Fit without fear: remarkable mathematical phenomena of deep learning through the prism of interpolation

Mikhail Belkin
Halıcıoğlu Data Science Institute, University of California San Diego, 10100 Hopkins Drive, La Jolla, CA 92093, USA
E-mail: mbelkin@ucsd.edu

In memory of Partha Niyogi, a thinker, a teacher, and a dear friend.

In the past decade the mathematical theory of machine learning has lagged far behind the triumphs of deep neural networks on practical challenges. However, the gap between theory and practice is gradually starting to close. In this paper I will attempt to assemble some pieces of the remarkable and still incomplete mathematical mosaic emerging from the efforts to understand the foundations of deep learning. The two key themes will be interpolation and its sibling over-parametrization. Interpolation corresponds to fitting data, even noisy data, exactly. Over-parametrization enables interpolation and provides flexibility to select a suitable interpolating model.

As we will see, just as a physical prism separates colours mixed within a ray of light, the figurative prism of interpolation helps to disentangle generalization and optimization properties within the complex picture of modern machine learning. This article is written in the belief and hope that clearer understanding of these issues will bring us a step closer towards a general theory of deep learning and machine learning.

CONTENTS

1 Preface 204
2 Introduction 204
3 The problem of generalization 206
4 The wonders of optimization 228
5 Odds and ends 235
6 Summary and thoughts 238
References 242

© The Author(s), 2021. Published by Cambridge University Press.
1. Preface

In recent years we have witnessed triumphs of machine learning in practical challenges from machine translation to playing chess to protein folding. These successes rely on advances in designing and training complex neural network architectures and on availability of extensive datasets. However, while it is easy to be optimistic about the potential of deep learning for our technology and science, we may still underestimate the power of fundamental mathematical and scientific principles that can be learned from its empirical successes.

In what follows, I will attempt to assemble some pieces of the remarkable mathematical mosaic that is starting to emerge from the practice of deep learning. This is an effort to capture parts of an evolving and still elusive picture with many of the key pieces still missing. The discussion will be largely informal, aiming to build mathematical concepts and intuitions around empirically observed phenomena. Given the fluid state of the subject and our incomplete understanding, it is necessarily a subjective, somewhat impressionistic and, to a degree, conjectural view, reflecting my understanding and perspective. It should not be taken as a definitive description of the subject as it stands now. Instead, it is written with the aspiration of informing and intriguing a mathematically minded reader and encouraging deeper and more detailed research.

2. Introduction

In the past decade theoretical machine learning has faced a crisis. Deep learning, based on training complex neural architectures, has become state of the art for many practical problems, from computer vision to playing the game of Go to natural language processing and even for basic scientific problems, such as, recently, predicting protein folding (Senior et al. 2020). Yet the mathematical theory of statistical learning extensively developed in the 1990s and 2000s struggled to provide a convincing explanation for its successes, let alone help in designing new algorithms or provide guidance in improving neural architectures. This disconnect resulted in significant tensions between theory and practice. The practice of machine learning was compared to ‘alchemy’, a pre-scientific pursuit, proceeding by pure practical intuition and lacking firm foundations (Rahimi and Recht 2017). On the other hand, a counter-charge of practical irrelevance, ‘looking for lost keys under a lamp post, because that’s where the light is’ (LeCun 2019) was levelled against the mathematical theory of learning.

In what follows, I will start by outlining some of the reasons why classical theory failed to account for the practice of ‘modern’ machine learning. I will proceed to discuss an emerging mathematical understanding of the observed phenomena, an understanding which points toward a reconciliation between theory and practice.

The key themes of this discussion are based on the notions of interpolation and over-parametrization, and the idea of a separation between the two regimes.
'Classical' under-parametrized regimes. The classical setting can be characterized by limited model complexity, which does not allow arbitrary data to be fitted exactly. The goal is to understand the properties of the (typically unique) classifier with the smallest loss. The standard tools include uniform laws of large numbers resulting in ‘what you see is what you get’ (WYSIWYG) bounds, where the fit of classifiers on the training data is predictive of their generalization to unseen data. Non-convex optimization problems encountered in this setting typically have multiple isolated local minima, and the optimization landscape is locally convex around each minimum.

‘Modern’ over-parametrized regimes. Over-parametrized setting deals with rich model classes, where there are generically manifolds of potential interpolating predictors that fit the data exactly. As we will discuss, some but not all of those predictors exhibit strong generalization to unseen data. Thus, the statistical question is understanding the nature of the inductive bias – the properties that make some solutions preferable to others despite all of them fitting the training data equally well. In interpolating regimes, non-linear optimization problems generically have manifolds of global minima. Optimization is always non-convex, even locally, yet it can often be shown to satisfy the so-called Polyak–Łojasiewicz (PL) condition guaranteeing convergence of gradient-based optimization methods.

As we will see, interpolation, the idea of fitting the training data exactly, and its sibling over-parametrization, having sufficiently many parameters to satisfy the constraints corresponding to fitting the data, taken together provide a perspective on some of the more surprising aspects of neural networks and other inferential problems. It is interesting to point out that interpolating noisy data is a deeply uncomfortable and counter-intuitive concept for statistics, both theoretical and applied, as it is traditionally concerned with over-fitting the data. For example, in their book on non-parametric statistics, Györfi, Kohler, Krzyzak and Walk (2002, page 21) dismissed a certain procedure on the grounds that it ‘may lead to a function which interpolates the data and hence is not a reasonable estimate.’ Similarly, the popular reference by Hastie, Tibshirani and Friedman (2001, page 194) suggests that ‘a model with zero training error is overfit to the training data and will typically generalize poorly.’

Likewise, over-parametrization is alien to optimization theory, which is traditionally more interested in convex problems with unique solutions or non-convex problems with locally unique solutions. In contrast, as we discuss in Section 4, over-parametrized optimization problems are in essence never convex, nor do they have unique solutions, even locally. Instead, the solution chosen by the algorithm depends on the specifics of the optimization process.

To avoid confusion, it is important to emphasize that interpolation is not necessary for good generalization. In certain models (e.g. Hastie, Montanari, Rosset and Tibshirani 2019), introducing some regularization is provably preferable to fitting the data exactly. In practice, early stopping is typically used for training neural
networks. It prevents the optimization process from full convergence and acts as a type of regularization (Yao, Rosasco and Caponnetto 2007). What is remarkable is that interpolating predictors often provide strong generalization performance, comparable to the best possible predictors. Furthermore, the best practice of modern deep learning is arguably much closer to interpolation than to the classical regimes (when training and testing losses match). For example, Ruslan Salakhutdinov (2017), in his tutorial on deep learning, stated that ‘The best way to solve the problem from practical standpoint is you build a very big system . . . basically you want to make sure you hit the zero training error.’ While more tuning is typically needed for best performance, these ‘overfitted’ systems already work well (Zhang et al. 2017). Indeed, it appears that the largest technologically feasible networks are consistently preferable for best performance. For example, in 2016 the largest neural networks had fewer than $10^9$ trainable parameters (Canziani, Paszke and Culurciello 2016), whereas the current (2021) state-of-the-art switch transformers (Fedus, Zoph and Shazeer 2021) have over $10^{12}$ weights – more than three orders of magnitude growth in under five years!

Just as a literal physical prism separates colours mixed within a ray of light, the figurative prism of interpolation helps to disentangle a blend of properties within the complex picture of modern machine learning. While significant parts are still hazy or missing and precise analyses are only being developed, many important pieces are starting to fall in place.

3. The problem of generalization

3.1. The setting of statistical learning

The simplest problem of supervised machine learning is that of classification. To construct a clichéd ‘cat versus dog’ image classifier, we are given data

$\{ (x_i, y_i), x_i \in \mathcal{X} \subset \mathbb{R}^d, y_i \in \{-1, 1\}, i = 1, \ldots, n \}$,

where $x_i$ is the vector of image pixel values and the corresponding label $y_i$ is (arbitrarily) $-1$ for ‘cat’ and $1$ for ‘dog’. The goal of a learning algorithm is to construct a function $f: \mathbb{R}^d \to \{-1, 1\}$ that generalizes to new data, that is, it accurately classifies images unseen in training. Regression, the problem of learning general real-valued predictions, $f: \mathbb{R}^d \to \mathbb{R}$, is formalized similarly.

This, of course, is an ill-posed problem which needs further mathematical elucidation before a solution can be contemplated. The usual statistical assumption is that both training data and future (test) data are independent identically distributed (i.i.d.) samples from a distribution $P$ on $\mathbb{R}^d \times \{-1, 1\}$ (defined on $\mathbb{R}^d \times \mathbb{R}$ for regression). While the i.i.d. assumption has significant limitations, it is the simplest and most illuminating statistical setting, and we will use it exclusively. Thus, from this point of view, the goal of machine learning in classification is simply to find a function, known as the Bayes optimal classifier, that minimizes the expected
Fit without fear

probability of misclassification:

\[ f^* = \arg \min_{f: \mathbb{R}^d \to \mathbb{R}} \mathbb{E}_{P(x,y)} l(f(x), y). \]  \hspace{1cm} (3.1)

Here \( l(f(x), y) = 1_{f(x) \neq y} \) is the Kronecker delta function called the 0–1 loss function. The expected loss of the Bayes optimal classifier \( f^* \) is called the Bayes loss or Bayes risk.

We note that the 0–1 loss function can be problematic due to its discontinuous nature, and it is entirely unsuitable for regression, where the square loss \( l(f(x), y) = (f(x) - y)^2 \) is typically used. For the square loss, the optimal predictor \( f^* \) is called the regression function.

3.2. The framework of empirical and structural risk minimization

While obtaining the optimal \( f^* \) may be the ultimate goal of machine learning, it cannot be found directly, as in any realistic setting we lack access to the underlying distribution \( P \). Thus the essential question of machine learning is how \( f^* \) can be approximated given the data. A foundational framework for addressing that question was given by Vapnik (1995) under the name of empirical and structural risk minimization.\(^1\) The first key insight is that the data itself can serve as a proxy for the underlying distribution. Thus, instead of minimizing the true risk \( \mathbb{E}_{P(x,y)} l(f(x), y) \), we can attempt to minimize the empirical risk

\[ \mathcal{R}_{\text{emp}}(f) = \frac{1}{n} \sum_{i=1}^{n} l(f(x_i), y_i). \]

Even in that formulation the problem is still under-defined, as infinitely many different functions minimize the empirical risk. However, it can be made well-posed by restricting the space of candidate functions \( \mathcal{H} \) to make the solution unique. Thus we obtain the following formulation of the empirical risk minimization (ERM):

\[ f_{\text{emp}} = \arg \min_{f \in \mathcal{H}} \mathcal{R}_{\text{emp}}(f). \]

Solving this optimization problem is called ‘training’. Of course, \( f_{\text{emp}} \) is only useful to the degree it approximates \( f^* \). While superficially the predictors \( f^* \) and \( f_{\text{emp}} \) appear to be defined similarly, their mathematical relationship is subtle due, in particular, to the choice of the space \( \mathcal{H} \), the ‘structural part’ of the empirical risk minimization.

According to the discussion in Vapnik (1995), ‘the theory of induction’ based on the structural risk minimization must meet the following two mathematical

\(^1\) While empirical and structural risk optimization are not the same, as we discuss below, both are typically referred to as ERM in the literature.
requirements.

ULLN. The theory of induction is based on the uniform law of large numbers. CC. Effective methods of inference must include capacity control.

A uniform law of large numbers (ULLN) indicates that for any hypothesis in $H$, the loss on the training data is predictive of the expected (future) loss:

$$\text{ULLN: } R(f) = \mathbb{E}_{P(x,y)} l(f(x), y) \approx R_{\text{emp}}(f) \text{ for all } f \in H.$$ 

We generally expect that $R(f) \geq R_{\text{emp}}(f)$, which allows ULNN to be written as a one-sided inequality, typically of the form

$$R(f) - R_{\text{emp}}(f) < O^*(\sqrt{\frac{\text{cap}(H)}{n}}) \text{ for all } f \in H. \quad (3.2)$$

Here $\text{cap}(H)$ is a measure of the capacity of the space $H$, such as its Vapnik–Chervonenkis (VC) dimension or the covering number (see Bousquet, Boucheron and Lugosi 2003), and $O^*$ can contain logarithmic terms and other terms of lower order. Inequality (3.2) holds with high probability over the choice of the data sample.

Inequality (3.2) is a mathematical instantiation of the ULLN condition and directly implies

$$R(f_{\text{emp}}) - \min_{f \in H} R(f) < O^*(\sqrt{\frac{\text{cap}(H)}{n}}).$$

This guarantees that the true risk of $f_{\text{emp}}$ is nearly optimal for any function in $H$, as long as $\text{cap}(H) \ll n$.

The structural condition CC is needed to ensure that $H$ also contains functions that approximate $f^*$. Combining CC and ULLN and applying the triangle inequality yields a guarantee that $R_{\text{emp}}(f_{\text{emp}})$ approximates $R(f^*)$ and the goal of generalization is achieved.

It is important to point out that the properties ULLN and CC are in tension to each other. If the class $H$ is too small, no $f \in H$ will generally be able to adequately approximate $f^*$. In contrast, if $H$ is too large, so that $\text{cap}(H)$ is comparable to $n$, the capacity term is large and there is no guarantee that $R_{\text{emp}}(f_{\text{emp}})$ will be close to the expected risk $R(f_{\text{emp}})$. In that case the bound becomes tautological (such as the trivial bound that the classification risk is bounded by 1 from above).

Hence the prescriptive aspect of structural risk minimization according to Vapnik is to enlarge $H$ until we find the sweet spot, a point where the empirical risk and the capacity term are balanced. This is represented by Figure 3.1.

---

2 This is the most representative bound, rates faster and slower than $\sqrt{n}$ are also found in the literature. The exact dependence on $n$ does not change our discussion here.
Figure 3.1. A classical U-shaped generalization curve. The optimal model is found by balancing the empirical risk and the capacity term (cf. Vapnik 1995, Figure 6.2).

This view, closely related to the ‘bias-variance dilemma’ in statistics (Geman, Bienenstock and Doursat 1992), had become the dominant paradigm in supervised machine learning, encouraging a rich and increasingly sophisticated line of mathematical research into uniform laws of large numbers and concentration inequalities.

3.3. Margins theory and data-dependent explanations.

However, even in the 1990s it had become clear that successes of AdaBoost (Freund and Schapire 1997) and neural networks were difficult to explain from the structural risk minimization or bias-variance trade-off paradigms. Leo Breiman (1995), a prominent statistician, posed the question ‘Why don’t heavily parametrized neural networks overfit the data?’ In particular, it was observed that increasing complexity of classifiers (capacity of \( \mathcal{H} \)) in boosting did not necessarily lead to the expected drop in performance due to over-fitting. Why did the powerful mathematical formalism of uniform laws of large numbers fail to explain the observed evidence?

An elegant explanation known as the margins theory was proposed by Schapire, Freund, Bartlett and Lee (1998). It is based on a more careful examination of the bound in (3.2), which identifies a serious underlying issue. We observe that the bound applies to any function \( f \in \mathcal{H} \). However, in the learning context, we are not concerned with all functions, only with those that are plausible predictors. Indeed, it is a priori clear that the vast majority of predictors in standard function classes (e.g. linear functions) are terrible predictors with performance no better than chance. Whether their empirical risk matches the true risk may be of importance to the theory of empirical processes or to functional analysis, but it is of little concern

\[ \text{Figure 3.1. A classical U-shaped generalization curve. The optimal model is found by balancing the empirical risk and the capacity term (cf. Vapnik 1995, Figure 6.2).} \]
to a ‘theory of induction’. The plausible candidate functions, those that are in an appropriate sense close to \( f^* \), form a much narrower subset of \( \mathcal{H} \). Of course, ‘closeness’ needs to be carefully defined to be empirically observable without the exact prior knowledge of \( f^* \).

To give an important special case, suppose we believe that our data are separable, so that \( \mathcal{R}(f^*) = 0 \). We can then concentrate our analysis on the subset of the hypothesis set \( \mathcal{H} \) with small empirical loss

\[
\mathcal{H}_\epsilon = \{ f \in \mathcal{H} : \mathcal{R}_{\text{emp}}(f) \leq \epsilon \}.
\]

Indeed, since \( \mathcal{R}(f^*) = 0 \), \( \mathcal{R}_{\text{emp}}(f^*) = 0 \) and hence \( f^* \in \mathcal{H}_\epsilon \).

The capacity \( \text{cap}(\mathcal{H}_\epsilon) \) will generally be far smaller than \( \text{cap}(\mathcal{H}) \) and we thus hope for a tighter bound. It is important to note that the capacity \( \text{cap}(\mathcal{H}_\epsilon) \) is a data-dependent quantity as \( \mathcal{H}_\epsilon \) is defined in terms of the training data. Thus we aim to replace (3.2) with a data-dependent bound:

\[
\mathcal{R}(f) - \mathcal{R}_{\text{emp}}(f) < O^*\left( \sqrt{\text{cap}(\mathcal{H}, X)} / n \right) \quad \text{for all } f \in \mathcal{H}, \tag{3.3}
\]

where class capacity \( \text{cap}(\mathcal{H}, X) \) depends on both the hypothesis class \( \mathcal{H} \) and the training data \( X \).

This important insight underlies the margins theory (Schapire et al. 1998), introduced specifically to address the apparent lack of over-fitting in boosting. The idea of data-dependent margin bounds has led to a line of increasingly sophisticated mathematical work on understanding data-dependent function space complexity with notions such as Rademacher complexity (Bartlett and Mendelson 2002). However, we note that as an explanation for the effectiveness of AdaBoost, the margins theory had not been universally accepted (see Buja, Mease and Wyner 2007 for an interesting discussion).

3.4. What you see is not what you get

It is important to note that the generalization bounds mentioned above, even the data-dependent bounds such as (3.3), are ‘what you see is what you get’ (WYSIWYG): the empirical risk that you see in training approximates and bounds the true risk that you expect on unseen data, with the capacity term providing an upper bound on the difference between expected and empirical risk.

However, it had gradually become clear (e.g. Neyshabur, Tomioka and Srebro 2015) that in modern machine learning, training risk and the true risk were often dramatically different and lacked any obvious connection. In an influential paper, Zhang et al. (2017) demonstrated empirical evidence showing that neural networks trained to have zero classification risk in training do not suffer from significant overfitting. Zhang et al. argued that these and similar observations are incompatible with the existing learning theory and ‘require rethinking generalization’. Yet their argument does not fully rule out explanations based on data-dependent bounds.
such as those in Schapire et al. (1998), which can produce non-trivial bounds for interpolating predictors if the true Bayes risk is also small.

A further empirical analysis in Belkin, Ma and Mandal (2018b) made such explanations implausible, if not outright impossible. The experiments used a popular class of algorithms known as kernel machines, which are mathematically predictors of the form

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x, x_i), \quad \alpha_i \in \mathbb{R}. \quad (3.4)$$

Here $K(x, z)$ is a positive definite kernel function (see e.g. Wendland 2004 for a review), such as the commonly used Gaussian kernel $K(x, z) = e^{-\|x-z\|^2/2}$ or the Laplace kernel $K(x, z) = e^{-\|x-z\|}$. It turns out that there is a unique predictor $f_{\text{ker}}$ of that form which interpolates the data:

$$f_{\text{ker}}(x_i) = y_i \quad \text{for all } i = 1, \ldots, n.$$ 

The coefficients $\alpha_i$ can be found analytically by matrix inversion $\alpha = K^{-1}y$. Here $K$ is the kernel matrix $K_{ij} = K(x_i, x_j)$ and $y$ is the vector containing the labels $y_i$.

Now consider a probability distribution $P$, ‘corrupted’ by label noise. Specifically (for a two-class problem) with probability $q$ the label for any $x$ is assigned from $\{-1, 1\}$ with equal probability, and with probability $1-q$ it is chosen according to the original distribution $P$. Note that $P_q$ can be easily constructed synthetically by randomizing the labels on the $q$ fraction of the training and test sets respectively.

It can be seen that the Bayes optimal classifier for the corrupted distribution $P_q$ coincides with the Bayes optimal $f_P^*$ for the original distribution:

$$f_{P_q}^* = f_P^*.$$
Furthermore, it is easy to check that the 0–1 loss of the Bayes optimal predictor \( f^*_P \) computed with respect to \( P_q \) (denoted by \( R_{P_q} \)) is bounded from below by the noise level:

\[
R_{P_q}(f^*_P) \geq \frac{q}{2}.
\]

Belkin et al. (2018b) showed empirically that interpolating kernel machines \( f_{\text{ker},q} \) (see (3.4)) with common Laplace and Gaussian kernels, trained to interpolate \( q \)-corrupted data, generalizes nearly optimally (approaches the Bayes risk) to the similarly corrupted test data. An example is shown in Figure 3.2.\(^4\) In particular, we see that the Laplace kernel tracks the optimal Bayes error very closely, even when as much as 80% of the data are corrupted (i.e. \( q = 0.8 \)).

Why is it surprising from the WYSIWYG bound point of view? For simplicity, suppose \( P \) is deterministic \( (R(f^*_P) = 0) \), which is essentially the case in Figure 3.2(b). In that case (for a two-class problem), \( R_{P_q}(f^*_P) = q/2 \). Hence we observe that

\[
R_{P_q}(f_{\text{ker},q}) \geq R_{P_q}(f^*_P) = \frac{q}{2}.
\]

On the other hand \( R_{\text{emp}}(f_{\text{ker},q}) = 0 \), and hence for the left-hand side of (3.3) we have

\[
R_{P_q}(f_{\text{ker},q}) - R_{\text{emp}}(f_{\text{ker},q}) = R_{P_q}(f_{\text{ker},q}) \geq \frac{q}{2}.
\]

To explain good empirical performance of \( f_{\text{ker},q} \), a bound like (3.3) needs to be both correct and non-trivial. Since the left-hand side is at least \( q/2 \) and observing that \( R_{P_q}(f_{\text{ker},q}) \) is upper-bounded by the loss of a random guess, which is 1/2 for a two-class problem, we must have

\[
\frac{q}{2} \leq O^* \left( \sqrt{\frac{\text{cap}(\mathcal{H}, X)}{n}} \right) \leq \frac{1}{2}.
\]

Note that such a bound would require the multiplicative coefficient in \( O^* \) to be tight within a multiplicative factor \( 1/q \) (which is 1.25 for \( q = 0.8 \)). No such general bounds are known. In fact, typical bounds include logarithmic factors and other multipliers making really tight estimates impossible. More conceptually, it is hard to see how such a bound can exist, as the capacity term would need to ‘magically’ know\(^5\) about the level of noise \( q \) in the probability distribution. Indeed, Nagarajan

\(^4\) For a ten-class problem in Figure 3.2(b), which makes the point even stronger. For simplicity, we only discuss a two-class analysis here.

\(^5\) This applies to the usual capacity definitions based on norms, covering numbers and similar mathematical objects. In principle, it may be possible to ‘cheat’ by letting capacity depend on complex manipulations with the data, e.g. cross-validation. This requires a different type of analysis (see Negrea, Dziugaite and Roy 2020, Zhou, Sutherland and Srebro 2020 for some recent attempts) and raises the question of what may be considered a useful generalization bound. We leave that discussion for another time.
and Kolter (2019) recently gave a strict mathematical proof of incompatibility of
generalization with uniform bounds under certain specific settings. Bartlett and
Long (2020) consequently proved that no good bounds can exist for a broad range
of models.
Thus we see that strong generalization performance of classifiers that interpolate
noisy data is incompatible with WYSIWYG bounds, independently of the nature
of the capacity term.

3.5. Giving up on WYSIWYG, keeping theoretical guarantees

So can we provide statistical guarantees for classifiers that interpolate noisy data?
Until very recently there had not been many. In fact the only common interpolating
algorithm with statistical guarantees for noisy data is the well-known 1-NN rule.\(^6\)
Below we will go over a sequence of three progressively more statistically powerful
nearest neighbour-like interpolating predictors, starting with the classical 1-NN
rule, and going to simplicial interpolation and then to general weighted nearest
neighbour/Nadaraya–Watson schemes with singular kernels.

3.5.1. The peculiar case of 1-NN

Given an input \(x\), 1-NN(\(x\)) outputs the label for the closest (in Euclidean or another
appropriate distance) training example.

While the 1-NN rule is among the simplest and most classical prediction rules
both for classification and regression, it has several striking aspects which are not
usually emphasized in standard treatments.

- It is an interpolating classifier, \(i.e. \mathcal{R}_{\text{emp}}(1-\text{NN}) = 0\).
- Despite ‘over-fitting’, classical analysis in Cover and Hart (1967) shows that
the classification risk of \(\mathcal{R}(1-\text{NN})\) is (asymptotically as \(n \to \infty\)) bounded
from above by \(2 \cdot \mathcal{R}(f^*)\), where \(f^*\) is the Bayes optimal classifier defined
by (3.1).
- Not surprisingly, given that it is an interpolating classifier, there is no ERM-
style analysis of 1-NN.

It seems plausible that the remarkable interpolating nature of 1-NN had been written
off by the statistical learning community as an aberration due to its high excess
risk.\(^7\) As we have seen, the risk of 1-NN can be a factor of two worse than the
risk of the optimal classifier. The standard prescription for improving performance

---

\(^6\) In the last two or three years there has been significant progress on interpolating guarantees
for classical algorithms such as linear regression and kernel methods (see discussion and refer-
ences below). However, traditional analyses nearly always used regularization, which precludes
interpolation.

\(^7\) Recall that the excess risk of a classifier \(f\) is the difference between the risk of the classifier and
the risk of the optimal predictor \(\mathcal{R}(f) - \mathcal{R}(f^*)\).
is to use $k$-NN, an average of $k$ nearest neighbours, which no longer interpolates. As $k$ increases (assuming $n$ is large enough), the excess risk decreases, as does the difference between the empirical and expected risks. Thus, for large $k$ (but still much smaller than $n$) we have, seemingly in line with the standard ERM-type bounds,

$$R_{\text{emp}}(k\text{-NN}) \approx R(k\text{-NN}) \approx R(f^*) .$$

It is perhaps ironic that an outlier feature of the 1-NN rule, shared with no other common methods in the classical statistics literature (except for the relatively unknown work by Devroye, Györfi and Krzyżak 1998), may be one of the cues to understanding modern deep learning.

### 3.5.2. Geometry of simplicial interpolation and the blessing of dimensionality

However, a modification of 1-NN different from $k$-NN maintains its interpolating property while achieving near-optimal excess risk, at least when the dimension is high. The algorithm is simplicial interpolation (Halton 1991) analysed statistically in Belkin, Hsu and Mitra (2018a). Consider a triangulation of the data, $\mathbf{x}_1, \ldots, \mathbf{x}_n$, that is a partition of the convex hull of the data into a set of $d$-dimensional simplices so that:

1. vertices of each simplex are data points,
2. for any data point $\mathbf{x}_i$ and simplex $s$, $\mathbf{x}_i$ is either a vertex of $s$ or does not belong to $s$.

The exact choice of the triangulation turns out to be unimportant as long as the size of each simplex is small enough. This is guaranteed by, for example, the well-known Delaunay triangulation. Given a multi-dimensional triangulation, we define $f_{\text{simp}}(x)$, the simplicial interpolant, to be a function which is linear within each simplex and such that $f_{\text{simp}}(\mathbf{x}_i) = y_i$. It is not hard to check that $f_{\text{simp}}$ exists and is unique.

It is worth noting that in one dimension simplicial interpolation based on the Delaunay triangulation is equivalent to 1-NN for classification. However, when the dimension $d$ is high enough, simplicial interpolation is nearly optimal both for classification and regression. Specifically, it was shown in Belkin et al. (2018a, Theorem 3.4) that simplicial interpolation benefits from a blessing of dimensionality. For large $d$, the excess risk of $f_{\text{simp}}$ decreases with dimension:

$$R(f_{\text{simp}}) - R(f^*) = O\left(\frac{1}{\sqrt{d}}\right).$$

Analogous results hold for regression, where the excess risk is similarly the difference between the loss of a predictor and the loss of the (optimal) regression function. Furthermore, for classification, under additional conditions $\sqrt{d}$ can be replaced by $e^d$ in the denominator.
Why does this happen? How can an interpolating function be nearly optimal despite the fact that it fits noisy data, and why does increasing dimension help?

The key observation is that incorrect predictions are localized in the neighbourhood of ‘noisy’ points, i.e. those points where $y_i = f_{simp}(x_i) \neq f^*(x_i)$. To develop an intuition, consider the following simple example. Suppose that $x_1, \ldots, x_{d+1} \in \mathbb{R}^d$ are vertices of a standard $d$-dimensional simplex $s_d$:

$$x_i = (0, \ldots, 1, \ldots, 0), \quad i = 1, \ldots, d, \quad x_{d+1} = (0, \ldots, 0).$$

Suppose also that the probability distribution is uniform on the simplex (the convex hull of $x_1, \ldots, x_{d+1}$) and the ‘correct’ labels are identically 1. As our training data, we are given $(x_i, y_i)$, where $y_i = 1$, except for the one vertex, which is ‘corrupted by noise’, so that $y_{d+1} = -1$. It is easy to verify that

$$f_{simp}(x) = \text{sign} \left( 2 \sum_{i=1}^{d} (x)_i - 1 \right).$$

We see that $f_{simp}$ coincides with $f^* \equiv 1$ in the simplex except for the set

$$s_{1/2} = \left\{ x : \sum_{i=1}^{d} x_i \leq 1/2 \right\},$$

which is equal to the simplex $\frac{1}{2}s_d$, and thus

$$\text{vol}(s_{1/2}) = \frac{1}{2^d} \text{vol}(s_d).$$

We see that the interpolating predictor $f_{simp}$ is different from the optimal, but the difference is highly localized around the ‘noisy’ vertex, while at most points within $s_d$ their predictions coincide. This is illustrated geometrically in Figure 3.3. The reasons for the blessing of dimensionality also become clear, as small neighbourhoods in high dimensions have smaller volume relative to the total space. Thus there is more freedom and flexibility for the noisy points to be localized.
3.5.3. Optimality of k-NN with singular weighting schemes

While simplicial interpolation improves on 1-NN in terms of the excess loss, it is still not consistent. In high dimensions $f_{\text{simp}}$ is near $f^*$ but does not converge to $f^*$ as $n \to \infty$. Traditionally, consistency and rates of convergence have been a central object of statistical investigation. The first result in this direction is Devroye et al. (1998), which showed statistical consistency of a certain kernel regression scheme, closely related to Shepard’s inverse distance interpolation (Shepard 1968).

It turns out that a similar interpolation scheme based on weighted $k$-NN can be shown to be consistent for both regression and classification and indeed to be optimal in a certain statistical sense; see Belkin et al. (2018a) for convergence rates for regression and classification and the follow-up work by Belkin, Rakhlin and Tsybakov (2019b) for optimal rates for regression. The scheme can be viewed as a type of Nadaraya–Watson predictor (Nadaraya 1964, Watson 1964). It can be described as follows. Let $K(\mathbf{x}, \mathbf{z})$ be a singular kernel, such as

$$K(\mathbf{x}, \mathbf{z}) = \frac{1}{\|\mathbf{x} - \mathbf{z}\|^\alpha}, \quad \alpha > 0,$$

with an appropriate choice of $\alpha$. Consider the weighted nearest neighbour predictor

$$f_{\text{sing}}(\mathbf{x}) = \frac{\sum_{i=1}^{k} K(\mathbf{x}, \mathbf{x}_{(i)}) y_{(i)}}{\sum_{i=1}^{k} K(\mathbf{x}, \mathbf{x}_{(i)})}.$$

Here the sum is taken over the $k$ nearest neighbours of $\mathbf{x}$, $\mathbf{x}_{(1)}, \ldots, \mathbf{x}_{(k)}$. While the kernel $K(\mathbf{x}, \mathbf{x}_{(i)})$ is infinite at $x = \mathbf{x}_{i}$, it is not hard to see that $f_{\text{sing}}(\mathbf{x})$ involves a ratio that can be defined everywhere due to the cancellations between the singularities in the numerator and the denominator. It is, furthermore, a continuous function.
of \( x \). Note that for classification it suffices to simply take the sign of the numerator \( \sum_{i=1}^{k} K(x, x_{(i)})y_{(i)} \) as the denominator is positive.

To better understand how such an unusual scheme can be consistent for regression, consider an example shown in Figure 3.4 for one-dimensional data sampled from a noisy linear model: \( y = x + \epsilon \), where \( \epsilon \) is normally distributed noise. Since the predictor \( f_{\text{sing}}(x) \) fits the noisy data exactly, it is far from optimal on the majority of data points. Yet the prediction is close to optimal for most points in the interval \([0, 1]! \) In general, as \( n \to \infty \), the fraction of those points tends to 1.

We will discuss this phenomenon further in connection with adversarial examples in deep learning in Section 5.2.

### 3.6. Inductive biases and Occam’s razor

The realization that, contrary to deeply ingrained statistical intuitions, fitting noisy training data exactly does not necessarily result in poor generalization, inevitably leads to a quest for a new framework for a ‘theory of induction’, a paradigm not reliant on uniform laws of large numbers and not requiring empirical risk to approximate the true risk.

While, as we have seen, interpolating classifiers can be statistically near-optimal or optimal, the predictors discussed above appear to be different from those widely used in machine learning practice. Simplicial interpolation, weighted nearest neighbour or Nadaraya–Watson schemes do not require training and can be termed direct methods. In contrast, common practical algorithms from linear regression to kernel machines to neural networks are ‘inverse methods’ based on optimization. These algorithms typically rely on algorithmic empirical risk minimization, where a loss function \( R_{\text{emp}}(f_w) \) is minimized via a specific algorithm, such as stochastic gradient descent (SGD) on the weight vector \( w \). Note that there is a crucial and sometimes overlooked difference between the empirical risk minimization as an algorithmic process and Vapnik’s ERM paradigm for generalization, which is algorithm-independent. This distinction becomes important in over-parametrized regimes, where the hypothesis space \( \mathcal{H} \) is rich enough to fit any data set\(^8\) of cardinality \( n \). The key insight is to separate ‘classical’ under-parametrized regimes, where there is typically no \( f \in \mathcal{H} \) such that \( R(f) = 0 \), and ‘modern’ over-parametrized settings, where there is a (typically large) set \( \mathcal{S} \) of predictors that interpolate the training data:

\[
\mathcal{S} = \{ f \in \mathcal{H} : R(f) = 0 \}. \tag{3.6}
\]

First observe that an interpolating learning algorithm \( \mathcal{A} \) selects a specific predictor \( f_{\mathcal{A}} \in \mathcal{S} \). Thus we are faced with the issue of the inductive bias: Why do solutions, such as those obtained by neural networks and kernel machines, generalize, while

---

\(^8\) Assuming that \( x_i \neq x_j \), when \( i \neq j \).
other possible solutions do not? Notice that this question cannot be answered through the training data alone, as any \( f \in S \) fits data equally well. While no conclusive recipe for selecting the optimal \( f \in S \) yet exists, it can be posited that an appropriate notion of functional smoothness plays a key role in that choice. As argued in Belkin, Hsu, Ma and Mandal (2019a), the idea of maximizing functional smoothness subject to interpolating the data represents a very pure form of Occam’s razor (see Vapnik 1995, Blumer, Ehrenfeucht, Haussler and Warmuth 1987).

Usually stated as

*Entities should not be multiplied beyond necessity,*

Occam’s razor implies that the simplest explanation consistent with the evidence should be preferred. In this case fitting the data corresponds to consistency with evidence, while the smoothest function is the ‘simplest’. To summarize, the ‘maximum smoothness’ guiding principle can be formulated as

\[
\text{Select the smoothest function, according to some notion of functional smoothness, among those that fit the data perfectly.}
\]

We note that kernel machines described above (see (3.4)) fit this paradigm precisely. Indeed, for every positive definite kernel function \( K(x, z) \), there exists a reproducing kernel Hilbert space \( \mathcal{H}_K \) (functional spaces, closely related to Sobolev spaces: see Wendland 2004), with norm \( \| \cdot \|_{\mathcal{H}_K} \), such that

\[
f_{\text{ker}}(x) = \arg \min_{\forall i, f(x_i) = y_i} \| f \|_{\mathcal{H}_K}.
\]  

3.7. The double descent phenomenon

A hint toward a possible theory of induction is provided by the double descent generalization curve (shown in Figure 3.5), a pattern proposed in Belkin et al. (2019a) as a replacement for the classical U-shaped generalization curve (Figure 3.1).

When the capacity of a hypothesis class \( \mathcal{H} \) is below the interpolation threshold, not enough to fit arbitrary data, learned predictors follow the classical U-curve from Figure 3.1. The shape of the generalization curve undergoes a qualitative change when the capacity of \( \mathcal{H} \) passes the interpolation threshold, i.e. becomes large enough to interpolate the data. Although predictors at the interpolation threshold typically have high risk, further increasing the number of parameters (capacity

---

9 The existence of non-generalizing solutions is immediately clear by considering over-parametrized linear predictors. Many linear functions fit the data; most of them generalize poorly.

10 We note that inductive biases are present in any inverse problem. Interpolation simply isolates this issue.
Fit without fear

Figure 3.5. Double descent generalization curve. Modern and classical regimes are separated by the interpolation threshold. (Source: Belkin et al. 2019a.)

of \( \mathcal{H} \) leads to improved generalization. The double descent pattern has been empirically demonstrated for a broad range of datasets and algorithms, including modern deep neural networks (Belkin et al. 2019a, Spigler et al. 2019, Nakkiran et al. 2020) and observed earlier for linear models (Loog et al. 2020). The ‘modern’ regime of the curve, the phenomenon that large number of parameters often do not lead to over-fitting, has historically been observed in boosting (Schapire et al. 1998, Wyner, Olson, Bleich and Mease 2017) and random forests, including interpolating random forests (Cutler and Zhao 2001) as well as in neural networks (Breiman 1995, Neyshabur et al. 2015).

Why should predictors from richer classes perform better given that they all fit data equally well? Considering an inductive bias based on smoothness provides an explanation for this seemingly counter-intuitive phenomenon as larger spaces will generally contain ‘better’ functions. Indeed, consider a hypothesis space \( \mathcal{H}_1 \) and a larger space \( \mathcal{H}_2, \mathcal{H}_1 \subset \mathcal{H}_2 \). The corresponding subspaces of interpolating predictors, \( \mathcal{S}_1 \subset \mathcal{H}_1 \) and \( \mathcal{S}_2 \subset \mathcal{H}_2 \), are also related by inclusion: \( \mathcal{S}_1 \subset \mathcal{S}_2 \). Thus, if \( \| \cdot \|_s \) is a functional norm, or more generally, any functional, we see that

\[
\min_{f \in \mathcal{S}_2} \| f \|_s \leq \min_{f \in \mathcal{S}_1} \| f \|_s.
\]

Assuming that \( \| \cdot \|_s \) is the ‘right’ inductive bias, measuring smoothness (e.g. a Sobolev norm), we expect the minimum norm predictor from \( \mathcal{H}_2 \),

\[
f_{\mathcal{H}_2} = \arg \min_{f \in \mathcal{S}_2} \| f \|_s,
\]

to be superior to that from \( \mathcal{H}_1 \),

\[
f_{\mathcal{H}_1} = \arg \min_{f \in \mathcal{S}_1} \| f \|_s.
\]

An illustration of double descent and its connection to smoothness is provided in Figure 3.6 within the random ReLU family of models in one dimension. A very similar random Fourier feature family is described in more mathematical detail.
Figure 3.6. Illustration of double descent for random ReLU networks in one dimension. (a) Classical under-parametrized regime (3 parameters). (b) Standard over-fitting, slightly above the interpolation threshold (30 parameters). (c) ‘Modern’ heavily over-parametrized regime (3000 parameters).

below.\(^\text{11}\) Figure 3.6(a) shows what may be considered a good fit for a model with a small number of parameters. Figure 3.6(b), with the number of parameters slightly larger than the minimum necessary to fit the data, shows textbook over-fitting. However, increasing the number of parameters further results in a far more reasonable-looking curve. While this curve is still piecewise linear due to the nature of the model, it appears completely smooth. Increasing the number of parameters to infinity will indeed yield a differentiable function (a type of spline), although the difference between 3000 and infinitely many parameters is not visually perceptible. As discussed above, over-fitting appears in a range of models around the interpolation threshold which are complex but not complex enough to allow smooth structure to emerge. Furthermore, low-complexity parametric models and non-parametric models (as the number of parameters approaches infinity) coexist within the same family on different sides of the interpolation threshold.

**Random Fourier features.** Perhaps the simplest mathematically and most illuminating example of the double descent phenomenon is based on random Fourier features (RFFs) (Rahimi and Recht 2007). The RFF model family \( \mathcal{H}_m \) with \( m \) parameters (assuming \( m \) is even) consists of functions \( f: \mathbb{R}^d \rightarrow \mathbb{C} \) of the form

\[
f(w, x) = \sum_{k=1}^{m} w_k e^{\sqrt{-1}\langle v_k, x \rangle},
\]

where the vectors \( v_1, \ldots, v_m \) are fixed weights with values sampled independently from the standard normal distribution on \( \mathbb{R}^d \). The vector \( w = (w_1, \ldots, w_m) \in \mathbb{C}^m = \mathbb{R}^{2m} \) consists of trainable parameters; \( f(w, x) \) can be viewed as a neural network with one hidden layer of size \( m \) and fixed first layer weights (see (3.11) below for a general definition of a neural network).

\(^{11}\) The random ReLU family consists of piecewise linear functions of the form \( f(w, x) = \sum_k w_k \min(v_k x + b_k, 0) \), where \( v_k, b_k \) are fixed random values. While it is quite similar to RFF, it produces better visualizations in one dimension.
Given data \( \{x_i, y_i\}, i = 1, \ldots, n \), we can fit \( f_m \in \mathcal{H}_m \) by linear regression on the coefficients \( w \). In the over-parametrized regime linear regression is given by minimizing the norm under the interpolation constraints\(^\text{12}\)

\[
f_m(x) = \arg\min_{f \in \mathcal{H}_m, f(w, x_i) = y_i} \|w\|.
\]

Rahimi and Recht (2007) have shown that

\[
\lim_{m \to \infty} f_m(x) = \arg\min_{f \in S \subset \mathcal{H}_K} \|f\|_{\mathcal{H}_K} =: f_{\text{ker}}(x).
\]

Here \( \mathcal{H}_K \) is the reproducing kernel Hilbert space corresponding to the Gaussian kernel \( K(x, z) = \exp(-\|x - z\|^2) \) and \( S \subset \mathcal{H}_K \) is the manifold of interpolating functions in \( \mathcal{H}_K \). Note that \( f_{\text{ker}}(x) \) defined here is the same function defined in (3.7). This equality is known as the representer theorem (Kimeldorf and Wahba 1970, Wendland 2004).

We see that increasing the number of parameters \( m \) expands the space of interpolating classifiers in \( \mathcal{H}_m \) and allows us to obtain progressively better approximations of the ultimate functional smoothness minimizer \( f_{\text{ker}} \). Thus adding parameters in the over-parametrized setting leads to solutions with smaller norm, in contrast to the under-parametrized classical world when more parameters imply norm increase. The norm of the weight vector \( \|w\| \) asymptotes to the true functional norm of the solution \( f_{\text{ker}} \) as \( m \to \infty \). This is verified experimentally in Figure 3.7. We see that the generalization curves for both the 0–1 loss and the square loss follow the double descent curve with the peak at the interpolation threshold. The norm of the corresponding classifier increases monotonically up to the interpolation peak and decreases beyond that. It asymptotes to the norm of the kernel machine, which can be computed using the following explicit formula for a function written in the form of (3.4) (where \( K \) is the kernel matrix):

\[
\|f\|_{\mathcal{H}_K}^2 = \alpha^T K \alpha.
\]

3.8. *When do minimum norm predictors generalize?*

As we have discussed above, considerations of smoothness and simplicity suggest that minimum norm solutions may have favourable generalization properties. This turns out to be true even when the norm does not have a clear interpretation as a smoothness functional. Indeed, consider an ostensibly simple classical regression set-up, where data satisfy a linear relation corrupted by noise \( \epsilon_i \):

\[
y_i = \langle \beta^*, x_i \rangle + \epsilon_i, \quad \beta^* \in \mathbb{R}^d, \quad \epsilon_i \in \mathbb{R}, \quad i = 1, \ldots, n.
\]

\( \text{12} \) As opposed to the under-parametrized setting when linear regression simply minimizes the empirical loss over the class of linear predictors.
Figure 3.7. Double descent generalization curves and norms for random Fourier features on a subset of MNIST (a ten-class hand-written digit image dataset). (Source: Belkin et al. 2019a.)

In the over-parametrized setting, when $d > n$, least-squares regression yields a minimum norm interpolator given by $y(x) = \langle \beta_{\text{int}}, x \rangle$, where

$$
\beta_{\text{int}} = \arg\min_{\beta \in \mathbb{R}^d} \|\beta\|,
$$

and $\beta_{\text{int}}$ can be written explicitly as

$$
\beta_{\text{int}} = X^\dagger y,
$$

where $X$ is the data matrix, $y$ is the vector of labels and $X^\dagger$ is the Moore–Penrose (pseudo-) inverse.\textsuperscript{13} Linear regression for models of the type in (3.8) is no doubt the oldest\textsuperscript{14} and best studied family of statistical methods. Yet, strikingly, predictors such as those in (3.9) have historically been mostly overlooked, at least for noisy data. Indeed, a classical prescription is to regularize the predictor by, for example,

\textsuperscript{13} If $XX^T$ is invertible, as is usually the case in over-parametrized settings, $X^\dagger = X^T(XX^T)^{-1}$. In contrast, if $X^T X$ is invertible (in the classical under-parametrized setting), $X^\dagger = (X^T X)^{-1}X^T$. Note that the $XX^T$ and $X^T X$ matrices cannot both be invertible unless $X$ is a square matrix, which occurs at the interpolation threshold.

\textsuperscript{14} Originally introduced by Gauss and, possibly later, Legendre! See Stigler (1981).
adding a ‘ridge’ $\lambda I$ to obtain a non-interpolating predictor. The reluctance to overfit inhibited exploration of a range of settings where $y(x) = \langle \beta_{\text{int}}, x \rangle$ provided optimal or near-optimal predictions. Very recently, these ‘harmless interpolation’ (Muthukumar, Vodrahalli, Subramanian and Sahai 2020b) or ‘benign over-fitting’ (Bartlett, Long, Lugosi and Tsigler 2020) regimes have become a very active direction of research, a development inspired by efforts to understand deep learning. In particular, Bartlett et al. (2020) provided a spectral characterization of models exhibiting this behaviour. In addition to the aforementioned papers, some of the first work toward understanding ‘benign overfitting’ and double descent under various linear settings include the work of Hastie et al. (2019), Belkin, Hsu and Xu (2020), Mitra (2019) and Xu and Hsu (2019). Importantly, they demonstrate that when the number of parameters varies, even for linear models over-parametrized predictors are sometimes preferable to any ‘classical’ under-parametrized model.

Notably, even in cases when the norm clearly corresponds to measures of functional smoothness, such as the cases of RKHS or closely related random feature maps, the analyses of interpolation for noisy data are subtle and have only recently started to appear, e.g. Liang and Rakhlin (2020) and Mei and Montanari (2019). For a far more detailed overview of the progress on interpolation in linear regression and kernel methods, see the parallel Acta Numerica paper by Bartlett, Montanari and Rakhlin (2021).

3.9. Alignment of generalization and optimization in linear and kernel models

While over-parametrized models have manifolds of interpolating solutions, minimum norm solutions, as we have discussed, have special properties which may be conducive to generalization. For over-parametrized linear and kernel models15 there is a beautiful alignment of optimization and minimum norm interpolation: gradient descent (GD) or stochastic gradient descent (SGD) initialized at the origin can be guaranteed to converge to $\beta_{\text{int}}$ defined in (3.9). To see why this is the case, we make the following observations.

- $\beta_{\text{int}} \in \mathcal{T}$, where $\mathcal{T} = \text{span} \{x_1, \ldots, x_n\}$ is the span of the training examples (or their feature embeddings in the kernel case). To see this, verify that if $\beta_{\text{int}} \notin \mathcal{T}$, orthogonal projection of $\beta_{\text{int}}$ onto $\mathcal{T}$ is an interpolating predictor with even smaller norm, a contradiction to the definition of $\beta_{\text{int}}$.

- The (affine) subspace of interpolating predictors $S$ (see (3.6)) is orthogonal to $\mathcal{T}$ and hence $\{\beta_{\text{int}}\} = S \cap \mathcal{T}$.

These two points together are in fact a version of the representer theorem briefly discussed in Section 3.7.

15 Kernel models are linear from the optimization point of view as they can be viewed as a fixed feature map followed by a linear method. Thus we will not make a distinction in the optimization context. They are, however, non-linear functions of the input space.
Now consider gradient descent for linear regression initialized within the span of training examples $\beta_0 \in T$. Typically, we simply choose $\beta_0 = 0$, as the origin has the notable property of belonging to the span of any vectors. It can be easily verified that the gradient of the loss function at any point is also in the span of the training examples and thus the whole optimization path lies within $T$. As the gradient descent converges to a minimizer of the loss function, and $T$ is a closed set, GD must converge to the minimum norm solution $\beta_{\text{int}}$. Remarkably, in the over-parametrized settings convergence to $\beta_{\text{int}}$ is true for SGD, even with a fixed learning rate (see Section 4.4). In contrast, under-parametrized SGD with a fixed learning rate does not converge at all.

3.10. Is deep learning kernel learning? Transition to linearity in wide neural networks

But how do these ideas apply to deep neural networks? Why are complicated non-linear systems with large numbers of parameters able to generalize to unseen data?

It is important to recognize that generalization in large neural networks is a robust pattern that holds across multiple dimensions of architectures, optimization methods and datasets. As such, the ability of neural networks to generalize to unseen data reflects a fundamental interaction between the mathematical structures underlying neural function spaces, algorithms and the nature of our data. It can be likened to the gravitational force holding the Solar System, not a momentary alignment of the planets.

This point of view implies that understanding generalization in complex neural networks has to involve a general principle, relating them to more tractable mathematical objects. A prominent candidate for such an object is that of kernel machines and their corresponding reproducing kernel Hilbert spaces. As we discussed above, random Fourier features-based networks, a rather specialized type of neural architectures, approximate Gaussian kernel machines. Perhaps general neural networks can also be tied to kernel machines? Strikingly, it turns out to be the case indeed, at least for some classes of neural networks.

One of the most intriguing and remarkable recent mathematical discoveries in deep learning is the constancy of the so-called neural tangent kernel (NTK) for certain wide neural networks due to Jacot, Gabriel and Hongler (2018). As the width of certain networks increases to infinity, they undergo transition to linearity

---

16 While details such as selection of activation functions, initialization methods, connectivity patterns or many specific parameters of training (annealing schedules, momentum, batch normalization, dropout, the list goes on ad infinitum) matter for state-of-the-art performance, they are almost irrelevant if the goal is simply to obtain passable generalization.
(using the term and following the discussion in Liu, Zhu and Belkin 2020a) and become linear functions of their parameters. Specifically, consider a model \( f(w, x) \), where the vector \( w \in \mathbb{R}^M \) represents trainable parameters. The tangent kernel at \( w \), associated to \( f \), is defined as follows:

\[
K_{(x,z)}(w) := \langle \nabla_w f(w; x), \nabla_w f(w; z) \rangle \quad \text{for fixed inputs } x, z \in \mathbb{R}^d .
\] (3.10)

It is not difficult to verify that \( K_{(x,z)}(w) \) is a positive semidefinite kernel function for any fixed \( w \). To see this, consider the ‘feature map’ \( \phi_w : \mathbb{R}^d \rightarrow \mathbb{R}^M \) given by

\[
\phi_w(x) = \nabla_w f(w; x).
\]

Equation (3.10) states that the tangent kernel is simply the linear kernel in the embedding space \( \mathbb{R}^M \), \( K_{(x,z)}(w) = \langle \phi_w(x), \phi_w(z) \rangle \).

The surprising and singular finding of Jacot et al. (2018) is that for a range of infinitely wide neural network architectures with linear output layer, \( \phi_w(x) \) is independent of \( w \) in a ball around a random ‘initialization’ point \( w_0 \). That can be shown to be equivalent to the linearity of \( f(w, x) \) in \( w \) (and hence transition to linearity in the limit of infinite width):

\[
f(w, x) = \langle w - w_0, \phi_{w_0}(x) \rangle + f(w_0, x).\]

Note that \( f(w, x) \) is not a linear predictor in \( x \): it is a kernel machine, linear in terms of the parameter vector \( w \in \mathbb{R}^M \). Importantly, \( f(w, x) \) has linear training dynamics, and that is how this phenomenon is usually described in the machine learning literature (e.g. Lee et al. 2019). However, the linearity itself is a property of the model unrelated to any training procedure.\(^{17}\)

To understand the nature of this transition to linearity, consider the Taylor expansion of \( f(w, x) \) around \( w_0 \) with the Lagrange remainder term in a ball \( B \subset \mathbb{R}^M \) of radius 1 around \( w_0 \). For any \( w \in B \) there exists \( \xi \in B \) such that

\[
f'(w, x) = f(w_0, x) + \langle w - w_0, \phi_{w_0}(x) \rangle + \frac{1}{2} \langle w - w_0, H(\xi)(w - w_0) \rangle.
\]

We see that the deviation from the linearity is bounded by the spectral norm of the Hessian:

\[
\sup_{w \in B} f(w, x) - f(w_0, x) - \langle w - w_0, \phi_{w_0}(x) \rangle \leq \frac{R^2}{2} \sup_{\xi \in B} \| H(\xi) \|.
\]

\(^{17}\) This is a slight simplification, as for any finite width the linearity is only approximate in a ball of a finite radius. Thus the optimization target must be contained in that ball. For the square loss it is always the case for a sufficiently wide network. For cross-entropy loss it is not generally the case: see Section 5.1.
A general (feed-forward) neural network with $L$ hidden layers and a linear output layer is a function defined recursively as

$$
\alpha^{(0)} = x, \\
\alpha^{(l)} = \phi_l(W^{(l)} \ast \alpha^{(l-1)}), \quad \alpha \in \mathbb{R}^{d_l}, \quad W^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}, \quad l = 1, 2, \ldots, L, \\
f(w, x) = \frac{1}{\sqrt{m}} v^T \alpha^{(L)}, \quad v \in \mathbb{R}^{d_L}.
$$

(3.11)

The parameter vector $w$ is obtained by concatenation of all weight vectors $w = (w^{(1)}, \ldots, w^{(L)}, v)$ and the activation functions $\phi_l$ are usually applied coordinate-wise. It turns out that these seemingly complex non-linear systems exhibit transition to linearity under quite general conditions (see Liu et al. 2020a), given appropriate random initialization $w_0$. Specifically, it can be shown that for a ball $B$ of fixed radius around the initialization $w_0$, the spectral norm of the Hessian satisfies

$$
\sup_{\xi \in B} \|H(\xi)\| \leq O^*\left(\frac{1}{\sqrt{m}}\right), \quad \text{where} \quad m = \min_{l=1, \ldots, L} (d_l).
$$

(3.12)

It is important to emphasize that linearity is a true emerging property of large systems and does not come from the scaling of the function value with the increasing width $m$. Indeed, for any $m$ the value of the function at initialization and its gradient are all of order $1$, $f(w_0, x) = \Omega(1)$, $\nabla f(w_0, x) = \Omega(1)$.

Two-layer network: an illustration. To provide some intuition for this structural phenomenon, consider a particularly simple case of a two-layer neural network with fixed second layer. Let the model $f(w, x), x \in \mathbb{R}$ be of the form

$$
\begin{align*}
f(w, x) &= \frac{1}{\sqrt{m}} \sum_{i=1}^{m} v_i \alpha(w_i x).
\end{align*}
$$

(3.13)

For simplicity, assume that $v_i \in \{-1, 1\}$ are fixed and $w_i$ are trainable parameters. It is easy to see that in this case the Hessian $H(w)$ is a diagonal matrix with

$$
(H)_{ii} = \frac{1}{\sqrt{m}} v_i \frac{d^2 \alpha(w_i x)}{d^2 w_i} = \pm \frac{1}{\sqrt{m}} x^2 \alpha''(w_i x).
$$

(3.14)

We see that

$$
\|H(w)\| = \frac{x^2}{\sqrt{m}} \max_i |\alpha''(w_i x)| = \frac{x^2}{\sqrt{m}} \| (\alpha''(w_1 x), \ldots, \alpha''(w_m x)) \|_\infty.
$$

In contrast, the tangent kernel is given by

$$
\|\nabla w f\| = \sqrt{\frac{1}{m} \sum_i x^2 (\alpha'(w_i x))^2} = \frac{x}{\sqrt{m}} \| (\alpha'(w_1 x), \ldots, \alpha'(w_m x)) \|.
$$
Assuming that \( \mathbf{w} \) is such that \( \alpha'(w_jx) \) and \( \alpha''(w_jx) \) are of all of the same order, and given the relationship between the 2-norm and \( \infty \)-norm in \( \mathbb{R}^m \), we expect
\[
\| \mathbf{b} \| \sim \sqrt{m} \| \mathbf{a} \|_\infty.
\]
Hence
\[
\| H(\mathbf{w}) \| \sim \frac{1}{\sqrt{m}} \| \nabla_w f \|.
\]
Thus we see that the structure of the Hessian matrix forces its spectral norm to be a factor of \( \sqrt{m} \) smaller compared to the gradient. If (following common practice) \( w_i \) are sampled i.i.d. from the standard normal distribution,
\[
\| \nabla_w f \| = \sqrt{K_{(w,w)}(x)} = \Omega(1), \quad \| H(\mathbf{w}) \| = O\left(\frac{1}{\sqrt{m}}\right).
\] (3.15)

If, furthermore, the second layer weights \( v_i \) are sampled with expected value zero, \( f(\mathbf{w}, x) = O(1) \). Note that to ensure the transition to linearity we need the scaling in (3.15) to hold in a ball of radius \( O(1) \) around \( \mathbf{w} \) (rather than just at the point \( \mathbf{w} \)), which in this case can be easily verified.

The example above illustrates how the transition to linearity is the result of the structural properties of the network (in this case the Hessian is a diagonal matrix) and the difference between the 2-norm and \( \infty \)-norm in a high-dimensional space. For general deep networks the Hessian is no longer diagonal and the argument is more involved, yet there is a similar structural difference between the gradient and the Hessian related to the different scaling of the 2-norm and \( \infty \)-norm with dimension.

Furthermore, transition to linearity is not simply a property of large systems. Indeed, adding a non-linearity at the output layer, \textit{i.e.} defining
\[
g(\mathbf{w}, x) = \phi(f(\mathbf{w}, x)),
\]
where \( f(\mathbf{w}, x) \) is defined by (3.13) and \( \phi \) is any smooth function with non-zero second derivative, breaks the transition to linearity independently of the width \( m \) and the function \( \phi \). To see this, observe that \( H_g \), the Hessian of \( g \), can be written in terms of the gradient and Hessian of \( f \) (\( \nabla_w f \) and \( H(\mathbf{w}) \), respectively) as
\[
H_g(\mathbf{w}) = \phi'(f) \underbrace{H(\mathbf{w})}_{O(1/\sqrt{m})} + \phi''(f) \nabla_w f \times (\nabla_w f)^T. \tag{3.16}
\]
We see that the second term in (3.16) is of order \( \| \nabla_w f \|^2 = \Omega(1) \) and does not scale with \( m \). Thus the transition to linearity does not occur and the tangent kernel does not become constant in a ball of a fixed radius even as the width of the network tends to infinity. Interestingly, introducing even a single narrow ‘bottleneck’ layer has the same effect even if the activation functions in that layer are linear (as long as some activation functions in at least one of the deeper layers are non-linear).
As we will discuss later in Section 4, the transition to linearity is not needed for optimization, which makes this phenomenon even more intriguing. Indeed, it is possible to imagine a world where the transition to linearity phenomenon does not exist, yet neural networks can still be optimized via the usual gradient-based methods.

It is thus even more fascinating that a large class of very complex functions turn out to be linear in parameters and the corresponding complex learning algorithms are simply training kernel machines. In my view this adds significantly to the evidence that understanding kernel learning is a key to deep learning, as we argued in Belkin et al. (2018b). Some important caveats are in order. While it is arguable that deep learning may be equivalent to kernel learning in some interesting and practical regimes, the jury is still out on the question of whether this point of view can provide a conclusive understanding of generalization in neural networks. Indeed a considerable amount of recent theoretical work has been aimed at trying to understand regimes (sometimes called the ‘rich regimes’: e.g. Woodworth et al. 2020, Ghorbani, Mei, Misiakiewicz and Montanari 2020) where the transition to linearity does not happen and the system is non-linear throughout the training process. Other work (going back to Warmuth and Vishwanathan 2005) argues that there are theoretical barriers separating function classes learnable by neural networks and kernel machines (Allen-Zhu and Li 2020, Pravesh and Roi 2020). Whether these analyses are relevant for explaining empirically observed behaviours of deep networks still requires further exploration.

Further discussion of these issues can be found in Section 6.2.

4. The wonders of optimization

The success of deep learning has heavily relied on the remarkable effectiveness of gradient-based optimization methods, such as stochastic gradient descent (SGD), applied to large non-linear neural networks. Classically, finding global minima in non-convex problems such as these has been considered intractable, yet in practice neural networks can be reliably trained.

Over-parametrization and interpolation provide a distinct perspective on optimization. Under-parametrized problems are typically locally convex around their local minima. In contrast, over-parametrized optimization landscapes are generically non-convex, even locally. Instead, as we will argue, throughout most (but not all) of the parameter space they satisfy the Polyak–Łojasiewicz condition, which guarantees both existence of global minima within any sufficiently large ball and convergence of gradient methods, including GD and SGD.

Finally, as we discuss in Section 4.4, interpolation sheds light on a separate empirically observed phenomenon, the striking effectiveness of mini-batch SGD (ubiquitous in applications) in comparison to standard gradient descent.
4.1. From convexity to the PL* condition

Mathematically, interpolation corresponds to identifying \( \mathbf{w} \) so that
\[
f(\mathbf{w}, \mathbf{x}_i) = y_i, \quad i = 1, \ldots, n, \quad \mathbf{x}_i \in \mathbb{R}^d, \quad \mathbf{w} \in \mathbb{R}^M.
\]
This is a system of \( n \) equations with \( M \) variables. Aggregating these equations into a single map,
\[
F(\mathbf{w}) = (f(\mathbf{w}, \mathbf{x}_1), \ldots, f(\mathbf{w}, \mathbf{x}_n)), \tag{4.1}
\]
and setting \( \mathbf{y} = (y_1, \ldots, y_n) \), we can write that \( \mathbf{w} \) is a solution for a single equation
\[
F(\mathbf{w}) = \mathbf{y}, \quad F: \mathbb{R}^M \to \mathbb{R}^n. \tag{4.2}
\]

When can such a system be solved? The question posed in such generality initially appears to be absurd. A special case, that of solving systems of polynomial equations, is at the core of algebraic geometry, a deep and intricate mathematical field. And yet, we can often easily train non-linear neural networks to fit arbitrary data (Zhang et al. 2017). Furthermore, practical neural networks are typically trained using simple first-order gradient-based methods, such as stochastic gradient descent (SGD).

The idea of over-parametrization has recently emerged as an explanation for this phenomenon based on the intuition that a system with more variables than equations can generically be solved. We first observe that solving (4.2) (assuming a solution exists) is equivalent to minimizing the loss function
\[
\mathcal{L}(\mathbf{w}) = \| F(\mathbf{w}) - \mathbf{y} \|^2.
\]
This is a non-linear least-squares problem, which is well studied under classical under-parametrized settings (see Nocedal and Wright 2006, Chapter 10). What property of the over-parametrized optimization landscape allows for effective optimization by gradient descent (GD) or its variants? It is instructive to consider the simple example in Figure 4.1. Figure 4.1(a) corresponds to the classical regime with many isolated local minima. We see that for such a landscape there is little hope that a local method such as GD can reach a global optimum. Instead we expect it to converge to a local minimum close to the initialization point. Note that in a neighbourhood of a local minimizer the function is convex and classical convergence analyses apply.

A key insight is that landscapes of over-parametrized systems look very different, like Figure 4.1(b). We see that there every local minimum is global and the manifold of minimizers \( \mathcal{S} \) has positive dimension. It is important to observe that such a landscape is incompatible with convexity even locally. Indeed, consider an arbitrary point \( s \in \mathcal{S} \) in the inset of Figure 4.1(b). If \( \mathcal{L}(\mathbf{w}) \) is convex in any ball \( \mathcal{B} \subset \mathcal{S} \) around \( s \), the set of minimizers within that neighbourhood, \( \mathcal{B} \cap \mathcal{S} \), must be a convex set in \( \mathbb{R}^M \). Hence \( \mathcal{S} \) must be a locally linear manifold near \( s \) for \( \mathcal{L} \) to be locally convex. It is, of course, not the case for general systems and cannot be expected, even at a single point.
Thus one of the key lessons of deep learning in optimization is

Convexity, even locally, cannot be the basis of analysis for over-parametrized systems.

But what mathematical property encapsulates the ability to optimize by gradient descent for landscapes, such as in Figure 4.1? It turns out that a simple condition proposed by Polyak (1963) is sufficient for efficient minimization by gradient descent. This PL condition (for Polyak and also Łojasiewicz, who independently analysed a more general version of the condition in a different context (Łojasiewicz 1963)) is a simple first-order inequality applicable to a broad range of optimization problems (Karimi, Nutini and Schmidt 2016).

We say that \( \mathcal{L}(w) \) is \( \mu \)-PL if the following holds:

\[
\frac{1}{2} \| \nabla \mathcal{L}(w) \|^2 \geq \mu(\mathcal{L}(w) - \mathcal{L}(w^*)) .
\]

(4.3)

Here \( w^* \) is a global minimizer and \( \mu > 0 \) is a fixed real number. Polyak’s original work (Polyak 1963) showed that the PL condition within a sufficiently large ball (with radius \( O(1/\mu) \)) implied convergence of gradient descent.

It is important to notice that, unlike convexity, the PL condition is compatible with curved manifolds of minimizers. However, in this formulation, the condition is non-local. While convexity can be verified pointwise by making sure that the Hessian of \( \mathcal{L} \) is positive semidefinite, the PL condition requires ‘oracle’ knowledge of \( \mathcal{L}(w^*) \). This lack of pointwise verifiability is perhaps why the PL condition has not been used more widely in the optimization literature.

However, simply removing the \( \mathcal{L}(w^*) \) from (4.3) addresses this issue in over-parametrized settings! Consider the following modification called PL* in Liu et al.
$
abla \mathcal{L}(w) \| \geq \mu \mathcal{L}(w),$

It turns out that the PL* condition in a ball of sufficiently large radius implies both existence of an interpolating solution within that ball and exponential convergence of gradient descent and, indeed, stochastic gradient descent.

It is interesting to note that PL* is not a useful concept in under-parametrized settings – generically, there is no solution to $F(w) = y$ and thus the condition cannot be satisfied along the whole optimization path. On the other hand, the condition is remarkably flexible: it naturally extends to Riemannian manifolds (we only need the gradient to be defined) and is invariant under non-degenerate coordinate transformations.

4.2. Condition numbers of non-linear systems

Why do over-parametrized systems satisfy the PL* condition? The reason is closely related to the tangent kernel discussed in Section 3.10. Consider the tangent kernel of the map $F(w)$ defined as an $n \times n$ matrix valued function

$$K(w) = DF^T(w) \times DF(w), \quad DF(w) \in \mathbb{R}^{M \times n},$$

where $DF$ is the differential of the map $F$. It can be shown that the square loss $\mathcal{L}(w)$ satisfies the PL* condition with $\mu = \lambda_{\text{min}}(K)$. Note that the rank of $K$ is less than or equal to $M$. Hence, if the system is under-parametrized, i.e. $M < n$, $\lambda_{\text{min}}(K)(w) \equiv 0$ and the corresponding PL* condition is always trivial.
In contrast, when $M \geq n$, we expect $\lambda_{\text{min}}(K(w)) > 0$ for generic $w$. More precisely, by parameter counting, we expect that the set of $w$ with singular tangent kernel \( \{ w \in \mathbb{R}^M : \lambda_{\text{min}}(K(w)) = 0 \} \) is of co-dimension $M - n + 1$, which is exactly the amount of over-parametrization. Thus we expect that large subsets of the space $\mathbb{R}^M$ have eigenvalues separated from zero, $\lambda_{\text{min}}(K(w)) \geq \mu$. This is depicted graphically in Figure 4.2. The shaded areas correspond to the sets where the loss function is $\mu$-PL*. In order to make sure that a solution to (4.1) exists and can be achieved by gradient descent, we need to make sure that $\lambda_{\text{min}}(K(w)) > \mu$ in a ball of radius $O(1/\mu)$. Every such ball in the shaded area contains solutions of (4.1) (global minima of the loss function).

But how can an analytic condition, such as a lower bound on the smallest eigenvalue of the tangent kernel, be verified for models such as neural networks?

4.3. Controlling the PL* condition of neural networks

As discussed above and illustrated in Figure 4.2, we expect over-parametrized systems to satisfy the PL* condition over most of the parameter space, yet explicitly controlling $\mu = \lambda_{\text{min}}(K)$ in a ball of a certain radius can be subtle. We can identify two techniques which help establish such control for neural networks and other systems. The first, Hessian control, uses the fact that near-linear systems are well-conditioned in a ball provided they are well-conditioned at the origin. The second, transformation control, is based on the observation that well-conditioned systems stay well-conditioned under composition with ‘benign’ transformations. Combining these techniques can be used to prove convergence of randomly initialized wide neural networks.

4.3.1. Hessian control

Transition to linearity, discussed in Section 3.10, provides a powerful (if somewhat crude) tool for controlling $\lambda_{\text{min}}(K)$ for wide networks. The key observation is that $K(w)$ is closely related to the first derivative of $F$ at $w$. Thus the change of $K(w)$ from the initialization $K(w_0)$ can be bounded in terms of the norm of the Hessian $H$, the second derivative of $F$, using essentially the mean value theorem. We can bound the operator norm to get the following inequality (see Liu et al. 2020a):

$$
\|K(w) - K(w_0)\| \leq O\left(R \max_{\xi \in B_R} \|H(\xi)\|\right) \quad \text{for all } w \in B_R,
$$

(4.4)

where $B_R$ is a ball of radius $R$ around $w_0$.

Using standard eigenvalue perturbation bounds we have

$$
|\lambda_{\text{min}}(K)(w) - \lambda_{\text{min}}(K)(w_0)| \leq O\left(R \max_{\xi \in B_R} \|H(\xi)\|\right) \quad \text{for all } w \in B_R.
$$

(4.5)

Recall (see (3.12)) that for networks of width $m$ with linear last layer $\|H\| = O(1/\sqrt{m})$. On the other hand, it can be shown (e.g. Du, Zhai, Poczos and
Singh 2019b and Du et al. 2019a for shallow and deep networks respectively) that \( \lambda_{\text{min}}(K)(w_0) = O(1) \) and is essentially independent of the width. Hence (4.5) guarantees that, given any fixed radius \( R \), for a sufficiently wide network \( \lambda_{\text{min}}(K)(w) \) is separated from zero in the ball \( B_R \). Thus the loss function satisfies the PL\(^*\) condition in \( B_R \). As we discussed above, this guarantees the existence of global minima of the loss function and convergence of gradient descent for wide neural networks with linear output layer.

4.3.2. Transformation control

Another way to control the condition number of a system is by representing it as a composition of two or more well-conditioned maps. Informally, due to the chain rule, if \( F \) is well-conditioned then so is \( \phi \circ F \circ \psi(w) \), where

\[
\phi: \mathbb{R}^n \rightarrow \mathbb{R}^n, \quad \psi: \mathbb{R}^m \rightarrow \mathbb{R}^m
\]

are maps with non-degenerate Jacobian matrices.

In particular, combining Hessian control with transformation control can be used to prove convergence for wide neural networks with non-linear last layer (Liu et al. 2020b).

4.4. Efficient optimization by SGD

We have seen that over-parametrization helps explain why gradient descent can reach global minima even for highly non-convex optimization landscapes. Yet in practice GD is rarely used. Instead, mini-batch stochastic methods, such as SGD or Adam (Kingma and Ba 2015) are employed almost exclusively. In its simplest form, mini-batch SGD uses the following update rule:

\[
w_{t+1} = w_t - \eta \nabla \left( \frac{1}{m} \sum_{j=1}^{m} \ell(f(w_t, x_{ij}), y_{ij}) \right).
\] (4.6)

Here \( \{(x_{i1}, y_{i1}), \ldots, (x_{im}, y_{im})\} \) is a mini-batch, a subset of the training data of size \( m \), chosen at random or sequentially, and \( \eta > 0 \) is the learning rate.

At first glance, from a classical point of view, it appears that GD should be preferable to SGD. In a standard convex setting GD converges at an exponential rate (referred as linear in the optimization literature), where the loss function decreases exponentially with the number of iterations. In contrast, while SGD requires a factor of \( n/m \) less computation than GD per iteration, it converges at a far slower sublinear rate (see Bubeck 2015 for a review), with the loss function decreasing proportionally to the inverse of the number of iterations. Variance reduction techniques (Roux, Schmidt and Bach 2012, Johnson and Zhang 2013, Defazio, Bach and Lacoste-Julien 2014) can close the gap theoretically but are rarely used in practice.
As it turns out, interpolation can explain the surprising effectiveness of plain SGD compared to GD and other non-stochastic methods.\footnote{Note that the analysis is for the convex interpolated setting. While bounds for convergence under the PL\(^*\) condition are available (Bassily, Belkin and Ma 2018), they do not appear to be tight in terms of the step size and hence do not show an unambiguous advantage over GD. However, empirical evidence suggests that analogous results indeed hold in practice for neural networks.}

The key observation is that in the interpolated regime SGD with fixed step size converges exponentially fast for convex loss functions. The results showing exponential convergence of SGD when the optimal solution minimizes the loss function at each point go back to the Kaczmarz method (Kaczmarz 1937) for quadratic functions, more recently analysed in Strohmer and Vershynin (2009). For the general convex case, it was first shown in Moulines and Bach (2011). The rate was later improved in Needell, Ward and Srebro (2014).

Intuitively, exponential convergence of SGD under interpolation is due to what may be termed ‘automatic variance reduction’ (Liu and Belkin 2020). As we approach interpolation, the loss at every data point nears zero, and the variance due to mini-batch selection decreases accordingly. In contrast, under classical under-parametrized settings, it is impossible to satisfy all of the constraints at once, and the mini-batch variance converges to a non-zero constant. Thus SGD will not converge without additional algorithmic ingredients, such as averaging or reducing the learning rate. However, exponential convergence on its own is not enough to explain the apparent empirical superiority of SGD. An analysis in Ma et al. (2018) identifies interpolation as the key to efficiency of SGD in modern machine learning, and provides a sharp computational characterization of the advantage in the convex case. As the mini-batch size \( m \) grows, there are two distinct regimes, separated by the critical value \( m^* \).
• **Linear scaling.** One SGD iteration with a mini-batch of size $m \leq m^*$ is equivalent to $m$ iterations of mini-batch of size one up to a multiplicative constant close to 1.

• **Saturation.** One SGD iteration with a mini-batch of size $m > m^*$ is as effective (up to a small multiplicative constant) as one iteration of SGD with mini-batch $m^*$ or as one iteration of full gradient descent.

For the quadratic model,

$$m^* = \frac{\max_{i=1}^n \{||x_i||^2\}}{\lambda_{\max}(H)} \leq \frac{\text{tr}(H)}{\lambda_{\max}(H)},$$

where $H$ is the Hessian of the loss function and $\lambda_{\max}$ is its largest eigenvalue. This dependence is graphically represented in Figure 4.3.

Thus we see that the computational savings of SGD with mini-batch size smaller than the critical size $m^*$ over GD are of order

$$\frac{n}{m^*} \approx n \frac{\lambda_{\max}(H)}{\text{tr}(H)}.$$ 

In practice, at least for kernel methods $m^*$ appears to be a small number, less than 100 (Ma et al. 2018). It is important to note that $m^*$ is essentially independent on $n$; we expect it to converge to a constant as $n \to \infty$. Thus small (below the critical batch size) mini-batch SGD has $O(n)$ computational advantage over GD.

To give a simple realistic example, if $n = 10^6$ and $m^* = 10$, SGD has a factor of $10^5$ advantage over GD, a truly remarkable improvement!

5. Odds and ends

5.1. **Square loss for training in classification?**

The attentive reader will note that most of our optimization discussions (as well as in much of the literature) involved the square loss. While training using the square loss is standard for regression tasks, it is rarely employed for classification, where the cross-entropy loss function is the standard choice for training. For two-class problems with labels $y_i \in \{1, -1\}$, the cross-entropy (logistic) loss function is defined as

$$l_{ce}(f(x_i), y_i) = \log(1 + e^{-y_if(x_i)}).$$

A striking aspect of cross-entropy is that in order to achieve zero loss we need to have $y_if(x_i) = \infty$. Thus interpolation only occurs at infinity, and any optimization procedure would eventually escape from a ball of any fixed radius. This presents difficulties for optimization analysis as it is typically harder to apply at infinity. Furthermore, since the norm of the solution vector is infinite, there can be no transition to linearity on any domain that includes the whole optimization path, no matter how wide our network is and how tightly we control the Hessian norm (see...
Section 3.10). Finally, analyses of cross-entropy in the linear case (Ji and Telgarsky 2019) suggest that convergence is much slower than for the square loss and thus we are unlikely to approach interpolation in practice.

Thus the use of the cross-entropy loss leads us away from interpolating solutions and toward more complex mathematical analyses. Does the prism of interpolation fail us at this juncture?

The accepted justification of the cross-entropy loss for classification is that it is a better ‘surrogate’ for the 0–1 classification loss than the square loss (e.g. Goodfellow, Bengio and Courville 2016, Section 8.1.2). There is little theoretical analysis supporting this point of view. To the contrary, very recent theoretical works (Mai and Liao 2019, Muthukumar et al. 2020a, Thrampoulidis, Oymak and Soltanolkotabi 2020) prove that in certain over-parametrized regimes, training using the square loss for classification is at least as good as or better than using other loss functions. Furthermore, extensive empirical evaluations conducted in Hui and Belkin (2021) show that modern neural architectures trained with the square loss slightly outperform same architectures trained with the cross-entropy loss on the majority of tasks across several application domains including natural language processing, speech recognition and computer vision.

A curious historical parallel is that current reliance on cross-entropy loss in classification is reminiscent of the predominance of the hinge loss in the era of the support vector machines (SVMs). At the time, the prevailing intuition had been that the hinge loss was preferable to the square loss for training classifiers, yet the empirical evidence had been decidedly mixed. In his remarkable thesis, Ryan Rifkin (2002) conducted an extensive empirical evaluation and concluded that ‘the performance of the RLSC [square loss] is essentially equivalent to that of the SVM [hinge loss] across a wide range of problems, and the choice between the two should be based on computational tractability considerations.’ We see that interpolation as a guiding principle points us in the right direction yet again. Furthermore, by suggesting the square loss for classification, it reveals shortcomings of theoretical intuitions and the pitfalls of excessive belief in empirical best practices.

5.2. Interpolation and adversarial examples

A remarkable feature of modern neural networks is the existence of adversarial examples. Bruna et al. (2014) observed that by adding a small, visually imperceptible perturbation of the pixels, an image correctly classified as ‘dog’ can be moved to the class ‘ostrich’ or to some other obviously visually incorrect class. Far from being an isolated curiosity, this turned out to be a robust and ubiquitous property among different neural architectures. Indeed, modifying a single, carefully selected pixel is frequently enough to coax a neural net into misclassifying an image (Su, Vargas and Kouichi 2019). The full implications and mechanisms for the emergence of adversarial examples are not yet fully understood and are an active area
Figure 5.1. ‘Raisin bread’. The ‘raisins’ are basins where the interpolating predictor $f_{\text{int}}$ disagrees with the optimal predictor $f^*$, surrounding ‘noisy’ data points. The union of basins is an everywhere dense set of zero measure (as $n \to \infty$).

of research. Among other things, the existence and pervasiveness of adversarial examples points to the limitations of the standard i.i.d. models, as these data are not sampled from the same distribution as the training set. However, it can be proved mathematically that adversarial examples are unavoidable for interpolating classifiers in the presence of label noise (Belkin et al. 2018a, Theorem 5.1). Specifically, suppose $f_{\text{int}}$ is an interpolating classifier and let $x$ be an arbitrary point. Assume that $f_{\text{int}}(x) = y$ is a correct prediction. Given a sufficiently large dataset, there will be at least one ‘noisy’ point $x_i, y_i$, such as $f^*(x_i) \neq y_i$, in a small neighbourhood of $x$ and thus a small perturbation of $x$ can be used to flip the label.

If, furthermore, $f_{\text{int}}$ is a consistent classifier, such as predictors discussed in Section 3.5.3, it will approach the optimal predictor $f^*$ as the data size grows.

Specifically, consider the set where predictions of $f_{\text{int}}$ differ from the optimal classification:

$$S_n = \{x: f^*(x) \neq f_{\text{int}}(x)\}.$$ 

From consistency, we have

$$\lim_{n \to \infty} \mu(S_n) = 0,$$

where $\mu$ is the marginal probability measure of the data distribution. On the other hand, as $n \to \infty$, $S_n$ becomes a dense subset of the data domain. This can be thought of as raisin bread (see Figure 5.1). They are the incorrect classification basins around each misclassified example, i.e. the areas where the output of $f_{\text{int}}$ differs from $f^*$. While the raisins permeate the bread, they occupy negligible volume inside.

\[19\] Any similarity to the ‘plum pudding’ model of the atom due to J. J. Thompson is purely coincidental.
This picture is indeed consistent with the extensive empirical evidence for neural networks. A random perturbation avoids adversarial ‘raisins’ (Fawzi, Moosavi-Dezfooli and Frossard 2016), but they are easy to find by targeted optimization methods such as PCG (Madry et al. 2018). I should point out that there are also other explanations for adversarial examples (Ilyas et al. 2019). It seems plausible that several mathematical effects combine to produce adversarial examples.

6. Summary and thoughts

We proceed to summarize the key points of this article, and conclude with a discussion of machine learning and some key questions still unresolved.

6.1. The two regimes of machine learning

The sharp contrast between the ‘classical’ and ‘modern’ regimes in machine learning, separated by the interpolation threshold in various contexts, has been a central aspect of the discussion in this paper. Figure 6.1 gives a concise summary of some of these differences in a single table.

6.2. Through a glass darkly

In conclusion, it may be worthwhile to discuss some of the many missing or nebulous mathematical pieces in the gradually coalescing jigsaw puzzle of deep learning.

Inverse and direct methods. To my mind, the most puzzling question of machine learning is why inverse methods, requiring optimization or inversion, generally perform better than direct methods such as nearest neighbours. For example, a kernel machine with a positive definite kernel $K(x, z)$ appears to perform consistently and measurably better than a Nadaraya–Watson (NW) classifier using the same kernel (or the same family of kernels), despite the fact that both have the same functional form

$$f(x) = \sum_{i=1}^{n} \alpha_i K(x_i, x), \quad \alpha_i \in \mathbb{R}.$$ 

The difference is that for a kernel machine $\alpha = (K)^{-1}y$, which requires a kernel matrix inversion, while NW (for classification) simply puts $\alpha = y$.

The advantage of inverse methods appears to be a broad empirical pattern, manifested, in particular, by successes of neural networks. Indeed, were it not the case that inverse methods performed significantly better, the machine learning landscape

$^{20}$ Regularization, e.g. $\alpha = (K + \lambda I)^{-1}y$, does not change the nature of the problem.
### Table 6.1: Differences between the ‘classical’ and ‘modern’ regimes.

<table>
<thead>
<tr>
<th></th>
<th>Classical (under-parametrized)</th>
<th>Modern (over-parametrized)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Generalization curve</td>
<td>U-shaped</td>
<td>Descending</td>
</tr>
<tr>
<td>Optimal model</td>
<td>Bottom of U (hard to find)</td>
<td>Any large model (easy to find)</td>
</tr>
<tr>
<td>Optimization landscape:</td>
<td>Locally convex</td>
<td>Not locally convex</td>
</tr>
<tr>
<td></td>
<td>Minimizers locally unique</td>
<td>Manifolds of minimizers</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Satisfies PL* condition</td>
</tr>
<tr>
<td>GD/SGD convergence</td>
<td>GD converges to local min.</td>
<td>GD/SGD converge to global min.</td>
</tr>
<tr>
<td></td>
<td>SGD w. fixed learning rate</td>
<td>SGD w. fixed learning rate</td>
</tr>
<tr>
<td></td>
<td>does not converge</td>
<td>converges exponentially</td>
</tr>
<tr>
<td>Adversarial examples</td>
<td>?</td>
<td>Unavoidable</td>
</tr>
<tr>
<td>Transition to linearity</td>
<td></td>
<td>Wide networks w. linear last layer</td>
</tr>
</tbody>
</table>

Figure 6.1. Differences between the ‘classical’ and ‘modern’ regimes.

would look quite different: there would be far less need for optimization techniques and, likely, less dependence on the availability of computational resources. I am not aware of any compelling theoretical analyses to explain this remarkable empirical difference.

**Why does optimization align with statistics?**  A related question is that of the inductive bias. In over-parametrized settings, optimization methods, such as commonly used SGD and Adam (Kingma and Ba 2015), select a specific point \( \mathbf{w}^* \) in the set of parameters \( S \) corresponding to interpolating solutions. In fact, given that \( \mathbf{w}^* \) depends on the initialization typically chosen randomly, e.g. from a normal distribution, we should view \( \mathbf{w}^* \) as sampled from some induced probability distribution \( \mu \) on the subset of \( S \) reachable by optimization.
Why do parameters sampled from $\mu$ consistently generalize to data unseen in training?

While there is significant recent work on this topic, including a number of papers cited here, and the picture is becoming clearer for linear and kernel methods, we are still far from a thorough theoretical understanding of this alignments in general deep learning. Note that interpolation is particularly helpful in addressing this question as it removes the extra complication of analysing the trade-off between the inductive bias and the empirical loss.

**Early stopping versus interpolation.** In this paper we have concentrated on interpolation as it provides insights into the phenomena of deep learning. Yet in practice, at least for neural networks, precise interpolation is rarely used. Instead, iterative optimization algorithms are typically stopped when the validation loss stops decreasing or according to some other early stopping criterion.

This is done both for computational reasons, as running SGD-type algorithms to numerical convergence is typically impractical and unnecessary, but also to improve generalization, as early stopping can be viewed as a type of regularization (e.g. Yao et al. 2007) or label denoising (Li, Soltanolkotabi and Oymak 2020) that can improve test performance.

For kernel machines with standard Laplacian and Gaussian kernels, a setting where both early stopping and exact solutions can be readily computed, early stopping seems to provide at best a modest improvement to generalization performance (Belkin et al. 2018b). Even for kernel machines, however, computational efficiency of training on larger datasets seems to require iterative methods similar to SGD (Ma and Belkin 2019, Meanti, Carratino, Rosasco and Rudi 2020), thus making early stopping a computational necessity.

Despite extensive experimental work, the computational and statistical trade-offs of early stopping in the non-convex over-parametrized regimes remain murky.

**Are deep neural networks kernel machines?** A remarkable recent theoretical deep learning discovery (discussed in Section 3.10) is that in certain regimes very wide neural networks are equivalent to kernel machines. At this point much of the theoretical discussion centres on understanding the ‘rich regimes’ (e.g. Woodworth et al. 2020, Ghorbani et al. 2020), often identified with ‘feature learning’, i.e. learning representations from data. In these regimes, tangent kernels change during the training, hence neural networks are not approximated by kernel machines, i.e. a feature map followed by a linear method. The prevalent view among both theorists and practitioners is that success of neural networks cannot be explained by kernel methods as kernel predictors. Yet kernel change during training does not logically imply useful learning and may be an extraneous side-effect. Thus the question of equivalence remains open. Recent, more sophisticated kernel machines show performance much closer to the state of the art on certain tasks (Arora et al. 2020, Shankar et al. 2020) but have not yet closed the gap with neural networks.
Without going into a detailed analysis of the arguments (unlikely to be fruitful in any case, as performance of networks has not been conclusively matched by kernels, nor is there a convincing ‘smoking gun’ argument why it cannot be), it is worth outlining three possibilities.

- Neural network performance has elements which cannot be replicated by kernel machines (linear optimization problems).

- Neural networks can be approximated by data-dependent kernels, where the kernel function and the reproducing kernel Hilbert space depend on the training data (e.g. on unlabelled training data like ‘warped RKHS’ (Sindhwani, Niyogi and Belkin 2005)).

- Neural networks in practical settings can be effectively approximated by kernels, such as neural tangent kernels corresponding to infinitely wide networks (Jacot et al. 2018).

I am hopeful that in the near future some clarity on these points will be achieved.

*The role of depth.* Last – and possibly least – we would be remiss to ignore the question of depth in a paper with deep in its title. Yet, while many analyses in this paper are applicable to multi-layered networks, it is the width that drives most of the observed phenomena and intuitions. Despite recent efforts, the importance of depth is still not well understood. Properties of deep architectures point to the limitations of simple parameter counting: increasing the depth of an architecture appears to have very different effects from increasing the width, even if the total number of trainable parameters is the same. In particular, while wider networks are generally observed to perform better than narrower architectures (Lee et al. 2020), even with optimal early stopping (Nakkiran et al. 2020), the same is not true with respect to the depth, where very deep architectures can be inferior (Nichani, Radhakrishnan and Uhler 2020). One line of enquiry is interpreting depth recursively. Indeed, in certain settings increasing the depth manifests similarly to iterating a map given by a shallow network (Radhakrishnan, Belkin and Uhler 2020). Furthermore, fixed points of such iterations have been proposed as an alternative to deep networks, with some success (Bai, Kolter and Koltun 2019). More weight for this point of view is provided by the fact that tangent kernels of infinitely wide architectures satisfy a recursive relationship with respect to their depth (Jacot et al. 2018).

**Acknowledgements**

I would like to thank Acta Numerica for the invitation to write this article and its careful editing. I would like to thank Daniel Hsu, Chaoyue Liu, Adityanarayanan Radhakrishnan, Steven Wright and Libin Zhu for reading the draft and providing numerous helpful suggestions and corrections. I am especially grateful to Daniel Hsu and Steven Wright for insightful comments that helped to clarify the exposition of key concepts.
The perspective outlined here has been influenced and informed by many illuminating discussions with collaborators, colleagues and students. Many of these discussions occurred in January–May 2017 and May–August 2019 during two excellent programmes on foundations of deep learning at the Simons Institute for the Theory of Computing at Berkeley. I thank the Institute for its hospitality.

Finally, I thank the National Science Foundation and the Simons Foundation for financial support.

References


W. Fedus, B. Zoph and N. Shazeer (2021), Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. Available at arXiv:2101.03961.


Y. LeCun (2019), The epistemology of deep learning. Available at [https://www.youtube.com/watch?v=gG5NCkMerHU&t=3210s](https://www.youtube.com/watch?v=gG5NCkMerHU&t=3210s).


C. Liu and M. Belkin (2020), Accelerating SGD with momentum for over-parameterized learning, in *8th International Conference on Learning Representations (ICLR 2020)*. Available at https://openreview.net/forum?id=r1gixp4FPH.


B. Neyshabur, R. Tomioka and N. Srebro (2015), In search of the real inductive bias: On the role of implicit regularization in deep learning, in *ICLR (Workshop) 2015*. Available at https://openreview.net/forum?id=6AzZb_7Qo0e.


Available at http://www.argmin.net/2017/12/05/kitchen-sinks/.


