$$\Delta_i p(t_i) \approx \frac{1}{K} \longrightarrow \Delta_i \approx \frac{1}{K p(t_i)} \tag{2.8}$$

This is a rough approximation, and a further proof is required to carry on with the analysis: instead, we need to consider an intermediate scale made by collecting together, for instance, $\sqrt{K}$ many points. Since $\sqrt{K} \ll K$ and $1 \ll \sqrt{K}$, this lets us both assume the second derivative is approximately constant in that region and that the length of the interval has random fluctuations much smaller than the length itself. Let us break down the claims into two:

- **The second derivative is approximately constant in the region**: to establish this, we must understand that the second derivative measures the curvature of a function. With many points, it is statistically more likely that the points sampled from the density $p(x)$ will be more evenly distributed over the domain. If the interval size is proportional to $\sqrt{K}$, then as K increases, the interval size also increases, but at a slower rate. This means that within each interval, the function will have a smaller and smaller curvature within that interval which is less likely to change dramatically. To put this into formulae, we will make use of the assumption made previously about the function being twice differentiable. This means we have a continuous second derivative, or in other words, the rate at which the slope of the function changes is smooth. Given a specific point δ_i, and considering an interval Δ_i around it, we can apply the Taylor's series expansion:

$$\hat{f}(x) = \hat{f}(t_i) + (x - t_i) * \hat{f}'(t_i) + 0.5 * (x - t_i)^2 * \hat{f}''(t_i) + \text{ higher order terms} \tag{2.9}$$

where x is in the interval Δ_i. Since Δ_i is small, the value $(x - t_i)$ is small. So, the square of a small number $(x - t_i)^2$ is even smaller. The higher-order terms become negligible, and thus the second derivative term contributes little to the value of $\hat{f}(x)$ in the interval Δ_i. This justifies our assumption that the second derivative is approximately constant within the interval Δ_i. The smaller the Δ_i, the more this approximation holds.

- **The length of the interval has random fluctuations much smaller than the length itself**: Suppose we have a sequence of intervals Δ_i, and let X_i represent the length of the $i - th$ interval. Each X_i is a random variable drawn from the same distribution with mean μ and variance σ^2. We know from the Law of Large Numbers that as we take more samples, the sample mean converges to the true mean. So, the average length of the intervals $(\Delta_1 + ... + \Delta_k)/k$ converges to the true mean μ as k grows large. This tells us that the average length of the intervals is stable as we take more and more samples.
  On the other hand, we have the Central Limit Theorem, which tells us that the sum of a large number of independent and identically-distributed random variables (like our Δ_i's) is approximately normally distributed. From here we get that the width of this distribution scales as the square root of the number of variables ($\sigma\sqrt{k}$ in this case). The fact that the width of the distribution grows slower than the number of variables justifies our assumption that the random fluctuations in the length of the interval are much smaller than the length itself. As we increase k, the average length of the intervals becomes more stable (due to the Law of Large Numbers), while the variability in the lengths (due to the Central Limit Theorem) does not grow as fast. Therefore, the random fluctuations are smaller relative to the length itself.

By our second bullet point, we may write:

$$\|\boldsymbol{\gamma}_2^2\| = \sum_{k=0}^{K} \gamma_k^2 \approx \sum_{k=0}^{K} \hat{f}''(t_k)^2 \Delta_k^2 \tag{2.10}$$

Applying the third bullet point, we can furthermore state:

$$\sum_{k=0}^{K} \hat{f}''(t_k)^2 \Delta_k^2 \approx \frac{1}{K} \sum_{k=0}^{K} \frac{\hat{f}''(t_k)^2}{p(t_k)} \Delta_k \tag{2.11}$$

The last equation is a discrete approximation of the integral of $\frac{\hat{f}''(x)^2}{p(x)}$, so we can conclude by stating:

$$\|\boldsymbol{\gamma}_2^2\| \approx \frac{1}{K} \int \frac{\hat{f}''(x)^2}{p(x)} dx \tag{2.12}$$

We typically consider the case where $p(x)$ is constant on an interval, and zero outside of it, so if we assume that $\hat{f}''(x) = 0$ outside our interval, then we may conclude that we are trying to minimize

$$\int_a^b \hat{f}''(x)^2 dx \tag{2.13}$$

And this gives us our key observation. By minimizing the L_2-norm of our parameters, we are actually minimizing the integral of the square of the second derivative of our interpolating function. Since second derivatives tell us the change in derivative, this can be interpreted as trying to find an interpolating function whose derivative changes as little as possible. This idea has been studied extensively and goes by the name of the *natural cubic spline* [9].
This exemplifies one of the reasons why people believe that double descent occurs: the interpolating functions found using very large models can be better behaved between the points we are interpolating, and thus can match the ground truth better if the model is structured in a way to build in an inductive bias towards the correct type of solution.

The phenomena, first shown to hold in Neural Networks by Belkin et al. [1], can be summarized by the following graph:

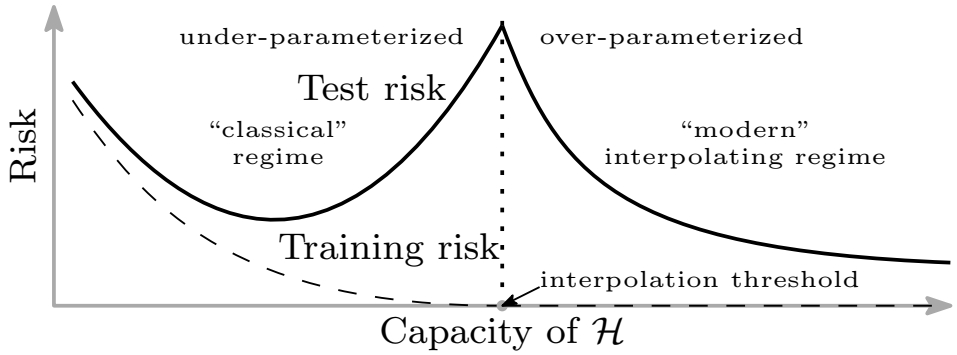


Figure 2.11: Double descent, reprinted from Belkin et al. [1].

3 Experimental results - Double Descent

In this chapter we will focus on reproducing the Double Descent phenomena discovered by Belkin et al. [1].

3.1 Experimental setting

We will carry out our experiments in the Julia programming language [3], a high-level, high-performance programming language developed specifically for applications in scientific and mathematical computation. It was designed to have a user-friendly and expressive syntax while addressing the performance issues and constraints commonly encountered in other languages, such as Python.

Dataset. To demonstrate the double descent risk curve, we train a number of differently parametrized neural networks on the famous MNIST dataset [18].

Dataset	Size of full training set	Feature dimension (d)	Number of classes
MNIST	$6 \cdot 10^4$	784	10

Table 3.1: Dataset used for the experiment

Model training. Each model is trained to minimize the squared loss on the given training set. Without regularization, such model is able to interpolate the training set when its capacity surpasses certain threshold (interpolation threshold). Since neural networks are sensible to weight initialization, we repeat the same experiment 20 times and report the mean for the risks.

Neural Network specifics. In our experiments, we use a fully connected neural network with a single hidden layer. The parametrization of the model is thus given by the number of hidden units of the model. We use stochastic gradient descent (SGD) with momentum (`lr = 0.01` and `m = 0.95`) to solve the optimization problem in this setting. The ERM optimization problem in this setting is generally very complex due to the non-convexity of the problem. Consequently, SGD, as mentioned previously, is known to be sensitive to initialization. To tackle this issue, we adopt what Belkin et al. proposed in the experimental setting for Neural Networks: we used a "weight reuse" scheme, where the trained weights and biases, obtained from training a smaller

neural network, are used as initialization for training sequentially larger networks. This procedure, guarantees training risk to be at least equal among two networks being sequentially larger. In the weight reuse scheme, we load, for all models after the first one, the parameters coming from the training of a smaller network with H1 hidden units. To train a larger network with H2 > H1 hidden units, we initialize the first H1 hidden units of the larger network to the weights learned in the smaller network. The additional weight of H1, since the models are in turn larger by only 1 hidden unit, is initialized with normally distributed random weight (mean 0 and variance 0.01) and bias = 0. The smallest network, with 1 hidden unit, is initialized through the procedure explained in the Glorot uniform distribution [10]. The training is carried out for 100 epochs throughout all models, as the procedure is fairly simple and guaranteed to converge given the easy task at hand.

3.2 Experimental Results

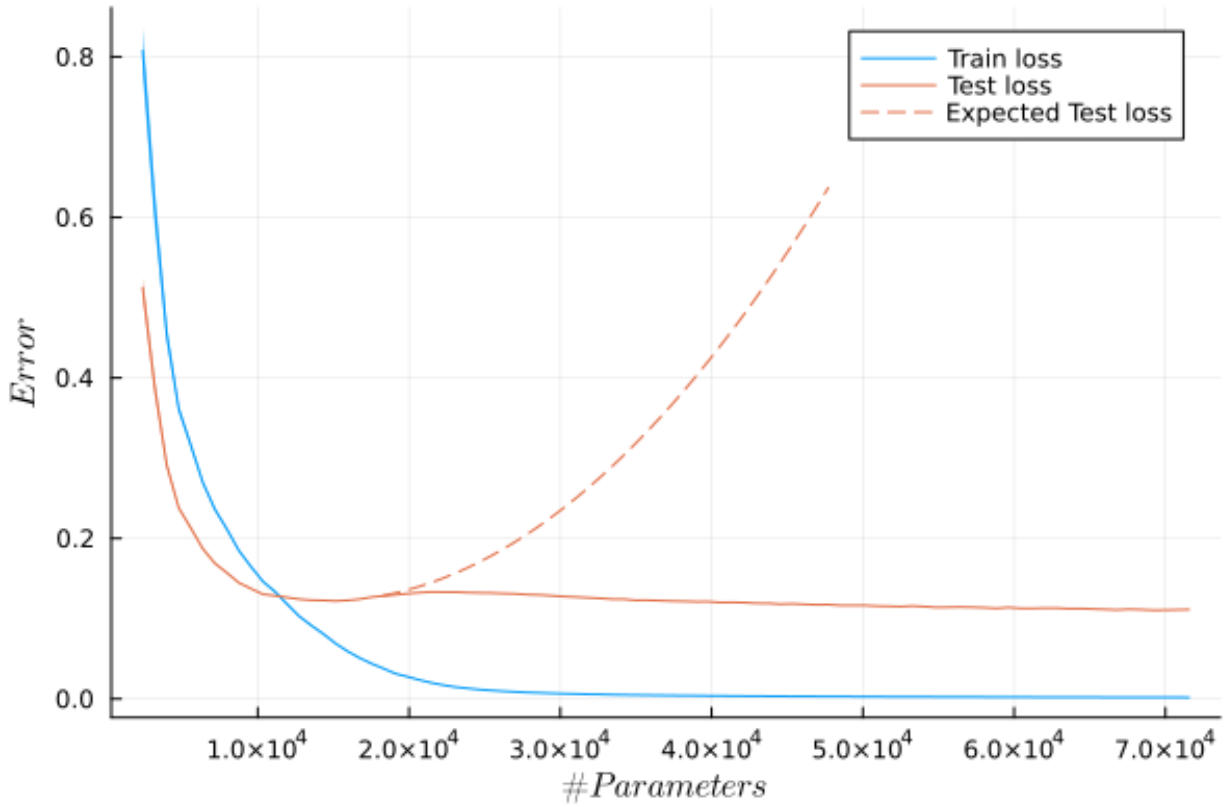


Figure 3.1: The double descent phenomena: the test loss, after an initial phase (in the so called *underparametrized regime*) of decrease, slows down and reverses its concavity, resulting in a slight bump around the so called *interpolation threshold*. The traditional statistical view would suggest a path following a trend similar to the generic upward sloping dashed line, but after going into the *overparametrized regime*, the model starts generalizing again, decreasing once more its test loss.

4 Theoretical background - Optimizing overparametrized networks

4.1 Overview of the issue at hand

The phenomenon of the ease of training in overparametrized regimes in neural networks has emerged as a topic of significant interest within the machine learning community. As we mentioned in the previous chapter of this thesis, overparametrization refers to a scenario where the number of parameters in a model exceeds the number of training data points. Conventional wisdom suggests that overparametrization would lead to overfitting, where the model memorizes the training data at the expense of its ability to generalize to unseen data. However, as we thoroughly explained before, recent findings contradict this traditional understanding, observing that overparametrized neural networks can often generalize better and can be easier to train.

Training neural networks involves minimizing a non-convex loss function in a high-dimensional space. This task is theoretically challenging but often manageable in practical scenarios ([5, 8, 24]) and although the task of optimizing general neural loss functions is known to be NP-hard [4], gradient-based methods often succeed in finding global minimizers [7], where the network parameters achieve zero or near-zero training loss. However, the ease of achieving such good performance is not universally guaranteed and heavily relies on various factors including network architecture design, choice of optimizer, variable initialization, and other considerations.
Given the advances made by academia in the field, it is crucial to understand the factors that contribute to the ease of training in the overparametrized regime. Several pieces of work have already started investigating these factors. For example, Ian Goodfellow et al. (2015) [11] uses the visualizations of optimization landscapes to reveal the characteristic features of neural network optimization problems, such as high dimensionality and non-convexity. In addition to this, Zhang et al. (2017) [25] discusses the implicit regularization effects in deep neural networks, which could explain their surprising generalization capabilities in the overparametrized regime. Furthermore, Neyshabur et al. (2019) [22] provides theoretical analysis on how overparametrization can improve generalization.

While the ease of training in the overparametrized regime seems to contradict the traditional understanding of the bias-variance tradeoff, it is in fact part of a more complex relationship, as

captured by the double descent curve. Understanding the nature of this relationship is key to designing more efficient and effective neural network models, towards the end goal of downscaling these networks and facilitating their democratization and adoption.

4.2 The *ease* of training

Overparametrization in neural networks is a crucial concern due to its seemingly paradoxical relationship with the bias-variance tradeoff and the model's ability to generalize. The crux of this issue lies in the high-dimensional nature of the parameter space in deep neural networks. In such high-dimensional spaces, our intuitive understanding, which is mostly derived from low-dimensional spaces, may not necessarily apply. High-dimensional spaces are notoriously difficult to navigate, both from a computational and a theoretical perspective. To appreciate the mathematical intuition behind the ease of training in overparametrized regimes, we need to study the optimization landscape of neural networks and the behavior of gradient-based optimization methods.

Optimization Landscape: The optimization problem for training a neural network is to minimize a loss function over the network's parameters. This loss function is non-convex due to the non-linear activations used in the network. Therefore, it contains numerous local minima and saddle points, which makes optimization challenging. In low dimensions, we would expect the optimization to get stuck in poor local minima. However, in high-dimensional spaces, the geometry is fundamentally different. Most critical points (where the gradient is zero) are high-dimensional saddle points, where the Hessian has both positive and negative eigenvalues [23]. Recent research has shown that the likelihood of encountering a poor local minimum decreases in higher dimensions, making it easier for the optimization to find good solutions.

Gradient Descent Dynamics: To further understand the ease of training in machine learning, it is crucial to comprehend the actions of gradient-based optimization algorithms, such as Stochastic Gradient Descent (SGD), in high-dimensional spaces. This understanding is particularly vital in an overparameterized situation, where the total number of model parameters surpasses the volume of the dataset. In such an overparameterized setup, the loss function is presented with a more substantial null space. Interestingly, SGD exhibits a natural predisposition towards flatter minima in the loss landscape [15]. These minima are of significance because they relate to solutions that offer more robustness and better generalization, primarily due to their extensive span in the parameter space. A common empirical justification [16] for this observed generalization difference states that adaptive gradient algorithms tend to settle at sharp minima, which are characterized by a large local curvature in their basin. These solutions generally underperform in terms of generalization whereas SGD is more likely to seek flat minima, which subsequently generalizes better. Nevertheless, contrary studies [13] present that deep neural networks can successfully generalize even from minima located in asymmetric basins. These basins are interesting as they possess both steep and flat directions. Even though these minima are locally sharp concerning their curvature, they still manage to generalize well, and SGD frequently converges to these minima. This contradiction indicates that the traditional differentiation of 'flat' and 'sharp' minima, based purely on their curvature, is insufficient to explain these new

findings. As a result, the reason for the observed generalization gap between adaptive gradient methods and SGD remains ambiguous. Even though an increase in the number of parameters might intuitively seem overwhelming, SGD has consistently demonstrated its ability to pinpoint desirable solutions.

Implicit Regularization: Overparametrization can lead to a form of implicit regularization during training. Implicit regularization is the phenomenon where a model or learning algorithm tends to prefer simpler, more structured solutions, even in the absence of explicit regularization terms in the loss function, usually referring to one that has a smaller norm or is sparser. This phenomenon was first observed in linear models and has since been extended to deep networks. Let us consider the case of linear regression, where we have more parameters than data points. Mathematically, we aim to solve the equation

$$Y = Xw$$

for w, where Y is the vector of target values, X is the matrix of input features, and w is the vector of weights. When X is a so called *fat matrix* (i.e., more columns than rows), there are infinitely many solutions that perfectly fit the data. However, when we solve this problem using a gradient-based algorithm like gradient descent, we don't obtain just any solution; we get the minimum norm solution. This is because gradient descent starts from zero (or small random initialization) and moves in the direction of steepest descent to minimize the loss. In doing so, it finds the solution in the smallest possible "L2" ball that touches the manifold of solutions. This is a form of implicit regularization: even though we did not explicitly ask for the solution with the smallest "L2" norm, gradient descent provides us that [26]. This phenomenon of implicit regularization extends to deep networks as well. For example [25] demonstrates that deep neural networks can fit random labels, implying they are not regularizing in the traditional sense (by restricting the hypothesis space). Instead, they suggest that the type of solutions found by the networks is influenced by the inductive bias of the learning algorithm (in this case SGD), which can be seen as a form of implicit regularization. In summary, overparametrization in machine learning models, from simple linear models to complex deep networks, can lead to a form of implicit regularization, where the learning algorithm tends to prefer simpler solutions. While the specific form and effects of this implicit regularization are still being studied, it is an essential factor in understanding why overparametrized models generalize well despite their capacity to fit even random labels. The exact form of this implicit regularization is not fully understood but is believed to be linked to the dynamics of SGD and the structure of the loss landscape.

From a computational perspective, the high-dimensionality of the problem implies a vast number of parameters to optimize, posing significant computational challenges. This can make it difficult to find a suitable minimum of the loss function, which is required for the model to perform well. To make things even worse, the theoretical perspective is even more challenging. In high-dimensional spaces, the geometry and topology can become counter-intuitive. For example, in the so called *Curse of Dimensionality* [2], most of the volume of a high-dimensional sphere is near its surface, not in the center as in the three-dimensional case. Moreover, most of the pairwise distances between points drawn randomly from a high-dimensional distribution are nearly equal. These factors significantly complicate the analysis and understanding of overparametrized models.

The high dimensionality of the problem also influences the type of analysis that can be carried out. Most of our tools for theoretical analysis, such as convex optimization and regularization theory, do not directly apply in high-dimensional non-convex problems, as encountered in deep learning. This necessitates the development of new theoretical tools, possibly inspired by fields like statistical physics, random matrix theory, and high-dimensional statistics, to better understand the behavior of overparametrized models.

5 Experimental results - Optimizing overparametrized networks

In this chapter, we will focus on drawing correlations between the aforementioned Double Descent phenomena and the ease of training in overparametrized networks.

5.1 Experimental setting

We will follow the procedure proposed by Goodfellow et al. (2015)[11]: this method, often referred to as a "linear interpolation" or "line search" approach, involves interpolating between two parameter sets of a neural network model optimized through Stochastic Gradient Descent (SGD) and evaluating the objective function at various points along this interpolation line. The idea is to investigate the geometry of the loss landscape, which can provide insights into the nature of the optimization problem.
Here's a detailed breakdown of how the technique works:

Define the start and end points: The parameters at the start and end points of the line are selected. The sets we chose to carry out this experiment are the Glorot initialized parameters (θ_0) and the final parameters (θ_f) after training the network.

Create an interpolation line: We then generate a series of interpolated parameters (θ) along the line between the start and end points using the linear combination between the start and end point such that

$$\theta = (1 - \alpha)\theta_0 + \alpha\theta_f \tag{5.1}$$

The α here is a coefficient that ranges between 0 and 1. When α is 0, $\theta = \theta_0$, and when α is 1, $\theta = \theta_f$.

Evaluate the objective function: With these interpolated parameters, we evaluate the objective function $J(\theta)$. This involves forwarding inputs through the neural network using these parameters and computing the loss. The loss represents how well the network performs with these parameters.

Repeat the process: We then vary the α from 0 to 1 in small increments to sample many points

along the interpolation line. For each α, we calculate a corresponding θ and evaluate $J(\theta)$.

The shape of the graph can give insights into the behavior of the optimization process. For instance, if the graph is nearly convex and has a single global minimum, this would indicate that the optimization problem is relatively "well-behaved", and that knowing the correct direction to move in parameter space would lead to an optimal solution. This technique is a simplification and makes some key assumptions:

Assumption of Linearity: The most prominent assumption is that the trajectory between two points in the parameter space can be approximated by a straight line. In reality, the path that an optimization algorithm like stochastic gradient descent (SGD) takes through parameter space is highly non-linear and can be affected by factors like the learning rate, batch size, choice of optimizer, and so on.

Assumption of Smoothness: The technique assumes that the loss landscape is smooth and continuous. In practice, while neural network loss landscapes are often observed to be surprisingly smooth, there may exist regions of non-smoothness caused by factors such as the use of non-differentiable activation functions (e.g., ReLU).

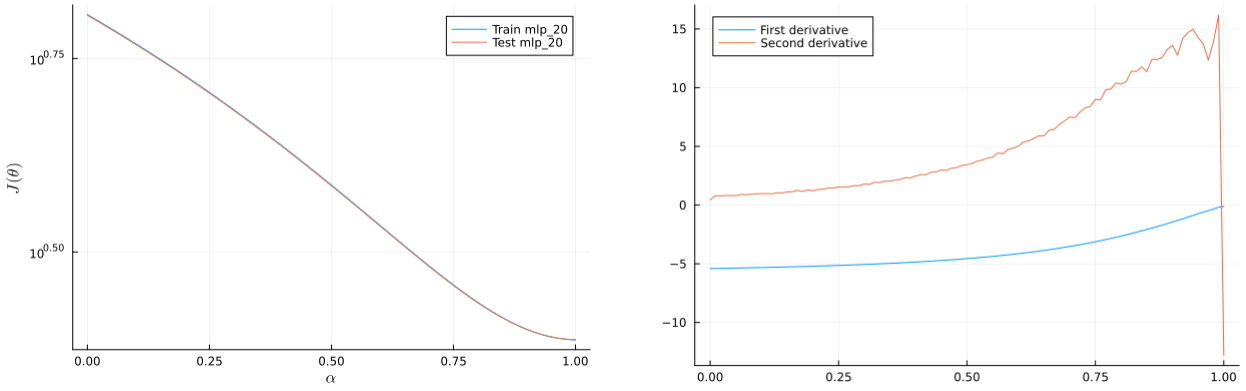
Assumption of Convexity: If the plot derived from the line search is interpreted as being indicative of the overall loss landscape, it might give the misleading impression that the optimization landscape is convex or near-convex. Neural network optimization problems are known to be non-convex, and the line search merely gives a 'slice' of the landscape which may appear convex due to the high dimensionality and potential for numerous local minima being 'averaged out' in the projection.

Assumption of Low Dimensionality: The technique inherently reduces the very high-dimensional parameter space of neural networks (which can often be in the millions or billions) to a one-dimensional line. This is a significant simplification and may not capture important aspects of the loss landscape in other directions.

Realistically, the optimization process does not just proceed along a straight line in the high-dimensional parameter space and the optimization path can be far more complex due to the non-linear nature of neural networks and their loss landscapes. Several other approaches to tackle this issues have been proposed [20], however, by simplifying the path to a straight line, this method provides a useful and understandable insight into the properties of the optimization process and gives us an idea of what the loss landscape could look loke in a very specific submanifold.
We repeated the process above among the three critical regimes defined in 2: the underparametrized regime, the interpolation threshold and the overparametrized regime. The setting for the training is the same one mentioned in 3, with the caveat that we now save the Glorot initialized set of parameters θ_0 and the final set of parameters θ_f to carry out the interpolation and the evaluation of the model with the interpolated weights.

5.2 Experimental results

5.2.1 Underparametrized regime

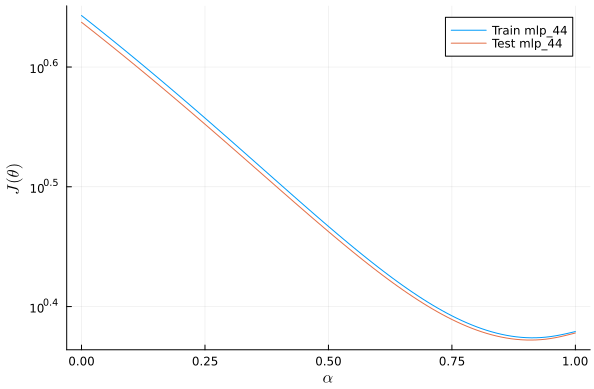


(a) Linear interpolation

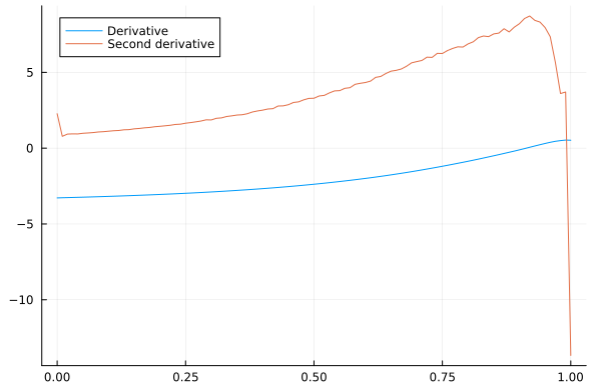
(b) First and Second derivatives of $J(\theta)$

Figure 5.1: Linear interpolation analysis for an underparametrized (20 hidden nodes) MLP. The function $J(\theta)$ hits a minimum at $\alpha = 1$, that is, the loss is minimized at the the fully trained set of parameters. The second derivative of the test function, representing the concavity, is increasing in magnitude up until the neighbors of 1, where it starts slowing down. The function is indeed convex and no bumps (or rather *barriers*) are appearing in the path.

5.2.2 At the interpolation threshold



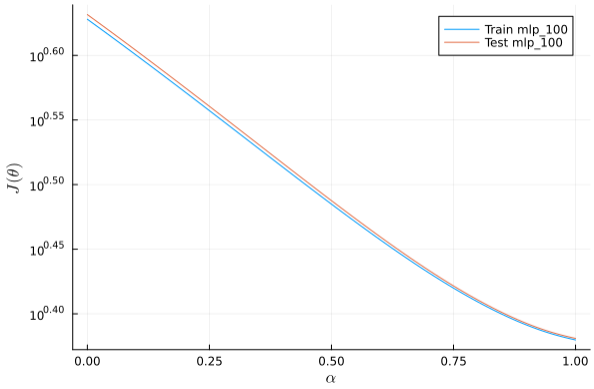
(a) Linear interpolation



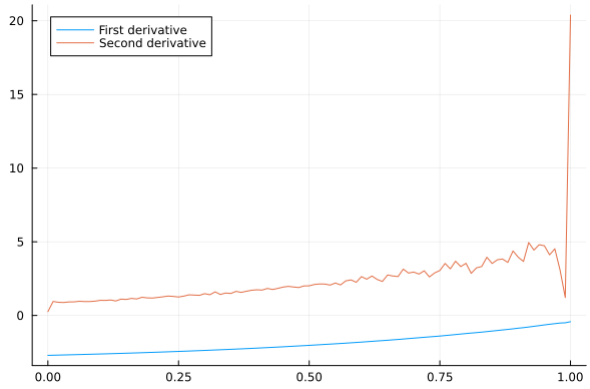
(b) First and Second derivatives of $J(\theta)$

Figure 5.2: Linear interpolation analysis for an MLP around the interpolation threshold (44 hidden nodes). The function $J(\theta)$ hits a minimum before $\alpha = 1$: that is, the loss is not anymore minimized at the the fully trained set of parameters, but prior to that, indicating how the model does not perfectly generalize the training data. The second derivative of the test function, representing the concavity, is increasing in magnitude up until $\alpha \sim 0.8$, where it starts plummeting. It is worth noting how the sign of the first derivative changes when approaching the SGD attained solution. Therefore, methods such as early stopping would have to be leveraged to reach solutions that have lower loss and thus generalize better.

5.2.3 Overparametrized regime



(a) Linear interpolation



(b) First and Second derivatives of $J(\theta)$

Figure 5.3: Linear interpolation analysis for an overparametrized (100 hidden nodes) MLP. The function $J(\theta)$, in similar fashion to the underparametrized regime, hits a minimum at $\alpha = 1$: that is, the loss is minimized at the the fully trained set of parameters. Interestingly, the second derivative of the test function indicates also that the linesearch approach is much smoother than what happens in the underparametrized regime.

6 Conclusion - Demystifying the Paradox of Overparametrization

6.1 Revisiting the Overparametrization Paradox

Throughout this thesis, we have unpacked the paradoxical nature of overparametrization in neural networks. Conventional wisdom would dictate that overparametrization, wherein the number of model parameters surpasses the number of training data points, should lead to overfitting. We expect that the model would memorize the training data to the detriment of its generalization ability. However, we have seen that this is not the case. Instead, empirical studies and theoretical analyses have shown that overparametrized models often generalize well and can be easier to train.

6.2 Bridging the Gap: The Role of the Optimization Landscape and Gradient Descent Dynamics

The paradoxical behavior of overparametrized models becomes less mysterious when we consider the optimization landscape and the dynamics of gradient descent in high-dimensional spaces. We explained how the non-convex optimization landscape in neural network training has numerous local minima and saddle points. However, as we move into higher dimensions, most critical points tend to be saddle points rather than local minima. This shift in the nature of critical points in high-dimensional spaces increases the likelihood of gradient descent finding a good solution.

Gradient descent dynamics further reveal why overparametrized models are easier to train. We examined how SGD exhibits a natural predisposition towards flatter minima in the loss landscape, which have been linked to better generalization. Although there are contradictions in the literature regarding the generalization difference between 'flat' and 'sharp' minima, it is evident that SGD, even in an overparametrized setup, is capable of seeking desirable solutions.

6.3 The Unseen Regulator: Implicit Regularization in Overparametrized Models

To further unravel the mystery of overparametrization, we dived into the notion of implicit regularization. Even without explicit regularization terms in the loss function, gradient descent provides us the minimum norm solution. This phenomena has been observed in both simple linear models and complex deep networks. The learning algorithm's inductive bias acts as a form of implicit regularization, leading to simpler solutions that generalize well. This adds another layer to our understanding of why overparametrized models defy traditional expectations.

Through our journey in this thesis, we have seen that the paradox of overparametrization is not so paradoxical after all. By examining the optimization landscape, the dynamics of gradient descent, and the role of implicit regularization, we have revealed the intricate mechanisms behind the surprising behavior of overparametrized models. The results suggest a complex interplay between model architecture, optimization methods, and inherent biases of the learning algorithm that drive the generalization performance of overparametrized models.

6.4 Future Directions and Applications

Despite these advances in our understanding, several aspects of overparametrization remain shrouded in mystery. For instance, although we have linked the ease of training to high-dimensional spaces and gradient descent dynamics, a concrete, mathematical description of this phenomenon is still elusive. Similarly, while we have observed the effect of implicit regularization in both simple linear models and complex deep networks, our understanding of this phenomenon is largely heuristic, relying on empirical observations rather than theoretical grounding.

One of the primary limitations of our current understanding is that it is centered on average case scenarios and broad characterizations. However, practical scenarios may deviate significantly from these average cases, and the complexity of the model, data, and training regime can dramatically alter the outcome. A future direction for research would be to move beyond average-case analysis and towards a more detailed understanding of the specific scenarios under which overparametrization is beneficial or detrimental.

Furthermore, while the phenomenon of implicit regularization is fascinating, we still lack a complete, formal characterization of this process. We understand that gradient descent introduces a bias towards simpler models, but the nature and extent of this simplicity remain unclear. Unraveling this could not only enhance our understanding of overparametrization but could also lead to novel regularization techniques and learning algorithms.

The high-dimensional nature of overparametrization presents a significant computational challenge. While we have a qualitative understanding that higher dimensions are "easier" for optimization, quantifying this ease and leveraging it for efficient computation is an open problem. This forms an interesting avenue for future research: can we design algorithms that effectively exploit the high-dimensionality for efficient optimization?

In conclusion, while overparametrization is a fascinating and complex topic that challenges our traditional understanding of learning, much remains to be explored. As we move forward, the hope is that a deeper understanding of these issues will enable us to build more robust, efficient, and understandable machine learning models, advancing the field and its countless applications.

Bibliography

[1] Mikhail Belkin et al. "Reconciling modern machine-learning practice and the classical bias–variance trade-off". In: *Proceedings of the National Academy of Sciences* 116.32 (July 2019), pp. 15849–15854. DOI: 10.1073/pnas.1903070116.

[2] Richard Bellman. "Dynamic programming and stochastic control processes". In: *Information and Control* 1.3 (1958), pp. 228–239. ISSN: 0019-9958. DOI: https://doi.org/10.1016/S0019-9958(58)80003-0.

[3] Jeff Bezanson et al. "Julia: A fresh approach to numerical computing". In: *SIAM review* 59.1 (2017), pp. 65–98.

[4] Avrim Blum and Ronald Rivest. "Training a 3-Node Neural Network is NP-Complete". In: *Advances in Neural Information Processing Systems*. Ed. by D. Touretzky. Vol. 1. Morgan-Kaufmann, 1988.

[5] Léon Bottou, Frank E. Curtis, and Jorge Nocedal. *Optimization Methods for Large-Scale Machine Learning*. 2018.

[6] Alfredo Canziani, Adam Paszke, and Eugenio Culurciello. *An Analysis of Deep Neural Network Models for Practical Applications*. 2017.

[7] Pratik Chaudhari et al. *Entropy-SGD: Biasing Gradient Descent Into Wide Valleys*. 2017.

[8] Lenaic Chizat and Francis Bach. *On the Global Convergence of Gradient Descent for Over-parameterized Models using Optimal Transport*. 2018.

[9] Joel Friedman. *Energy in cubic splines, Power Series as algorithms, and the Initial Value problem*. University of British Columbia, Vancouver, 2020.

[10] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Ed. by Yee Whye Teh and Mike Titterington. Vol. 9. Proceedings of Machine Learning Research. Chia Laguna Resort, Sardinia, Italy: PMLR, 13–15 May 2010, pp. 249–256.

[11] Ian J. Goodfellow, Oriol Vinyals, and Andrew M. Saxe. *Qualitatively characterizing neural network optimization problems*. 2015.

[12] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. New York, NY, USA: Springer New York Inc., 2001.

[13] Haowei He, Gao Huang, and Yang Yuan. *Asymmetric Valleys: Beyond Sharp and Flat Local Minima*. 2019.

[14] Jared Kaplan et al. "Scaling Laws for Neural Language Models". In: *CoRR* abs/2001.08361 (2020).

[15] Nitish Shirish Keskar and Richard Socher. *Improving Generalization Performance by Switching from Adam to SGD*. 2017.

[16] Nitish Shirish Keskar et al. "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima". In: *CoRR* abs/1609.04836 (2016).

[17] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep learning". In: *Nature* 521.7553 (May 2015), pp. 436–444. DOI: 10.1038/nature14539.

[18] Yann LeCun and Corinna Cortes. "The MNIST database of handwritten digits". In: 2005.

[19] Dmitry Lepikhin et al. "GShard: Scaling Giant Models with Conditional Computation and Automatic Sharding". In: *CoRR* abs/2006.16668 (2020).

[20] Hao Li et al. *Visualizing the Loss Landscape of Neural Nets*. 2018.

[21] Preetum Nakkiran et al. *Deep Double Descent: Where Bigger Models and More Data Hurt*. 2019.

[22] Behnam Neyshabur et al. *Towards Understanding the Role of Over-Parametrization in Generalization of Neural Networks*. 2018.

[23] Levent Sagun, Leon Bottou, and Yann LeCun. *Eigenvalues of the Hessian in Deep Learning: Singularity and Beyond*. 2017.

[24] Ilya Sutskever et al. "On the importance of initialization and momentum in deep learning". In: *Proceedings of the 30th International Conference on Machine Learning*. Ed. by Sanjoy Dasgupta and David McAllester. Vol. 28. Proceedings of Machine Learning Research 3. Atlanta, Georgia, USA: PMLR, 17–19 Jun 2013, pp. 1139–1147.

[25] Chiyuan Zhang et al. *Understanding deep learning requires rethinking generalization*. 2017.

[26] Hui Zou and Trevor Hastie. "Regularization and Variable Selection via the Elastic Net". In: *Journal of the Royal Statistical Society. Series B (Statistical Methodology)* 67.2 (2005), pp. 301–320. ISSN: 13697412, 14679868.