The Role of Architecture, Data Structure and Algorithm in Machine Learning : a Statistical Physics Approach

Maria Refinetti

THÈSE DE DOCTORAT
DE SORBONNE UNIVERSITÉ

Spécialité : Physique

École doctorale n°564: Physique en Île-de-France

réalisée

au Laboratoire de Physique de l'École Normale Supérieure de Paris

sous la direction de Florent KRZAKALA

présentée par

Maria REFINETTI

pour obtenir le grade de :

DOCTEUR DE SORBONNE UNIVERSITÉ

Sujet de la thèse :

The Role of Architecture, Data Structure and Algorithm in Machine Learning: a Statistical Physics Approach

soutenue le 24 Juin 2022

devant le jury composé de :

M.	David SAAD	Rapporteur
M.	Alessandro LAIO	Rapporteur
M^{me}	Leticia CUGLIANDOLO	Examinatrice SU
M^{me}	Sara SOLLA	Examinatrice
M.	Federico RICCI-TERSENGHI	Examinateur
M.	Florent KRZAKALA	Directeur de thèse

Acknowledgements

The three years I spent in Paris were the happiest years of my life thanks to many outstanding people for whom one page of thank you is much too short. This PhD was an incredible experience and I feel honoured to have been part of this unique academic environment.

Firstly, I would like to thank my parents, who provided me with the best conditions to live in this wonderful city and w hose encouragements and support I am grateful for. M y sister who spent the hardest times with me and stood by me despite everything. Thank you for always keeping your spirits up, for our long confinement walks near the Seine, and for always being there for me!

Thank you Florent for taking me on board the PhD, and for being so welcoming, friendly and comforting from start. I learned a lot from you, not only about physics but about music, movies, politics, history and life in general. I wish we could have gotten to know each other better as I always enjoyed our chats very much and you still have so much to teach me! I am grateful to Giulio for taking care of me as a second supervisor. You always made me feel included, made me participate in projects and even found me a spot at CSD. Thank you Giulio, you really made a big difference in my life during these years! I also want to thank Lenka, who motivated me to always do my best. You pushed me to never be satisfied until I understood every minute detail of my results, and to be a better scientist overall.

Francesca, Rudy and Florentin, who made many days special. I will cherish forever the time we spent together: laughing, discussing life and plans for the future and going to concerts and theatres. You made these years unforgettable! Stephane who I admire. You were a role model throughout and showed me how much one could do during the PhD. I never met someone as complete as you are. Ruben O., Silvia, Stefano, Luca S., Giovanni, Luca P., Gaspard, Ruben Z., Hugo, Gabriele, Bruno, Cedric, Marylou, Jean, Antoine and Alia; some of our interactions were longer than others, but each one of you was fun, kind and welcoming. You made me feel important and part of a group. M y PhD would not have been the same without all these special people.

Lastly, my most sincere thanks go to Sebastian w ho from the first day has been the best friend I could have hoped for. This PhD would not have been possible without your help, your support and the intellectual thrill of discussing with you. From the first days at ENS, to our confinement zooms, to my visits to Trieste, all these experiences shaped my life and the person I am today. You were there when I needed help and always pushed me to do my best. I am deeply grateful for all you did and hope that our scientific collaboration is not over.

This Thesis in a Nutshell

Over the last decades, machine learning revolutionised our daily lives from recommendation systems [330] to image recognition, medical data analysis [179, 202, 318], text completion and translation [148, 297], self driving cars [122, 324] and algorithmic trading [310]. Its groundbreaking successes defy classical statistics and the underlying mechanisms driving them remain, for the most part, obscure [49]. To bridge the gap between theory and practice, a community of mathematicians, physicists and computer scientists joined forces to develop an understanding of the three key components in deep learning: the architecture, the data and the algorithm [325]. My thesis falls within this vast research program with results providing insights into each one of these aspects and their interplay: [76, 201] for the architecture, [258, 260] for the data and [77, 259] for the algorithm. The main theoretical tools are those of statistical physics well adapted to describe the jillions of adjustable parameters in deep networks.

In [76], we target architecture and the benign effect of overparametrisation in deep learning. We focus on the lazy regime of deep networks [63, 145] where the weights barely move from their initial values during training. Through a bias-variance decomposition of the test error, we demonstrate that fluctuations stemming from from the noise corrupting the labels, and from the initialisation of the weights are responsible for the "double-descent" of the test error i.e its peak at the interpolation threshold and its decay upon overparametrisation. From these results, which we validate through numerical experiments, we can compare the effect of ensembling methods, overparametrisation and regularisation. Follow-ups [191, 246] perform more fine grained bias-variance decomposition. In a later collaboration [201], we rigorously extend the analysis to the general case of convex losses with generic convex regularisation.

Deep neural networks, however, solve complex tasks by extracting features from the data they are trained with and, thus, cannot be fully described by lazy-methods [19, 172]. Establishing to what extent the former outperform the later requires understanding the interplay between architecture and data structure and assessing how well both methods capture input features. We make progress in [260] by studying Gaussian mixtures classification at varying signal-to-noise ratio (SNR). We describe the dynamics of neural networks (NN), by extending the works [114, 269, 270] to the case where inputs are conditional on the labels and have non trivial covariance. The resulting equations reveal that that two layer NN with a few hidden nodes, achieve oracle like performances for all SNR. In sharp contrast, lazy methods are unable to classify the mixture at low SNR

because the transformation of the inputs in feature space remains linear. Thus, a non-separable mixture in input space remains non-separable in feature space where linear methods perform no better than random chance. The non-linear effects which allow lazy methods to learn, only kick-in at sufficiently high SNR.

Going a step further, it is key to understand which input features NN learn and how. An ideal framework to do so is unsupervised learning, where feature extraction is crucial. In [258], we focus on shallow non-linear autoencoders (AE), the simplest unsupervised learning architecture, which are trained to reconstruct their inputs. Leveraging again on [114, 269, 270], we characterise their training dynamics on synthetic data and demonstrate that the results carry over to benchmark datasets through numerical experiments. The analytical description reveals that AE learn the leading principal components of their inputs sequentially and uncovers the need to untie the weights in sigmoidal AE and to train the biases in ReLU AE. Building on previous results for linear networks, we finally propose a modification of the vanilla SGD algorithm which allows to learn the exact principal components.

In deep learning applications, however, practitioners rarely resort to vanilla SGD considered so far. They speed up and improve optimisation through add-ons among which, learning rate schedules, i.e. protocols to change the learning rate during training, are ubiquitously used [147, 222, 245, 292]. In [77], we analytically study the effects of learning rate scheduling in optimisation problems where the loss function is high-dimensional and highly non-convex as is the case in deep learning. We find that, with a power-law decay of the learning rate $\eta(t) = t^{-\beta}$, optimisation is fastest with $\beta < 1$, increasing with decreasing landscape complexity and smaller than in convex setups where $\beta = 1$ is generally optimal. When we add a signal to be recovered, optimisation occurs in two phases. In a first *exploration* phase, the dynamics navigate through rough parts of the landscape and the learning rate should be kept large. As soon as the signal is recovered, the dynamics enter a convex basin. In this *convergence* phase the convex criterion $\beta = 1$ is optimal. Numerical experiments confirm that our conclusions hold in a common regression task involving neural networks.

The scrutiny of alternative algorithms, introduced to simplify back-propagation and overcome some of its pitfalls, can deepen our understanding of the role of the algorithm in deep learning. One such algorithm, *Direct Feedback Alignement* (DFA) allows to speed up the training of neural networks [237] and to increase their robustness against adversarial attacks [54] by simultaneously injecting the error at every layer during the backward pass. In [259] we provide theoretical understanding of how and when DFA succeeds. We distinguish two phases of learning: an *alignment* phase, during which the algorithm limits the expressivity

of the network, is followed by a *memorisation* phase where the expressivity is recovered and the network learns the task. Thus, good performance is often not hindered by a considerable simplification of back-propagation's backward pass. The results also reveal why DFA notoriously fails to train convolutional neural networks, a manifestation of the interplay between algorithm and architecture. Through numerical studies in deep networks, we verify that our conclusions carry over to realistic setups.

The works, many of which were published at prestigious machine Learning conferences such as ICML, were conducted in collaboration with various institutions, including ENS Paris, EPFL (Lausanne), King's College (London) and SISSA (Trieste), and private companies, including Meta AI (Paris) and Lighton (Paris).

Papers & Pre-prints (in chronological ordering):

[76] Double trouble in double descent: Bias and variance(s) in the lazy regime.
D'Ascoli, Refinetti, Biroli & Krzakala, Sec. 3

[259] Align, then memorise: the dynamics of learning with feedback alignment.
Refinetti, D'Ascoli, Ohana & Goldt, Sec. 5.2

[260] Classifying high-dimensional Gaussian mixtures: Where kernel methods fail and neural networks succeed.
Refinetti, Goldt, Krzakala & Zdeborová, Sec. 4.2.

[258] The dynamics of representation learning in shallow, non-linear autoencoders.
Refinetti & Goldt, Sec. 4.7.

[201] Fluctuations, Bias, Variance & Ensemble of Learners: Exact Asymptotics for Convex Losses in High-Dimension.
Loureiro, Gerbelot, Refinetti, Sicuro & Krzakala, Sec. 3.1.6.

[77] Optimal learning rate schedules in high-dimensional non-convex optimisation problems.
D'Ascoli, Refinetti & Biroli, Sec. 5.1.

Two projects I was involved in during my PhD are outside the scope of this thesis and will not be discussed. These are of independent interest and we provide a short summary below.

[23] Epidemic mitigation by statistical inference from contact tracing data.
Baker, Biazzo, Braunstein, Catania, Dall'Asta, Ingrosso, Krzakala, Mazza, Mézard,

Muntoni, Refinetti, Sarao Mannelli & Zdeborová

In brief This work develops probabilistic inference methods which increase the ability of contact-tracing to mitigate pandemics such as COVID-19. We estimate the risk that an individual is infected based on the list of his recent contacts and their own risk levels, as well as personal information such as results of tests or presence of syndromes. This allows to optimise testing and quarantining strategies and provides an efficient way to control the epidemic, especially in ranges of epidemic spreading where manual tracing of all contacts of infected people becomes practically impossible but before a lock-down becomes inevitable. Our approaches translate into fully distributed algorithms compatible with privacy preserving standards since they only require communication between individuals who have recently been in contact.

[257] Bootstrapping traceless symmetric O(N) scalars.

Reehorst, Refinetti & Vichi

In brief Recently, numerical bootstrap techniques [249] determined the critical exponents of numerous physical systems to new levels of precisions [91, 92, 158, 256]. Here, we use these techniques on the correlation functions of a traceless symmetric tensors of $O(N)$ with two indexes t_{ij}. We bound, for generic N, operator dimensions for all the relevant representations. Of particular interest is the $N = 4$ case which has been conjectured to describe a phase transition in the antiferromagnetic real projective model ARP^3. Lattice simulations strongly suggest the existence of a second order phase transition, while an effective field theory approach does not predict any fixed point. Through a set of assumptions we can constrain operator dimensions to a closed region compatible with the lattice predictions. This region is still present when considering a mixed system involving t and the lowest dimension scalar singlet.

Contents

Chapter 1

Architecture, algorithm and data : the three building blocks of Machine Learning

1.1 Introduction

Today, machine leaning is central to many, if not most, daily life activities: from machine translation [148, 297], to recommendation systems [330], from medical applications [179, 202, 318] to self driving cars [122, 324]. This impressive leap forward was in part due to the skyrocketing amount of data created world-wide, Fig. 1.1, accompanied by fast increase in memory capacity which enabled storage of thousands of millions of bytes of data. With so much data available, artificial intelligence (AI) underwent a paradigm shift. It deviated from *symbolic* or *deterministic* AI [210, 211, 224], fashionable in the 80s, to modern AI or *machine learning* (ML) used today [172]. For a given task, e.g. differentiating images of cats from images of dogs, symbolic AI programs a set of rules, or representations, into a machine [178]. For example, one such rule could be to assign a picture to a cat if the animal displays displays pointy ears and to a dog otherwise. However, as is already clear in this simple example, keeping track of all exceptions, i.e. German shepherds have pointy ears, and devising all rules by hand is a gargantuan task.

Machine learning (ML) takes a contrasting approach in which the rules are learned directly from the raw data by the computer using artificial neural networks (ANN). In full generality, an ANN is simply a function, denoted $\phi_\theta(x)$, mapping an input x i.e. the image of a cat or a dog, to an output y i.e. $+1$ if the network believes it received a cat image as input and -1 for a dog. The function $\phi_\theta(x)$ depends on the network's parameters, or *weights* w. ML provides a way of finding the best

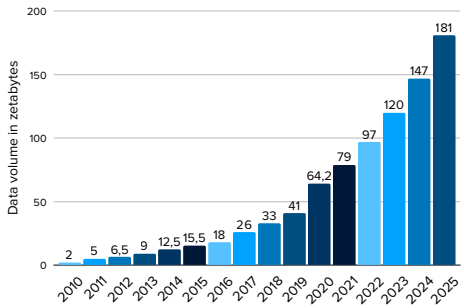


Figure 1.1: **The data created, captured, copied, and consumed globally has exploded.** Data taken from here.

choice of parameters to perform a given task like, for instance, discriminating cats from dogs. Initially, these parameters are chosen randomly, as one has no idea of what the correct choice is. In ML we start by collecting a large amount of data, i.e. various images of cats and dogs. During *training*, we iteratively feed these images to the network and update its weights with a general purpose rule in order to minimise a *loss* function i.e. the number of images for which the network makes an incorrect prediction. At the end of training, after many iterations of this step, the parameters reach a *good* value. When presented with an unseen image, the computer correctly predicts whether it is of a cat or of a dog. Having access to large amounts of data is crucial as it takes around a hundred thousand images of cats and dogs for the computer to classify images as well as the human eye. Fast computing power is also needed to perform the jillion training operations and the advent of "Graphical Processing Units" [238] (GPUs) greatly favoured the development of modern AI. In this manuscript, we are mainly interested in *supervised* learning, where the true *label* y^*, i.e. $+1$ for images of cat and -1 for those of dogs, of an input is used to train the machine. At each step, the weights are then updated to reduce the difference between the output $\phi_\theta(x)$ and the correct label. This contrasts with *unsupervised learning* where only the inputs are used for training and the network learns the relevant rules directly from them. We defer a discussion of unsupervised learning to Sec. 4.7.

Population loss vs. empirical loss

Ideally, we would like to minimise the number of times the network gives makes wrong prediction on images it hasn't seen before. If we were to define a joint probability distribution, p_{x,y^*} over the input space $\mathcal{X} \subseteq \mathbb{R}^D$ of images of cats and dogs and label space $\mathcal{Y} = \{\pm 1\}$, the aim of machine learning is to minimise the expected number of wrong predictions on new images, i.e. the *population loss* or

population error:

$$\mathcal{L}_{\text{pop}} = \mathbb{E}_{p_{x,y^*}} \ell(\phi_\theta(x), y^*). \tag{1.1}$$

In practice, one does not have access to the population error as the input-label probability distribution is inaccessible. Thus, the practitioner approximates the expectation by a sum over a finite *training* dataset of P samples leading to the minimisation of the *empirical error* defined as:

$$\mathcal{L}_{\text{emp}} = \frac{1}{P} \sum_{\mu} \ell(\phi_\theta(x_\mu), y^*_\mu), \tag{1.2}$$

where $\ell(\phi_\theta(x_\mu), y^*_\mu)$ is the error function evaluated on a single sample (x_μ, y^*_μ). As we will see, minimisation of the empirical error does not guarantee good *generalisation*, i.e. low population error.

Indeed, even if the network correctly learns to classify all pictures of cats and dogs it has seen during training, it might miss-classify a new image. For instance, suppose a leash is present in all the dog pictures in the training dataset. Then, instead of finding the "correct" criterion to distinguish cats from dogs, the network might learn to assign an image to a cat whenever there is no leash in the picture thus miss-classifying all new dog images showing no leash. The leash is an example of *spurious* feature i.e. a pattern irrelevant to the task that the network exploits while we humans avoid. This example emphasises the need to train the network on a diversified dataset i.e. one containing pictures with and without a leash. *Overfitting* can also prevent good generalisation. When the network *overfits* the training data it first learns the correct rules to classify cats and dogs. Then it exploits distinctive features of training images, such as "leash = dog" to improve its score on the training set thereby degrading its predictive power on new images of cats and dogs.

Architecture, data and algorithm as buildings blocks of machine learning

The fundamental concept behind ML is astonishingly simple. However, the devil is in the details. How should one choose the mapping ϕ_θ, i.e. which network's *architecture* is most suited for a given task? How should the parameters w of the network be updated during training in order to produce the best performance, i.e. which training *algorithm* to use? And, which features of the data x impact the answer to any one of these questions?

Despite the ubiquity of ANN in daily life, these questions remain largely unanswered. The design of efficient technologies, as those implemented today,

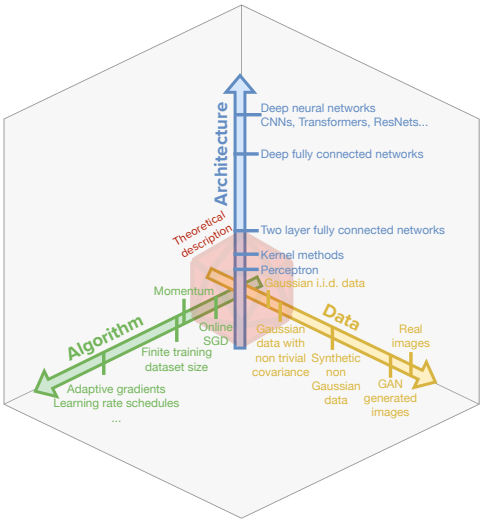


Figure 1.2: Developing a theory of machine learning requires understanding the role and interplay of the architecture, data structure and algorithm. This can be pictured as a three dimensional space where the question we address fall in the continuum defining their interplay Zdeborová [325]. In all three aspects, theoretical descriptions lag behind practice.

requires much expertise and hours of careful engineering [119, 225]. Besides some basic principles, which we discuss now, there remains a lack of a priori understanding of why certain choices lead to certain outcomes, a mystery often dubbed the "BlackBox of AI" [49]. Not only is training these networks challenging, their performance is also quite sensitive to small changes in design and to rare events. For instance, minute changes in YouTube's recommendation algorithm can cause mysterious scrambling of the ordering of videos leading to great frustration among Youtubers. Another example is self-driving cars which are very sensitive to rare events with which they have not been confronted before [126, 309].

A theoretical understanding of when, why and how neural networks are successful is crucial to make them more secure, reliable and faster to deploy. To develop this theory we must understand the three backbones of modern machine learning: the role played by the network's architecture, the training algorithm and the data as well as the interplay between the three [325]. These building axis span a three dimensional space, Fig. 1.2. Their interplay defines a continuum, and the theoretical knowledge we currently have can be seen as a volume in this space where each new insight is a point. In the three directions, theory lags behind practice. The aim of this manuscript is to show how tools from physics can help to increasing the known volume and our understanding of each one of these aspects.

Organisation of the manuscript
In the rest of this section, we give an introduction to architecture, data and algorithm in sequence. For each we provide a summary of the results obtained during the thesis. These results, as well as their derivation are detailed in later chapters. Chap. 3 concerns architecture and explains how the traditional bias-variance trade-off can be reconciled with overparametrisation in deep learning. It is mostly based on [76] and takes the results for arbitrary loss from [201]. Chap. 4 discusses data. Sec. 4.2 describes how the interplay between data and architecture impacts performance. Sec. 4.7 analyses which features are learned by a given ANN architecture and how. The results are those of [260] and [258]. Chap. 5 considers the algorithm. Sec. 5.1 investigates the impact learning rate scheduling has on performance and optimisation. Sec. 5.2 studies an alternative algorithm, direct feedback alignment, which overcomes some pitfalls of today's most widely used optimisation algorithm, back-propagation [168]. The discussions can be found in [259] and [77].

1.2 Architecture

Suppose you want to design a machine able to classify images of cats and dogs, of identifying the friends in pictures, or of recognising the sentiment you are conveying in a Whatsapp text. One way to undertake the task is to understand which *features*, or *patterns*, in the data are relevant and to program these rules into a computer [178]. This however requires much domain expertise and careful scrutiny of the type of data being analysed. The key idea of deep learning is to learn the discriminating features directly from the raw data using a general purpose learning procedure [119].

Artifical neural networks (ANN) offer to do so. They are composed of elementary constituents, the "atoms" of ANNs: the *neurons*. Neurons are grouped into simple units, or *layers*, which are stacked on top of each others like Lego bricks and connected by adjustable parameters, called *weights*. Each layer performs a simple non-linear transformation of its input. As an input is propagated through the network, it is is sequentially transformed into more and more abstract representations which amplify important features for the task and suppress the irrelevant ones (such as the background in pictures of cats and dogs). An image, for instance, is presented to the network as a vector of pixels. The first layers, i.e. top layers, often detect the presence and location of edges in the picture. Subsequent layers detect patterns as combinations of these edges, regardless of the edges position in the image, then part of familiar objects are obtained by combining these motives and so on.

At the end of the training, an ANN performs a mapping of the inputs which extracts the features needed to classify the image. Crucial to the development of deep learning is that these *features extractors* learn the relevant patterns and rules themselves, without the need for practitioner to add any domain specific knowledge or rules. Modern deep networks are often composed of tens or hundreds of layers, giving rise to the terminology *deep learning*, and hundred millions of adjustable weights. These are fixed by the learning procedure using hundred thousands inputs via an algorithm described in Sec. 1.4.

In a Lego construction, there are various different brick types and even more ways to arrange the bricks together to form different buildings, objects etc. Similarly, today, there are numerous types of ANN layers including convolutional layers [96, 172, 279], mostly used in task involving images, attention layers [307] used when the inputs are sequences of smaller components (such as words in a sentence), normalisation layers [17, 143], residual layers [133, 140, 295] and many more. These can be are stacked together in innumerable ways to form different *architectures*. Each architecture is best suited for a given type of task. For all these complex architectures however, theoretical results are scarce. For now, most analysis are centerer on the simplest architecture type i.e. fully connected networks which we describe now.

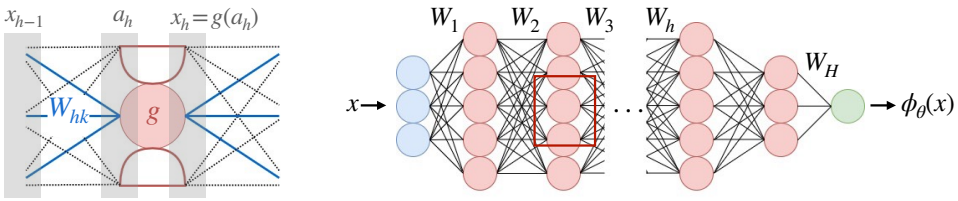


Figure 1.3: **A fully connected network is composed of fully connected layers.** Each layer takes an input x_{h-1} and maps it to the output $x_h = g(a_h)$ with $a_h = w_h x_{h-1}$. g is an element wise non-linearity. By stacking multiple layers on top of each others FC networks transform an input x into and output $\phi_\theta(x)$ and learn complex representation of the inputs.

Fully connected networks

For the purpose of this manuscript, we focus on the simplest type of layer, namely *fully-connected* layers which when stacked together form a *fully-connected* network (FCN) also called a *multi-layer perceptron* (MLP) and illustrated in Fig. 1.3 (Right). The number of layers in an FCN, i.e. its depth, is denoted H, and each layer comprises of K_h elementary units. Thus, as sketched on Fig. 1.3 (Left), a layer at depth h in the network takes its input $x_{h-1} \in \mathbb{R}^{K_{h-1}}$ and transforms it into an

output x_h:

$$x_h = g(a_h + b_h) \qquad a_h \equiv \sum_{k=1}^{K_h} w_{hk} x_{(h-1)k} \in \mathbb{R}^{K_h}, \tag{1.3}$$

where x_0 are the inputs of the network and $y \equiv x_H$ the outputs. $g : \mathbb{R} \to \mathbb{R}$ is an element-wise non-linear function and we refer to a_h as *pre-activations*. b_h is an additive bias at each layer. The stacking of these simple transformations leads to extremely intricate functions of the inputs even at moderate number of layers.

State-of-the art (SOTA) neural networks, which achieve wonders in applications, often comprise of tens of layers and hundreds of thousands parameters. However, an understanding of how to choose, for a given problem, the ideal number of layers, their width, and their activation function is still lacking. For instance, the most popular activation function nowadays is the rectified linear unit, $\mathrm{ReLU}(x) = \max(0, x)$, but in past decades, neural nets used different non-linearities, such as $\tanh(x)$ or the sigmoid, $1/1+e^{-x}$. The change occurred on purely evidence experience after noting that the ReLU typically learns much faster in networks with many layers and avoids certain pitfalls of training deeper networks, such as the vanishing gradient problem LeCun et al. [172]. In recent years, theoreticians sought to develop theoretical arguments to study deep FC networks and the role different components play in their performance. Nevertheless, only two layer networks are for now analytically tractable and even so, only in some specific limits. Below, we describe special choices of FCN which are the focus of many theoretical studies.

The perceptron and linear methods

The easiest way of predicting an outcome y from inputs x is to use linear methods. For the sake of simplicity, suppose you are given the task of separating data points coming from two different clusters as shown on the left of Fig. 1.4. A naive way to solve the problem is to draw a line separating the two clusters and assign a new observation x to the yellow cluster if it falls above the line, called *decision boundary*, and to the red cluster if it falls below the line. By defining the decision boundary as $w \in \mathbb{R}^D$, and its offset, or *bias*, $b \in \mathbb{R}$, this simple protocol can be pictured as a one layer neural network, dubbed *perceptron* [93, 98], whose output is:

$$\phi^{\mathrm{P}}_{\theta}(x) = w \cdot x + b \tag{1.4}$$

An observation x is assigned to the yellow cluster if $\phi^{\mathrm{P}}_{\theta} > 0$ and to the red cluster otherwise. Whenever there exists a line (or hyper-plane in higher dimensional problems), parametrised by $\tilde{w}$, that separates the two clusters, the problem is said

to be *linearly separable* and linear methods perform well. These methods are also very common in regression tasks where one aims at predicting a response variable $y^* \in \mathbb{R}$ from an inputs x. Pictured on a 2D plane, linear regression draws a line and assigns to an input a response on the line i.e. given by $\phi^{\mathrm{P}}_{\theta}(x)$ (see Fig. 1.5 left).

A key result, *representer theorem* [152] states that the optimal choice of weights w_{opt}, minimising the empirical loss $\mathcal{L}_{\mathrm{emp}}$, can be written as a weighted sum of the inputs in the training-set:

$$w_{\mathrm{opt}} = \sum_{\mu=1}^{N} \beta_{\mu} x_{\mu}. \tag{1.5}$$

Thus, instead of minimising the empirical risk over the parameter's w, we can minimise it with respect to the coefficients in the weighted sum β and write the output ϕ_{θ} as:

$$\phi_{\theta}(x) = \sum_{\mu} \beta_{\mu} x_{\mu} \cdot x + b. \tag{1.6}$$

This dual representation is can be much more convenient as it requires estimating N (the dimension of the training set) unknown parameters β instead of D (the input dimension). It is particularly useful when using kernel methods which we now discuss.

Kernel methods

Linear methods are extremely efficient for solving linearly separable problems and

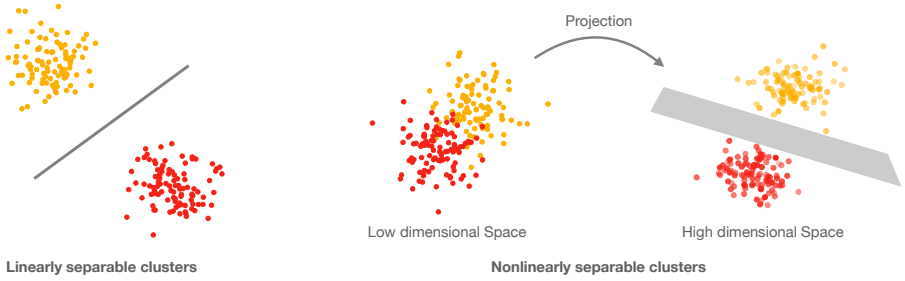


Figure 1.4: **Sketch of kernel methods** *(Left)* When clusters are linearly separable, we can determine which cluster a new point belongs to by comparing its position to the separating line. *(Right)* When the clusters cannot be separated by a line, Kernel methods project the points to a higher dimensional space in which the clusters become separable.

simple regression tasks, where the relation between the response and the inputs is linear. However, most problems of interest do not fall into this category and one

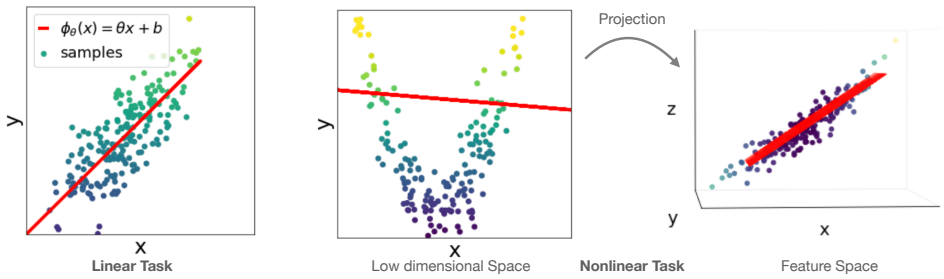


Figure 1.5: **Sketch of kernel methods for regression tasks** *(Left)* When the task is linear, i.e. the underlying relation linking y^* to x is of the form $y^* = \tilde{w}x + \tilde{b}$, then linear methods work well. *(Right)* When the task is not linear, Kernel methods project the points to a higher dimensional space in which a linear method can be successfully applied.

must resort to more sophisticated methods. Kernel methods, first introduced by [8], were advanced in the context of ML by [42]. See [41, chap. 6] and [19, chap. 7] for an introduction and a comprehensive overview of Kernel methods.

Consider, for a given task, a memory based approach which keeps in memory the inputs x_μ and their associated label y^*_μ for all pairs in the training dataset. When a new input x is presented, the method finds the training input x_ν to which it is closest and assigns it the corresponding label $y(x) = y^*_\nu$. The similarity between two inputs x and x' is measured via the *kernel function* $k(x, x') \in \mathbb{R}$. A new input x is assigned the label y^*_ν of x_ν in the train-set for which $k(x, x_\mu)$ is smallest. The simplest example of kernel function is the linear kernel, which measures similarity as the scalar product between two inputs $k(x, x') = x \cdot x'$. As we now explain, kernel methods are connected to linear methods, and offer an intuitive solution for improving the latter's performance in a myriad of settings [11, 99, 156, 174, 186, 209, 282, 286, 287].

Consider the classification problem illustrated on the right of Fig. 1.4 for which the clusters cannot be well separated by a line. To solve the task, one approach consists in projecting the points to a well chosen higher dimensional space where the clusters become linearly separable. In this new space, called *feature space*, one can use linear methods to find a plane separating the clusters. Then, by projecting a new input x in feature space and observing its position relative to the plane, we determine which cluster it belongs to. Denote $\psi : \mathbb{R}^D \to \mathbb{R}^P$ the projection of an input $x \in \mathbb{R}^D$ to a higher $P > D$ (possible infinite) dimensional space. Then we can

write the output of this procedure as:

$$\phi_\theta^{\mathrm{K}}(x) = \sum_{i=1}^{P} w_i \psi(x)_i. \tag{1.7}$$

The input is assigned to the positive cluster if $\phi_\theta(x) > 0$ and to the negative cluster otherwise.

This method, however, requires estimating P unknown parameters w which is hopeless if one chooses P large, or infinite, as is often done in practice. Representer's theorem 1.5 offers an elegant solution to the problem by expressing the optimal choice of w as a weighted sum over the projected training inputs $\{\psi(x_\mu)\}$:

$$w_{\text{opt}} = \sum_\mu \beta_\mu \psi(x_\mu) \Rightarrow \phi_\theta^{\mathrm{K}}(x) = \sum_\mu \beta_\mu \psi(x_\mu) \cdot \psi(x) = \sum_\mu \beta_\mu k(x_\mu, x). \tag{1.8}$$

We introduced the kernel function as the scalar product between the projections of the inputs in feature space $k(x, x') = \psi(x) \cdot \psi(x')$. This formulation turns the question of estimating P parameters w into finding N parameters β. Crucially, it allows to choose a very high dimensional, possibly infinite, feature space. In addition, expressing the problem only in terms of the kernel function allows to use more general similarity measures, beyond $k(x, x') = \psi(x) \cdot \psi(x')$, which cannot be written as a scalar product of feature transformations. Choosing an appropriate kernel function, or transformation ψ, is often non trivial and depends on the task at hand [283].

As the size of modern datasets increases, kernel methods become numerically expensive as they require memorising all points in the training set. Random features [253, 254] (RF) were introduced to resolve this issue by mapping inputs into feature space with $\psi = g(w_1 x)$. The parameters w_1 are chosen randomly from some probability distribution, typically $w_{ij} \sim \mathcal{N}(0, 1)$. They perform a linear projection of the inputs before an element wise non linearity g is applied. Rahimi and Recht [253] demonstrated that RF can approximate kernel methods to arbitrary precision and offer a convenient solution for large training datasets. We can identify RF with a two layer network in which the first layer's weights are fixed and while those of the second are adjustable:

$$\phi_\theta(x) = \sum_{i=1}^{P} w_{2i} g(w_{1i} \cdot x). \tag{1.9}$$

Other analytically tractable architectures

Besides linear methods and kernel methods, neural networks are analytically

understood only in particular limiting cases, some of which we discuss now.

To understand properties of deep, wide, architectures results have emerged in the limit of networks have very large hidden layers and many more parameters than the input dimensions. They are best understood by considering two layer neural networks with output:

$$\phi_\theta(x) = \frac{1}{\sqrt{K}} \sum_{i=1}^{K} w_{2i} g(w_{1i} x). \tag{1.10}$$

The first limit, coined *lazy regime* [63, 145], studies the case where the weights are initialised of order 1. If K is much larger than D, the sum runs over a large number of terms, and any small change in the weights will add up to produce a sizeable change in overall output. Thus, the network's parameters do not move much in order to fit the data they are trained with. As a consequence, we can linearise ϕ_θ around its initialisation where weights are independent of the inputs:

$$\phi_\theta(x) - \phi_{\theta_0}(x) \approx \nabla \phi_{\theta_0}(x)(w - w_0). \tag{1.11}$$

In the limit $K \to \infty$, $D/K \to 0$, Jacot et al. [144] showed that this approximation becomes exact and the network essentially performs a kernel method where the projection ψ is given by the gradient of ϕ_θ around its initialisation i.e. $\nabla \phi_{\theta_0}(x)$ [9, 10, 52, 86, 145, 181]. This has drastic consequences on learning since, in this limit, the transformation of the inputs performed by neural networks does not depend on the data one wishes to analyse. It reveals that, when training deep, overparametrised models, for which the same argument holds, one needs to take care not to operate in the lazy regime. From a theoretical perspective, this limit allows to use tools from kernel theory to study deep ANNs and thus, even if it does not give a complete picture, understand some aspects of ML.

In applications, the weights of neural networks move far away from their initialisation in order to capture relevant features of the data. The *mean-field* (MF) limit of neural networks goes beyond the lazy regime and aims at understanding feature learning [63, 104, 316]. It describes two layer neural networks defined in Eq. 1.10 for which the weights are initialised of order $O(1/\sqrt{K})$ and the width K is again taken to infinity. In sharp contrast with the lazy regime, the parameters of the network must depart considerably from their initial value to produce a sizeable change in the output allowing the network to adapt to the data and the task at hand. Learning in the MF regime converges to a well defined limit and is described theoretically by recognising that in the $K \to \infty$ limit, the sum over the hidden nodes can be replaced by an integral over a density [60, 214, 265]. We can then study

the network's performance in a variety of tasks and understand, for instance, how different aspects of the inputs, the task, the activation function or algorithm impact performance [18, 62, 73, 101, 109, 110, 185, 243, 300, 312, 320].

The distinction between a regime where the weights evolve to learn the features in the data, i.e. a *feature learning* regime, and one where they do not, i.e. a *lazy* regime, is also possible at finite width K. Chizat et al. [61] proposed to rescale the overall output of the network by a scalar α:

$$\tilde{f}_\theta^\alpha = \alpha \left[f_\theta - f_{\theta_0} \right]. \tag{1.12}$$

Choosing α of order one leads to a lazy network and $\alpha = O(K^{-1/2})$ to a mean-field limit.

The last limit we discuss is the work-horse of the analysis in Secs. 4.2 4.7 5.2 and was first studied in the seminal works [38, 269, 270]. It describes two layer networks with small hidden layer, i.e. K of order 1:

$$f_\theta = \sum_{i=1}^{K} w_{2i} g\left(\frac{w_{1i} \cdot x}{\sqrt{D}}\right), \tag{1.13}$$

where the inputs dimension is large i.e. $D \to \infty$. As we will see, in the limit where the number of training data is infinite (i.e. scales linearly with input dimension), the learning dynamics of these networks are described by a set of ordinary differential equations (ODE). These provide insights into how, and how well, the neural network learns a given task (see [114–116, 118, 258–260] and the sections referenced above).

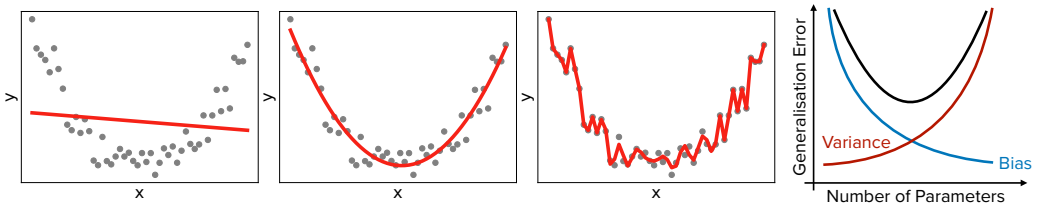


Figure 1.6: **The conventional bias-variance trade off** To fit data points, the model should neither be too simple to describe the underlying process generating the points (*left*) nor too complex to avoid being sensitive to noise in measurements (*right*). The right number of parameters is in between the two (*middle*) where both the bias and the variance of the model are low. Here $y^* = x^2 + \xi$ with ξ a random noise sampled i.i.d. from $\mathcal{N}(0, \sigma^2)$. (*right-most*) The generalisation error displays a "U"-shaped curve as the variance increases with the number of parameters while the bias decreases.

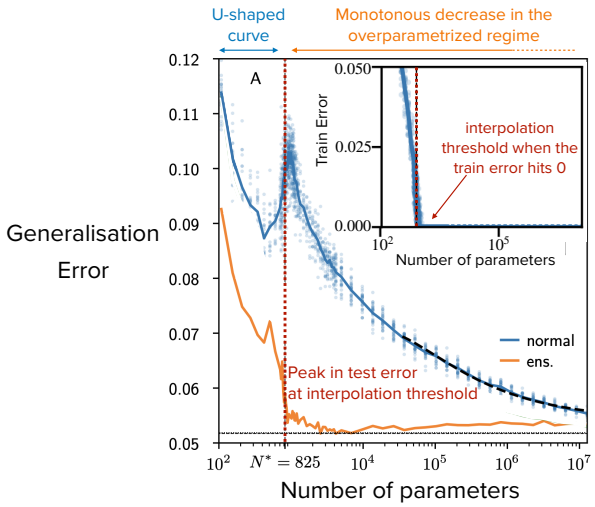


Figure 1.7: **The test error of DNNs displays a "double-descent"** As the number of parameters in the model is increased, the test error curve of modern DNN display a "U-shaped" curve characteristic of the bias-variance trade-off. After the peak at the interpolation threshold, where the training error goes to 0 (*inset*), the test error decreases monotonically in the overparametrised regime. The best performance is reached in the deeply overparamtetrised regime. Ensembling (*orange line*) suppresses the peak. (Plot taken from [102].)

Reconciling the bias-variance trade-off with overparametrisation in deep NNs

Theoreticians can gain insights into what effects certain choices of network's architecture, for instance the number of trainable parameters, have on performance by understanding how to reconcile basic principles of statistics with modern deep learning. The bias-variance trade-off, at the basis of any statistics class, teaches that the model chosen to explain data points should be carefully chosen as shown on Fig. 1.6. If it has too few parameters, it is too simple to capture the underlying physical process and the systematic error the model makes, i.e. the bias, is high. In contrast, by including too many parameters in the model, we risk overfitting i.e. fitting also the noise in the training data. The variance of the fit is high and we expect it to generalise poorly. In between these two extremes, there is a sweet-spot where the model is simple enough to be robust to noise yet complex enough to grasp the underlying input-response relation having generated the data. This results in a U-shaped curve of the population error. Deep learning however, challenges this paradigm [230, 233, 294]. Modern deep networks have hundreds of thousands of parameters, can fit all samples in the training set and yet generalise to unseen data well, at odds with conventional statistics [4, 31, 131, 213, 232]. Their performance, seen on Fig. 1.7 displays the conventional bias-variance U-shaped curve until a peak in test error at the *interpolation threshold*, where network achieves

0 train loss. It generally occurs when a number of parameters is roughly equal to the number of data points [103]. Then, instead of deteriorating at high model complexity, performance improves with increasing number of parameters resulting in a trend coined *double descent* [230, 233]. Reconciling both paradigms is crucial if we want to understand the role of depth and width in modern ANNs. This question has attracted attracted a growing amount of theoretical attention in the last few years, see e.g. [4, 31, 131, 213, 232].

In my first project [76], we make progress by focusing on the lazy-regime of neural networks and study the related RF model [253] trained with quadratic loss. In a recent collaboration [201] we rigorously extend the results to generic convex losses. The RF model is an ideal candidate for our analysis partly, but not only, for its connection to the lazy regime [63, 145, 253]. Moreover, its test error displays a double descent and it allows to disentangle the input dimension from the number of parameters, an impossible distinction in simpler linear methods. Lastly, through the randomness inherent to the choice of features, RF also offer a good basis to study ensembling which improves performance by making a prediction based on the averaged output of networks trained independently [102]. If the average is taken over sufficiently many networks, ensembling suppresses the peak at the interpolation threshold (orange line on Fig. 1.7).

An analysis of bias and variances in RF starts by recognising that there are various sources of randomness in the model: the additive noise corrupting the labels, the random sampling of the features and the random sampling of inputs in the trainset. Each contributes to the test error via its associated variance. By disentangling out the individual contributions, we can study how each source of fluctuations impacts the generalisation performances of RF. This analysis, detailed in Sec. 3.1, reveals that the variances associated to the noise corrupting the labels and the initialisation of the random features are responsible for the peak of the test error near the interpolation threshold and its decrease upon overparametrisation, see Fig. 3.3. The variance stemming from the choice of training set and the bias both display a kink at the interpolation threshold, and remain constant in the overparametrised regime. Hence, ensembling and overparametrisation improve performance by the suppressing fluctuations from the label noise and of the weights initialisation. Subsequent results [2, 176, 191] perform a more fine grained decomposition of the test error in terms of bias and variances.

1.3 Data

Today neural networks are used on astonishingly different data types, from financial data [141], to images [244], to medical data [179, 202, 318] and many more. Each data type is analysed using a specific architecture and algorithm, often chosen on a trial-and-error basis. Understanding which features in the inputs are relevant for a given task, how these are captured by a network and why is key to develop a theory of ML. Firstly for practical applications since answering these questions would allow practitioners to better select the inputs in the training set as well as simplify the choice of architecture and algorithm based on input features. Then, from a fundamental perspective, the answers could reveal some intrinsic characteristics in the data such as symmetries and similarities and teach us more on what we humans capture when interacting with the outside world.

In this manuscript, we approach the question from a physics perspective: we make assumptions on how the data is generated, i.e. we consider specific *generative models* and describe properties of typical input sampled from this model. This contrasts with *worst-case* approaches which study how rare and adverse inputs impact learning. A generative model consists of simplifying assumptions on the input-label distribution and often depends on parameters which can be tuned to study a given phenomenon in a controlled way. It must be complete enough to capture the phenomenon yet simple enough to be analytically tractable. To validate that the conclusions drawn from the model carry over to realistic settings, theoreticians can test their predictions on real data. Over the years, researchers came up with a multitude of generative models. These remain nevertheless far too simple to account for the complexity of real inputs. We now review some of the simplest and most widely used.

Gaussian data & the teacher-student model

The simplest generative model one can think of is to assume each component in the input $x \in \mathbb{R}^D$ is drawn i.i.d. from a standard normal distribution i.e. $x_i \sim \mathcal{N}(0,1)$. This will seem as a crude simplification as it amounts to assuming that the inputs are white noise. Nonetheless, many key results were derived in this setting in particular in a *teacher-student* model [1, 98, 208]. The teacher-student (TS) model assumes that the true labels y^* are given by a teacher function i.e. $y^* = \tilde{\phi}_{\tilde{w}}(x)$. The student network is trained to reproduce the out-

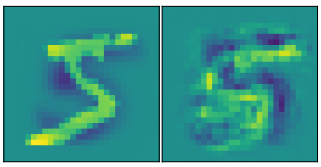


Figure 1.8: *(Left)* Original image. *(Right)* Input x sampled from a Gaussian distribution with the same covariance as the original image.

put of her teacher. The teacher's parameter $\tilde{w}$ can be seen as a signal in parameter's space which the student has to recover. Learning a target function such as the teacher is a widely studied setup in the theory of neural networks [5, 15, 20, 22, 85, 97, 109, 114, 276, 293, 303, 321, 333].

The next to simplest model is to assume the inputs are still Gaussian, but have non trivial correlation among components i.e. $x \sim \mathcal{N}(0, \Omega)$ with $\Omega \in \mathbb{R}^{D\times D}$ a covariance matrix. We can check on the right-hand side of figure 1.8 that this simplification allows to describe images well enough to recognise a digit taken from dataset of hand-written digits (MNIST [65, 170]).

Gaussian mixture model

The Gaussian mixture (GM) model [261] assumes the inputs x are sampled from a Gaussian mixture distribution. One first draws a label from a set of possible values $y^* \in \{1, ..., N_c\}$ with some probability $p(y^* = i)$ for each value. Many times we choose all values equally probable so that $p(y^* = i) = 1/N_c$. Given the value of the label, an input x is sampled from a Gaussian $x \sim \mathcal{N}(\mu_{y^*}, \Omega_{y^*})$. Note that here, the generative process is reversed with respect to the teacher-student model in the sense that in the latter, the realisation of the input determines the label. For the GM model instead, the realisation of the label determines which distribution the input is drawn from. The GM model is often used in classification tasks where the inputs belong in N_c classes, specified by the labels. Mixture models generalise the GM model by taking the distribution of x, given the y^* to be generic $p_x^{y^*}$.

Understanding the interplay between data structure and architecture in Gaussian mixture classification

The role of data structure cannot be disentangled from the one of architecture. Simple networks detect less features than more complex networks and different architectures capture different features [119, 124, 150]. So rather than asking what features in the inputs are important for the task, the right question is: given an architecture and a task, what are the input's features that allow the network to learn? One way to answer the question study how well neural networks perform on inputs drawn from a well chosen generative model. By tuning the parameters in the model, we can study the effect different features have on training.

In [260], detailed in Sec. 4.2, we explore the interplay between data structure and architecture by comparing how kernel methods and two layer NN capture input features at different noise levels. The motivation for this study is the observation that, in the lazy regime (c.f. 1.11), deep neural networks act as kernel methods. This raises the question of whether NN can only perform well if kernel methods

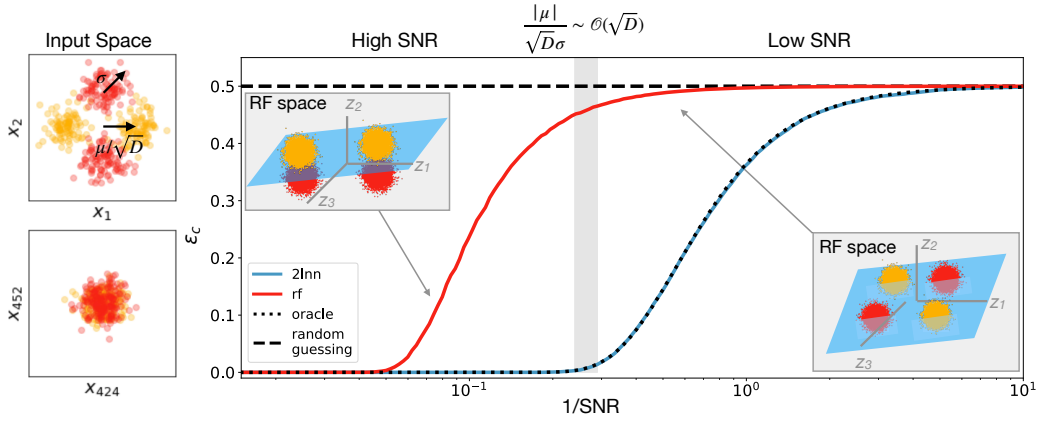


Figure 1.9: **RF and 2LNN on high-dimensional Gaussian mixture classification** *(Left)* Example of nonlinearly separable distribution with 4 clusters. The first 2 components of each centres are organised as a XOR while the other $D-2$ components are 0. The signal to noise ratio (SNR) is tuned by varying the width σ of each cluster. *(Right)* For this model, an oracle which has knowledge of the means and widths of the clusters attains minimal error. Two layers NN learn the mixture at all SNR and achieve oracle like performances. Random features instead require high SNR to learn. At low SNR the transformation in feature space remains linear. The mixture is only mapped into a non-separable mixture in feature space at high SNR. For more details see Sec. 4.2.

can also do so. Clearly, this cannot be the case as not all tasks performed by NN are achievable by kernel methods [145]. Consequently, understanding the precise conditions and input features for which the performance of NN are superior to those of kernel methods becomes a central question in deep learning [18, 73, 101, 185, 243, 300, 312, 320]. Recent works [62, 109, 110] give evidence of the superior learning ability of wide NN in the mean-field limit and Kernel methods. In Sec. 4.2, we focus on the opposite ODE limit of neural networks, (c.f. 1.13), in which the hidden layer is small, i.e. of order one. We compare the ability of kernel methods and neural networks to classify Gaussian mixtures in high dimensions and tune the hardness of the task, i.e. the prominence of the features, via the *signal-to-noise* ratio (SNR) i.e. the ratio between the distance separating the clusters and their width.

In this setup, two layer neural networks, with only a few hidden neurons, learn the mixture and achieve near optimal performance for all values of SNR. In contrast, for the same number of samples, kernel methods and random features require the signal in the input to be strong in order to perform better than chance. We explain this failure by realising that, if the SNR is too low, the transformation in feature space performed by kernel methods is linear. Thus, if a mixture is non separable in input space, it remains non-separable in feature space. At high SNR, the non-linear

effects of the transformation kick-in and performance improves. Fig. 1.9 illustrates this picture for the example of a non-separable mixture with the same covariance matrix for all clusters $\Omega_{y^*} = \sigma^2 \mathrm{I}_D$. The first 2 directions of the centers are arranged in a XOR like manner, a distance $\mu/\sqrt{D} = 1$ from the origin, and the remaining $D-2$ components are 0. The SNR is tuned by changing the width of the clusters σ. At low SNR, the mixture in feature space remains non-separable. As the SNR increases, the non-linear contributions to the projection in feature space allow for the RF to learn the task. The results and the theoretical analysis apply to more general mixtures with non-trivial covariance.

Gaussian equivalence theorem

The relevance of this result goes beyond illustrating some limitations of lazy methods. For these simple networks, Gaussian mixtures turn out to be a successful model to capture learning on real data [117, 118, 200, 213]. The similarity between training on synthetic Gaussian (and GM) data and inputs used in practice can be summarised in a somewhat surprising result, the *Gaussian equivalence theorem* (GET) [118]. The GET states that training simple networks, such as two layers FC and random features, on Gaussian inputs with well chosen mean and covariance, is indistinguishable from training on real data for a plethora of input types and tasks. It was first proposed by [106, 117, 213]. It was rigorously proved by [215] for the special case of the square loss and by [118, 138] for general convex penalties.

The GET applies, for instance, in the hidden manifold model of Goldt et al. [115]. This model builds on the observation that images are composed of a large number of pixels and are thus high dimensional objects. However, it is most likely that, when classifying images of cats and dogs, the features determining the animal are actually much fewer than the images dimensions and define a lower dimensional manifold on which the images live. We can formulate this intuition through a generative model in which the label, i.e. cat or dog, depends on a low dimensional variables $c \in \mathbb{R}^{D_c}$ with $D_c < D$. In a teacher-student setup, the teacher acts on c to give the label $y^* = \tilde{\phi}_{\tilde{w}}(c)$. The network does not have knowledge of the low dimensional vector c and is given as inputs $x = \psi(c)$ with $\psi : \mathbb{R}^{D_c} \to \mathbb{R}^D$ transforming c into x in high dimensions. In the cat vs. dog example, we can picture c as a vector containing all the necessary information for classification, x as the image with many pixels and ψ the function mapping one c to an x. As predicted by the GET, in this model, the performance of a simple network is the same when trained on x and on Gaussian inputs with the same mean and covariance as x [118]. This remains true for very complex mappings ψ, including those of generative adversarial networks which transform c into synthetic images indistinguishable

to human eyes from the real images [67, 199]. Nonetheless, it remains hard to believe that, in all cases, networks do not use any information beyond the first two moments of their inputs. The precise conditions under which the GET applies are subject to much recent interest [142].

Autoencoders as a tool to study feature learning

Another way to obtain insights into how networks learn relevant features of their inputs, and what these features are, is to study unsupervised learning, and more specifically *autoencoders* (AE) [119, 135, 159, 192] as we do in [258] and Sec. 4.7. In contrast to supervised learning, autoencoders are trained using only the inputs x which they aim to reconstruct. The task is thus to minimise the distance between x, and the network's output $\phi_\theta(x)$. To ensure good feature extraction, we take intermediate layers considerably smaller than the input dimension, thereby forcing the network to develop a compressed representation of its inputs. Autoencoders of this type, called under-complete and shown on the right, provide an ideal framework to study feature learning as the latent representation encompasses what the network considers the most valuable features in the data. Understanding training with unsupervised learning is also relevant for practical applications. Indeed, the performance of ANNs, especially with limited number of samples, can be improved by first pre-training each layer to reproduce the output of the previous layer [35, 134, 136, 255].

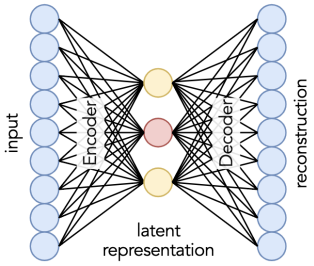


Figure 1.10: **Sketch of a shallow under-complete autoencoder**

In Refinetti and Goldt [258], we analyse the performance and training dynamics of shallow autoencoders with a single hidden layer in reconstruction tasks. For *linear* AE, Eckart and Young [89] establish that the optimal reconstruction error is obtained by a network whose weights are given by (a rotation of) the K leading eigenvectors of the input covariance matrix. The performance is the one of principal component analysis and dubbed PCA error. It is interesting to ask oneself how autoencoders learn these components during training as was recently done from various perspectives [24, 25, 46, 111, 165, 236, 239, 252, 276].

Still, neural networks rely on non-linearities in their hidden layers to extract good features from data [119]. Adding the non-linearity to the theoretical description of AEs is thus key to understand how more realistic models capture characteristics of their inputs. Although, it is known that the performances of non-linear AE is also limited by the PCA reconstruction error [24, 46], we show in Sec. 4.7 that their learning dynamics is rich and raises a series of questions: can

non-linear AE reach the PCA error, and if so, how do they do it? How do the representations they learn depend on the architecture of the network, or on the learning algorithm used to train them?

Unsupervised learning, requires the inputs to display some structure which the network can exploit and thus to have non-trivial covariance matrix. We describe the training dynamics of non-linear shallow AEs on these inputs using similar techniques as those for supervised learning in the ODE limit [38, 269, 270], detailed in Sec. 4.1. The analysis reveals that the network learns input features sequentially in order of importance. This is intuitive as one expects the most relevant features to carry the most information needed for reconstruction. However, not all AEs are able to reconstruct their inputs and we find constrains on their architecture which allow to do so: untying the network's encoder weights from those of the decoder is crucial to achieve good performance as is adding a bias in ReLU autoencoders. In addition, we introduce a slightly unconventional algorithm which allows AE to learn a one-to-one correspondence between nodes in the hidden layer and features in the inputs, i.e. leading eigenvectors of the covariance matrix.

These results, derived for synthetic datasets, where the inputs are Gaussian, carry over to networks trained on realistic inputs. They give access, at each stage, to the marginal benefit of learning higher order modes of input data and reveal a fundamental limit of shallow AE which do not differentiate between realistic inputs and Gaussian inputs with well chosen covariance.

1.4 Algorithm

Gradient descent and stochastic gradient descent

In training a neural network, the practitioner is faced with the *credit assignment dilemma* i.e. the question of how to update the network's parameters to minimise the the empirical loss (Eq. 1.2) given only the inputs and the target labels.

The most common way is to use gradient based methods, and in particular *gradient descent* (GD) [266]. GD updates the parameters of the network, at layer h and step s, via the rule:

$$w_h^{s+1} = w_h^s - \eta \frac{\partial \mathcal{L}_{\text{emp}}}{\partial w_h}, \tag{1.14}$$

where η is the *learning rate*. We can picture the empirical loss function as a hilly landscape in the high dimensional space of weight values, sketched in Fig. 1.11, where the minimum is a configuration of low average error. Then, the negative gradient vector indicates the direction of steepest descent in this landscape. The

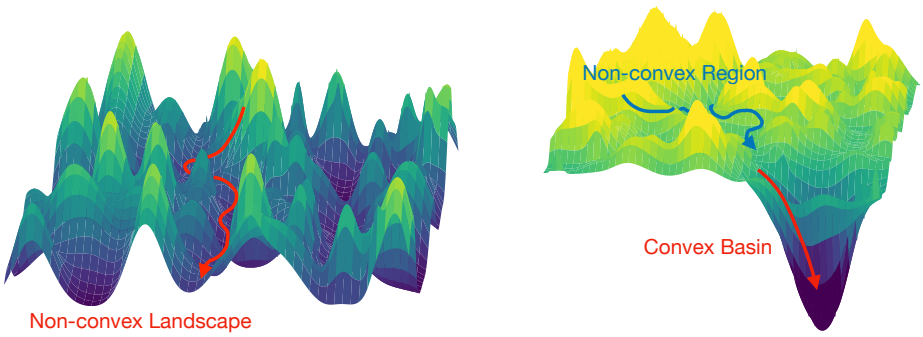


Figure 1.11: **The empirical loss as a hilly landscape**. Gradient based methods descent in the landscape.

update equation above takes a current configuration of weights to a new configuration closer to a minimum. The learning rate η determines the size of the step taken in the downward direction. Note that the hilly, i.e. *non-convex*, nature of the loss makes optimisation hard as the algorithm can get stuck in *critical* points, i.e. valleys where the gradient is small and the error is still high [119].
Gradient descent computes the error over the entire training set. As a consequence, it is often computationally expensive and requires large memory capacity, especially on modern dataset consisting of many thousands of samples [44]. *Stochastic gradient descent* (SGD) was designed to circumvent this limitation [43]. In SGD, instead of computing the loss $\mathcal{L}_{\text{emp}}$ over the whole dataset and then making a step, one computes it over a mini-batch of B samples and makes several steps before seeing the whole dataset. It is clear that SGD introduces noise in the optimisation as the average over the small set of examples gives a noisy estimate of the average over all examples. The direction of each step on the hilly landscape becomes less precise.

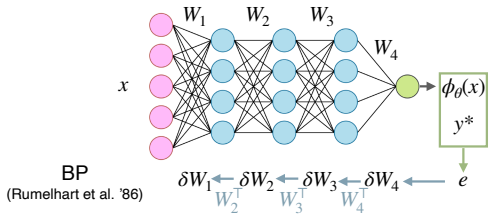


Figure 1.12: **Back-Propagation propagates the error from the last layer to the first.** Using the chain rule, BP allows to compute the gradient at each layer h efficiently by using the computations performed at layer $h+1$.

Back Propagation

A fundamental building block in modern ML is the back-propagation (BP) algorithm allowing to compute the updates Eq. 1.1 efficiently [173, 267]. It was introduced in 1986 by Rumelhart et al. [267] and is currently the most common algorithm to train ANNs.

BP computes the gradient $\partial \mathcal{L}_{\text{emp}}/\partial w^h$ by using the *chain rule* for differentiation. "Back" in the name arises because the gradient is computed across the network from the last layer to the first and the computation at a given layer uses the results at the previous layer. To illustrate BP, consider the pre-activations and post-activations defined in Eq. 1.3. The chain rule allows to rewrite 1.14 as:

$$\mathrm{d}w_{hij} = -\eta \frac{\partial \mathcal{L}_{\text{emp}}}{\partial a_{hj}} \frac{\partial a_{hj}}{\partial w_{hij}} = -\eta \frac{\partial \mathcal{L}_{\text{emp}}}{\partial a_{hj}} x_{(h-1)i} \equiv -\eta \delta_{hj} x_{(h-1)i}, \tag{1.15}$$

where we defined the *error* factor $\delta_{hj} = \partial \mathcal{L}_{\text{emp}}/\partial a_{hj}$. The computation of δ_{hj} uses the error factor at the next layer $h+1$ and proceeds back sequentially from the last layer to the first giving rise to the name "back-propagation of the error", BP for short. To clarify, we show the computation for squared error (se) $\ell \equiv$ se defined as:

$$\text{se}(y, y^*) \equiv \frac{1}{2}(y - y^*)^2. \tag{1.16}$$

The error factor at the last layer and the gradient of the loss with respect to the last layer's weights are given by:

$$\delta_{Hi} = (\phi_\theta(x) - y^*) g_H'(a_{Hi}) \Rightarrow \frac{\partial \mathcal{L}_{\text{emp}}}{\partial w_{Hij}} = (\phi_\theta(x) - y^*) g_H'(a_{Hi}) x_{(H-1)j}, \tag{1.17}$$

where g_H is a non-linearity applied to the output and we set the bias to 0 for simplicity. For the hidden layers, using the chain rule makes the computation simple:

$$\begin{aligned} \delta_{hj} &= \frac{\partial \mathcal{L}_{\text{emp}}}{\partial a_{hj}} = \sum_k \frac{\partial \mathcal{L}_{\text{emp}}}{\partial a_{(h+1)k}} \frac{\partial a_{(h+1)k}}{\partial a_{hj}} \\ &= g'(a_{hj}) \sum_k^{K_{h+1}} \delta_{(h+1)k} w_{(h+1)jk}. \end{aligned} \tag{1.18}$$

Crucially, the gradient at an intermediate layer h is given by a sum over the error vectors at the next layer $h+1$ weighted by the parameters $w_{(h+1)}$. Thus, as Fig. 1.12 shows, BP transmits the information of the error back through the network from the output layer all the way to the input layer using the network's weights. Contrasted with the naive approach of computing the gradient at each layer separately, the

backward flow of error allows efficient calculations. Putting the pieces together, the weights are updated via:

$$\mathrm{d}w_{ij}^{h} = -\eta x_{(h-1)i} g'(a_{hj}) \sum_{k=1}^{K_{k+1}} w_{(h+1)jk} \delta_{(h+1)k} \tag{1.19}$$

Note the similarity between the computations of the gradients and of the network's output Eq. 1.3. This symmetry leads to the terminology *forward pass* for the computation of output from the inputs and *backward pass* for that of the gradients in the opposite direction. Also note that the forward pass always occurs before the backward pass as the latter requires the knowledge of the activations a_{hj} and of the outputs x_{hj} computed during the former.

Learning rate schedules and how they improve BP performance

BP works incredibly well, and its invention was consequential to the success of modern ML [170–173]. However, training a network with BP requires a careful choice of *hyper-parameters* i.e. training variables fixed by the developer such as the choice of learning rate, parameters initialisation, weight decay to control the norm of the weights, batch-size etc. Tuning these parameters is still mostly done *a posteriori* on a trials-and-errors basis and requires hours of careful adjustments [119, chap 11]. In addition, variations of BP have been proposed to improve its performance such as weight-decay [161], different type of optimizers [87, 127, 153], momentum [267, 298, 299], learning rate schedules [147, 222, 245, 292] and many more. There remains, however, an overall lack of understanding about which parameters and add-ons are best suited for a given problem.

Of particular importance is the choice of learning rate and *learning rate schedule* which changes the learning rate during training i.e. changes the size of the step taken in the downward direction as the network moves on the hilly landscape (for an introduction see e.g. [50, 262]). Learning rate schedules have been developed in recent years and today their use is ubiquitous among practitioners. Thus, it is relevant to give some theoretical understanding about which scheduling technique is most effective for a given problem and why.

In convex problems, i.e. setups where the landscape looks like a basin, the optimal learning rate schedule generally goes as $\eta(t) \sim 1/t$ [182, 194]. However, we know that deep neural networks and other high-dimensional modern optimisation problems operate in highly non-convex, i.e. rugged, loss landscapes [48, 64]. In [77] and Chap. 5.1, we develop theoretical insights into the role of learning rate scheduling in this setting. To do so, we focus on a specific gradient based algorithm called the Langevin algorithm and decay the learning rate as $\eta(t) = t^{-\beta}$. The

Langevin algorithm is picked because it allows to mimik the noise in optimisation induced by SGD [59, 139, 182, 223] and on the problems we study, is analytically tractable in the infinite dimensional limit [72, 207]. We begin by considering models where the loss function presents an extensive, i.e. very large, number of critical points. We show that to speed up optimisation, i.e. descent towards the minimum of the loss, without getting stuck in saddles, the optimal choice of decay rate is $\beta < 1$ contrary to convex setups where $\beta = 1$ is generally optimal. As the non-convexity, i.e. the ruggedness, of the landscape increases, the value of the optimal exponent decreases. We also consider the problem of recovering a signal, i.e. a *ground truth* configuration of weights. In parameter's space, the signal represents a point of minimum loss around which the landscape is convex, i.e. a downward pit. In this case, learning occurs in two phases: an *exploration* phase where the dynamics navigates through rough parts of the landscape, followed by a *convergence* phase where the signal is detected and the dynamics enter a convex basin (c.f. 1.11). The optimal protocol is to keep the learning rate large during the exploration phase in order to escape the non-convex region as quickly as possible, then use the convex criterion $\beta = 1$ to converge rapidly to the solution. It allows to speed up convergence and find lower loss solutions, and is reminiscent of schedules used in practice [100, 328]. Lastly, we show that the conclusions drawn for this theoretical setup also hold in a common regression task involving neural networks trained with SGD.

Alternative training algorithm to go beyond BP

The careful tuning of hyper-parameters is only one of the issues of BP which, despite its popularity and practical success, suffers from several other limitations. For instance, it is clear from Eq. 1.18, that BP uses the same weights in the forward pass as those in the backward pass. In biological neural networks, however, it is highly unlikely that synapses propagating information in one direction are the same as those propagating it in the other direction. As a consequence, BP is considered a biologically implausible learning algorithm [69, 123]. In addition, and maybe more importantly for practical applications, the sequential updates of weights during the backward pass, see Fig. 1.13 top line, prevents an efficient parallelisation of training. This constrain becomes ever more restraining as state-of-the-art networks grow larger and deeper.

In light of these shortcomings algorithms which do not compute the exact gradient in Eq. 1.18 but an approximation of it are recently attracting increasing interest. Feedback alignment (FA) [190], sketched in the middle row of Fig. 1.13, is one such algorithm that replaces the weights w_{h+1} in Eq. 1.18 by a random *feedback*

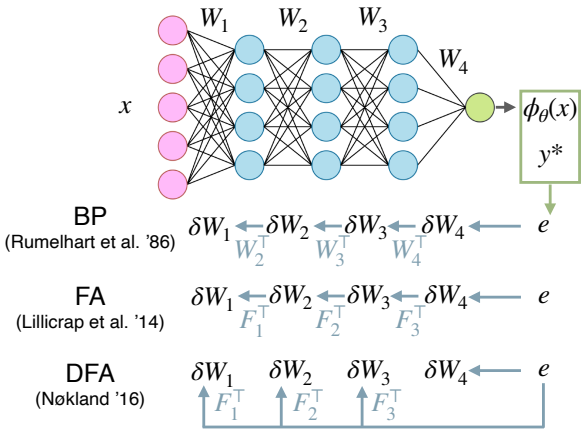


Figure 1.13: **Back-Propagation propagates the error from the last layer to the first.** Using the chain rule, BP allows to compute the gradient at each layer h efficiently by using the computations performed at layer $h+1$.

matrix F, leading to:

$$\delta_{(FA)hj} = g'(a_{hj}) \sum_k \delta_{(h+1)k} F_{hjk} \tag{1.20}$$

In this way, FA dispenses of the biologically unreasonable requirement of symmetric weights in the forward and backward pass. It is not a priori clear that the replacement of the true weights by a random matrix does not prevent the network from learning. Lillicrap et al. [190] demonstrated that FA allows to train neural networks successfully.

Direct feedback alignment (DFA), an algorithm proposed by Nøkland [237] and sketched on the last row of Fig. 1.13, takes this idea one step further. DFA propagates the error directly from the output layer to each hidden layer of the network through random feedback connections F^h:

$$\delta^h_{(DFA)j} = g'(a_{hj}) \Delta F^h_j, \tag{1.21}$$

with $\Delta \equiv \phi_\theta(x) - y^*$ as before. When using DFA, the backward pass can be done in parallel as all layers are updated simultaneously. While it was initially unclear whether DFA could scale to challenging datasets and complex architectures [29, 112], recently Launay et al. [167] obtained performances comparable to fine-tuned BP when using DFA to train a number of state-of-the-art architectures on problems ranging from neural view synthesis to natural language processing. Yet, both FA and DFA notoriously fails to train convolutional networks [29, 128, 166, 227], which

are the most widely used for vision tasks i.e. tasks taking images as inputs.

These varied results underline the need for a theoretical understanding of how and when feedback alignment works. Such an understanding, besides being useful in practice, can shed light on the interplay between architecture, data and algorithm. Does DFA train FC networks as well as BP on all data structures? When it does, can we conclude that the information carried by the weights in the backward pass is superfluous? If not, what part of it is essential to train the network? Why is the information required by FC networks not sufficient to train convolutional NN? What makes them intrinsically different from FC networks with respect to the algorithm?

These questions motivate our work [259], detailed in Sec. 5.2, where we provide theoretical insights into how and when feedback alignment works. We start by extending the theoretical description of Sec. 4.1 for shallow non-linear networks to study the dynamics of training with DFA. Through this analysis, we show that learning with DFA proceeds in two steps. First, in an *alignment phase*, the forward weights w adapt to the feedback weights and their configuration induce the DFA gradient Eq. 1.21 to be aligned with the true BP gradient Eq. 1.18. Thus, both algorithms take steps in the same direction in the landscape. In a second *memorisation phase*, the network sacrifices some alignment in order minimise the loss. The constrains imposed during the alignment phase lead to a *degeneracy breaking* effect: out of all same loss solutions in the landscape, DFA converges to the one maximising the alignment between the DFA gradient and the BP gradient. We then move to the analysis of deep linear networks. Although the function implemented by these networks is linear, their dynamics is rich [275]. We establish a condition for gradient alignment: the conditioning of the *alignment matrices*, described in Sec. 5.2. In particular, this condition allows us to analyse the impact of data structure on DFA, and suggests an explanation for the failure of DFA to train convolutional layers. Through various numerical experiments, we demonstrate that these theoretical findings, namely the Align-then-Memorise phases of learning, degeneracy breaking and layer-wise alignment, extend to deep non-linear networks trained on standard vision datasets.

Chapter 2

L'algorithme, l'architecture et les données : les trois piliers du machine learning

Au cours des dernières décennies, l'apprentissage machine (*machine learning* (ML) en anglais) a révolutionné notre quotidient, depuis les systèmes de recommandation [330] à la reconnaissance d'images, de l'analyse de données médicales [179, 202, 318] à la traduction automatique [148, 297], des voitures autonomes [122, 324] aux marchés financiers [310]. Ses succès fulgurants défient les statistiques classiques et la plupart des mécanismes sous-jacents qui les rendent possible restent inexpliqués [49]. Pour combler le gouffre entre la théorie et la pratique, des mathématiciens, des physiciens et des informaticiens ont unis leur forces pour développer une compréhension des trois composants clés du deep learning: l'architecture, les données et l'algorithme [325]. Ma thèse s'inscrit dans ce vaste programme de recherche et les résultats concernent chacun de ces aspects ainsi que leurs interactions: [76, 201] pour l'architecture, [258, 260] pour les données et [77, 259] pour l'algorithme. Les principaux outils théoriques sont ceux de la physique statistique qui se porte bien à la description des millions de paramètres constituants les réseaux de neurones artificiels profonds.

Dans mon premier travail [76], nous étudions l'architecture et, plus spécifiquement, l'effet bénin de la surparamétrisation en deep learning. Selon la statistique classique, le nombre de paramètres utilisés dans le modèle ne doit être ni trop grand ni trop petit de sorte que le modèle soit assez expressif pour décrire les relations entre les entrées et les cibles mais ne soit pas trop sensible au bruit aléatoire dans les données. Il est donc stupéfiant que dans la pratique les réseaux de neurones avec un nombre de paramètres élevé produisent d'excellents résul-

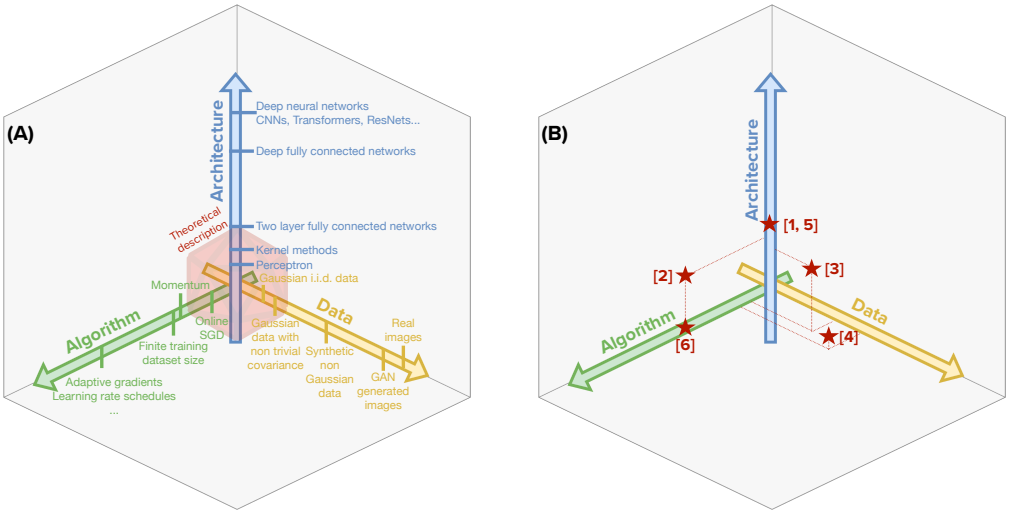


Figure 2.1: Pour développer une théorie de l'apprentissage machine nous devons comprendre le rôle de l'architecture, de la structure des données et de l'algorithme ainsi que leurs interactions. Ces trois axes définissent un espace en trois dimensions dans lequel s'inscrivent les problématiques abordées au cours de ma thèse (B). Les descriptions théoriques sont en retard sur la pratique dans ces trois aspects (A).

tats. Nous nous focalisons sur le régime paresseux des réseaux profonds [63, 145] (*lazy regime* en anglais) ainsi nommé car, pendant l'entraînement, les poids du réseaux restent proche de leurs valeurs initiales. Par une décomposition de l'erreur de généralisation en terme de biais et variances, nous montrons que les fluctuations issues du bruit dans les cibles, et du choix aléatoire des poids initiaux sont responsables de la "double descente" de l'erreur de généralisation c'est-à-dire de son pic au seuil d'interpolation et suivi par une décroissance dans le régime surparamétré. Ces résultats nous permettent également de comparer l'effet des méthodes d'*ensembling*, de surparamétrisation et de régularisation. Nous validons nos conclusions par des expériences numériques sur des réseaux profonds. Cette recherche a suscité de l'intérêt dans la communauté et par la suite plusieurs travaux l'ont approfondie [191, 246] en effectuant une décomposition en biais et variance plus fine. Dans une collaboration postérieure [201], nous étendons l'analyse de manière rigoureuse au cas général des fonctions de cout convexes.

Néanmoins, les réseaux de neurones profonds résolvent des tâches complexes en reconnaissant des caractéristiques intrinsèques aux données d'entrainement. Ainsi, ils ne peuvent être entièrement décrits par les méthodes paresseuses qui ne discernent pas ces structures [19, 172]. Pour établir dans quelle mesure les réseaux de neurones surpassent les méthodes paresseuses, nous devons comprendre l'interaction entre l'architecture et la structure des données ainsi qu'évaluer la

capacité des deux méthodes à capturer les caractéristiques des données. C'est ce que nous faisons dans [260] en étudiant le problème de la classification des mixtures de Gaussiennes (*Gaussian mixtures* (GM) en anglais) à différents nivaux de signal à bruit (*Signal-to-Noise* (SNR) en anglais). Nous étendons les travaux [114, 269, 270] au cas où les entrées sont conditionnelles aux cibles et ont une matrice de covariance non triviale. Ceci nous permet de décrire la dynamique d'entrainement des réseaux de neurones à deux couches avec une petite couche cachée. Les équations qui résultent de cette analyse révèlent que ces réseaux atteignent des performances optimales pour toute valeur de signal-à-bruit. À l'opposé, les méthodes paresseuses sont incapables d'apprendre la tache à faible SNR et nécessitent d'un signal de forte intensité pour atteindre de bonnes performances. Ceci est du à la classification linéaire que ces méthodes effectuent dans un espace à haute dimension dans lequel elles projettent les donnés. Si le niveau du signal est en dessous d'une valeur seuil, cette transformation est linéaire. Ainsi, une mixture non-séparable dans l'espace des entrées demeure non-séparable après transformation et une classification linéaire ne peut que deviner la cible au hasard. Les effets non-linéaires de la transformation permettant aux méthodes paresseuses d'apprendre, ne se déclenchent qu'à un niveau du signal suffisamment élevé.

Pour aller plus loin dans la description du machine learning, il est essentiel de comprendre quelles caractéristiques des données sont relevées par les réseaux de neurones. Pour ce faire, l'apprentissage non supervisé est un cadre idéal car il garantie l'extraction de structure dans les entrés. Dans [258], nous considérons les auto-encodeurs non linéaires peu profonds (AE), l'architecture la plus simple pour l'apprentissage non supervisé. Les AE sont entraînés à reconstruire leurs entrées. Nous caractérisons leur dynamique d'entraînement sur des données synthétiques de manière théorique et vérifions ensuite par des expériences numériques que les résultats s'appliquent également lorsque les réseaux sont entrainés sur des données réelles. La description analytique révèle que les AE apprennent les composantes principales de leur données séquentiellement, par ordre décroissant d'importance. Elle met également en évidence des contraintes architecturales néscéssaires à l'apprentissage comme celle de délier les poids dans les AE sigmoïdaux et d'entrainer les biais dans les AE ReLU. Enfin, nous nous basons sur des résultats précédents pour les réseaux linéaires, et proposons une modification de SGD permettant d'obtenir une correspondance un à un entre les poids du résaux et les composantes principales des données.

Dans les applications de deep learning la version basique de SGD, considérée jusqu'à présent est rarement utilisée. En effet, l'optimisation est accélérée et améliorée grâce à des modules complémentaires. Notamment, les *learning rate*

schedules, c'est-à-dire des protocoles qui modifient la taille du pas de temps au cour de l'entrainement, sont systématiquement utilisés [147, 222, 245, 292]. Dans [77], nous étudions leur effets de manière analytique dans des problèmes d'optimisation où la fonction de cout est de grande dimensions et non-convexe comme dans le deep learning. Nous décelons qu'avec une décroissance en loi de puissance du pas de temps $\eta(t) = t^{-\beta}$, l'optimisation est plus rapide en prenant $\beta < 1$. Cet exposant diminue lorsque la complexité du problème augmente et est inférieure au choix optimal dans le cas convex $\beta = 1$. Si on ajoute au problème un signal à retrouver, l'optimisation se déroule en deux phases. Dans une première phase d'*exploration*, la dynamique navigue à travers des regions rugueuse du paysage et on doit maintenir un grand pas de temps. Dés que le signal est repéré, commence la phase de *convergence* où la dynamique entre dans un bassin convexe et le critère $\beta = 1$ est optimal. Au travers d'expériences numériques, nous confirmons que nos conclusions sont valables dans des tâches de régression sur des réseaux de neurones.

Nous pouvons également approfondir notre compréhension du rôle de l'algorithme dans le deep learning en étudiants des algorithmes d'entrainement alternatifs qui ont souvent été introduits pour simplifier la passe arrière de *back-propagation* (BP) et pour résoudre certains de ses défauts. En particulier, le *Direct Feedback Alignement* (DFA) permet d'accélérer l'entraînement des réseaux de neurones [237] et d'augmenter leur robustesse contre les attaques adverses [54]. Au cours de la passe-arrière, cet algorithme transmet le signal d'erreur depuis la sortie à toutes les couches en simultané au moyen de matrices aléatoires. Dans [259], nous menons une analyse théorique qui permet de comprendre quand et pourquoi DFA permet d'entrainer des réseaux avec succès. Nous identifions deux phases d'apprentissage: une phase d'*alignement*, pendant laquelle l'expressivité du réseau est limitée par l'algorithme, est suivie d'une phase de *mémorisation* où le réseau apprend la tâche. Ainsi, une simplification drastique de la passe arrière de BP permet encore d'obtenir une bonne performance. Nos résultats révèlent également la raison pour laquelle la DFA échoue à entrainer des réseaux de neurones de convolution (*convolutional neural networks* (CNN) en anglais). Ceci est à nouveau une manifestation de l'interaction entre l'algorithme et l'architecture. Au travers d'études numériques sur des réseaux profonds, nous confirmons que nos résultats rentent valides dans des applications réalistes.

La plus part de ces recherches ont menées à des publications dans des conférences prestigieuses telles que ICML. Elles ont été faites en collaboration avec plusieures institutions, dont l'ENS Paris, l'EPFL (Lausanne), King's College (Londres) et SISSA (Trieste), et des entreprises privées, dont Meta AI (Paris) et Lighton

(Paris).

Publications & Pre-prints (en ordre chronologique):

[76] Double trouble in double descent: Bias and variance(s) in the lazy regime.
D'Ascoli, Refinetti, Biroli & Krzakala, Sec. 3

[259] Align, then memorise: the dynamics of learning with feedback alignment.
Refinetti, D'Ascoli, Ohana & Goldt, Sec. 5.2

[260] Classifying high-dimensional Gaussian mixtures: Where kernel methods fail and neural networks succeed.
Refinetti, Goldt, Krzakala & Zdeborová, Sec. 4.2.

[258] The dynamics of representation learning in shallow, non-linear autoencoders.
Refinetti & Goldt, Sec. 4.7.

[201] Fluctuations, Bias, Variance & Ensemble of Learners: Exact Asymptotics for Convex Losses in High-Dimension.
Loureiro, Gerbelot, Refinetti, Sicuro & Krzakala, Sec. 3.1.6.

[77] Optimal learning rate schedules in high-dimensional non-convex optimisation problems.
D'Ascoli, Refinetti & Biroli, Sec. 5.1.

Au cours de mon doctorat j'ai pris part à deux projets qui sortent du cadre de ce manuscrit et ne seront pas discutés. Nous en fournissons un bref résumé ci-dessous et encourageons le lecteur intéressé à les parcourir indépendamment.

[23] Epidemic mitigation by statistical inference from contact tracing data.
Baker, Biazzo, Braunstein, Catania, Dall'Asta, Ingrosso, Krzakala, Mazza, Mézard, Muntoni, Refinetti, Sarao Mannelli & Zdeborová
En bref Ce travail développe des méthodes d'inférence probabiliste permettant d'augmenter l'efficacités des méthodes de traçage pour blocker la propagation de pandémies telles que la COVID-19. Nous estimons le risque d'infection d'un individu au travers de la liste de ses contacts récents et de leurs propres niveaux de risque, ainsi qu'au travers d'informations personnelles commes les résultats des tests ou la présence de syndromes. Cette estimation permet d'optimiser les stratégies de test et d'isolement et fournit un moyen efficace pour contrôler la propagation de l'épidémie. Elle est notamment efficace pour des tots de propagation épidémique où le traçage

manuel de tous les contacts de personnes infectées est infaisable mais avant qu'un confinement ne devienne inévitable. Notre approche se traduit par des algorithmes entièrement distribués et compatibles avec les normes de protection des données personnelles puisqu'ils nécessitent uniquement d'une communication entre des individus ayant été en contact récemment.

[257] Bootstrapping traceless symmetric O(N) scalars.

Reehorst, Refinetti & Vichi

In brief Au cours des dernières années, les les exposants critiques de nombreux systèmes physiques ont été déterminés très précisément [91, 92, 158, 256] grace à des techniques de bootstrap numériques [249]. Dans ce travail, nous appliquons ces techniques aux fonctions de corrélation d'un tenseur symétrique de $O(N)$ à deux indices et trace nulle t_{ij}. Ainsi, pour N générique, nous bornons les dimensions (*scaling dimensions* en anglais) des opérateurs de toutes les representations pertinentes (*relevant representations* en anglais). Le cas $N = 4$ est particulièrement intéressant du fait que, selon une conjecture communément admise, il décrirait une transition de phase dans le modèle projectif réel antiferromagnétique ARP^3. Dans ce modèle, les simulations sur des lattices suggèrent l'existence d'une transition de phase de deuxième ordre alors qu'une approche de théorie des champs effective ne prédit aucun point fixe. Un ensemble d'hypothèses nous permet de contraindre les dimensions de l'opérateur à une région fermée compatible avec les prédictions des simulations sur des lattices. Cette région est toujours présente lorsque l'on considère un système mixte impliquant t et le *singlet* scalaire de dimension la plus basse.

Chapter 3

Architecture

3.1 Reconciling the bias-variance trade-off with over-parametrisation in deep NNs

3.1.1 Overview

As we have seen in Sec. 1.2, one of the main puzzles for theoreticians is to understand the excellent generalization performance of heavily overparametrized deep neural networks able to fit random labels [329]. Such *interpolating* estimators—that can reach zero training error— have attracted a growing amount of theoretical attention in the last few years, see e.g. [4, 31, 131, 213, 232]. Instead, of following the characteristic U-shape curve of the bias-variance trade-off, the performance of deep neural networks [230, 233, 294] as well as other machine learning models [31], follow a *double descent* curve.

This curve displays two regimes : the *classical* U-curve is superseded at high complexity by a *modern* interpolating regime where the test error decreases monotonically with overparametrization [49]. Between these two regimes, i.e. at the *interpolation threshold* where training error vanishes, a *peak* occurs in absence of regularization, sometimes called the *jamming* peak [294] due to similarities with a well-studied phenomenon in the Statistical Physics litterature [93, 94, 164, 193, 241, 326]. Some mechanisms playing an important role for the good performance of deep neural networks in the overparametrised regime include the implicit regularization of stochastic gradient descent which allows to converge to the minimum norm solution, and the convergence to mean-field limits [4, 11, 63, 212, 264]. However, a complete picture of why overparametrised networks perform well remains lacking.

In this subsection, we present a detailed investigation of the double descent phenomenon, and its theoretical explanation in terms of bias and variance in the *lazy* regime [63] introduced in Chap. 1.2. As we discussed, in this regime,

neural networks behave like kernel methods [74, 145, 175] or equivalently random projection methods [11, 63, 253]. This mapping makes the training analytically tractable, allowing, for example, to prove convergence to zero error solutions in overparametrized settings.

Optimization plays an important role in neural networks, and in particular for the double descent phenomenon, by inducing implicit regularization [234] and fluctuations of the learnt estimator [102]. Disentangling the variance stemming from the randomness of the optimization process from that the variance due to the randomness of the dataset is a crucial step towards a unified picture, as suggested in [232]. Here, we address this issue and attempt to reconcile the behavior of bias and variance with the double descent phenomenon by providing a precise and quantitative theory in the *lazy* regime.

We focus on an analytically solvable model of random features (RF) defined in Eq. 1.9, introduced by [253]. As a reminder, RF can be viewed either as a randomized approximation to kernel ridge regression, or as a two-layer neural network whose first layer contains fixed random weights. The latter providing a simple model for lazy learning via Eq. 1.11. In this setting, we quantitatively disentangle the contributions to the test error of the bias and the various sources of variance of the estimator, stemming from the sampling of the dataset, from the additive noise corrupting the labels, and from the initialization of the random feature vectors. Our theoretical analysis also provides a sharp asymptotic formula for the effect of *ensembling* (averaging the predictions of indepently initialized estimators) on these various terms. We show in particular how the over-fitting peak at the interpolation threshold can be attenuated by ensembling, as observed in real neural networks [102]. We also compare the effect of ensembling, overparametrizing and optimally regularizing. The conclusions stemming from this analysis include that the over-fitting near the interpolation threshold is entirely due to the variances due to the additive noise in the ground truth and the initialization of the random features. Also, the data sampling variance and the bias both display a phase transition at the interpolation threshold, and remain constant in the overparametrized regime. Hence, the benefit of ensembling and overparametrization beyond the interpolation threshold is solely due to a reduction of the noise and initialization variances. Finally, we present numerical results on a classic deep learning scenario in the lazy learning regime to show that these findings, obtained for simple random features and i.i.d. data, are relevant to realistic setups involving correlated random features and realistic data. The analytical results we present are obtained using a heuristic method from Statistical Physics called the Replica Method [217], which despite being non-rigorous has shown its remarkable efficacy in many machine

learning problems [3, 93, 285, 327] and random matrix topics, see e.g. [6, 195, 301]. The rigorous proof of these result, for arbitrary loss function is given in [201].

Related work The analysis in this subsection builds on two contributions by Geiger *et al.* [102], and Mei and Montanari [213]. Geiger et al. [102] carried out a series of experiments in order to shed light on the generalization properties of neural networks. The work discussed here, d'Ascoli et al. [76], carried out during my PhD, is inspired by their observations and scaling theory about the role of the variance due to the random initialization of the weights in the double-descent curve. They argued that the decrease of the test error in the limit of very wide networks is due to this source of variance, which vanishes inversely proportional to the width of the network. They then used ensembling to empirically support these findings in more realistic situations. Also related is the work of Neal et al. [232] who empirically disentangles the various sources of variance in the process of training deep neural networks.

On the analytical side, we build on the results of Mei and Montanari [213], which provide a precise expression of the test error of the RF model in the high-dimensional limit where the number of random features, the dimension of the input data and the number of data points are sent to infinity with their relative ratios fixed. The double descent was also studied analytically for various types of linear models, both for regression [4, 32, 131, 229] and classification [80, 106, 155]. An example of a practical method that uses ensembling in kernel methods is detailed in [84]. Note that this analysis performs an average over the sampling of the random feature vectors in contrast to [331] where the average is taken over the sampling of the data set.

Reproducibility The codes necessary to reproduce the results in this subsection and obtain new ones are given at https://github.com/mariaref/Random_Features.git.

Model

This work is centered around the RF model first introduced in [253] defined in Eq. 1.9. Although simpler settings such as linear regression display the double descent phenomenology [131], this model is more appealing in several ways. First, the presence of two layers allows to freely disentangle the dimensionality of the input data from the number of parameters of the model. Second, it closely relates to the lazy regime of neural networks, as described above. Third, and most importantly for our specific study, the randomness of the fixed first layer weights mimics the randomness due to weight initialization in neural networks.

Let us repeat Eq. 1.9 which formulates the RF model as a two-layer neural

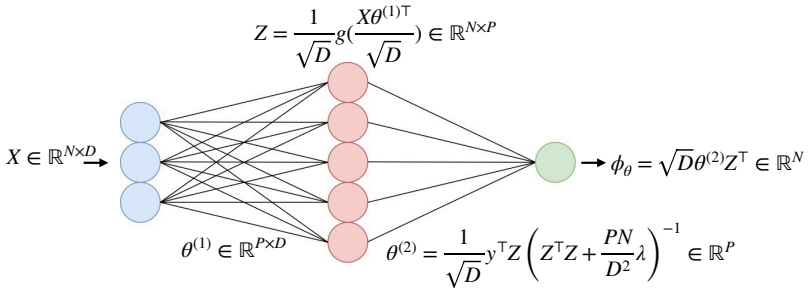


Figure 3.1: Illustration of a Random Feature (RF) network. The first layer weights are fixed and initialized as i.i.d. centered Gaussian variables of unit variance. The second layer weights are trained via ridge regression.

network whose first layer contains fixed random weights (see Fig. 3.1):

$$\phi_\theta(x) = \sum_{i=1}^{P} w_i^{(2)} g(w_i^{(1)} x). \tag{3.1}$$

As in the introduction, $w_i^{(1)}$ is the i^{th} *random feature vector*, i.e the i^{th} column of the random feature matrix $w^{(1)} \in \mathrm{R}^{P\times D}$ whose elements are drawn i.i.d from $\mathcal{N}(0,1)$ and g is a pointwise activation function, which we will take to be ReLU : $x \mapsto \max(0,x)$. For convenience, we define the output of the network as $\phi(x) \equiv \hat{y}$, the post-activation of the first layer as:

$$Z \equiv \frac{g(a^{(1)})}{\sqrt{D}} = \frac{g(\frac{X\theta^{(1)\top}}{\sqrt{D}})}{\sqrt{D}}. \tag{3.2}$$

The training data is collected in a matrix $\boldsymbol{X} \in \mathrm{R}^{N\times D}$ whose elements are drawn i.i.d from $\mathcal{N}(0,1)$. We assume that the training labels y are given by a linear ground truth corrupted by some additive Gaussian noise:

$$\begin{aligned} &y_\mu = \langle \boldsymbol{\beta}, \boldsymbol{X}_\mu \rangle + \epsilon_\mu, \quad ||\boldsymbol{\beta}|| = F, \quad \epsilon_\mu \sim \mathcal{N}(0,\tau), \\ &\mathrm{SNR} = F/\tau. \end{aligned} \tag{3.3}$$

The generalization to non-linear functions can also be performed as in [213].

The second layer weights, i.e the elements of $w^{(2)}$, are fixed by the means of

ridge regression i.e. by minimising the regularised squared loss:

$$\begin{aligned}\mathcal{L}^{\mathrm{RF}}_{\mathrm{emp}}(w^{(2)}) &\equiv \frac{1}{N}\sum_{\mu=1}^{N}\left(y_\mu - \sum_{i=1}^{P} w_i^{(2)} g\left(\frac{\langle w_i^{(1)}, \boldsymbol{X}_\mu\rangle}{\sqrt{D}}\right)\right)^2 + \frac{P\lambda}{D}\|w^{(2)}\|_2^2, \\ \hat{\theta}^{(2)} &\equiv \underset{w^{(2)}\in\mathrm{R}^P}{\arg\min}\ \mathcal{L}^{\mathrm{RF}}_{\mathrm{emp}}(w^{(2)}).\end{aligned} \tag{3.4}$$

Note that as $P\to\infty$, this is equivalent to kernel ridge regression with respect to the following kernel:

$$K\left(\boldsymbol{x},\boldsymbol{x}'\right) = \mathbb{E}_{w^{(1)}\sim\mathcal{P}}\left[g(\langle \boldsymbol{x}, w^{(1)}\rangle/\sqrt{D})g\left(\left\langle \boldsymbol{x}', w^{(1)}\right\rangle/\sqrt{D}\right)\right],$$

where $\mathcal{P} = \mathrm{Unif}\left(\mathbb{S}^{D-1}(\sqrt{D})\right)$.

The key quantity of interest is the *test error*, or population loss 1.1, of this model, defined as the mean square error evaluated on a fresh sample $x\sim\mathcal{N}(0,1)$ corrupted by a new noise $\tilde{\epsilon}$:

$$\mathcal{L}^{\mathrm{RF}}_{\mathrm{pop}} = \mathbb{E}_x\left[\left(\langle\boldsymbol{\beta},\boldsymbol{x}\rangle + \tilde{\epsilon} - \phi(\boldsymbol{x})\right)^2\right], \quad \tilde{\epsilon}\sim\mathcal{N}(0,\tilde{\tau}). \tag{3.5}$$

3.1.2 Analytical results

In this subsection, we present the main result of this work, which is an analytical expression for the terms appearing in the decomposition of the test error.

Decomposition of the test error

The test error can be decomposed into its bias and variance components:

$$\mathbb{E}_{w^{(1)},\boldsymbol{X},w^{(2)},\epsilon}\left[\mathcal{L}^{\mathrm{RF}}_{\mathrm{pop}}\right] = \mathcal{E}_{\mathrm{Noise}} + \mathcal{E}_{\mathrm{Init}} + \mathcal{E}_{\mathrm{Samp}} + \mathcal{E}_{\mathrm{Bias}} + \tilde{\tau}^2. \tag{3.6}$$

The first three terms contribute to the variance, the fourth is the bias, and the final term $\tilde{\tau}^2$ is simply the Bayes error. It does not play any role and will be set to zero in the rest of the paper: the only reason it was included is to avoid confusion with $\mathcal{E}_{\mathrm{Noise}}$ defined below.

Noise variance: The first term is the variance associated with the additive noise corrupting the labels of the dataset which is learnt, $\boldsymbol{\epsilon}$:

$$\mathcal{E}_{\mathrm{Noise}} = \mathbb{E}_{\boldsymbol{x},\boldsymbol{X},w^{(1)}}\left[\mathbb{E}_\epsilon\left[\phi_w(\boldsymbol{x})^2\right] - \left(\mathbb{E}_\epsilon\left[\phi_w(\boldsymbol{x})\right]\right)^2\right]. \tag{3.7}$$

Initialization variance: The second term encodes the fluctuations stemming from

the random initialization of the random feature vectors, $w^{(1)}$:

$$\mathcal{E}_{\text{Init}} = \mathbb{E}_{\boldsymbol{x},\boldsymbol{X}}\left[\mathbb{E}_{w^{(1)}}\left[\mathbb{E}_{\epsilon}\left[\phi_w(\boldsymbol{x})\right]^2\right] - \mathbb{E}_{w^{(1)},\epsilon}\left[\phi_w(\boldsymbol{x})\right]^2\right]. \tag{3.8}$$

Sampling variance: The third term measures the fluctuations due to the sampling of the training data, $\boldsymbol{X}$:

$$\mathcal{E}_{\text{Samp}} = \mathbb{E}_{\boldsymbol{x}}\left[\mathbb{E}_{\boldsymbol{X}}\left[\mathbb{E}_{w^{(1)},\boldsymbol{\epsilon}}\left[\phi_w(\boldsymbol{x})\right]^2\right] - \mathbb{E}_{\boldsymbol{X},w^{(1)},\boldsymbol{\epsilon}}\left[\phi_w(\boldsymbol{x})\right]^2\right]. \tag{3.9}$$

Bias: Finally, the fourth term is the bias, i.e. the error that remains once all the sources of variance have been averaged out. It can be understood as the approximation error of our model and takes the form:

$$\mathcal{E}_{\text{Bias}} = \mathbb{E}_{\boldsymbol{x}}\left[\left(\langle\boldsymbol{\beta},\boldsymbol{x}\rangle - \mathbb{E}_{\boldsymbol{X},w^{(1)},\epsilon}\left[\phi_w(\boldsymbol{x})\right]\right)^2\right]. \tag{3.10}$$

Note that since we are performing deterministic ridge regression, the noise induced by SGD, which can play an important role outside the lazy regime for deep neural networks, cannot be captured.

Main result

Consider the high-dimensional limit where the input dimension D, the hidden layer dimension P (which is equal to the number of parameter in our model) and the number of training points N go to infinity with their ratios fixed:

$$N, P, D \to \infty, \quad \frac{P}{D} = O(1), \quad \frac{N}{D} = O(1). \tag{3.11}$$

We obtain the following result:

$$\mathbb{E}_{\boldsymbol{x},\boldsymbol{\epsilon},w^{(1)},\boldsymbol{X}}\left[\langle\boldsymbol{\beta},\boldsymbol{x}\rangle\phi_w(\boldsymbol{x})\right] = F^2\Psi_1, \tag{3.12}$$

$$\mathbb{E}_{\boldsymbol{x},w^{(1)},\boldsymbol{X}}\left[\mathbb{E}_{\boldsymbol{\epsilon}}\left[\phi_w(\boldsymbol{x})\right]^2\right] = F^2\Psi_2^v, \tag{3.13}$$

$$\mathbb{E}_{\boldsymbol{x},w^{(1)},\boldsymbol{X}}\left[\mathbb{E}_{\boldsymbol{\epsilon}}\left[\phi_w(\boldsymbol{x})^2\right] - \mathbb{E}_{\boldsymbol{\epsilon}}\left[\phi_w(\boldsymbol{x})\right]^2\right] = \tau^2\Psi_3^v, \tag{3.14}$$

$$\mathbb{E}_{\boldsymbol{x},\boldsymbol{X}}\left[\mathbb{E}_{\boldsymbol{\epsilon},\boldsymbol{\Theta}}\left[\phi_w(\boldsymbol{x})\right]^2\right] = F^2\Psi_2^e, \tag{3.15}$$

$$\mathbb{E}_{\boldsymbol{x},\boldsymbol{X}}\left[\mathbb{E}_{\boldsymbol{\epsilon},\boldsymbol{\Theta}}\left[\phi_w(\boldsymbol{x})^2\right] - \mathbb{E}_{\boldsymbol{\epsilon},w^{(1)}}\left[\phi_w(\boldsymbol{x})\right]^2\right] = \tau^2\Psi_3^e, \tag{3.16}$$

$$\mathbb{E}_{\boldsymbol{x}}\left[\mathbb{E}_{\boldsymbol{\epsilon},w^{(1)},\boldsymbol{X}}\left[\phi_w(\boldsymbol{x})\right]^2\right] = F^2\Psi_2^d, \tag{3.17}$$

where the terms $\{\Psi_1, \Psi_2^v, \Psi_3^v, \Psi_2^e, \Psi_3^e, \Psi_2^d\}$, whose full analytical expressions are deferred to Appendix A.2, are computed following the steps below:

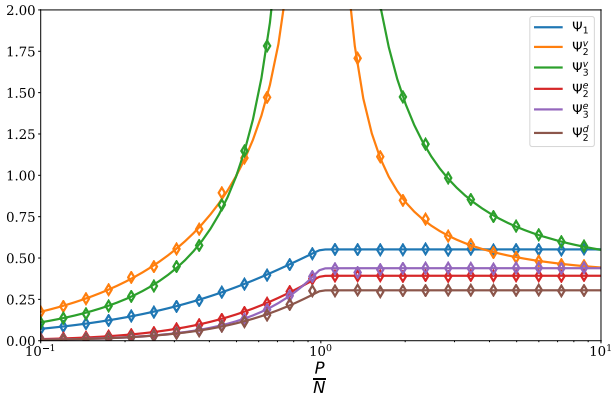


Figure 3.2: All the terms entering our analytical expressions for the decomposition of the error (Equations 3.12-3.17), as function of the overparametrization ratio P/N for $\lambda = 10^{-5}$ and $N/D = 1$. Numerical estimations obtained for a finite size $D = 200$ (diamonds) shows that the asymptotic predictions are extremely accurate even at moderate sizes.

1. *Mapping to a random matrix theory problem.* The first step is to express the right-hand sides of Equations 3.12-3.17 as traces over random matrices. This is achieved by replacing our model with its asymptotically equivalent Gaussian covariate model [213], in which the non-linearity of the activation function is encoded as an extra noise term. This enables to take the expectation value with respect to the test sample $\boldsymbol{x}$.
2. *Mapping to a statistical physics model.* The random matrix theory problem resulting from the solution of ridge regression (3.4) involves inverse random matrices. In order to evaluate their expection value, we use the formula:

$$M_{ij}^{-1} = \lim_{n\to 0} \int \prod_{\alpha=1}^{n} \prod_{i=1}^{D} d\eta_i^\alpha \eta_i^1 \eta_j^1 \, \mathrm{e}^{-\frac{1}{2}\eta_i^\alpha M_{ij} \eta_j^\alpha},$$

   which is based on the Replica Trick [51, 217]. The Gaussian integrals over $\boldsymbol{\epsilon}, w^{(1)}, \boldsymbol{X}$ can then be straightforwardly performed and lead to a Statistical Physics model for the auxiliary variables η_i^α.
3. *Mean-Field Theory.* The model for the η_i^α variables can then be solved by introducing as order parameters the $n \times n$ overlap matrices $Q^{\alpha\beta} = \frac{1}{P}\sum_{i=1}^{P} \eta_i^\alpha \eta_i^\beta$ and using replica theory [217], see Appendix A.3 for the detailed computation*.

The Ψ's may also be estimated numerically at finite size by evaluating the traces of the random matrices appearing in the Gaussian covariate model at the end of

*In order to obtain the asymptotic formulas for the Ψ's we need to compute (what are called in the Statistical Physics jargon) fluctuations around mean-field theory.

step 1. Figure 3.2 shows that results thus obtained are in excellent agreement with the asymptotic expressions even at moderate sizes, e.g. $D = 200$, proving the robustness of steps 2 and 3, which differ from the approach presented in [213].

The indices v, e, d in $\{\Psi_1, \Psi_2^v, \Psi_3^v, \Psi_2^e, \Psi_3^e, \Psi_2^d\}$ stand for *vanilla*, *ensemble* and *divide and conquer*. The *vanilla* terms are sufficient to obtain the test error of a single RF model and were computed in [213]. The *ensemble* and *divide and conquer* terms allow to obtain the test error obtained when averaging the predictions of several different learners trained respectively on the same dataset and on different splits of the original dataset (see Sec. 3.1.4). Figure 3.2 shows that the *vanilla* terms exhibit a radically different behavior from the others: at vanishing regularization, they diverge at $P = N$ then decrease monotonically, whereas the others display a kink followed by a plateau. This behavior will be key to the following analysis.

3.1.3 Analysis of Bias and Variances

The results of the previous subsection, allow to rewrite the decomposition of the test error as follows:

$$\mathcal{E}_{\text{Noise}} = \tau^2 \Psi_3^v, \tag{3.18}$$

$$\mathcal{E}_{\text{Init}} = F^2(\Psi_2^v - \Psi_2^e), \tag{3.19}$$

$$\mathcal{E}_{\text{Samp}} = F^2\left(\Psi_2^e - \Psi_2^d\right), \tag{3.20}$$

$$\mathcal{E}_{\text{Bias}} = F^2\left(1 - 2\Psi_1 + \Psi_2^d\right). \tag{3.21}$$

These contributions, together with the test error, are shown in Fig. 3.3 in the case of small (top) and large (bottom) regularization.

Interpolation Threshold

The peak at the interpolation threshold is completely due to noise and initialization variance, which both diverge at vanishing regularization. In contrast, the sampling variance and the bias remain finite and exhibit a phase transition at $P = N$, which is revealed by a kink at vanishing regularization. Adding regularization smooths out these singular behaviours: it removes the divergence and irons out the kink.

Overparametrized regime

In the overparametrized regime, the sampling variance and the bias do not vary substantially (they remain constant for vanishing regularization). The decrease of the test error is entirely due to the decrease of the noise and initalization variances for $P > N$. In the limit $P/N \to \infty$, the initialization variance vanishes, whereas there

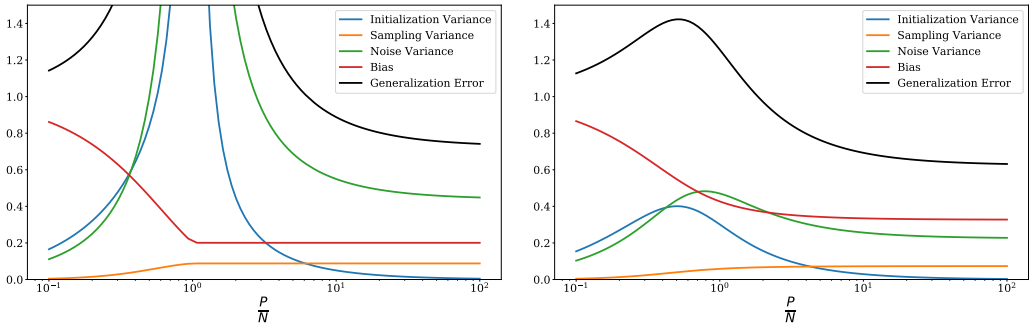


Figure 3.3: Decomposition of the test error into the bias and the various sources of variance as function of the overparametrization ratio P/N for $N/D = 1$, SNR $= F/\tau = 1$. Two values of the regularization constant are used: $\lambda = 10^{-5}$ (**left**) and $\lambda = 10^{-1}$ (**right**). Notice the contrasting behaviors at the interpolation threshold: the noise and initialization variances diverge then decrease monotonically whereas the sampling variance and the bias display a kink followed by a plateau. These singular behaviors are smoothed out by regularization.

remains an irreducible noise variance.

Discussion

In conclusion, we find that the *origin of the double descent curve lies in the behavior of noise and initialization variances*. The benefit of overparametrizing stems only from reducing these two contributions.

These results are qualitatively similar to the empirical decomposition of [232] for real neural networks. The divergence of the test error as $(P/N - 1)^{-1}$ at the interpolation threshold is in agreement with the results of [213]*. As for the decrease of the test error in the over-parametrized regime, we find consistently with the scaling arguments of [102] that the initialization error asymptotically decays to zero inversely proportional to the width (see Appendix A.1 for more details). The interpretation of our results differ from those of [213] where the authors relate the over-fitting peak occurring at $P = N$ to a divergence in both the variance and the bias terms. This is due to the fact the bias term, as defined in that paper, also includes the initialization variance†. When the two are disentangled, it becomes clear that it is only the latter which is responsible for the divergence: the bias is, in fact, well-behaved at $P = N$.

*Note that for classification problems the singularity is different [102].

†For a given set of random features this is legitimate, but from the perspective of lazy learning the randomness in the features corresponds to the one due to initialization, which is an additional source of variance.

Intuition

The phenomenology described above can be understood by noting that the model essentially performs linear regression, learning a vector $\boldsymbol{a} \in \mathrm{R}^P$ on a projected dataset $\boldsymbol{Z} \in \mathrm{R}^{N \times P}$ (the activations of the hidden nodes of the RF network). Since $\boldsymbol{a}$ is constrained to lie in the space spanned by $\boldsymbol{Z}$, which is of dimension $\min(N, P)$, the model gains expressivity when P increases while staying smaller than N.

At $P = N$, the problem becomes fully determined: the data is perfectly interpolated for vanishing λ. Two types of noise are overfit: (i) the *stochastic noise* corrupting the labels, yielding the divergence in noise variance, and (ii) the *deterministic noise* stemming from the non-linearity of the activation function which cannot be captured, yielding the divergence in initialization variance. However, by further increasing P, the noise is spread over more and more random features and is effectively averaged out. Consequently, the test error decreases again as P increases.

When we make the problem deterministic by averaging out all sources of randomness, i.e. by considering the bias, we see that increasing P beyond N has no effect whatsoever. Indeed, the extra degrees of freedom, which lie in the null space of $\boldsymbol{Z}$, do not provide any extra expressivity: at vanishing regularization, they are killed by the pseudo-inverse to reach the minimum norm solution. For non-vanishing λ, a similar phenomenology is observed but the interpolation threshold is reached slightly after $P = N$ since the expressivity of the learner is lowered by regularization.

3.1.4 On the effect of ensembling

In order to further study the effect of the variances on the test error, we follow [102] and study the impact of ensembling. In the lazy regime of deep neural networks, the initial values of the weights only affect the gradient at initialization, which corresponds to the vector of random features. Hence, we can study the effect of ensembling in the lazy regime by averaging the predictions of RF models with independently drawn random feature vectors.

Expression of the test error

Consider a set of $K > 1$ RF networks whose first layer weights are drawn independently. These networks are trained independently on the *same* training set. In the analogy outlined above, they correspond to K independent inizializations of the neural network. At the end of training, one obtains K estimators $\{\phi_{w_k}\}_{k \in [K]}$. When a new sample x is presented to the system, the output is taken to be the

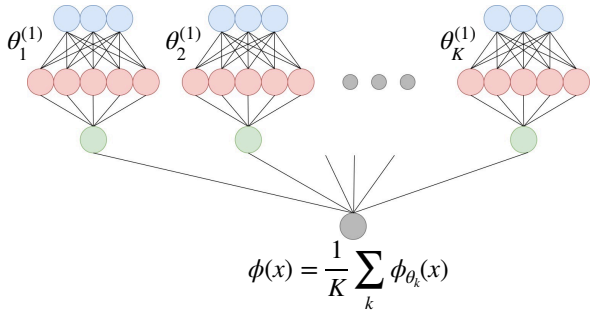


Figure 3.4: Illustration of the ensembling procedure over K Random Features networks trained on the same data but with different realizations of the first layer, $\{w_1^{(1)}, ...w_K^{(1)}\}$.

average over the outputs of the K networks, as illustrated in Fig. 3.4. By expanding the square and taking the expectation with respect to the random initalizations, the test error can then be written as:

$$\begin{aligned}\mathbb{E}_{\{w^{(1)k}\}}\left[\mathcal{L}^{\text{RF}}_{\text{pop}}\right] &= \mathop{\mathbb{E}}_{\boldsymbol{x}\{w^{(1)k}\}}\left[\left(\langle\boldsymbol{\beta},\boldsymbol{x}\rangle - \frac{1}{K}\sum_k \phi_{w_k}(\boldsymbol{x})\right)^2\right] \\ &= \mathbb{E}_{\boldsymbol{x}}\left[\langle\boldsymbol{\beta},\boldsymbol{x}\rangle^2\right] - \frac{2}{K}\sum_{i=1}^{K}\mathbb{E}_{\boldsymbol{x},w_i}\left[\langle\boldsymbol{\beta},\boldsymbol{x}\rangle\phi_{w_i}(\boldsymbol{x})\right] + \frac{1}{K^2}\sum_{i,j=1}^{K}\mathbb{E}_{\boldsymbol{x},w_i,w_j}\left[\phi_{w_i}(\boldsymbol{x})\phi_{w_j}(\boldsymbol{x})\right].\end{aligned} \tag{3.22}$$

The key here is to isolate in the double sum the $K(K-1)$ *ensemble* terms $i \neq j$, which involve two different initalizations and yield $\mathbb{E}_{\boldsymbol{x}}\left[\mathbb{E}_{w^{(1)}}\left[\phi_{w^{(1)}}(\boldsymbol{x})\right]^2\right]$, from the K *vanilla* terms which give $\mathbb{E}_{\boldsymbol{x},\boldsymbol{\Theta}}\left[\phi_{w^{(1)}}(\boldsymbol{x})^2\right]$. This allows to express the test error in terms of the quantities defined in (3.12) to (3.17) and leads to the analytic formula for the test error valid for any $K \in \mathbb{N}$:

$$\mathbb{E}_{\{w^{(1)k}\},\boldsymbol{X},w^{(2)},\epsilon}\left[\mathcal{L}^{\text{RF}}_{\text{pop}}\right] = F^2\left(1-2\Psi_1^v\right) + \frac{1}{K}\left(F^2\Psi_2^v + \tau^2\Psi_3^v\right) + \left(1-\frac{1}{K}\right)\left(F^2\Psi_2^e + \tau^2\Psi_3^e\right). \tag{3.23}$$

We see that ensembling amounts to a linear interpolation between the *vanilla* terms Ψ_2^v, Ψ_3^v, for $K = 1$, and the *ensemble* terms Ψ_2^e, Ψ_3^e for $K \to \infty$.

The effect of ensembling on the double descent curve is shown in Fig. 3.5. As K increases, the overfitting peak at the interpolation threshold is diminished. This observation is very similar to the empirical findings of [102] in the context of real neural networks. Our analytic expression agrees with the numerical results

obtained by training RF models, even at moderate size $D = 200$.

Note that a related procedure is the *divide and conquer* approach, where the dataset is partitioned into K splits of equal size and each one of the K differently initialized learners is trained on a distinct split. This approach was studied for kernel learning in [84], and is analyzed within our framework in Appendix A.1.

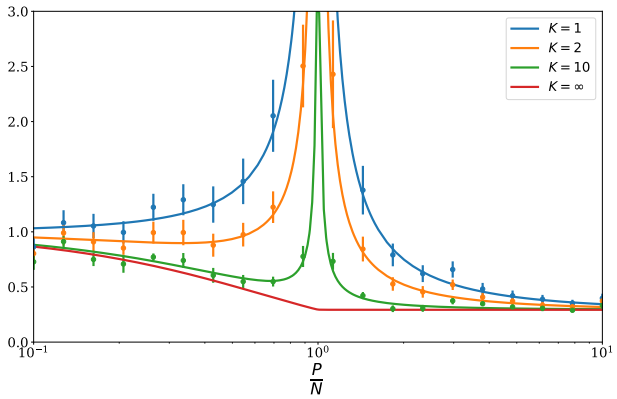


Figure 3.5: Test error when ensembling $K = 1, 2, 10$ differently initialized RF models as function of the overparametrization ratio P/N. We fixed $\lambda = 10^{-5}$, $N/D = 1$, $SNR = 10$. For comparison, we show the results of numerical simulations at finite $D = 200$: the vertical bars depict the standard deviation over 10 runs. The variability observed here was absent in Fig. 3.2 because we are considering the true RF model rather than the asymptotically equivalent Gaussian covariate model. This shows that most of this variability is caused by the finite-size deviation between the two models. Note that our analytic expression (3.23) gives us access to the limit $N \to \infty$, where the divergence at $P = N$ is entirely suppressed.

Ensembling reduces the double trouble

The bias-variance decomposition of the test error makes the suppression of the divergence explicit. The bias and variances contribution read for the averaged estimator:

$$\mathcal{E}_{\text{Noise}} = \tau^2 \left(\Psi_3^e + \frac{1}{K} (\Psi_3^v - \Psi_3^e) \right), \tag{3.24}$$

$$\mathcal{E}_{\text{Init}} = \frac{F^2}{K} (\Psi_2^v - \Psi_2^e), \tag{3.25}$$

$$\mathcal{E}_{\text{Samp}} = F^2 \left(\Psi_2^e - \Psi_2^d \right), \tag{3.26}$$

$$\mathcal{E}_{\text{Bias}} = F^2 \left(1 - 2\Psi_1 + \Psi_2^d \right). \tag{3.27}$$

These equations show that ensembling only affects the noise and initialization variances. In both cases, their divergence at the interpolation threshold (due to

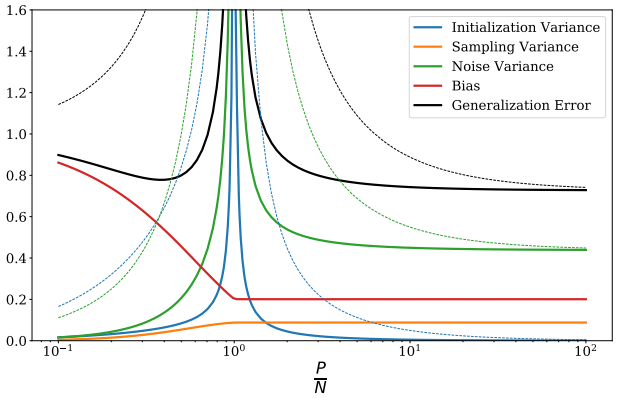


Figure 3.6: Decomposition of the test error into the bias and the various sources of variance as function of the overparametrization ratio P/N for $\lambda = 10^{-5}$, $N/D = 1$, SNR $= 1$. The thin dashed lines are taken from Fig. 3.3 (top) where we had $K = 1$; the thick solid lines show how ensembling at $K = 10$ suppressed the divergences of the noise and initialization variances.

Ψ_2^v, Ψ_3^v) is suppressed as $1/K$, see Fig. 3.6 for an illustration. At $P > N$, ensembling and overparametrizing have a very similar effect: they wipe out these two troubling sources of randomness by averaging them out over more random features. Indeed, we see in Fig. 3.5 that in this overparametrized regime, sending $K \to \infty$ has the same effect as sending $P/N \to \infty$: in both cases the system approaches the kernel limit. At $P < N$, this is not true: as shown in [146], the $K \to \infty$ predictor still operates in the kernel limit, but with an effective regularization parameter $\tilde{\lambda} > \lambda$ which diverges as $P/N \to 0$. This (detrimental) implicit regularization increases the test error.

Ensembling vs. overparametrization

As we have shown, ensembling and overparametrizing have similar effects in the lazy regime. But which is more powerful: ensembling K models, or using a single model with K times more features? The answer is given in Fig. 3.7 for $K = 2$ where we plot our analytical results while varying the number of data points, N. Two observations are particularly interesting. First, overparametrization shifts the interpolation threshold, opening up a region where ensembling outperforms overparametrizing. Second, overparametrization yields a higher asymptotic improvement in the large dataset limit $N/D \to \infty$, but the gap between overparametrizing and ensembling is reduced as P/D increases. At $P \gg D$, where we are already close to the kernel limit, both methods yield a similar improvement. Note that from the point of view of efficiency, ridge regression involves the inversion of a $P \times P$ matrix, therefore ensembling is significantly more efficient.

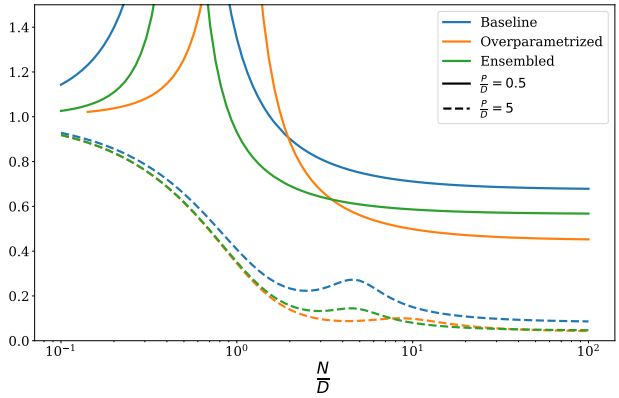


Figure 3.7: Comparison of the test error of a RF model (blue) with that obtained by doubling the number features (orange) or ensembling over two initializations of the features (green), as function of N/D. The parameters are $\lambda = 10^{-5}$, SNR $= 10$, $P/D = 0.5$ (solid lines) and $P/D = 5$ (dashed lines).

Ensembling vs. optimal regularization

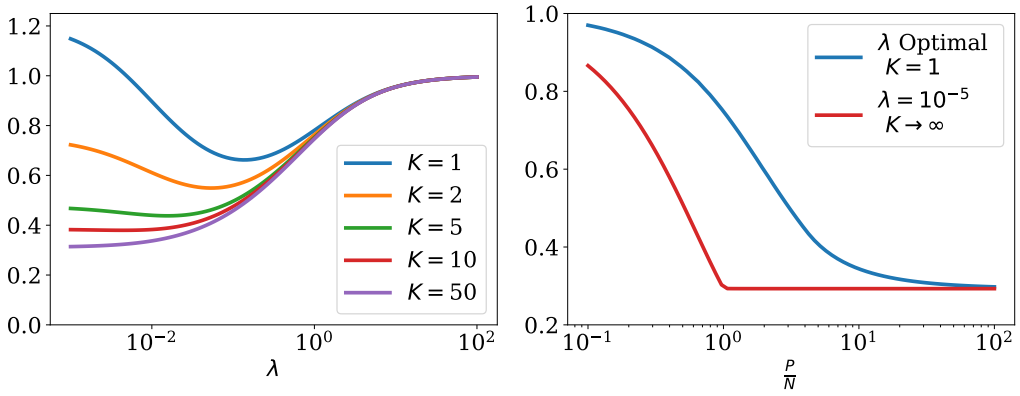


Figure 3.8: **Left:** Test error as a function of λ for various values of K and parameters $P/D = 2$, $N/D = 1$, SNR $= 10$. **Right:** Comparison of test error for an optimal regularized system with $K = 1$ and the system with $K \to \infty$ with $\lambda = 10^{-5}$. Optimization performed over 50 values of λ from 10^{-5} to 10^{2}. Parameters are $N/D = 1$, SNR $= 10$.

In all the results presented above, we keep the regularization constant λ fixed. However, by appropriately choosing the value of λ at each value of P/N, the performance is improved. As Fig. 3.7 (left) reveals, the optimal value of λ decreases with K since the minimum of the test error shifts to the left when increasing K. In other words, ensembling is best when the predictors one ensembles upon are individually under-regularized, as was observed previously for kernel learning in [332]. Figure 3.8 (right) shows that an infinitely ensembled model ($K \to \infty$) always

performs better than an optimally regularized single model ($K = 1$).

3.1.5 Numerical experiments on neural networks

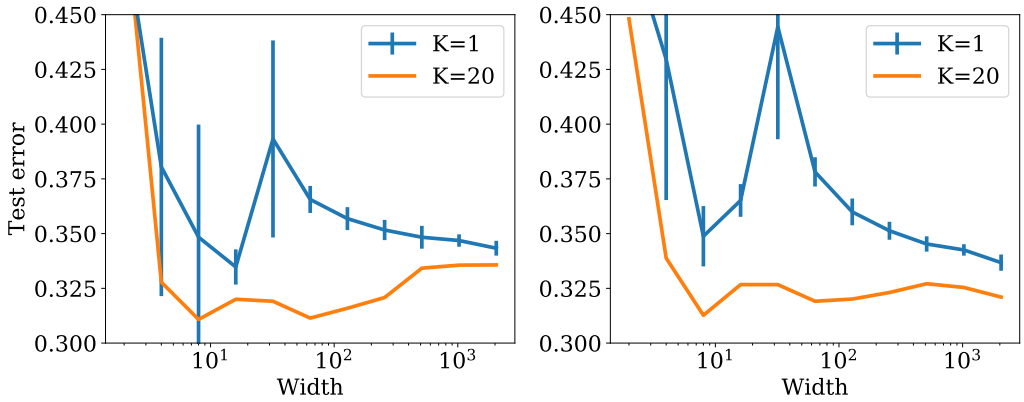


Figure 3.9: Test error on the binary 10-PCA CIFAR10 as function of the number of nodes per layer of the 5-layer neural network trained until convergence with the full-batch *Adam*. We compare the test error of a single predictor ($K = 1$), averaged over 20 initializations of the weights (the standard deviation is depicted as vertical bars), with the ensembling predictor at $K = 20$. **Left:** $\alpha = 10$. **Right:** $\alpha = 100$, where we are closer to the lazy regime and the ensembling curve flattens beyond the interpolation threshold, which occurs around 30 nodes per layer.

Finally, we investigate whether the phenomenology described here holds for realistic neural networks learning real data in the lazy regime. We follow here the protocol used in [102, 104] and train a 5-layer fully-connected network on the CIFAR-10 dataset. We keep only the first ten PCA components of the images, and divide the images in two classes according to the parity of the labels. We perform 10^5 steps of full-batch gradient descent with the *Adam* optimizer and a learning rate of 0.1, and scale the weights as prescribed in [145].

We gradually go from the usual *feature learning* regime to the *lazy learning* regime using the trick introduced in [63], which consists in scaling the output of the network by a factor α and replacing the learning function $f_\theta(\boldsymbol{x})$ by $\alpha(f_\theta(\boldsymbol{x}) - f_{\theta_0}(\boldsymbol{x}))$. For $\alpha \gg 1$, one must have that $\theta - \theta_0 \sim 1/\alpha$ in order for the learning function to remain of order one. In other words, the weights are forced to stay close to their initialization, hence the name *lazy learning*.

Results are shown in Fig. 3.9. Close to the lazy regime ($\alpha = 100$, right panel), a very similar behavior as the RF model is observed. The test error curve* obtained when ensembling $K = 20$ independently initialized networks becomes roughly

*Note that we are considering a binary classification task here: the error is defined as the fraction of misclassified images.

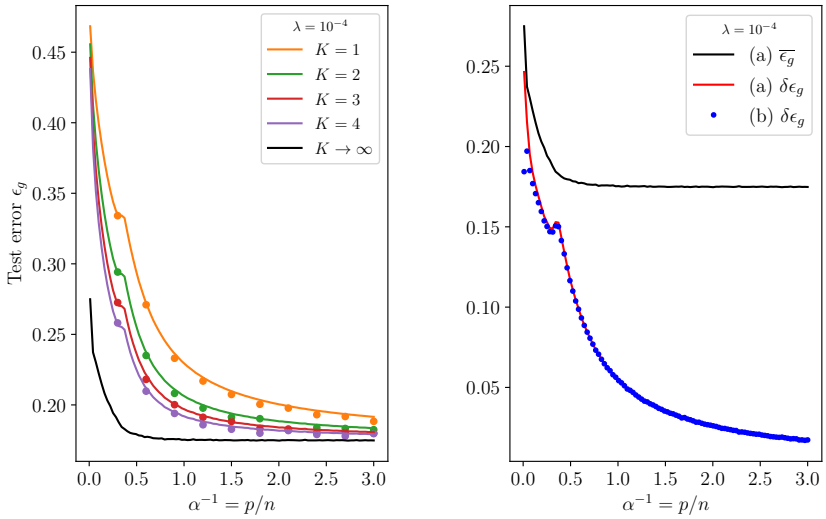


Figure 3.10: *Left.* Test error for logistic regression with $\lambda = 10^{-4}$ and different values of K as function of $P/N = 1/\alpha$ with $N/D = 2$ and $\rho = 1$. Dots represent the average of the outcomes of 10^3 numerical experiments. Here we adopted $g(x) = \mathrm{erf}(x)$ and estimator $\phi(v) = \mathrm{sign}(\sum_k v_k)$. *Right.* Decomposition of the $K = 1$ test error $\mathcal{L}_{\mathrm{pop}} = \overline{\mathcal{L}}_{\mathrm{pop}} + \delta\mathcal{L}_{\mathrm{pop}}$ for the estimator **(a)**, with $N/D = 2$ and $\lambda = 10^{-4}$. We plot also the contribution $\delta\mathcal{L}_{\mathrm{pop}}$ corresponding to the estimator **(b)**: we numerically observed that such decomposition coincides in the two cases. Note also the presence of a kink in $\delta\mathcal{L}_{\mathrm{pop}}$ at the interpolation transition.

flat after the interpolation threshold (which here is signalled by the peak in the test accuracy). As we move away from the lazy regime ($\alpha = 10$, left panel), the same curve develops a *dip* around the interpolation threshold and increases beyond $P > N$ as observed previously in [102]. This may arguably be associated to the beneficial effect of *feature learning*, as discussed in [104] where the transition from lazy to feature learning was investigated.

3.1.6 Extensions

The results derived in the previous section allow to gain intuition on how to reconcile the classical bias-variance tradeoff with the double descent trend of the test error of over-parametrised networks. In particular, we have seen how over-parametrisation improves performance by taming the label noise and initialisation variances.

A slightly finer decomposition of the variance in terms of the different sources of randomness in the problem was later proposed by Adlam and Pennington [2]. Lin and Dobriban [191] show that such decomposition is not unique, and can be more generally understood from the point of view of the *analysis of variance*

(ANOVA) framework. Their conclusions are similar to the previous ones and reveal that the divergence at the interpolation threshold of the error is due to the random noise in the training labels and a joint contribution of the randomness in the choice of weights and of inputs sampled for training.

Sec. 3 only considers the squared L2-loss, or mse, defined in Eq. 3.5 with $L2$ regularisation. The tools employed do not allow to study the case where the networks are learned jointly.

In a recent work, Loureiro et al. [201] tackle these issues by considering the case of general convex loss, any convex regularisation, and jointly trained learners. The estimator is then given by:

$$\phi(x) = \hat{f}\left(\frac{Z_1(x)w_1^{(2)\top}}{\sqrt{P}}, \dots, \frac{Z_K(x)w_K^{(2)\top}}{\sqrt{P}}\right), \tag{3.28}$$

where $\hat{f} : \mathbb{R}^K \to \mathcal{Y}$ is an activation function. In the particular case in which the predictors are *independently trained*, minimisation of the regularised empirical risk fixes the weights $w^{(2)}$ as:

$$w_k^{(2)} = \underset{w\in\mathbb{R}^P}{\arg\min}\left[\frac{1}{N}\sum_{\mu=1}^{N}\ell\left(y^\mu, \frac{Z_k(x^\mu)w^\top}{\sqrt{P}}\right) + \frac{\lambda}{2}\|w\|_2^2\right] \tag{3.29}$$

with a convex loss function $\ell : \mathcal{Y} \times \mathbb{R} \to \mathbb{R}$ (e.g., the logistic loss) and ridge penalty whose strength is given by $\lambda \in \mathbb{R}^+$. The analysis also includes the case in which the learners are jointly trained with a generic convex penalty.

The results not only allow to study binary classification problems with the logistic loss and cross-entropy loss but are also are fully mathematically rigourous. They are proven through a *Approximate Message Passing* (AMP) proof technique [30, 83], leveraging on recently introduced progresses in [107, 200] which enables to capture the full complexity of the problem and obtain the asymptotic joint distribution of the ensemble of predictors.

Although the asymptotic statistics of the single learner, i.e. $K = 1$, in the general case had been studied in [81, 106, 200], they are not enough to quantify the correlation between different learners, induced by the training on the same dataset. Doing so is required to compute, e.g., the test error associated with an ensembling predictor or the impact of fluctuations in the test error. Both these quantities crucially depend on the average correlation between two independent learners which Loureiro et al. [201] give an exact asymptotic characterisation of.

In the general case, we can also understand the role played by fluctuations in the double-descent behaviour i.e. the non-monotonic behaviour of the generalisation

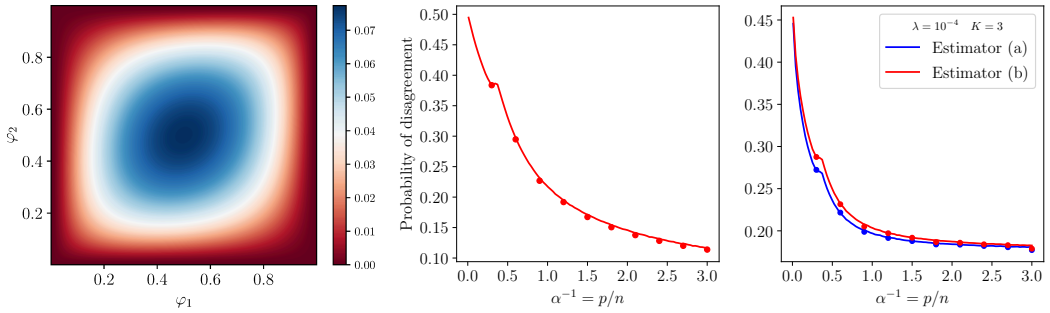


Figure 3.11: *Left.* Joint probability density of the confidence score $\varphi_k(x) = \left(1 + \exp\left(-z_k w_k^{(2)}/\sqrt{P}\right)\right)^{-1}$ of two learners for $P/N \simeq 0.13$. *Center.* Probability that two learners give discordant predictions using logistic regression as function of $P/N = 1/\alpha$ with $N/D = 2$, $\rho = 1$, and $\lambda = 10^{-4}$. *Right.* Test error for logistic regression using the estimators in eq. (3.31) and $K = 3$, with the same parameters. We adopted $g(x) = \mathrm{erf}(x)$. We observe that the test error obtained using **(a)** is always smaller than the one obtained using **(b)**. (*Center and right*) Dots represent the average of the outcomes of 10^3 numerical experiments.

performance of interpolators. To do so, we decompose the error in terms of fluctuation-free term $\overline{\mathcal{L}}_{\text{pop}}$ and a fluctuation term $\delta\mathcal{L}_{\text{pop}}$:

$$\mathcal{L}_{\text{pop}}(K=1) = \overline{\mathcal{L}}_{\text{pop}} + \delta\mathcal{L}_{\text{pop}}. \tag{3.30}$$

This decomposition confirms that, similar to the ridge case, the interpolation peak arises from the model overfitting the particular realisation of the random weights. Fig. 3.10 gives an example for max-margin classification.

Another question of interest in the context of ensembling is: given an ensemble of predictors $\{\phi_{w_k}\}_{k\in[K]}$, what is the best way of combining them to produce a point estimate? In our setting, this amounts to choosing the function $\hat{f}$ in Eq. 3.28. We compare, in terms of generalisation performance, two popular choices, majority vote and score averaging:

$$\textbf{(a)} \qquad \hat{f}(\boldsymbol{v}) = \mathrm{sign}\left(\sum_k v_k\right), \tag{3.31a}$$

$$\textbf{(b)} \qquad \hat{f}(\boldsymbol{v}) = \mathrm{sign}\left(\sum_k \mathrm{sign}(v_k)\right). \tag{3.31b}$$

In a sense, **(a)** provides an estimator based on the average of the individual outputs, whereas **(b)**, which corresponds to a majority rule if K is odd [129], is a function of the average of the estimators of the single learners. For both choices of the estimator we use $\Delta(y, \phi(x)) = \delta_{\hat{y},\phi(x)}$ to measure the test error. Fig. 3.11

(right) compares the test error obtained using **(a)** and **(b)** for $K = 3$ with vanishing regularisation $\lambda = 10^{-4}$. As the estimator **(a)** has better performances than the estimator **(b)**, we conclude that score averaging consistently outperforms the majority vote predictor. However, for a large number of learners $K \gg 1$ these two predictors agree, see Fig. 3.11 (right).

Ensembling can also be used as a tool for uncertainty quantification. In particular, we can connect the correlation between two learners to the probability of disagreement, and show that it decreases with overparametrisation. By comparing with the results in Fig. 3.11 (center), it is evident that the benefit of ensembling in reducing the test error correlates with the tendency of learners to disagree, notably for small P/N, as stressed by Krogh and Vedelsby [163]. Fig. 3.11 (left) provides a full characterisation of the joint probability density of the confidence score between two independent learners. Finally, we observe a constant value of $\overline{\mathcal{L}}_{\text{pop}}$ beyond the interpolation threshold, compatibly with the numerical results of Geiger et al. [105].

Chapter 4

Data

In the previous chapter, we mostly considered i.i.d. Gaussian inputs, i.e. white noise, a crude approximation of real data since, by definition, they are completely structureless. Thus, the analysis was insensitive to particular features in the inputs. In this chapter, we make a step forward and give some intuition on which features in the inputs are relevant for learning a task in the feature learning regime. We start, in Sec. 4.2, by studying the simple case of gaussian mixture classification where the inputs have some low dimensional structure. We tune the hardness of the problem via a signal-to-noise ratio (SNR). We will see that feature learning allows small two-layer neural networks to learn the task while lazy methods fail to capture the underlying structure for small SNR. This emphasises the need to understand the interplay between input features and architecture. If badly chosen the later can prevent capturing of relevant structure in the data and thus good performance.

Second, in Sec. 4.7, we consider unsupervised learning, and in particular autoencoders (AE) which are trained to reproduce their inputs. To do so, they crucially exploit the structure in the inputs. We provide an analytic theory of the training dynamics of shallow non-linear AE where the hidden layer is significantly smaller than the input dimensions. This bottleneck forces the network to develop a concise representation of the inputs which helps understand which features in the data are captured by the network and are most important for the reconstruction task. The derived equations give bounds on how strong the input structure must be to achieve successful reconstruction. They also allow to assess the long times training dynamics and to unravel constrains on the network's architecture: tight weight AE fail to learn as do ReLU AE if the bias is not trained. Importantly, we show theoretically that relevant features in the data are learned sequentially, by order of importance.

Of course real images are not Gaussians, nor Gaussian mixtures and have complex relations between pixels. We must therefore address the question of whether our results, derived for a specific generative model, carry over to learning on realistic datasets and, if so, whether these relations are important for the network to perform well.

In Sec. 4.7.4, we argue that the equations derived for learning with AE on synthetic Gaussian inputs describe well the training dynamics on real images suggesting that, for the reconstruction task, shallow AE do not exploit information in the inputs beyond the second moment. This remains true for the supervised learning setup where training 2LNN on benchmark datasets is well described by a Gaussian mixture model, see App. C.2. This is another realisation of the interplay between the architecture and data structure. Even though images are more elaborate than Gaussians, shallow AE, and 2LNN with few hidden nodes, are not powerful enough to capture the additional complexity in the inputs for these simple tasks. Understanding the precise conditions under which networks capture which features in the data, beyond the Gaussian, is an exciting direction which we discuss further in the afore mentioned Sec. 4.7.4.

4.1 Theory for the dynamics of online learning of shallow networks

To start, we illustrate the main theoretical tools needed for the analysis of the next two sections by focussing on the simple setting of Gaussian i.i.d. inputs $x_i \sim \mathcal{N}(0,1)$. Namely, we show how to establish a theoretical description of the dynamics of learning of two layers neural networks (2LNN) with $K \sim O(1)$ hidden nodes. The networks are trained in the limit of an infinite number of training samples, i.e. $N \to \infty$, with online (or one pass) SGD in which at each step of the algorithm a new data point is drawn from the input's distribution p_x. The online limit is extensively used to analyse the dynamics of non-linear networks for both supervised and unsupervised learning, and it has been shown experimentally to capture the dynamics of deep networks trained with data augmentation well [231]. The description is valid in the high-dimensional limit where the input dimensions are taken to infinity i.e. $D \to \infty$ with $t = N/D \sim O(1)$. We follow the derivation of the seminal works [38, 269, 270] which was extended and made rigorous by [114]. We also illustrate the computations for the slightly more general setting where the inputs have non-trivial covariance matrix as it will be useful in deriving the results in the rest of the chapter. The approach here-after has several extensions to different data distributions [117, 118, 321] and [258–260] which we discuss in this

manuscript.

Setup Consider a classical *teacher-student* setup [98] in which the inputs $x \in \mathbb{R}^D$ are drawn component wise i.i.d. from a standard Gaussian distribution and the labels are given by a teacher network:

$$y^* = \tilde{\phi}_{\tilde{w}}(x) = \sum_{m=1}^{M} \tilde{v}^m \tilde{g}(\tilde{a}^m), \qquad \tilde{a}^m \equiv \sum_{i=1}^{D} \frac{\tilde{w}_i^m x_i}{\sqrt{D}}, \tag{4.1}$$

where $\tilde{g}$ is the teacher non-linearity applied component wise and the sum is over the teacher's M hidden nodes with $M \sim O(1)$. The student is also a 2LNN with $K \sim O(1)$ hidden nodes and has output:

$$\phi_\theta(x) = \sum_{k=1}^{K} v^k g(a^k), \qquad a^k = \sum_{i=1}^{D} \frac{w_i^k x_i}{\sqrt{D}}. \tag{4.2}$$

The student is trained to reproduce her teacher by minimising the mean-squared error:

$$\text{mse} = \frac{1}{2} \left(\tilde{\phi}_{\tilde{w}}(x) - \phi_w(x)\right)^2 \equiv \Delta^2, \tag{4.3}$$

where we defined the error on a single input $\Delta \equiv y^* - \phi_w(x)$. Using the vanilla BP algorithm described in Sec. 1.4, the update of the first and second layer of weights at training step μ is given by:

$$w_{\mu+1}^k - w_\mu^k \equiv \mathrm{d}w^k = \frac{\eta_w}{\sqrt{D}} v_\mu^k g'\left(a_\mu^k\right) \Delta_\mu x_\mu \tag{4.4}$$

$$v_{\mu+1}^k - v_\mu^k \equiv \mathrm{d}v^k = \eta_v g\left(a_\mu^k\right) \Delta_\mu. \tag{4.5}$$

Note that we introduced different learning rates for the first and second layer weights, η_w and η_v respectively.

Statics We are mostly interested in the generalisation performances of the student and thus focus on the population mean-squared error:

$$\text{pmse} = \mathop{\mathbb{E}}_x \text{mse} = \frac{1}{2} \mathop{\mathbb{E}}_x \left(\tilde{\phi}_{\tilde{w}}(x) - \phi_w(x)\right)^2. \tag{4.6}$$

To evaluate the expectation over the inputs distribution notice that the input x only enters in the pmse via its low dimensional projections on the teacher's and student's weights, $\tilde{a} \in \mathbb{R}^M$ and $a \in \mathbb{R}^K$ respectively. We can hence replace the high-dimensional expectation over the inputs with a low dimensional expectation over the joint distribution of the local fields $\{\tilde{a}, a\}$. A crucial observation is that due to the Gaussianity of the inputs, the local fields are jointly Gaussian with 0 mean

and second moments:

$$\begin{aligned} Q^{k\ell} &= \mathop{\mathbb{E}}_{x} a^k a^\ell = \frac{w^k \cdot w^\ell}{D}, \\ R^{km} &= \mathop{\mathbb{E}}_{x} a^k \tilde{a}^m = \frac{w^k \cdot \tilde{w}^m}{D}, \\ T^{mn} &= \mathop{\mathbb{E}}_{x} \tilde{a}^m \tilde{a}^n = \frac{\tilde{w}^m \cdot \tilde{w}^n}{D}. \end{aligned} \tag{4.7}$$

These macroscopic operators are often dubbed *order parameters* in statistical physics and play an important role in the analysis. Intuitively the *teacher-student* overlaps R measure the similarity between the student weights and the teacher weights, the *student-student* overlap Q and the *teacher-teacher* overlap T, measure the overlap between different weights of the student and the teacher respectively. In the high-dimensional $D \to \infty$ limit, the pmse is a function only of Q, R, T and of the second layer weights of both teacher and student:

$$\lim_{D\to\infty} \text{pmse}(x) \to \text{pmse}(Q, R, T, v, \tilde{v}). \tag{4.8}$$

Thus, by establishing a closed set of ordinary differential equations (ODEs) describing the dynamics of $(Q, R, T, v, \tilde{v})$ in the $D \to \infty$ limit, we obtain a complete description of the training dynamics, i.e. the generalisation performance, of the student at all times.

Dynamics Let us illustrate the rationale by deriving the dynamical equations for the student-student overlaps Q. From its definition we have:

$$\begin{aligned} Q^{k\ell}_{\mu+1} - Q^{kl}_{\mu} &= \frac{1}{D}\left\{ \mathrm{d}w^k_\mu w^\ell_\mu + w^k_\mu \mathrm{d}w^\ell_\mu + \mathrm{d}w^k_\mu \mathrm{d}w^\ell_\mu \right\} \\ &= \frac{\eta_w}{D} v^k_\mu g'\left(a^k_\mu\right) \Delta_\mu a^\ell + (k \leftrightarrow \ell) + \frac{\eta_w^2}{D} v^k_\mu v^\ell_\mu g'\left(a^k_\mu\right) g'\left(a^\ell_\mu\right) \Delta^2, \end{aligned} \tag{4.9}$$

where we used $\sum_i x^2_{\mu i} = D$ in the $D \to \infty$ limit.

By defining $t \equiv \mu/D$, one can show [114] that in the high-dimensional limit, Q_μ concentrates towards $Q(t)$ given by the solution of the ODE:

$$\frac{\mathrm{d}}{\mathrm{d}t} Q^{kl} = f_{Q^{kl}}(t), \tag{4.10}$$

where $f_{Q^{kl}}(t)$ is found by taking the expectation over x of Eq. 4.9:

$$f_{Q^{kl}} = \eta_w v^k(t) \mathop{\mathbb{E}}_{x} g'\left(a^k\right) a^\ell \Delta + (k \leftrightarrow \ell) + \eta_w^2 v^k_\mu v^\ell_\mu g'\left(a^k\right) g'\left(a^\ell\right) \Delta^2. \tag{4.11}$$

That we should take the expectation over the inputs distribution is intuitive as it

allows to describe a *typical* trajectory, independent of the particular realisation of $\{x_\mu\}$. In addition, the inputs only appear in the averages through the local fields. We can thus replace $\mathbb{E}_x$ by expectations over the distribution of $\{a, \tilde{a}\}$ which is low dimensional and only a function of the order parameters Q, R and T. It follows that the integrals are at most four dimensional. For some activation functions such as $g(x) = \text{erf}(x/\sqrt{2})$, the later have an explicit analytical expression [269, 270], see e.g. App. B.2.

Performing a similar argument for R yields:

$$\frac{\mathrm{d}}{\mathrm{d}t} R^{km} = \eta_w v^k \mathop{\mathbb{E}}_{\{a,\tilde{a}\}} g'\left(a^k\right) \tilde{a}^m \Delta. \tag{4.12}$$

To find a well defined high dimensional limit for v, we have to scale the learning rates as: $\eta_v = \eta/D$ and $\eta_w = \eta$. Then we obtain the last equation of a closed set of ODEs describing the dynamics at all times:

$$\frac{\mathrm{d}}{\mathrm{d}t} v^k = \eta \mathop{\mathbb{E}}_{\{a,\tilde{a}\}} g(a^k) \Delta. \tag{4.13}$$

Case of non-trivial covariance

Consider now a slightly more sophisticated generative model in which the inputs are Gaussian with non trivial covariance $\mathbb{E}_x x_i x_j = \Omega_{ij}$. Then, the order parameters become:

$$R^{km} = \frac{1}{D} \sum_{ij} w_i^k \Omega_{ij} \tilde{w}_j^m \qquad Q^{k\ell} = \frac{1}{D} \sum_{ij} w_i^k \Omega_{ij} w_j^\ell, \qquad T^{mn} = \frac{1}{D} \sum_{ij} \tilde{w}_i^m \Omega_{ij} \tilde{w}_j^n. \tag{4.14}$$

Rotating the dynamics An immediate issue arises due to the presence fo the covariance matrix in the contraction between the weights. Indeed, when trying to compute the update of the order parameters the input appears contracted with Ωw instead of w and cannot be identified with a local field. A key idea to make progress is to rotate the dynamics in the eigen-basis of the covariance matrix. To do so, we start by defining the eigen-decomposition of the covariance matrix and the rotation of any vector $z \in \{w, \tilde{w}, x\}$ onto this basis as:

$$\Omega_{ij} = \frac{1}{D} \sum_\tau \rho_\tau \Gamma_{i\tau} \Gamma_{j\tau}, \qquad z_\tau = \frac{1}{\sqrt{D}} \sum_i \Gamma_{i\tau} z_i. \tag{4.15}$$

Note the normalisation of the eigenvectors as $\sum_\tau \Gamma_{i\tau}\Gamma_{j\tau} = D\delta_{ij}$ and $\sum_i \Gamma_{i\tau}\Gamma_{i\tau'} = D\delta_{\tau\tau'}$. In this basis, the order parameters reduce to sums over D terms, instead of

D^2 terms:

$$R^{km} = \frac{1}{D}\sum_\tau \rho_\tau w_\tau^k \tilde{w}_\tau^m \qquad Q^{k\ell} = \frac{1}{D}\sum_\tau \rho_\tau w_\tau^k w_\tau^\ell, \qquad T^{mn} = \frac{1}{D}\sum_\tau \rho_\tau \tilde{w}_\tau^m \tilde{w}_\tau^n. \quad (4.16)$$

For illustration, let us focus on the dynamics of the teacher-student overlaps R. The evolution of the rotated weights w_τ follows directly from Eq. 4.5 and yields the update for R:

$$\frac{\mathrm{d}}{\mathrm{d}t}R^{km} = \eta_w \sum_\tau \rho_\tau v^k \frac{\tilde{w}_\tau^m}{D}\left\{\sum_n \tilde{v}^n \mathbb{E}\, g'\left(a^k\right)\tilde{g}\left(\tilde{a}^n\right)x_\tau - \sum_j v^j \mathbb{E}\, g'\left(a^k\right) g\left(a^j\right) x_\tau\right\}. \quad (4.17)$$

The expectations in blue cannot be simply expressed in terms of local fields. A crucial simplification occurs by noting that the rotated inputs x_τ and the local fields $\{a, \tilde{a}\}$ are weakly correlated:

$$\mathop{\mathbb{E}}_x a^k x_\tau = \frac{1}{\sqrt{D}}\rho_\tau w_\tau^k, \qquad \mathop{\mathbb{E}}_x \tilde{a}^m x_\tau = \frac{1}{\sqrt{D}}\rho_\tau \tilde{w}_\tau^m. \quad (4.18)$$

Both these correlations are of order $O(1/\sqrt{D})$. To compute expectations of the type $\mathbb{E}\, g'\left(a^k\right) a^j x_\tau$, we can therefore use the lemma for weakly correlated variables derived in App. B.1:

$$\begin{aligned}
\mathbb{E}\, g'\left(a^k\right) g\left(a^j\right) x_\tau =& \frac{1}{Q^{kk}Q^{jj} - Q^{kj2}}\frac{\rho_\tau}{\sqrt{D}}\Big[\mathbb{E}\, g'(a^k)a^k g(a^j)\left(w_\tau^k Q^{jj} - w_\tau^j Q^{kj}\right) \\
&+ \mathbb{E}\, g'(a^k)a^j g(a^j)\left(w_\tau^j Q^{kk} - w_\tau^k Q^{kj}\right)\Big] \\
\mathbb{E}\, g'\left(a^k\right) \tilde{g}\left(\tilde{a}^m\right) x_\tau =& \frac{1}{Q^{kk}T^{mm} - R^{km2}}\frac{\rho_\tau}{\sqrt{D}}\Big[\mathbb{E}\, g'(a^k)a^k \tilde{g}\left(\tilde{a}^m\right)\left(w_\tau^k T^{mm} - \tilde{w}_\tau^m R^{km}\right) \\
&+ \mathbb{E}\, g'(a^k)\tilde{a}^m \tilde{g}\left(\tilde{a}^m\right)\left(\tilde{w}_\tau^m Q^{kk} - w_\tau^k R^{km}\right)\Big]
\end{aligned} \quad (4.19)$$

The equation for R now involves terms of the form $\sum_\tau \rho_\tau^2 \tilde{w}_\tau^m w_\tau^j / D$ and $\sum_\tau \rho_\tau^2 \tilde{w}_\tau^m \tilde{w}_\tau^n / D$. The later is a constant of motion we do not need to worry about. The former, instead, is dynamical and cannot be expressed simply as order parameters. Defining an additional operator is not an option as it would result in an infinite sequence of operators with higher and higher powers of ρ. To find a closed set of equations, in terms of a finite number of quantities, we must therefore make another manipulation.

Integral representation The trick is to introduce the order-parameter densities $r(\rho, s)$, $q(\rho, s)$ and $t(\rho)$. These are continuous functions of ρ and, for r and q, of

the normalised number of steps $s \equiv \mu/D$, which we interpret as a continuous time variable in the limit $D \to \infty$:

$$r^{km}(\rho, s) = \frac{1}{\varepsilon_\rho}\frac{1}{D}\sum_\tau w_\tau^k \tilde{w}_\tau^m \mathbb{1}\left(\rho_\tau \in [\rho, \rho + \varepsilon_\rho[\right), \tag{4.20}$$

$$q^{k\ell}(\rho, s) = \frac{1}{\varepsilon_\rho}\frac{1}{D}\sum_\tau w_\tau^k w_\tau^\ell \mathbb{1}\left(\rho_\tau \in [\rho, \rho + \varepsilon_\rho[\right), \tag{4.21}$$

$$t^{mn}(\rho) = \frac{1}{\varepsilon_\rho}\frac{1}{D}\sum_\tau \tilde{w}_\tau^m \tilde{w}_\tau^n \mathbb{1}\left(\rho_\tau \in [\rho, \rho + \varepsilon_\rho[\right), \tag{4.22}$$

$$\tag{4.23}$$

where t^{mn} is constant, $\mathbb{1}(.)$ is the indicator function and the limit $\epsilon_\rho \to 0$ is taken after the thermodynamic limit $D \to \infty$. We recover the order parameters as an integral over the densities, weighted by the distribution $p_\Omega(\rho)$ of eigenvalues of the covariance matrix:

$$R^{km}(s) = \int \mathrm{d}\rho \rho p_\Omega(\rho) r^{km}(\rho, s), \tag{4.24}$$

$$Q^{k\ell}(s) = \int \mathrm{d}\rho \rho p_\Omega(\rho) q^{k\ell}(\rho, s), \tag{4.25}$$

$$T^{mn}(s) = \int \mathrm{d}\rho \rho p_\Omega(\rho) t^{mn}(\rho). \tag{4.26}$$

The dynamical equation for r^{km} follows directly from Eq. 4.17 and Eq. 4.19:

$$\frac{\partial r^{km}}{\partial t} = \eta_w \rho v^k \left\{ \sum_n \frac{\tilde{v}^n}{Q^{kk}T^{nn} - R^{kn2}} \left[I_3^{kkn}\left(r^{km}T^{nn} - t^{mn}R^{kn}\right) + I_3^{knn}\left(t^{mn}Q^{kk} - r^{km}R^{kn}\right)\right] \right.$$
$$\left. - \sum_\ell \frac{v^\ell}{Q^{kk}Q^{\ell\ell} - Q^{k\ell 2}} \left[I_3^{kk\ell}\left(r^{km}Q^{\ell\ell} - r^{\ell m}R^{k\ell}\right) + I_3^{k\ell\ell}\left(r^{\ell m}Q^{kk} - r^{km}R^{k\ell}\right)\right] \right\}, \tag{4.27}$$

where we defined the integral $I_3^{bcd} = \mathbb{E}_z g'(z^b) z^c g(z^d)$ with $z^b = a^b$ if $b \in \{k, \ell, j\}$ and $z^b = \tilde{a}^b$ if $b \in \{m, n, p\}$.

Evolution of the student-student overlaps The evolution of the student-student overlap Q is very similar to the one of R and we skip detailed computations to avoid redundancy. The only additional elements are the terms quadratic in the learning rate appearing in $\mathrm{d}Q^{k\ell}$ i.e.:

$$\frac{\eta^2}{D^2}\sum_\tau \rho_\tau \mathbb{E}\,\Delta^2 g'(a^k) g'(a^\ell) x_\tau^2 \overset{D\to\infty}{\approx} \frac{\eta^2}{D}\sum_\tau \frac{\rho_\tau^2}{D} \mathbb{E}\,\Delta^2 g'(a^k) g'(a^\ell) \equiv \frac{\eta^2}{D}\gamma \mathbb{E}\,\Delta^2 g'(a^k) g'(a^\ell). \tag{4.28}$$

where we used $\mathbb{E}_x x_\tau^2 = \rho_\tau$ and that x_τ and the local fields are weakly correlated to factorise the expectation. Lastly, we defined the constant of motion $\gamma = \sum_\tau \rho_\tau^2 / D$. Together with the equations for the second layer weights v^k:

$$\frac{\mathrm{d}}{\mathrm{d}t} v^k(t) = \eta \mathop{\mathbb{E}}_x \Delta g(a^k), \tag{4.29}$$

these equations provide a closed set of dynamical equations which allow to track the training dynamics and evaluate the generalisation performances Eq. 4.6 at all times. As we will see in the next two sections, these equations also allow to gain insight into the asymptotic solutions reached by the network and into some aspects of learning with neural networks.

4.2 Understanding the interplay between data structure and architecture in Gaussian mixture classification

4.3 Overview

In Chap. 3, we mostly focused on the random feature model Eq. 1.9 which can seen as an approximation of kernel learning [281]. Part of the motivation for this choice was the observation that, as discussed in the introduction, the weights of strongly over-parameterised 2LNN, initialised in the *lazy regime* remain almost constant throughout training [9, 10, 52, 86, 145, 183]. Nevertheless, these networks can achieve good performance. The RF model, is obtained by going a step further and fixing the first-layer weights of a 2LNN at their initial values. This behaviour is to be contrasted with the other regime discussed in Sec. 1.2, i.e. the *feature learning regime*, where the weights of the first layer move significantly during training. Recent empirical studies showed that on some benchmark data sets in computer vision, kernels derived from neural networks achieve comparable performance to neural networks [11, 99, 174, 186, 209, 286].

These results raise the question of whether neural networks only learn successfully if random features can also learn successfully, and have led to a renewed interest in the exact conditions and input type under which neural networks trained with gradient descent, achieve a better performance than random features [18, 73, 101, 185, 243, 300, 312, 320]. Chizat and Bach [62] studied the implicit bias of wide two-layer networks trained on data with a low-dimensional structure. They derived strong generalisation bounds, which, when both layers of the network are trained, are independent of ambient dimensions, indicating that the network is able to adapt to the low dimensional structure. In contrast, when only the output

layer of the network is trained, the network does not possess such an adaptivity, leading to worse performance. Ghorbani et al. [109, 110] analysed in detail how data structure breaks the curse of dimensionality in wide two-layer neural networks in the mean-field (MF) limit (Sec. 1.2), but not in learning with random features, leading to better performance of the former.

Both these works, however, study 2LNN in the MF limit of wide hidden layer, defined in Sec. 1.2. Here, we continue exploring the interplay between architecture and data structure, and the limitation of lazy methods to capture the later, by studying the opposite limit of only a few neurones in the hidden layer. These, despite having much fewer parameters than their mean-field counterpart, are able to capture the structure in the data and learn the task even at high noise levels. In contrast, random features, and lazy methods, do not and require much higher signal-to-noise ratio (SNR) to perform well. We focus on high-dimensional gaussian mixture classification. The setup is close to the one of Sec. 4.1 i.e. the network are trained using online SGD with infinite number of samples N on infinite dimensional inputs with $t = {}^{N}/_{D} \sim O(1)$.

We start with the study these 2LNN with a few hidden neurones $K \sim O(1)$. To do so, we extend the derivation of Sec. 4.1, where the label $y(x)$ is a function of the input x, to a setup where the input is conditional on the label. Solving these equations for their asymptotic fixed point, i.e. taking $t \to \infty$ *after* $D \to \infty$, yields the final classification error of the 2LNN. We then characterise analytically the performance of RF on the same task by analysing how Gaussian mixtures are transformed under P random features in the regime $P, D \to \infty$ with $\gamma \equiv {}^{P}/_{D}$ fixed. In the high-dimensional limit the performance at large γ converges to the one of the corresponding kernel [90, 189, 198, 246, 253, 254], and we can thus recover the performance of kernel learning by taking γ large enough. Finally, Computing the asymptotic generalisation of random features allows us to compare their performance to the performance of 2LNN for various signal-to-noise ratios.

Note that we will not be concerned by formal rigorous derivations. The theoretical claims of this section are however amenable to rigorous theorems. In particular the ODEs analysis could be formalised rigorously using the technique of [114, 308]. Our results are valid for generic Gaussian Mixtures with $O(1)$ clusters and we will focus on the particular example of the XOR-like mixture in order to make the problematic clear.

A paradigmatic example

The results can be illustrated through the simple example already presented in Chap. 1.3 which we recall here. Consider a data distribution, sketched on Fig. 4.1

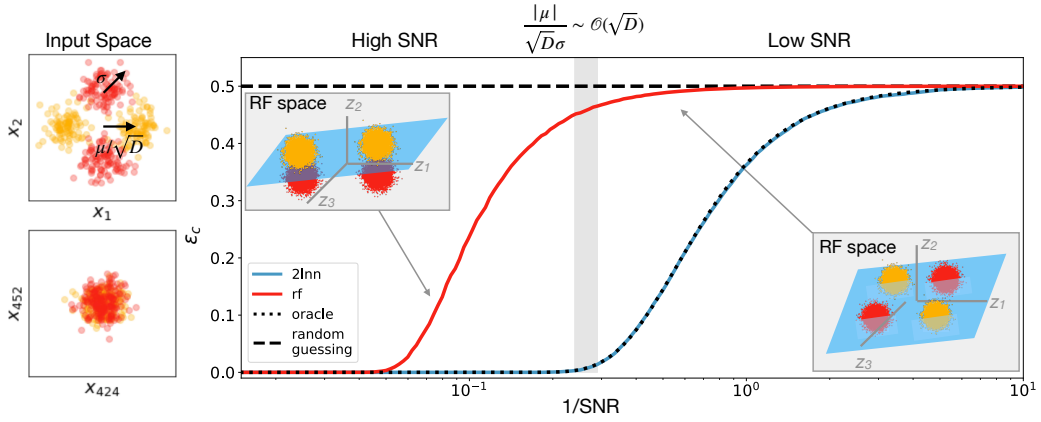


Figure 4.1: **Random Features and 2LNN on high-dimensional Gaussian mixture classification** ***(Left)*** Consider a data distribution that is a mixture of four Gaussians in D dimensions. The first two components of the centroids organized in a XOR-like manner as shown, while the other $D-2$ directions of the centroids are set to zero. The signal-to-noise is SNR $= |\mu|/\sqrt{D}\sigma$. We used $\mu = \sqrt{D}$ so the SNR is effectively given by the inverse of the width of the Gaussian: SNR $= 1/\sigma$. ***(Right)*** A two-layer neural network with $K = 4$ hidden neurones and ReLU non-linearity trained using stochastic gradient descent, achieves a long-time test error close to the optimal (oracle) error 4.33 for the whole range of SNR. The test error is obtained analytically using techniques described in Sec. 4.4. In sharp contrast, random features (RF), whose performance is given by Eq. 4.45, require a high SNR to perform as well as the oracle. They performs better than chance when SNR $\gg \sqrt{D}/\min(P,N)^{1/4}$, and thus requires a diverging SNR in the high-dimensional limit. The insets show the mixture *after* applying random features; only at high SNR does the mixture become linearly separable. *Parameters:* RF error is computed with $D = 10000$ and $P = 2D$ and the 2LNN's with $D = 1000$. For both methods $\eta = 0.1$, $|\mu|/\sqrt{D} = 1$.

(left), where inputs $x = (x_r) \in \mathbb{R}^D$ are distributed in a mixture of four Gaussians, those in red, yellow cluster have labels $y^* = \ 1, -1$, respectively. The first two components of the means are organised as in the top diagram, a distance $|\mu|/\sqrt{D} = 1$ from the origin, while the remaining $D-2$ components are zero, yielding a XOR-like pattern. Each Gaussian cloud has standard deviation $\sigma \mathrm{I}_D$, as Gaussian noise is added to all components of the input. We can define an SNR of $|\mu|/\sqrt{D\sigma^2}$.

Clearly, the mixture cannot be linearly separated in direct space. As we already argued and now show analytically, neural networks with a few neurones have no problems learning a good partitioning of the space in this situation, reaching oracle-like performance in the process. Kernel methods, however, manage to do so only if the centres are extremely well separated, and completely fail when they are too close.

We compare the performance of a two-layer neural network (2LNN) $\phi_\theta^{\mathrm{2LNN}}$,

defined in Eq. 1.13:

$$\phi_\theta^{\text{2LNN}} = \sum_{k=1}^{K} v^k g(a^k) \qquad a^k \equiv \frac{w^k \cdot x}{\sqrt{D}}, \tag{4.30}$$

where the number of hidden units K is kept of order 1 compared to the input dimension $D \to \infty$. We then train the 2LNN using online SGD and study the high-dimensional limit $t \equiv {}^{N}/_{D} = O(1)$. We obtain the final performance of the 2LNN by taking $t \to \infty$ (after $D \to \infty$) of the ODEs describing the training dynamics, see Sec. 4.4. The blue line gives the 2LNN's final classification error i.e. the expected number of mistaken classifications:

$$\epsilon_c(\theta) = \mathbb{E}\, \Theta\left[-y^* \phi_\theta(x)\right], \tag{4.31}$$

where Θ is the Heaviside step function given by:

$$\Theta\left[-y^* \phi_\theta(x)\right] \equiv \begin{cases} 1 & \text{if} \quad \text{sign}(\phi_\theta(x)) = y^* \\ 0 & \text{otherwise} \end{cases} . \tag{4.32}$$

The expectation $\mathbb{E}$ is computed over the Gaussian mixture for a network with fixed parameters w. The classification error of the 2LNN is very close to that of an *oracle* with knowledge of the means of the mixture that assigns to each input the label of the nearest mean, achieving a classification error of

$$\epsilon_c^{\text{oracle}} = {}^{1}/_{2} \left(1 - \text{erf}\left({}^{|\mu|}/_{2\sigma\sqrt{D}}\right)^2\right). \tag{4.33}$$

We compare the performance of the 2LNN to the performance of RF [253, 254] defined in Eq. 1.9, where we first project the inputs x to a higher-dimensional feature space, where features $z = (z_i) \in \mathbb{R}^P$ are given by

$$z_i = \psi(b_i), \qquad b_i \equiv \sum_{r=1}^{D} \frac{1}{\sqrt{D}} F_{ir} x_r. \tag{4.34}$$

Compared with the original definition Eq. 1.9, we denoted the random, fixed projection matrix $F \in \mathbb{R}^{P \times D}$ to avoid confusion with the 2LNN's trained parameters $\{w\} \in \mathbb{R}^{K \times D}$. $\psi : \mathbb{R} \to \mathbb{R}$ is the usual element-wise non-linearity. Then, like in the previous section, the features are fit by training a linear model on the features:

$$\phi_w^{\text{RF}}(x) = \frac{1}{\sqrt{P}} \sum_{i=1}^{P} w_i z_i, \tag{4.35}$$

where we denoted the trainable parameters as $w \in \mathbb{R}^P$ (as opposed to v in Eq. 1.9 and the previous section) to avoid ambiguity. The weights w are trained using SGD, where we again draw a fresh sample from the mixture to evaluate the gradients at each step. The performance of RF is shown in red in Fig. 4.1. While RF achieve low classification error at high SNR, there is a wide range of SNR where random features do significantly worse than the 2LNN. The insets give the intuition behind this result: at high SNR, RF map the inputs into linearly separable mixture in random feature space (left) while at lower snr, the transformed mixture is not linearly separable in RF space (right), leading to poor performance.

We emphasise that we study random features in the *high-dimensional limit* where we let $N, D \to \infty$ with their ratio $t = {}^{N}/_{D} \sim O(1)$ as before, while also letting the number of random features $P \to \infty$ with their ratio $\gamma \equiv {}^{P}/_{D} \sim O(1)$ fixed. This regime has been studied in a series of recent works [67, 80, 155, 177, 188, 203, 220]. While we concentrate on random features, we note that we can recover the performance of kernel methods [253, 254] by sending $\gamma \to \infty$. Indeed, as $\gamma = {}^{P}/_{D}$ grows, the gram matrix converges to the limiting kernel gram matrix in the high-dimensional regime; detailed studies of the convergence in this regime can be found in [90, 189, 198, 246]. We can thus recover the performance for any general distance or angle based kernel method, e.g. the NTK of Jacot et al. [145], by considering γ large enough in our computations with random features. Note, however, that this must be done with some care. Our results for random projections are given for $N > P$. As discussed by Ghorbani et al. [109], Mei et al. [215], the relevant dimension for random features performances is, rather than P, the minimum between N and P. Since we focus here in the regime where increasing P beyond $O(D)$, and therefore γ beyond $O(1)$, is not allowed. Indeed, we shall see that Lazy training methods such as kernels or random projections require asymptotically $N = O(D^2)$ samples to beat a random guess, while neural-networks achieves oracle-like performances with only $N = O(D)$ samples.

Reproducibility We provide code to reproduce our plots and solve the equations of Sec. 4.4 at this GitHub.

Further related work

Separation between kernels & 2LNN Barron [28] already discussed the limitations of approximating functions with a bounded number of random features within a worst-case analysis. Yehudai and Shamir [320] construct a data distribution that can be efficiently learnt by a single ReLU neuron, but not by random features. Wei et al. [312] studied the separation between 2LNN & RF and show the *existence* of a small ($K \sim O(1)$) network that beats kernels on this data distribution, and study

the *dynamics* of learning in the same mean-field limit as Chizat and Bach [62] and Ghorbani et al. [109, 110]. Likewise, Li et al. [185] show separation between kernels & neural networks in the mean-field limit on the phase retrieval problem. Geiger et al. [101] investigated numerically the role of architecture and data in determining whether lazy or feature learning perform better. Paccolat et al. [243] studied how neural networks can compress inputs of effectively low-dimensional data.

Gaussian mixture classification is a well-studied problem in statistical learning theory, and its supervised version was recently considered in a series of works from the perspective of Bayes-optimal inference [80, 177, 203]. Mignacco et al. [220, 221] studied the dynamics of stochastic gradient descent on a finite training set using dynamical mean-field theory for the perceptron, which corresponds to the case $K = 1, v^1 = 1$ in Eq. 4.30. Liao and Couillet [188] and Couillet [67] studied mixture classification with kernel in an unsupervised setting using random matrix theory.

Dynamics of 2LNN The equations describing the dynamics of learning, derived for this study, build on the results presented in Sec. 4.1 and on the references therein. All the referenced works, though, consider the label y^* as a function of the input x, or as a function of a latent variable from which x is generated. Here, we extend this type of analysis to a case where the input is conditional on the label, a point of view taken implicitly by Cohen et al. [66].

The reduction of the dynamics to a set of low-dimensional ODEs should be contrasted with the "mean-field" approach, discussed in Sec. 1.2, where the number of hidden neurones K is sent to infinity while the input dimension D is kept finite. In this limit, the neural networks are still a more expressive function class than the corresponding reproducing kernel Hilbert space [60, 214, 265, 290]. The evolution of the network parameters in this limit can be described by a high-dimensional partial differential equation. This analysis was used in the aforementioned works by Ghorbani et al. [109, 110].

4.4 Neural networks for GM classification

Setup

We draw inputs $x = (x_i) \in \mathbb{R}^D$ from a high-dimensional Gaussian mixture, where all samples from one Gaussian are assigned to one of two possible labels $y^* = \pm 1$, which are equiprobable. The data distribution is thus

$$q(x, y^*) = q(y^*)q(x|y^*), \quad q(x|y^*) = \sum_{\alpha \in \mathcal{S}(y^*)} \mathcal{P}_\alpha \mathcal{N}_\alpha(x) \tag{4.36}$$

where $\mathcal{N}_\alpha(x)$ is a multivariate normal distribution with mean $\mu^\alpha/\sqrt{D}$ and covariance Ω^α. The index set $\mathcal{S}(y^*)$ contains all the Gaussians that are associated with the label y^*. We choose the constants $\mathcal{P}_\alpha$ such that $q(x, y^*)$ is correctly normalised. To simplify notation, we focus on binary classification, which can be learnt using a student with a single output unit. Extending our results to C-class classification, where the student has C output heads, is straightforward.

Training The network is trained using stochastic gradient descent on the population mean-squared error pmse for technical reasons related to the analysis. The update equations for the weights at the μth step of the algorithm, $\mathrm{d}w_i^k \equiv (w_i^k)_{\mu+1} - (w_i^k)_\mu$, read

$$\mathrm{d}w_i^k = -\frac{\eta}{\sqrt{D}} v^k \Delta g'(a^k) x_i - \frac{\eta}{\sqrt{D}} \lambda w_i^k, \tag{4.37a}$$

$$\mathrm{d}v^k = -\frac{\eta}{D} g(a^k)\Delta - \frac{\eta}{D}\lambda v^k\,, \tag{4.37b}$$

where $\Delta = \sum_{\ell=1}^K v^\ell g(a^\ell) - y^*$ and $\lambda \in \mathbb{R}$ is, here again, a L_2-regularisation constant. Initial weights are taken i.i.d. from the normal distribution with standard deviation σ_0. The different scaling of the learning rates η for first and second-layer weights guarantees the existence of a well-defined limit of the SGD dynamics as $D \to \infty$. We make the crucial assumption that at each step of the algorithm, we use a previously unseen sample (x, y^*) to compute the updates in Eq. 4.37. This limit of infinite training data is variously known as online learning or one-shot/single-pass SGD.

Theory for the learning dynamics of 2LNN on mixture of Gaussians

The derivation of the dynamical equations of motion follows closely the one described in Sec. 4.1. The additional complexity is both in the dependence of the inputs on the labels, which requires expectations to be conditioned on the label realisation and in the non-zero mean of the inputs, which has to be treated with care.

Statics Since we are training on the quadratic error, the first step of our analysis is to rewrite the prediction mean-squared error pmse as a sum over the error made on inputs from each Gaussian α in the mixture,

$$\begin{aligned} \mathrm{pmse}(\theta) &= \mathop{\mathbb{E}}_{q(x,y^*)} \left(y^* - \phi_\theta(x)\right)^2 \\ &= \sum_{y^*} \sum_{\alpha \in \mathcal{S}(y_i^*)} q(y_i^*) \mathcal{P}_\alpha \mathop{\mathbb{E}}_{\alpha} \left[\sum_k v^k g(a^k) - y\right]^2 \end{aligned} \tag{4.38}$$

where the average $\mathop{\mathbb{E}}_{\alpha}$ is taken over the αth normal distribution $\mathcal{N}_\alpha$ for fixed param-

eters θ. Then, within each cluster, we can apply the tools of Sec. 4.1 remembering that here the the means of the inputs are not zero. Again, the input x only enters the expression via products with the student weights $a = (a^k)$ and we can replace the high-dimensional averages over x with an average over the K "local fields" a^k. Since the a^k are jointly Gaussian when averages are evaluated over just a single distribution in the mixture. We write the first two moments of the local fields as $M = (M_\alpha^k)$ and $Q = (Q_\alpha^{k\ell})$, with

$$M_\alpha^k \equiv \mathop{\mathbb{E}}_\alpha a^k = \frac{1}{D}\sum_r w_r^k \mu_r^\alpha, \tag{4.39a}$$

$$Q_\alpha^{k\ell} \equiv \mathop{\mathsf{Cov}}_\alpha \left(a^k, a^\ell\right) = \frac{1}{D}\sum_{r,s} w_r^k \Omega_{rs}^\alpha w_s^\ell. \tag{4.39b}$$

Any average over a Gaussian distribution is a function of only the first two moments of that distribution, so the pmse can be written as a function of the "order parameters" M and Q and of the $K \sim O(1)$ second-layer weights $v = (v^k)$. Likewise, the classification error ϵ_c 4.31 can also be written as a function of the order parameters only: $\lim_{D\to\infty} \epsilon_c(\theta) \to \epsilon_c(Q, M, v)$. The order parameters have a clear interpretation: M_α^k encodes the overlap between the kth student node and the mean of the α cluster, and plays a similar role to the teacher-student overlap in the vanilla teacher-student scenario. Q_α^{kl} instead tracks the overlap between the various student weight vectors, with the input-input covariance Ω^α intervening. The strategy for our analysis, like in Sec. 4.1, is thus to derive equations that describe how the order parameters (Q, M, v) evolve during training, which will in turn allow us to compute the pmse of the network at all times.

Dynamics We derived a closed set of ordinary differential equations that describe the evolution of the order parameters in the case where each Gaussian in the mixture has the same covariance matrix Ω. We proceed here with a brief statement of the equations and defer the detailed derivation to Sec. C.2. The approach is most easily illustrated with the second-layer weights v^k. The key idea to compute the average change in the weight v^k upon an SGD update 4.37b, $\mathrm{d}v^k$, which can be decomposed into a contribution from every Gaussian in the mixture,

$$\mathbb{E}\,\mathrm{d}v^k = \sum_{\alpha\in\mathcal{S}(+)} \mathcal{P}_\alpha \mathrm{d}v_{\alpha^+}^k + \sum_{\alpha\in\mathcal{S}(-)} \mathcal{P}_\alpha \mathrm{d}v_{\alpha^-}^k\,, \tag{4.40}$$

where the change $\mathrm{d}v_{\alpha^+}^k$ is obtained directly from Eq. 4.37b,

$$\mathrm{d}v_\alpha^k = \frac{\eta}{D}\mathop{\mathbb{E}}_\alpha y_\alpha g(a^k) - \frac{\eta}{D}\sum_j v^j \mathop{\mathbb{E}}_\alpha g(a^k) g(a^j) - \frac{\eta}{D}\lambda v^k. \tag{4.41}$$

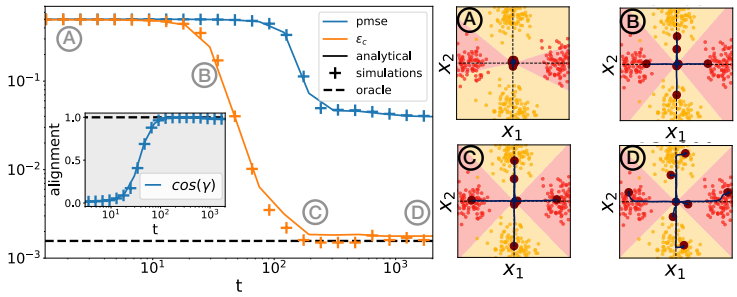


Figure 4.2: **How the 2LNN learns a XOR-like GM (Left)** Evolution of the prediction mean-squared error pmse 4.38 and the classification error 4.31 of a 2LNN with $K = 8$ neurones trained on the XOR-like mixture of Fig. 4.1. We plot the test errors as obtained from a single simulation with $D = 1000$ (crosses) and from integration of the ODEs of Sec. 4.4. The dashed black line is the classification error of an oracle with knowledge of the means μ^α, Eq. 4.33. The inset shows the mean angle of the network weights to the means of the mixture. **(Right)** Projections of the first layer weights (dots) onto the plane spanned by the means of the XOR-like mixture at different times during training. Shaded areas indicate the decision boundaries of the network, where its output $\phi_\theta(x)$ changes sign. *Parameters:* $K = 8, D = 1000, \sigma = 0.05, \eta = 0.1$ weights initialised with s.t.d. $\sigma_0 = 1$, $\lambda = 10^{-2}$.

The averages that remain to be computed only involve the true label and the local fields a. The former is a constant within each Gaussian while the latter are jointly Gaussian. It follows, that also these averages can be expressed in terms of only the order parameters and the equation closes. As in Sec. 4.1, in the high-dimensional limit $D \to \infty$ the normalised number of samples $t \equiv N/D$ can be interpreted as a continuous time, which allows the dynamics of v^k to be captured by the ODE C.19.

Due to the presence of a non-trivial covariance, the order parameters Q require the additional step which consists in diagonalising the sum $Q^{k\ell} \sim \sum_{r,s}^{D} w_r^k \Omega_{rs} w_s^\ell$ by introducing the integral representation

$$Q^{kl} = \int \mathrm{d}\rho \; p_\Omega(\rho) \; \rho q^{kl}(\rho), \tag{4.42}$$

where $p_\Omega(\rho)$ is the spectral density of Ω, and $q^{kl}(\rho)$ is a density whose time evolution can be characterised in the thermodynamic limit. We relegate the full expression of the equation of motion for $q^{kl}(\rho)$ to Eq. C.18 of the appendix. Crucially, it involves only averages that can be expressed in terms of the order parameters 4.39, and hence the equation closes. Likewise, the order parameter M can be rewritten in terms of a density as $M^{\alpha k} = \int \mathrm{d}\rho \; p_\Omega(\rho) m^k(\rho)$. The dynamics of m^k is described by Eq. C.13.

Solving the equations of motion The equations are valid for any mean and covariance matrix Ω. Solving them requires evaluating multidimensional integrals

of dimension up to 4, e.g. $\mathbb{E}_{\alpha} g'(a^k)a^\ell y^*$, which can be efficiently estimated using Monte-Carlo (MC) methods. We provide a ready-to-use numerical implementation on the GitHub.

Comparing theory and simulation On the left of Fig. 4.2, we plot the evolution of the pmse 4.38 and the classification error 4.31 of a 2LNN with $K = 8$ neurones trained on the XOR-like mixture of Fig. 4.1. We plot the test errors obtained from integration of the order parameters with solid lines, and the same quantities computed using a test set during the simulation with crosses. The agreement between ODE predictions and a single run of SGD is good, even at intermediate system size ($D = 1000$). In the App. C.2, we give additional plots for the simulated dynamics of the individual order parameters and find very good agreement with predictions obtained from the ODEs (cf. Fig. C.1). Note that although we initialise the weights of the student randomly and independently of the means, there is an initial overlap between student weights and the means of order $1/\sqrt{D}$ due to finite-size fluctuations. To capture this with the ODEs, we initialise them in a regime of weak recovery, where $M_\alpha^k \neq 0$. For a detailed discussion of the early period of learning up to weak recovery, see Arous et al. [14].

How 2LNNs learn the XOR-like mixture A closer look at the learning dynamics on the right of Fig. 4.2 reveals several phases of learning. There we show the first-layer weight vectors of the 2LNN, projected into the plane spanned by the four means of the mixture, at four different times during training. The regions shaded in red and yellow indicate the decision boundaries of the network, which correspond to the line where the network's output $\phi_\theta(x)$ changes its sign. A 2LNN with $K \geq 4$ neurones can approach the classification error of the oracle 4.33 if its weight vectors approach the four means, with corresponding second-layer weights. Panel (C) shows that network reaches this configuration. However, this configuration does not minimise the mean-squared error used during training 4.37, so eventually the weights depart slightly from the means to converge to a solution with lower mean squared error (D). This is confirmed by the inset on the left of Fig. 4.2, where we see that the average angle of the network weights to the means has a maximum around $t = 300$, before decaying slightly at the end of training.

Predicting the long-time performance of 2LNN

Direct integration of the ODEs is numerically expensive. A more straight forward way to extract information from the ODEs is to find their asymptotic fixed point, which fully characterises the $t \to \infty$ performance of the network. However, the number of equations is already 26 for a 2LNN with 4 neurones trained on the

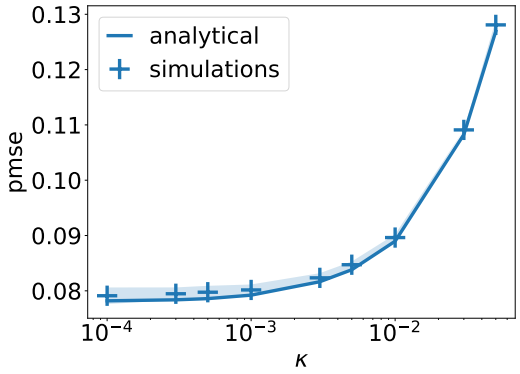


Figure 4.3: Prediction mean-squared error 4.38 on the XOR-like mixture versus weight decay. Results obtained from a fixed point analysis of the ODEs with 4 degrees of freedom discussed in Sec. 4.4 ($\sigma^2 = 0.1, K = 4, \eta = 0.1, 10^4$) Monte-Carlo samples. The shaded area indicates standard deviation over 10 runs.

XOR mixture, and scales like K^2. The key to finding fixed points efficiently is thus to make an *ansatz* with fewer degrees of freedom for the matrices Q and M which solve the equations. For example, one could impose $Q^{kk} = Q$ and $Q^{k\ell} = C, k \neq \ell$. By exploiting the symmetries of the XOR-like mixture, we find that the fix points of the equations can be described by only $K = 4$ parameters: $K/2$ angles between weight vectors and means, and $K/2$ norms, as described in Appendix C.2. Finding the fixed-point of this reduced four-dimensional system allows to compute, for example, the dependence of the generalisation error as a function the regularisation in Fig. 4.3. The agreement between simulation and analytical predictions is again good, and we find that increasing the regularisation only increases the test error of the student. This is in agreement with previous work on two-layer networks in the same limit in the teacher-student setup, where L_2-regularisation was also found to hurt performance [271].

The impact of over-parametrisation

We also studied the effect of over-parametrisation, which we define as the number of additional neurones a student has on top of the $K = 4$ neurones that it needs to reach the oracle's performance on the XOR mixture. We show in the inset of Fig. 4.4 that over-parametrisation does *not* improve final performance, since the remaining error of the student is dominated by "spill-over" of points from one mixture into adjacent quadrants. However, over-parametrisation leads to an "implicit acceleration" effect: over-parametrised networks are much more likely to converge to a solution that approaches the oracle's performance, as we show in the main of Fig. 4.4. The term "implicit acceleration" was coined by Arora et al. [12]

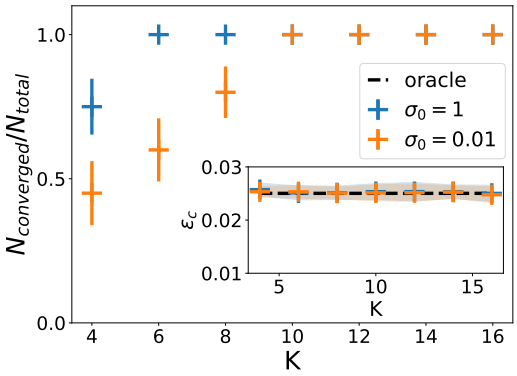


Figure 4.4: Fraction of simulations that converged to the optimal solution for the 2LNN out of 20 simulations for increasing values of K. Overparametrisation increases the probability of finding the optimal solution but does not affect classification performances. Simulations started with initial weights of std.dev. σ_0. The inset shows the classification error of the networks that converged. *Parameters:* $D = 800, \eta = 0.1, \lambda = 0, \sigma^2 = 0.1$, run time 10^5.

for similar effects in deep neural networks, and analysed for two-layer networks in the teacher-student setup by Livni et al. [196], Safran and Shamir [272]. A complete understanding of the phenomenon remains an open problem, which we leave for future work.

4.5 Random features on GM classification

To understand the performance of random features on Gaussian Mixtures classification, we analyse the performances of the linear model 4.35 trained with online SGD with the squared loss on the random features z 4.34 [53, 296].

First, we assume that we have enough samples, so that $N \gg P$, and discuss the situation when $N \ll P$ later. For any finite D, P, running the algorithm up to convergence then corresponds to taking the limit $t \to \infty$. The random features' weights converge to an estimate $\hat{W}$ which can be computed analytically, see Eq. C.51, and allows to precisely characterise the test error:

$$\text{pmse}_{t\to\infty} = \frac{1}{2}\Big(1 - \sum_{\tau} \frac{\tilde{\Phi}_\tau^2}{\rho_\tau}\Big), \tag{4.43}$$

where ρ_τ are the eigenvalues of the feature's covariance matrix $\Omega_{ij} = \mathbb{E}\, z_i z_j$, with associated eigenvector Γ_τ. $\tilde{\Phi}_\tau \equiv \sum_{i=1}^{P} \Gamma_{\tau i}\Phi_i/\sqrt{P}$ is the input-label covariance after rotation into the eigenbasis of Ω (see Appendix C.4). Crucially, the test error and $\hat{W}$ *only* depend on the first two moments of the features. The formula for these moments, as well as the one for the classification error can be obtained when P, D

are large using the Gaussian equivalence of [118]. Indeed, the distribution of the features z remains a mixture of distributions (see App. C.3). We then define

$$M_\alpha = \sum_{i=1}^{P} \frac{\hat{w}_i \, \mathbb{E}_\alpha[z_i]}{\sqrt{P}}, \quad Q_\alpha = \sum_{i=1}^{P} \frac{\hat{w}_i \hat{w}_j}{P} \operatorname*{Cov}_\alpha(z, z), \tag{4.44}$$

and we find for D, P large enough, that

$$\epsilon_{ct\to\infty} = \frac{1}{2}\left(1 - \sum_\alpha \mathcal{P}_\alpha \, y \, \operatorname{erf}\left(\frac{M_\alpha}{\sqrt{2Q_\alpha}}\right)\right). \tag{4.45}$$

As discussed in App. C.3, in the case of ReLU activation function, the feature distribution $p(z_i)$ is a truncated Gaussian. Hence, at large D, P, both the mean of z_i and the population covariance $\operatorname{Cov}(z_i, z_j)$ can be obtained analytically in terms of the matrix F and means μ, see Eq. C.37 and C.40 for the full result.

We used this formula to obtain precisely the error 4.43, and the results are shown in Fig. 4.5. We see that the RF error is a function of $\sigma D^{1/2}/P^{1/4} = \sigma(D/\gamma)^{1/4}$ leading to the conclusion that – as discussed in Fig. 4.1 – the "transition" from the high to low SNR regime happens when $\sigma^{-1} \approx D^{1/2}/P^{1/4}$. This scaling further reveals that $P \approx D^2$ features are required in order to obtain good performance. The validity of Eq. 4.45 is verified in Fig. C.3. Reaching this performance, however, requires the number of samples N to be larger than P, so $N > O(D^2)$. In the so-called *high-dimensional* regime analysed in this paper, where $N \approx D$, such performances remain out of reach. The scaling analysis can be easily generalised; as discussed by Ghorbani et al. [109], Mei et al. [215], the relevant dimension for RF performances is, rather than P, the minimum between N and P.

The classification error is thus a function of $\sigma D^{1/2}/\min(N,P)^{1/4}$. If N is $O(D)$, then even in the kernel limit when $P\to\infty$, the performance degrades to no more than a random guess as soon as

$$\sigma \gg N^{1/4}/D^{1/2}, \tag{4.46}$$

and therefore for any value of σ when $D, N\to\infty$ with fixed N/D. In a nutshell, for any fixed σ, lazy training methods such as random features or kernels will fail to beat a random guess in the high-dimensional limit. This, and the requirement of at least $N = O(D^2)$ samples to learn, are to be contrasted with the the oracle-like performance achieved by a simple neural net with only $N = O(D)$ samples.

Why do random features fail? The linear regime of features maps

This raises the question of *why* the random feature and kernel methods fail in the high-dimensional setting. As we shall see – and this has been already discussed in different contexts by El Karoui [90], Mei and Montanari [213] – this can be

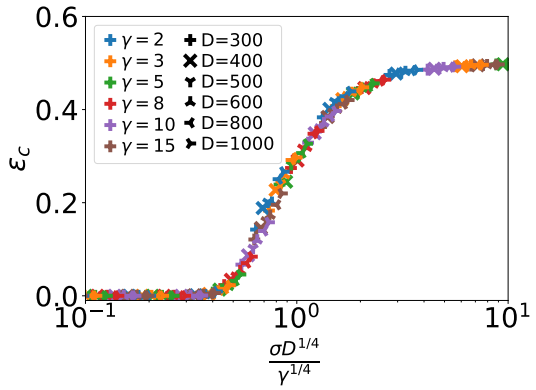


Figure 4.5: **Evolution of the classification error of random features (RF) for various values of σ, $\gamma = P/D$ and D** on the XOR mixture of Fig. 4.1, in the limit of large number of samples $N \gg P$. All these different cases can be collapsed into a single master curve by plotting the classification error versus $\sigma D^{1/2}/P^{1/4} = \sigma D^{1/4}/\gamma^{1/4}$, showing that for large D, P, it should be a function $\epsilon_c = f(\sigma D^{1/2}/P^{1/4})$. For a finite input dimension D, increasing the number of features allows RF to perform increasingly better as they approach the Kernel limit. Analytical predictions are obtained by the linear regression analysis of Eq. 4.45 for RF. *Parameters:* $\eta = 0.1$, $|\mu|/\sqrt{D} = 1$. Note however that this analysis requires $N \gg P$. Given random features are sensitive to the minimum of P and N [109, 215] the effective scaling variable is rather $\sigma D^{1/2}/\min(P,N)^{1/4}$ (see text).

understood analytically from the fact that the feature map is effectively linear when when $P = O(D)$. This section is now dedicated to computing the moments $\mathbb{E}_\alpha[z_i]$ and $\mathrm{Cov}_\alpha(z,z)$ analytically in this region, revealing that this effective linearity is indeed the underlying reason for the failure of RF in this regime.

Since the mixture remains a mixture after the application of random features, our main task is to compute the new means and variances of the distribution in the transformed space. We thus focus on transformation of a random variable drawn from a single Gaussian $x_r = \mu_r/\sqrt{D} + \sigma w_r$, where w_r is a standard Gaussian, in the kernel, or random feature, space.

For generic activation function the first two moments of the features can be obtained in the well studied low signal-to-noise regime SNR$\sim O(1)$. Key to do so, is the observation that $F_{ir}\mu_r/D \sim O(1/\sqrt{D})$. The activation function can thus be expanded in orders of $1/D$ and its action is essentially linear. We define the constants

$$a \equiv \mathbb{E}\,\psi(\sigma\zeta)\,, \quad b \equiv \mathbb{E}\,\zeta\psi(\sigma\zeta)\,, \quad c^2 \equiv \mathbb{E}\,\psi(\sigma\zeta)^2 \tag{4.47}$$

with the expectation taken over the standard Gaussian random variable ζ. To

leading order, the mean and covariance of the features are given by (cf. Sec. C.3):

$$\mathbb{E}\, z_i = a + b \sum_{r=1}^{D} \frac{F_{ir}\mu_r}{\sigma D} \tag{4.48}$$

$$\operatorname{cov}(z_i, z_j) = \begin{cases} c^2 - a^2, & i = j\,, \\ b^2 \sum_r \frac{F_{ir}F_{jr}}{D} & i \neq j. \end{cases} \tag{4.49}$$

This computation immediately reveals the reason random features cannot hope to learn in the low SNR regime: the transformation of the means is only linear; hence a Gaussian mixture that is not linearly separable in input space will remain so even after random features. In other words, if the centres of the Gaussian are too close, the kernel fails to map the data non-linearly to a large dimensional space. In contrast, in the high-SNR regime, where the centres are separated enough, the non-linearity kicks-in and the data becomes separable in feature space.

Relation to kernel methods The same argument explains the failure of kernel methods: if two centres $\mathbf{x}$ and $\mathbf{y}$ are close, the kernel function $K(\mathbf{x}, \mathbf{y})$ can be expanded to low order and the kernel is essentially linear, leading to bad performances. The connection can be made explicit using the convergence of random features to a kernel [253, 254]:

$$K(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{P} \sum_{i=1}^{P} \mathop{\mathbb{E}}_{F} \left[\psi\left(\sum_{r=1}^{D} \frac{x_r F_{ir}}{\sqrt{D}} \right) \psi\left(\sum_{s=1}^{D} \frac{y_s F_{is}}{\sqrt{D}} \right) \right], \tag{4.50}$$

At low SNR, the constants a, b, c can be obtained from the kernel via

$$c^2 = \mathbb{E}\, K(\sigma\boldsymbol{\omega}_1, \sigma\boldsymbol{\omega}_1), \quad a^2 = \mathbb{E}\, K(\sigma\boldsymbol{\omega}_1, \sigma\boldsymbol{\omega}_2),$$
$$b^2 = D\sigma^2 \left[-a^2 + \mathbb{E}\, K\left(\frac{\boldsymbol{\mu}}{\sqrt{D}} + \sigma\boldsymbol{\omega}_1, \frac{\boldsymbol{\mu}}{\sqrt{D}} + \sigma\boldsymbol{\omega}_2 \right) \right]$$

where the average is taken over two standard Gaussian random vectors $\omega_1, \omega_2 \in \mathbb{R}^D$. This relation, similar in nature to one of El Karoui [90], allows to express the statistical properties of the features directly from the kernel function.

4.6 Neural networks vs random features

We now collect our results for a comparison of the performance of 2LNN and RF on the XOR-like mixture from Fig. 4.1. We look at three different regimes for the SNR, illustrated in the first column of Fig. 4.6. The second column visualises the mixture after the Gaussian random features transformation with $\psi(x) = \max(0, x)$ 4.34. The third and fourth columns show the evolution of the pmse of 2LNN and RF,

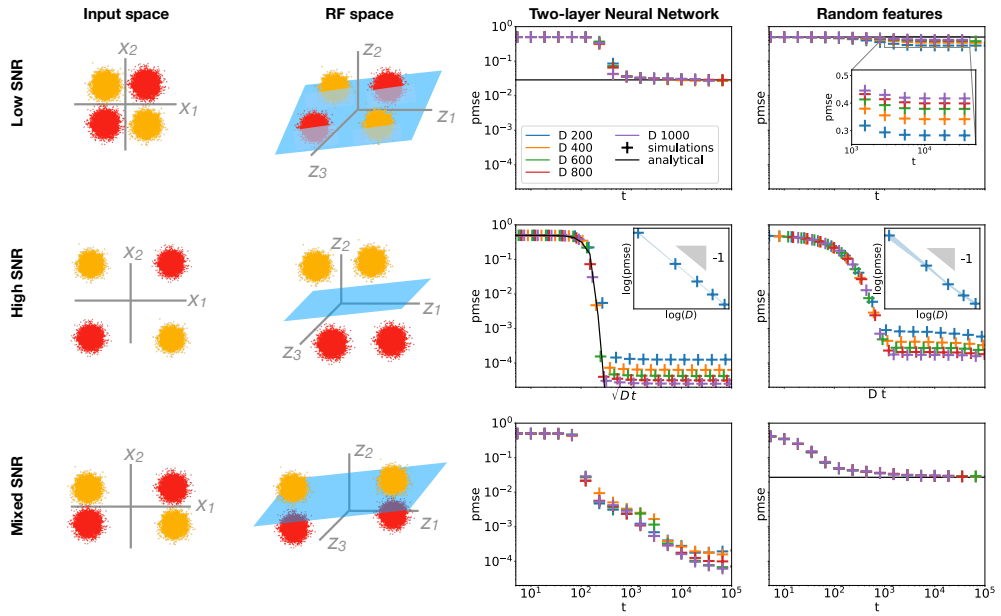


Figure 4.6: **The performance of 2LNN and RF on the XOR-like mixture with different signal-to-noise ratios.** The *first and second columns* show the XOR-like mixture in input space and random feature space, resp. The *third and fourth columns* show the pmse 4.38 of 2LNN and RF during training, resp. In the low SNR regime (top row), while the 2LNN learns a non trivial function of the inputs, RF cannot perform better than random chance since the XOR-like mixture remains one in feature space. Both networks learn to classify the XOR-mixture in the high SNR regime (middle row) as the clusters become well separated in feature space. For mixed SNR (bottom row), even though both networks do better than random guessing, 2LNN outperform RF as the distance between opposite sign clusters remains of order one in feature space. In all plots, crosses are obtained from simulations with input dimension $D = 1000$. Solid black lines in the 2LNN plots are obtained using tools from Sec. 4.4 (long-time performance of Sec. 4.4 for the first row and integration of the odes for the second). Solid black lines in the RF plots indicate the test error obtained from the analysis of Sec. 4.5 with $D = 10000$. $\sigma^2 = 0.05, P = 2D, \eta = 0.1, K = 10, \sigma_0 = 10^{-2}$.

respectively, during training with online SGD. Since overparametrisation does not impact the 2LNN's performance in these tasks, Sec. 4.4, we train a $K = 10$ network to increase the number of runs that converge.

At **low** SNR (a) the distance of each Gaussian to the origin is $O(1)$ and the standard deviation $\sigma \sim O(1)$ as well. The two-layer neural network learns to predict the correct labels almost as well as the oracle 4.33. Its performance does not depend on D, and using the long-time solution of Sec. 4.4, we can predict its asymptotic error (black line) which agrees well with simulations (crosses). In contrast, random features display an asymptotic error that approaches random guessing as the input dimensions increases (inset). This is clear from Eq. 4.49: in the large D limit,

random features only produce a *linear* transformation of their input. The XOR therefore remains a XOR in RF space, leading linear regression's failure to do better than chance.

At **high** SNR (b), the distance between the clusters scales as $\sqrt{D}$ while the σ remains fixed. The asymptotic error of the 2LNN thus decreases with D and the network is able to learn perfectly in the $D \to \infty$ limit (black line). The error of random features also approaches 0 as $D \to \infty$, since the mixture is now well separated in random feature space, too.

We finally consider a regime of **mixed** SNR (c) where the mixture is well-separated in one dimension, but very close in the other dimension. We achieve this by setting $\mu_1^0 \sim \sqrt{D}, \mu_2^0 \sim D$ for the mean of the first mixture, etc. Random features then achieve a non-trivial generalisation error, which can be understood by considering the means of the features z_i. The large component μ_2, induces the activation function to perform a non-linear transformation of the centres and allows for opposite sign centroids to be separated by a hyper-plane in feature space. The small component μ_1, causes the distance between opposite sign centroids, which is of order $O(1)$ in input space, to remain of order one in feature space, for all D. This leads to a finite generalisation error of RF which remains invariant with increasing input dimension. In this regime, the 2LNN still achieve better performance than the random features, thereby completing the picture we developed in Fig. 4.1.

4.7 Autoencoders as a tool to study feature learning

4.7.1 Overview

In the previous section, we quantified how strong the data structure has to be in order for different architectures to classify high dimensional mixtures of Gaussian. We controlled the strength of the features in the data by tuning the signal-to-noise ratio. The larger the SNR, the more salient the features.

Even if the results give valuable insights into the data-architecture interplay and also describes learning on some benchmark datasets, the type of structure remains extremely simple. Real data is expected to be more complex than mixture of Gaussians and to have no clear SNR measure allowing to a priori determine wether lazy methods can learn a given task or not.

Studying unsupervised learning is one way to uncover which features are learned in more general contexts, why and how. In this paradigm, the network is trained using only the inputs and exploits their structure to perform a given task. Autoencoders (AE) are the simplest neural network for unsupervised learning, and thus offer an ideal framework for studying feature learning.

They are trained to reconstruct their inputs by minimising the distance between an input $x \in \mathcal{X}$, and the network output $\hat{x} \in \mathcal{X}$. As we discussed in Sec. 1.3, the key idea for learning good features with AE is to make the intermediate layer in the middle of the network (significantly) smaller than the input dimension. This *bottleneck* forces the network to develop a compressed representation of its inputs, together with an encoder and a decoder describing how to go from the inputs to the compressed representation and vice-versa. AE of this type are called under-complete, see Fig. 4.7.

In order to study AE from a theoretical perspective we focus on shallow autoencoders with a single hidden layer of neurons. An immediate question arises: how well can these networks perform on a reconstruction task? which features in the data do they capture to attain this performance? and how do they capture the latter?

For *linear* autoencoders, Eckart and Young [89] established that the optimal reconstruction error is obtained by a network whose weights are given by (a rotation of) the K leading principal components of the data, *i.e.* the K leading eigenvectors of the input covariance matrix. The reconstruction error of such a network is therefore called the PCA error and only the first two moments, i.e. the mean and covariance, of the inputs are exploited.

How linear autoencoders learn these components when trained using stochastic gradient descent (SGD) remains an interesting question. Indeed, although these networks only perform linear transformations of their inputs, their learning dynamics is non-linear and rich enough to give insights that carry over to the non-linear case [5, 24, 162, 169, 275]. A series of recent works analysed linear autoencoders in more detail, focusing on the loss landscape [165], the convergence and implicit bias of SGD dynamics [236, 252, 276], and on recovering the exact principal components [25, 111, 239].

However, we have seen in Sec. 1 that part of the power of neural networks as feature extractors relies on the non-linearities in the hidden layers. Adding a non-linearity to autoencoders is hence a key step on the way to more realistic models of neural networks. This addition has been hindered mostly by the theoretical challenge of analysing non-linear models on structured data, crucial ingredient to study feature learning. The first step in this direction was taken by Nguyen [235], who analysed over-complete autoencoders, in the MF limit, where the number of hidden neurones grows polynomially with the input dimension, focusing on the special case of weight-tied autoencoders, where encoder and decoder have the same weights.

Here, we leverage recent developments in the analysis of supervised learning

on complex input distributions and in particular in the description of the training dynamics of networks in the ODE limit described in Sec. 4.1 and developed in [115, 118, 199, 260].

We study the learning dynamics of under-complete, non-linear autoencoders with both tied and untied weights. While, their reconstruction error is still limited by the PCA reconstruction error [24, 46], their learning dynamics is rich, and different from the one of linear AE. In particular, we tackle the questions of whether non-linear AE trained with SGD can reach the PCA error, and if so, how do they do it? How do the representations they learn depend on the architecture of the network, or the learning rule used to train them? The answers to all these question also provides insights on the relevant features in the data and on their relative importance for reconstruction.

The first step to tackle these question is to build on the tools of Sec. 4.1 and derive the asymptotically exact ODEs that describe the generalisation dynamics of shallow, non-linear autoencoders trained using online SGD (4.7.3). In deriving these equations, we will establish a precise requirement on the magnitude of the features required for the AE to learn to reconstruct their inputs. Using the ODEs, we can understand how AE learn important features of their data. We will see that, as in the linear case, these features are the principal components of the covariance matrix of the data. These are learned sequentially, in order of importance modelled, in this case, by the eigenvalue associated to a given PC (see Sec.4.7.3). The ODEs also provide insights into the long-time dynamics of learning and highlight another instance of data/architecture interplay in ML: to extract good features and achieve low reconstruction error, it is the necessity to untie the weights of decoder and encoder (4.7.3) and train the bias for ReLU autoencoders (4.7.3). We notice that, due to the rotational symmetry inherent to the SGD step, the network only learns a rotation of the leading PCs, making the output of the hidden layer hard to interpret. We thus suggest a modification of vanilla SGD that beaks the rotational symmetry of neurones and yields the exact principal components (4.7.3). Finally, we end this section in Sec. 4.7.4, by verifying the pertinence of the generative model introduced to study unsupervised learning with shallow AE. In particular, we demonstrate that the equations and the insights developed for Gaussian data capture learning on realistic dataset suc as CIFAR10 with great accuracy and discuss connections with recent results on Gaussian universality in neural network models already discussed in Chap. 1.3.

Reproducibility We provide code to solve the dynamical equations of 4.7.3 and to reproduce our plots at this GitHub.

4.7.2 Setup

Architecture We study the performance and learning dynamics of shallow non-linear autoencoders with K neurones in the hidden layer. Given the usual D-dimensional input $x = (x_i)$, the output of the autoencoder is given by

$$\phi_w^{\mathrm{AE}}(x_i) \equiv \hat{x}_i = \sum_k^K d_i^k g\left(a^k\right), \qquad a^k \equiv \sum_{i=1}^D \frac{e^k x}{\sqrt{D}} \tag{4.51}$$

where $e^k, d^k \in \mathbb{R}^D$ are the encoder and decoder weight of the kth hidden neurone, resp. With respect to the previous sections, we changed the notation from $\{w^1, w^2\}$ for the weights of the network to $\{e, d\}$ for clarity and to highlight the role of the encoder, respectively decoder in reconstructing the inputs. We keep $g(\cdot)$ a non-linear function. Here again, we study this model in the thermodynamic limit where we let the input dimension $D \to \infty$ while keeping the number of hidden neurones K finite, as shown in 4.7 (a). The performance of a *given* autoencoder is measured by the population reconstruction mean-squared error,

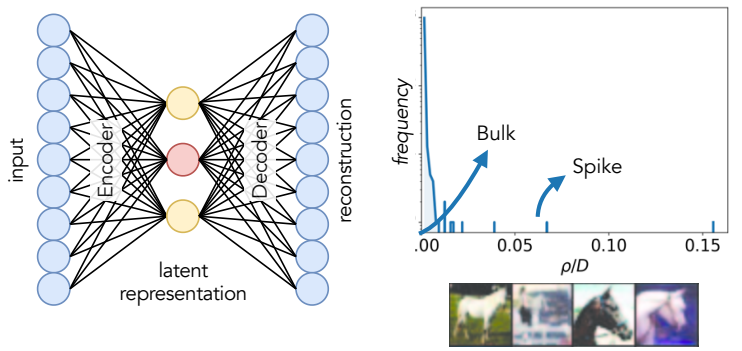
$$\mathrm{pmse} \equiv \frac{1}{D} \sum_i \mathbb{E}(x_i - \hat{x}_i)^2, \tag{4.52}$$

where the expectation $\mathbb{E}$ is taken with respect to the data distribution. Nguyen [235] analysed shallow autoencoders in a complementary mean-field limit, where the number of hidden neurones K grows polynomially with the input dimension. Here, by focusing on the case $K < D$, we study the case where the hidden layer is a bottleneck. Nguyen [235] considers tied autoencoders, where a single set of weights is shared between encoder and decoder, $e^k = d^k$. We will see in 4.7.3 that untying the weights is critical to achieve a non-trivial reconstruction error in the regime $K < D$.

Data model We derive our theoretical results for inputs drawn from a spiked Wishart model [250, 315], which has been widely studied in statistics to analyse the performance of unsupervised learning algorithms. In this model, inputs are sampled according to

$$x_\mu = \mathbf{A} c_\mu + \sqrt{\sigma}\xi_\mu, \tag{4.53}$$

where μ is an index that runs over the inputs in the training set, x_μ is a D-dimensional vector whose elements are drawn i.i.d. from a standard Gaussian distribution and $\sigma > 0$. The matrix $\mathbf{A} \in \mathbb{R}^{D \times M}$ is fixed for all inputs, while we sample $c_\mu \in \mathbb{R}^M$ from some distribution for each μ. Different choices for $\mathbf{A}$ and the distribution of c allow modelling of Gaussian mixtures, sparse codes, and non-negative sparse coding.



(a) Autoencoder architecture (b) Typical inputs and their spectrum

Figure 4.7: **Shallow autoencoders and their inputs (a)** We analyse two-layer autoencoders with non-linear activation function in the hidden layer. **(b)** Top: Rescaled eigenvalues of the covariance matrix of inputs drawn from the spiked Wishart model in 4.53, which can be divided into a bulk and a finite number of outliers. Bottom: example inputs drawn from CIFAR10 [160], a benchmark data set we use for our experiments with realistic data in 4.7.4.

Spectral properties of the inputs The spectrum of the covariance of the inputs is determined by $\mathbf{A}$ and the distribution of c. It governs the dynamics and the performance of autoencoders (by construction, we have $\mathbb{E}\, x_i = 0$). If we set $c_\mu = 0$ in 4.53, inputs would be i.i.d. Gaussian vectors with an average covariance of $\mathbb{E} x_i x_j = \sigma^2 \delta_{ij}$, where δ_{ij} is the Kronecker delta. The *empirical* covariance matrix of a finite data set with $P \sim \mathcal{O}(D)$ samples has a Marchenko–Pastur distribution with a scale controlled by σ [218, 250]. By sampling the mth element of c as $c_m \sim \mathcal{N}(0, \tilde{\rho}_m)$, we ensure that the input-input covariance has M eigenvalues $\rho_m = D\tilde{\rho}_m$ with the columns of $\mathbf{A}$ as the corresponding eigenvectors. The remaining $D - M$ eigenvalues of the empirical covariance, i.e. the *bulk*, still follow the Marchenko-Pastur distribution. We further ensure that the reconstruction task for the autoencoder is well-posed by letting the outlier eigenvalues scale as $\rho_m \sim \mathcal{O}(D)$ (or $\tilde{\rho}_m \sim \mathcal{O}(1)$), resulting in spectra such as the one shown in 4.7(b). If the largest eigenvalues were of order one an autoencoder with a finite number of neurones K could not obtain a pmse better than random guessing as the input dimension $D \to \infty$. Thus, if the features in the inputs are not salient *enough* shallow non linear AE, trained in the online limit, fail to reconstruct their inputs.

Training As in Sec. 4.1, we train the autoencoder on the quadratic error in the online limit of SGD. The SGD weight increments for the network parameters are

then given by:

$$\mathrm{d}e_i^k = -\frac{\eta_W}{D}\left(\frac{1}{\sqrt{D}}\sum_j d_j^k \Delta_j g'(a^k)x_i\right) - \frac{\lambda}{D}e_i^k, \tag{4.54a}$$

$$\mathrm{d}d_i^k = -\frac{\eta_V}{D}g(a^k)\Delta_i - \frac{\lambda}{D}d_i^k, \tag{4.54b}$$

where $\Delta_i \equiv (\hat{x}_i - x_i)$ and $\lambda \geq 0$ is the ℓ_2 regularisation constant. In order to obtain a well defined limit in the limit $D \to \infty$, we rescale the learning rates as $\eta_e = \eta/D$, $\eta_d = \eta$ for some fixed $\eta > 0$.

4.7.3 Results

Dynamical equations to describe feature learning

The theoretical derivation of the dynamical equations builds on the one of described in Sec. 4.1 and extends it to the case of unsupervised learning. The starting point is the observation that the pmse defined in 4.52 depends on the inputs only via their low-dimensional projections on the network's weights

$$a^k \equiv \frac{e^k x}{\sqrt{D}}, \qquad lft^k \equiv \frac{d^k x}{D}. \tag{4.55}$$

This allows us to replace the high-dimensional expectation over the inputs in 4.52 with a K-dimensional expectation over the local fields $a = (a^k)$ and $\tilde{a} = (lft^k)$. For Gaussian inputs drawn from (4.53), $(a, \tilde{a})$ are jointly Gaussian and their distribution is entirely characterised by their second moments:

$$T_1^{k\ell} \equiv \mathbb{E}\, lft^k lft^\ell = \frac{1}{D^2}\sum_{i,j} d_i^k \Omega_{ij} d_j^\ell, \tag{4.56}$$

$$Q_1^{k\ell} \equiv \mathbb{E}\, a^k a^\ell = \frac{1}{D}\sum_{i,j} e_i^k \Omega_{ij} e_j^\ell, \tag{4.57}$$

$$R_1^{k\ell} \equiv \mathbb{E}\, lft^k a^\ell = \frac{1}{D^{3/2}}\sum_{i,j} d_i^k \Omega_{ij} e_j^\ell. \tag{4.58}$$

These macroscopic overlaps, the *order parameters*, together with $T_0^{kl} = \sum_i d_i^k d_i^\ell / D$, are sufficient to evaluate the pmse. To analyse the dynamics of learning, the goal is then to derive a closed set of differential equations which describe how the order parameters evolve as the network is trained using SGD Eq. 4.54. Solving these equations then yields performance of the network at all times. Note that feature learning requires the weights of the encoder and the decoder move far away from their initial values. Hence, we cannot resort to the framework of the neural tangent

kernel or *lazy learning* [61, 145].

Derivation: a sketch

Here, we sketch the derivation of the equation of motion for the order parameter T_1 and skip precise computations as it is very close to the one of Sec. 4.1; a complete derivation for T_1 and all other order parameters is given in D.2. We define the eigen-decomposition of the covariance matrix $\Omega_{rs} = {}^1/_D \sum_{\tau=1}^D \Gamma_{s\tau}\Gamma_{r\tau}\rho_\tau$ and the rotation of any vector $z \in \{e^k, d^k, x\}$ onto this basis as $z_\tau \equiv {}^1/\sqrt{D} \sum_{s=1}^D \Gamma_{\tau s} z_s$. The eigenvectors are normalised as $\sum_i \Gamma_{\tau i}\Gamma_{\tau' i} = D\delta_{\tau\tau'}$ and $\sum_\tau \Gamma_{\tau i}\Gamma_{\tau j} = D\delta_{ij}$. Using this decomposition, we can re-write T_1 and its update at step μ as:

$$\left(T_1^{k\ell}\right)_{\mu+1} - \left(T_1^{k\ell}\right)_{\mu} = \frac{1}{D^2}\sum_\tau \rho_\tau \left(\mathrm{d}d_\tau^k d_\tau^\ell + d_\tau^k \mathrm{d}d_\tau^\ell + \mathrm{d}d_\tau^k \mathrm{d}d_\tau^\ell\right).$$

We introduce the order-parameter density $t(\rho, s)$ that depends on ρ and on the normalised number of steps $s = {}^\mu/_D$, which we interpret as a continuous time variable in the limit $D \to \infty$,

$$t^{k\ell}(\rho, s) = \frac{1}{D^2 \epsilon_\rho} \sum_\tau d_\tau^k d_\tau^\ell \mathbf{1}, \tag{4.59}$$

where $\mathbb{1}(.)$ is the indicator function and the limit $\epsilon_\rho \to 0$ is taken after the thermodynamic limit $D \to \infty$. Now, inserting the update for d Eq. 4.54 in the expression of $t^{k\ell}$, and evaluating the expectation over a fresh sample x, we obtain the dynamical equation:

$$\begin{aligned}\frac{\partial t^{k\ell}(\rho, s)}{\partial s} = \eta \Bigg(& \frac{\mathbb{E}\, a^k g(a^k)}{Q_1^{kk}} \frac{\rho}{D} r^{\ell k}(\rho, s) \\ & - \sum_a t^{\ell k}(\rho, s) \mathbb{E}\, g(a^k) g(a^a) \Bigg) + (k \leftrightarrow \ell).\end{aligned} \tag{4.60}$$

The order parameter is recovered by integrating the density $t^{\ell k}(\rho, s)$ over the spectral density μ_Ω of the covariance matrix: $T_1^{k\ell} = \int \mathrm{d}\mu_\Omega(\rho)\, \rho/D\, t^{k\ell}(\rho, s)$. We can finally close the equations by evaluating the remaining averages such as $\mathbb{E}\, a^k g(a^k)$. For some activation functions, such as the ReLU or sigmoidal activation $g(x) = \mathrm{erf}\left({}^x/\sqrt{2}\right)$, these integrals have an analytical closed form. In any case, the averages only depend on the second moments of the Gaussian random variables $(a, \tilde{a})$, which are given by the order parameters, allowing us to close the equations on the order parameters.

Separation between bulk modes and principal components

We can exploit the separation of bulk and outlier eigenvalues to decompose the integral Eq. 4.59 into an integral over the *bulk* of the spectrum and a sum over the outliers. This results in a decomposition of the overlap T_1 as:

$$\mathrm{T}_1 = \sum_{i \in \text{outliers}}^{M} \tilde{\rho}_i t_i + \mathrm{T}_{\text{bulk}}, \quad \mathrm{T}_{\text{bulk}} = \frac{1}{D} \int_{\rho \in \text{bulk}} \mathrm{d}\mu_\Omega(\rho) \rho t(\rho), \tag{4.61}$$

and likewise for other order parameters. Leveraging the fact that the bulk eigenvalues are of order $\mathcal{O}(1)$, and that we take the $D \to \infty$ limit, we can write the equation for T_{bulk} as:

$$\frac{\partial T^{kl}_{\text{bulk}}}{\partial s} = -\eta \sum_a T^{al}_{\text{bulk}} \mathbb{E}\, g(a^k) g(a^a) + (k \leftrightarrow \ell). \tag{4.62}$$

On the other hand, the M outlier eigenvalues, i.e. the spikes, are of order $\mathcal{O}(D)$. We thus need to keep track of all the terms in their equations of motion. Similarly to 4.63, they are written:

$$\frac{\partial t^{k\ell}_\tau}{\partial s} = \eta \left(\frac{\mathbb{E}\, a^k g(a^k)}{Q^{kk}_1} \tilde{\rho}_\tau r^{\ell k}_\tau - \sum_a t^{\ell k}_\tau \mathbb{E}\, g(a^k) g(a^a) \right) + (k \leftrightarrow \ell) \tag{4.63}$$

Note that the evolution of $\{t_i\}_{i=1,..,M}$ is coupled to the one of T_{bulk} only through the expectations of the local fields. A similar derivation, which we defer to D.2, can be carried out for R_1 and Q_1.

Empirical verification

In 4.8, we plot the pmse of an autoencoder with $K = 3$ hidden neurones during training on a dataset drawn from 4.53 with $M = 3$. We plot the pmse both obtained from training the network using SGD Eq. 4.54 (crosses) and obtained from integrating the equations of motion that we just derived (lines). The agreement between the theoretical description in terms of bulk and spike modes captures the training dynamics well, even at an intermediate input dimension of $D = 1000$. In the following, we analyse these equations, and hence the behaviour of the autoencoder during training, in more detail.

Nonlinear autoencoders learn principal components sequentially

The dynamical equations of the spike mode order parameters Eq. D.29, (D.33) and

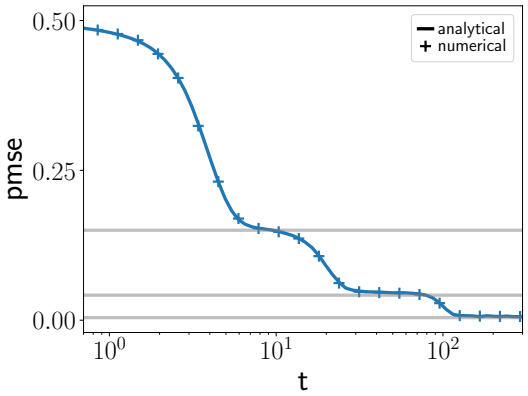


Figure 4.8: **The dynamical equations describe the learning dynamics of autoencoders** Prediction mean-squared error pmse Eq. 4.52 of an autoencoder with $K=3$ hidden neurones during training on with stochastic gradient descent on the spiked Wishart model in the one-pass limit. We obtained the pmse directly from a simulation (crosses) and from integration of the dynamical equations describing learning that we derive in 4.7.3 (line). Horizontal lines indicate the PCA reconstruction error with increasing numbers of principal components. *Parameters:* $M=3, \eta=1$, $D=1000$.

Eq. D.34 take the form:

$$\frac{\partial q^{k\ell}(\rho, s)}{\partial s} = \eta \tilde{\rho}\{\cdots\}, \tag{4.64}$$

etc. The learning rate of each mode is rescaled by the corresponding eigenvalue. Therefore the principal component corresponding to each mode is learnt sequentially, from the largest to the smallest. Indeed, the pmse of the sigmoidal autoencoder shown in 4.8 goes through several sudden decreases, preceded by plateaus with quasi-stationary pmse. Each transition is related to the network "picking up" an additional principal component. By comparing the error of the network on the plateaus to the PCA reconstruction error with an increasing number of principal components (solid horizontal lines in 4.8), we confirm that the plateaus correspond to stationary points where the network has learnt (a rotation of) the first leading principal components. Whether or not the plateaus are clearly visible in the curves depends on the separation between the eigenvalues of the leading eigenvectors.

This sequential learning of principal components has also been observed in several other models of learning. It appears in unsupervised learning rules such as Sanger's rule [273] (aka as generalised Hebbian learning, cf. D.1) that was analysed by Biehl and Schlösser [37]. Gidel et al. [111], Gunasekar et al. [125] found a sudden transition from the initial to the final error in linear models $\hat{x}_i = \sum_{j=1}^{D} e_{ij} x_j$, but step-wise transitions for factorised models, i.e. linear autoencoders of the form

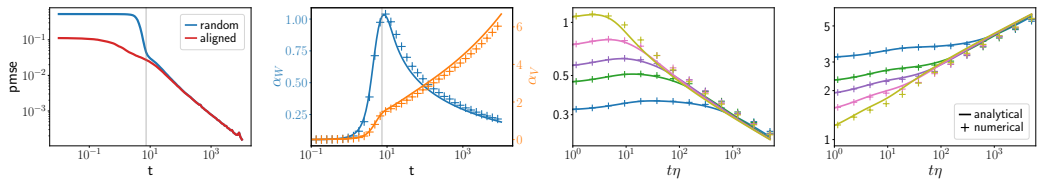


Figure 4.9: **Two phases of learning in sigmoidal autoencoders. (a)** Prediction error pmse Eq. 4.52 of a sigmoidal autoencoder with a single hidden neurone starting from random initial conditions (blue). The error first decays exponentially, then as a power law. During this second phase of learning, the pmse is identical to the pmse of an autoencoder whose weights are proportional to the leading eigenvector Γ of the input covariance at all times: $e(t) \propto a_e(t)\Gamma, \quad d(t) \propto a_d(t)\Gamma$, cf. 4.66. **(b)** Dynamics of the scaling constants a_e and a_d obtained through simulations of an autoencoder starting from random weights (crosses) and from integration of a reduced set of dynamical equations, D.47. **(c,d)** The norm of the encoder (left) and decoder (right) weights. Their weights shrink, respectively grow, as a power law $\sqrt{\rho}(\eta t)^{\pm\delta}$ with exponent $\delta \simeq 1/6$. This allows the sigmoidal network to remain in the linear region of its activation function and to achieve PCA performance. Solid lines are obtained from integration of a reduced set of dynamical equations, D.47, crosses are from simulations for various combinations of learning rates and leading eigenvalue ρ (different colours). *Parameters:* $D = 500, \eta = 1(a,b), M = 1, K = 1$.

$\hat{x}_i = \sum_{k=1}^{K}\sum_{j=1}^{D} d_i^k e_j^k x_j$. Saxe et al. [277] also highlighted the sequential learning of principal components, sorted by singular values.

The evolution of the bulk order parameters obeys

$$\begin{aligned} \frac{\partial T^{kl}_{\text{bulk}}}{\partial s} &= -\eta \sum_a T^{al}_{\text{bulk}} \mathbb{E}\, g(a^k) g(a^a) + (k \leftrightarrow \ell) \\ \frac{\partial R^{kl}_{\text{bulk}}}{\partial s} &= -\eta \sum_a R^{al}_{\text{bulk}} \mathbb{E}\, g(a^k) g(a^a), \qquad \frac{\partial Q^{kl}_{\text{bulk}}}{\partial s} = 0 \end{aligned} \tag{4.65}$$

As a consequence, in the early stages of training, to first approximation, the bulk component of R_1 and T_1 follow an exponential decay towards 0. The characteristic time of this decay is given by the expectation $\mathbb{E}\, g(a^k) g(a^\ell)|_{s=0}$ which only depends on $Q_1|_{s=0}$. In addition, the last equation implies that the evolution of Q_1 is entirely determined by the dynamics of the spike modes.

The long-time dynamics of learning: align, then rescale

At long training times, we expect a sigmoidal autoencoder with untied weights to have retrieved the leading PCA subspace since it achieves the PCA reconstruction error. This motivates an ansatz for the dynamical equations where the network

weights are proportional to the eigenvectors of the covariance Γ^k,

$$e_i^k(t) = \alpha_e^k(t)/\sqrt{D}\, \Gamma_i^k \qquad d_i^k = \alpha_d^k(t)\Gamma_i^k, \tag{4.66}$$

where $\alpha_e, \alpha_d \in \mathbb{R}^K$ are scaling constants that evolve in time. With this ansatz, the order parameters are diagonal, and we are left with a reduced set of $2K$ equations describing the dynamics of α_e^k and α_d^k which are given in D.47 together with their detailed derivation.

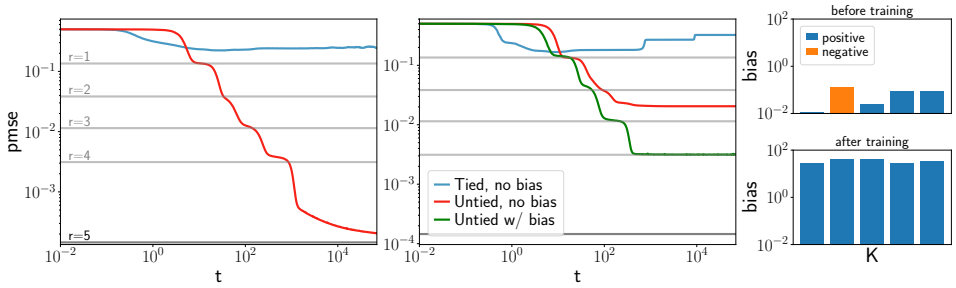


Figure 4.10: **Shallow autoencoders require untied weights, and ReLU autoencoders also need biases. (a)** Prediction error pmse Eq. 4.52 of sigmoidal autoencoders with untied weights (red) and with tied weights $d^k = e^k$ (blue). Horizontal lines indicate PCA errors for rank r. **(b)** Same plot for ReLU autoencoders with three different architectures: tied weights, no bias (blue), untied weights without bias (red) and untied weights with bias (green). Only the latter architecture achieves close to PCA error. **(c)** Distribution of biases in a ReLU network before (top) and after training (bottom). The biases increase throughout training, hence pushing the network into the linear region of its activation function. *Parameters:* $D = 1000, K = 5, \eta = 1$.

We first verify the validity of the reduced equations Eq. D.47 to describe the long-time dynamics of sigmoidal autoencoders. In 4.9(a), we show the generalisation dynamics of a sigmoidal autoencoder starting from random initial conditions (blue). We see two phases of learning: after an exponential decay of the pmse up to time $t \sim 10$, the pmse decays as a power-law. This latter decay is well captured by the evolution of the pmse of a model which we initialised with weights aligned to the leading PCs. We deduce that during the exponential decay of the error, the weights align to the leading PCA directions and stay aligned during the power-law decay of the error.

After having recovered the suitable subspace, the network adjusts the norm of its weights to decrease the error. A look at the dynamics of the scale parameters α_e and α_d during this second phase of learning reveals that the encoder's weights shrink, while the decoder's weights grow, cf. 4.9(b). Together, these changes lead to the power-law decay of the pmse. Note that we chose a dataset with only one

leading eigenvalue so as to guarantee that the AE recovers the leading principal component, rather than a rotation of them. A scaling analysis of the evolution of the scaling constants α_e^k and α_d^k shows that the weights decay, respectively grow, as a power-law with time, $\alpha_e^k \propto 1/\sqrt{\rho}\,(\eta t)^{-\delta}$ and $\alpha_d^k \propto 1/\sqrt{\rho}\,(\eta t)^{\delta}$, see 4.9(c-d).

We can understand this behaviour by recalling the linearity of $g(x) = \mathrm{erf}\,(x/\sqrt{2})$ around the origin, where $g(x) \sim \sqrt{2/\pi}\; x$. By shrinking the encoder's weights, the autoencoder thus performs a linear transformation of its inputs, despite the non-linearity. It recovers the linear performance of a network given by PCA if the decoder's weights grow correspondingly for the reconstructed input $\hat{x}$ to have the same norm as the input x.

We can find a relation between the scaling constants α_d^k and α_e^k at long times using the ansatz 4.66 in the expression of the pmse:

$$\mathrm{pmse} = \sum_{k=1}^{K} \underbrace{\left\{ \tilde{\rho}_k + \alpha_d^{k2} \mathbb{E}\, g(a^k)^2 - 2\mathbb{E}\, \tilde{a}^k g(a^k) \right\}}_{f(\alpha_d^k, \alpha_e^k)} + \sum_{k>K}^{D} \tilde{\rho}_k, \tag{4.67}$$

where the second term is simply the rank K PCA error. The first term should be minimised to achieve PCA error, i.e. $f(\alpha_e^k, \alpha_d^{k*}) = 0$ for all k. For a linear autoencoder, we find $\alpha_d^{k*} = 1/\alpha_e^k$: as expected, any rescaling of the encoder's weights needs to be compensated in the decoder. For a sigmoidal autoencoder instead, we find

$$\alpha_d^{k*} = \frac{1}{\alpha_e^k} \sqrt{\frac{\pi}{2}\left(1 + \tilde{\rho}^k \alpha_e^{k2}\right)} \tag{4.68}$$

Note, that at small α_e^k we recover the linear scaling $\alpha_d^{k*} \sim \left(g'(0)\alpha_e^k\right)^{-1}$.

The importance of untying the weights for sigmoidal autoencoders The need to let the encoder and decoder weights grow, resp. shrink, to achieve PCA error makes learning impossible in sigmoidal autoencoders with tied weights, where $d^k = e^k$. Such an autoencoder evidently cannot perform the rescaling required to enter the linear regime of its activation function, and is therefore not able able to achieve PCA error. Even worse, the learning dynamics shown in 4.10(a) show that a sigmoidal autoencoder with tied weights (blue) hardly achieves a reconstruction error better than chance, in stark contrast to the same network with untied weights (red).

The importance of the bias in ReLU autoencoders

Sigmoidal autoencoders achieve the PCA error by exploiting the linear region of their activation function. We can hence expect that in order to successfully train a ReLU AE, which is linear for all positive arguments, it is necessary to add biases at the hidden layer. The error curves for ReLU autoencoders shown in 4.10(b) show

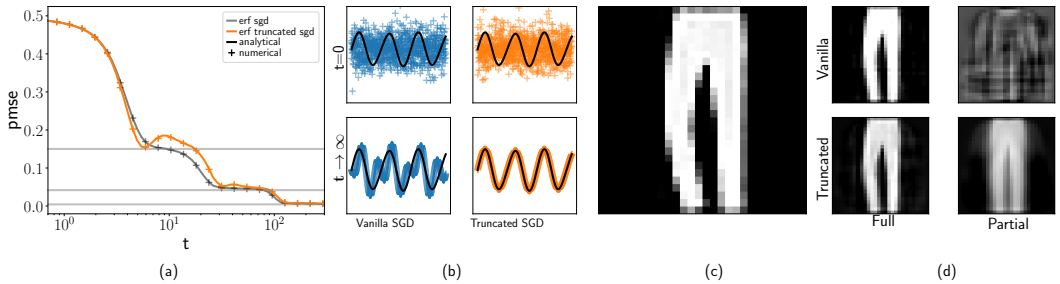


Figure 4.11: **Breaking the symmetry between neurones yields the exact principal components of the data. (a)** Prediction mean-squared error Eq. 4.52 of an autoencoder trained using vanilla SGD (eq. 4.54, blue) and truncated SGD (eq. 4.70, orange). Grey horizontal lines indicate PCA reconstruction errors. **(b)** We show the weight vector of the first neurone of the autoencoder before and after training (top vs bottom) with vanilla and truncated SGD (left and right, resp.). In contrast to vanilla SGD, truncated SGD recovers the true leading principal component of the inputs, a sinusoidal wave (black). **(c)** Example image taken from Fashion MNIST. **(d)** The left column shows reconstructions of the image from (c) using all $K = 64$ neurones of an autoencoder trained with vanilla (top) and truncated SGD (bottom) on the full Fashion MNIST database. The right column shows reconstructions using only the first 5 neurones of the same autoencoders. *Parameters:* $\eta = 1$, $K = 4$ *((a) and (b))*, $K = 64$ *((c) and (d))*, $B = 1, P = 60000$.

indeed that ReLU autoencoders achieve an error close to the PCA error only if the weights are untied and the biases are trained. If the biases are not trained, we found that a ReLU autoencoder with K hidden neurones generally achieves a reconstruction error of roughly $K/2$, presumably because only $K/2$ nodes have a positive pre-activation and can exploit the linear region of ReLU. Training the biases consistently resulted in large, positive biases at the end of training, 4.10(c). The large biases, however, result in a small residual error that negatively affects the final performance of a ReLU network when compared to the PCA performances, as can be seen from linearising the output of the ReLU autoencoder:

$$\hat{x} = \sum_k d^k \max(0, a^k + b^k) = \underbrace{\sum_k d^k a^k}_{\text{PCA}} + \underbrace{\sum_k d^k b^k}_{\text{residual}}, \tag{4.69}$$

where $b^k \in \mathbb{R}$ is the bias of the kth neurone.

Breaking the symmetry of SGD yields the exact principal components

Linear autoencoders do not retrieve the exact principal components of the inputs, but some rotation of them due to the rotational symmetry of the square loss [41]. While this does not affect performance, it would still be desirable to obtain the

leading eigenvectors exactly to identify the important features in the data.

The symmetry between neurones can be broken in a number of ways. In linear autoencoders, previous work considered applying different amounts of regularisation to each neurone [25, 165, 219, 248] or optimising a modified loss [239]. In unsupervised learning, Sanger's rule [273] breaks the symmetry by making the update of the weights of the first neurone independent of all other neurones; the update of the second neurone only depends on the first neurone, etc. This idea was extended to linear autoencoders by Bao et al. [25], Oftadeh et al. [239]. We can easily extend this approach to non-linear autoencoders by training the decoder weights following:

$$\mathrm{d}d_i^k = -\frac{\eta_V}{D} g(a^k) \left(x_i - \sum_{\ell=1}^{k} d_i^\ell g(a^\ell) \right), \tag{4.70}$$

where the sum now only runs up to k instead of K. The update for the encoder weights e^k is unchanged from 4.54, but the error term is changed to $\Delta_i^k = x_i - \sum_{\ell=1}^{k} d_i^\ell g(a^\ell)$.

We can appreciate the effect of this change by considering a fixed point $\{e^*, d^*\}$ of the vanilla SGD updates 4.54 with $\mathrm{d}e^{*k} = 0$ and $\mathrm{d}d^{*k} = 0$. Multiplying this solution by an orthogonal matrix $\mathbf{O} \in O(K)$ i.e. $\{\tilde{e}, \tilde{d}\} = \{\mathbf{O}e^*, \mathbf{O}d^*\}$ yields another fixed point, since

$$\begin{aligned} \mathrm{d}\tilde{v}_i^k &= -\frac{\eta_V}{D} g(lft^k) \left(x_i - \sum_{\ell_1 \ell_2 = 1}^{K} \underbrace{\sum_{\ell=1}^{K} O^{\ell\ell_2} O^{\ell\ell_3}}_{\delta^{\ell_2 \ell_3}} d_i^{*\ell_2} g(a^{*\ell_3}) \right) \\ &= -\frac{\eta_V}{D} g(lft^k) \left(x_i - \sum_{\ell=1}^{K} d_i^{*\ell} g(a^{*\ell}) \right) = 0 \end{aligned} \tag{4.71}$$

for the decoder, and likewise for the encoder weights. For truncated SGD Eq. 4.70, the underbrace term in 4.71 is no longer the identity and the rotated weights $\{\tilde{e}, \tilde{d}\}$ are no longer a fixed point.

We show the pmse of a sigmoidal autoencoder trained with vanilla and truncated SGD in 4.11(a), together with the theoretical prediction from a modified set of dynamical equations. The theory captures both learning curves perfectly, and we can see that using truncated SGD incurs no performance penalty - both autoencoders achieve the PCA reconstruction error. Note that the reconstruction error can increase temporarily while training with the truncated algorithm since the updates do not minimise the square loss directly. In this experiment, we trained the

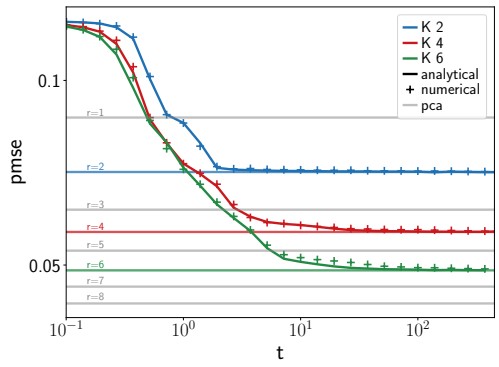


Figure 4.12: **Theory vs. simulations for sigmoidal autoencoders trained on CIFAR10.** Prediction mean-squared error pmse Eq. 4.52 of a sigmoidal autoencoder with $K = 2, 4, 6$ hidden neurones trained on 10 000 greyscale CIFAR10 images [160] (crosses). Solid lines show the pmse predicted from integrating the dynamical equations of 4.7.3, using the empirical covariance matrix of the training inputs. Horizontal lines indicate the PCA error with varying rank r. *Parameters:* $D = 1024, \eta = 1$.

networks on a synthetic dataset where the eigenvectors are chosen to be sinusoidal (black lines in 4.11 b). As desired, the weights of the network trained with truncated SGD converges to the exact principal components, while vanilla SGD only recovers a linear combination of them.

The advantage of recovering the exact principal components is illustrated with the partial reconstructions of a test image from the FashionMNIST database [317]. We train autoencoders with $K = 64$ neurones to reconstruct FashionMNIST images using the vanilla and truncated SGD algorithms until convergence. We show the reconstruction using all the neurones of the networks in the left column of 4.11(d). Since the networks achieve essentially the same performance, both reconstructions look similar. The partial reconstruction using only the first five neurones of the networks shown on the right are very different: while the partial reconstruction from truncated SGD is close to the original image, the reconstruction of the vanilla autoencoder shows traces of pants mixed with those of a sweatshirt.

4.7.4 Representation learning on realistic data

We have derived a series of results on the training of non-linear AEs derived in the case of synthetic, Gaussian datasets. Most of our simplifying assumptions are not valid on real datasets, because they contain only a finite number of samples, and because these samples are finite-dimensional and not normally distributed. However, it turns out that the dynamical equations of 4.7.3 describe the dynamics

of an autoencoder trained on more realistic images rather well.

In 4.12, we show the pmse of autoencoders of increasing size trained on 10k greyscale images from CIFAR10 [160] with crosses. The solid lines are obtained from integration the equations of motion of D.2, using the empirical covariance of the training images.

The accuracy of the equations to describe the learning dynamics at all times implies that the low-dimensional projections a and $\tilde{a}$ introduced in 4.55 are very close to being normally distributed. This is by no means obvious, since the images clearly do not follow a normal distribution and the weights of the autoencoder are obtained from training on said images, making them correlated with the data. Instead, 4.12 suggests that from the point of view of a shallow, sigmoidal autoencoder, the inputs can be replaced by Gaussian vectors with the same mean and covariance without changing the dynamics of the autoencoder. In D.1 in the appendix, we show that this behaviour persists also for ReLU autoencoders and, as would be expected, for linear autoencoders. Nguyen [235] observed a similar phenomenon for tied-weight autoencoders with a number of hidden neurones that grows polynomially with the input dimension D.

This behaviour is an example of the Gaussian equivalence principle, or Gaussian universality, that received a lot of attention recently in the context of supervised learning with random features [189, 215, 284] or one- and two-layer neural networks [117, 118, 138, 199]. These works showed that the performance of these models is asymptotically well captured by an appropriately chosen Gaussian model for the data. The result closest to our setup was obtained by Goldt et al. [118], who showed that for two-layer networks, the Gaussian equivalence was, to a first approximation, a result of projecting high-dimensional inputs to a low-dimensional set of local fields a in the limit where the number of hidden neurones $K \sim \mathcal{O}(1)$ compared to the input dimension. Intriguingly, we found that the Gaussian equivalence in autoencoders extends to essentially any number of hidden neurones, for example $K = D/2$, as we show in D.1 in the appendix.

We finally verified that several insights developed in the case of Gaussian data carry over to real data. In particular, we demonstrate in D.2 that sigmoidal AE require untied weights to learn, ReLU networks require adding biases to the encoder weights, and that the principal components of realistic data are also learnt sequentially.

Chapter 5

Algorithm

In Chap. 3 and Chap. 4, we understood how theoretical tools can be used to gain insight into role of architecture and data structure in learning with artificial neural networks. We also described the importance of understanding the interplay between the two aspects. However, in all discussions, we bypassed the question of the impact of the algorithm used in training. Indeed, we considered either training with vanilla SGD (Sec. 4.2, Sec. 4.7) or performed a static analysis of the solution to which NN converge (Chap. 3).

In training deep neural networks, making the proper choice of algorithm, including that of the various hyper-parameters involved, is crucial. Back-propagation is certainly the most widely used algorithm, but to perform best practitioners spend hours adjusting the gigantic number of add-ons such as learning-rate schedule, regularisation scheme, weights initialisation, optimiser, etc. Each one of these requires fixing multiple hyper-parameters. These procedure is straining and time consuming. Gaining theoretical insights into the effect different choices have on learning is key to facilitate the task and improve performance.

In this view, we analyse in Sec. 5.1 the role of using a learning rate schedule in optimisation problems. We focus on power law decays of the learning rate $\eta(t) = \eta_0/t^\beta$. The results provide an optimal exponent β_{opt} with which to decay the learning rate and theoretic intuition behind how this value depends on the complexity of the loss landscape.

Sec. 5.2 analyses instead alternative algorithms to vanilla BP, which overcome some of the latter's pitfall. First of all, we know from Sec. 1.4 that BP requires the backward pass to be done sequentially, for the last layers to the firsts. This prevents efficient parallelisation of the updates increasing the training time of deep NN. In addition, in BP, the same weights are used in the forward and the backward pass. Since, it is unlikely that in biological neural networks the same synapses transmit

the information forward and backward, BP remains a biologically implausible algorithm. Additional problems include that of vanishing gradients, sensitiveness to external attacks, etc. In light of these shortcomings, in recent years different algorithms which only approximate the BP gradient have been proposed. Sec. 5.2 focuses on the study of one such algorithm: *direct feedback-alignment* (DFA). By analysing learning with DFA on two layer neural networks described in Sec. 4.1 and on deep linear networks, we establish precise conditions under which networks trained with DFA perform well. We then verify that the conclusions hold in deep non-linear models of neural networks.

We will see however, that we cannot study algorithm without considering its interplay with architecture and data structure. Indeed, even though DFA successfully trains fully-connected neural networks on various input distributions, it notoriously fails on convolutional neural networks. The analytical results of Sec. 5.2 provide some insight as to why. They also demonstrate that structure in the inputs prevent NN trained with DFA to learn. Thus, even though theoreticians can study either one of the architecture, the data structure or the algorithm by keeping the other two fixed, the real challenge is to understand the interplay between them i.e. answering the question of given a task, which is the best architecture-algorithm combination to solve it?

5.1 Learning rate schedules and how they improve BP performance

5.1.1 Overview

When training neural networks with BP, one is required to fix various *hyper-parameters*, i.e. variables that configure various aspect of optimisation. Among which the learning rate, i.e. the size of the step taken by the algorithm at every update, has plays a key role and a wrong choice can completely prevent a network from learning. Until now, only few guidelines on how to choose the appropriate learning rate for a given problem exist and the choice is almost always done on a trial-and-error basis. In this section, based on [77], we use statistical physics tools to provide insights into what is the impact of the learning rate, and of learning rate schedules, on optimisation. Such analysis is key to develop guidelines to chose hyper-parameters and accelerate the design of machine learning methods. Learning rate schedules allow to change the learning rate during training according to some protocol and are used across all areas of modern machine learning. They provide more flexibility than fixed learning rate methods since the training step size is

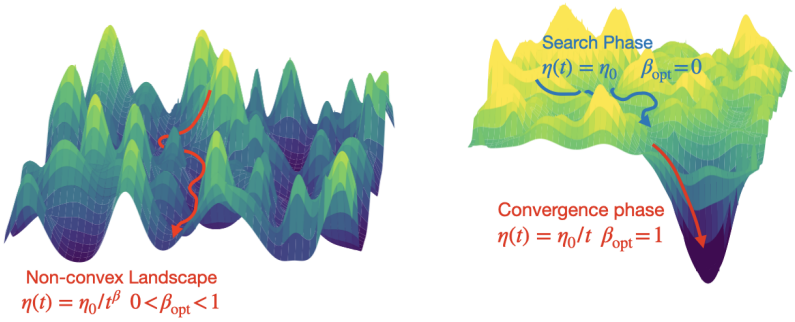


Figure 5.1: **The optimal learning rate schedule depends on the structure of the landscape.** *(Left):* in the purely non-convex landscapes of Sec. 5.1.3, the learning rate must be decayed as $\eta(t) = \eta_0/t^\beta$ with $\beta < 1$ to speed up optimisation. *(Right):* the landscapes of Secs. 5.1.4 and 5.1.5 feature basins of attraction due to the presence of a signal to recover. One must first keep a large constant learning rate to escape the rough parts of the landscape as quickly as possible, then decay the learning rate as $\eta(t) = \eta_0/t$ once inside a convex basin.

adjusted depending on the phase of optimisation one is at. Despite their wide spread, very little is known on which schedule is most suited for a given problem.

This question has been thoroughly studied for convex problems, where the optimal learning rate schedule generally goes as $\eta(t) \sim 1/t$ [182, 194]. However, deep neural networks and other high-dimensional modern optimisation problems are known to operate in highly non-convex loss landscapes [48, 64]. In this section we make some progress towards a theory to understand the impact of scheduling in these settings.

Here, we focus on the analytical study of learning-rate schedules in gradient-based algorithms and on the high-dimensional inference problem of retrieving a ground truth signal $w^\star \in \mathrm{R}^N$ from observations via a noisy channel. In the teacher-student jargon of previous chapters, the teacher's weights are the signal which much be recovered by her student. When the noise dominates the signal, the loss simply boils down to a Gaussian random function on the N-dimensional sphere ($N \to \infty$). This optimisation problem has been studied in the literature for constant learning rate, both using rigorous methods and techniques from statistical physics, see [13, 33, 79, 206, 327] and references therein.

Setup Learning rate decay is generally used to reduce the noise induced by optimisation schemes used in practice. For example, SGD with batch size B typically induces a noise which scales as the learning rate divided by batch size η/B [147, 222, 245, 292]. To mimick this optimisation noise, we focus on Langevin dynamics [59, 139, 182, 223]. Given a loss function $\mathcal{L}$ and a temperature T, this consists in minimising $\mathcal{L}$ by updating the estimate $w \in \mathbb{R}^N$ of the signal from a

random initial condition according to the equation:

$$\frac{\mathrm{d}w_i(t)}{\mathrm{d}t} = -\eta(t)\left(\frac{\partial \mathcal{L}(w, w^\star)}{\partial w_i} + \xi_i(t) + z(t)w_i(t)\right), \tag{5.1}$$

where $\xi(t) \in \mathrm{R}^N$ is a Gaussian noise with 0 mean and variance $\langle \xi_i(t)\xi_j(t')\rangle = 2T\delta_{ij}\delta(t-t')$, and the Lagrange multiplier $z(t)$ is used to enforce the spherical constraint $\|w\|^2 = N$ which we impose throughout the paper ($z(t)$ can be thought of as a weight decay that evolves during training to keep the norm of the estimator fixed). The temperature T represents the strength of the noise inherent to the optimisation algorithm, i.e. $1/B$ for SGD (we consider $T < 1$ in the following). To study scheduling, we decay the learning rate as $\eta(t) = \eta_0/t^\beta$, as commonly chosen in the literature [228, 319]. Note that here we are considering gradient-flow – our results are confirmed by experiments performed with gradient descent.

We consider two models for the loss $\mathcal{L}$: the (planted) *Sherrington-Kirkpatrick* (SK) model [288], where the signal is scrambled by a random matrix, and the more involved *spiked matrix-tensor* (SMT) model [207], where the signal is additionally observed through its contraction with a random tensor of order p. The first setup is analytically tractable both at infinite and finite dimensions [26, 71], and its landscape features a number of critical points which grows linearly with the dimension. The second setup is more involved and requires a mean-field treatment in the infinite dimensional limit [72, 207, 274]. The number of critical points grows exponentially with the dimension and has been studied analytically with the *Kac-Rice* method [34, 263]. This distinction allows us to grasp how the amount of non-convexity impacts the optimal decay of the learning rate.

Here, we begin by considering the purely non-convex setup where the signal is undetectable (left panel of Fig. 5.1). The loss is then a Gaussian random function on the N-dimensional sphere with zero mean and a covariance $\mathbb{E}[\mathcal{L}(w)\mathcal{L}(w')] \propto (w \cdot w')^p$. We determine the optimal learning rate to reach the lowest value of the loss function on an arbitrarily large (but finite) time in the high-dimensional limit. For the $p = 2$ case, corresponding to the spherical SK model, we find $\beta = 1/2$ whereas for $p > 2$ we obtain $\beta = 2/5$. The higher degree of non-convexity of the latter requires the learning rate to be decayed more slowly; we generalise these findings by leveraging results from out-of-equilibrium physics. Note that inverse square root decay is commonly used among practitioners in state-of-the-art endeavours such as training Transformers [307]; this analysis provides theoretical evidence for its soundness in a particular class of non-convex landscapes.

We then study the influence of a detectable signal (right panel of Fig. 5.1), and we determine the optimal learning rate schedule to find the signal in the shortest

amount of time. In this case, a crossover time emerges between two phases [43]: a *search* phase, where the signal is weak and the dynamics travel through a rugged landscape, followed by a *convergence* phase the signal is detected and the problem becomes locally convex. We show that the optimal schedule is to keep a large constant learning rate during the first phase to speed up the search, then, once in the convex basin, to decay the learning rate as $1/t$. This protocol allows to speed up convergence and find lower loss solutions, and is reminiscent of schedules used in practice.

We conclude this section with experiments that verify that these insights are reflected in practice when training neural networks on a teacher-student regression task with SGD.

Related work Typically, learning rate schedules consist in a large learning rate phase followed by a decay phase. A body of works have shown that this allows to learn easy patterns early on and complex patterns later [184, 323]. Although stepwise decays of the learning rate were used for a long time [100, 133], most recent works have turned to smooth decays such as inverse square root [307] and cosine annealing [197], which involve less hyperparameters to tune. Other possibilities include cyclical learning rates [291] and automatic schedulers [180].

The use of a warmup [121] before decaying the learning rate has shown to be effective in avoiding instabilities arising from large learning rates [113, 120]. Another common practice is to use adaptive optimizers, which select a different learning rate for each learning parameter [87, 153, 328], although these have been shown to often degrade generalization [58, 151, 314].

On the theoretical side, several works have studied Langevin dynamics for mean-field spin glasses. Particularly relevant to us are those which focus on the spherical SK setup [26, 71], as well as those showing the existence of a search and a convergence phase for the SMT model [207]. However, to the best of our knowledge, no previous works have studied these kind of highly non-convex optimisation problems in the context of a non-constant learning rate. Our analysis is based on common methods in theoretical physics which have been to a large extent made rigorous in recent years [33, 34, 79, 274], and is confirmed by numerical experiments.

Reproducibility The code to reproduce the figures in this paper is available at `https://github.com/mariaref/nonconvex-lr`.

5.1.2 The speed-noise trade-off in a simple convex problem

Before studying non-convex problems, it is instructive to recall the effect learning rate decay has on optimisation in a simple 1D convex basin of curvature κ, for

which $\mathcal{L}(w) = \frac{1}{2}\kappa w^2$. The Langevin equation (Eq. 5.1) can easily be solved and yields (see App. E.1):

$$\langle \mathcal{L}(t) \rangle = \underbrace{\frac{\kappa w(t_0)^2}{2} e^{-2\kappa \int_{t_0}^{t} \mathrm{d}\tau \eta(\tau)}}_{\bar{\mathcal{L}}(t)} + \underbrace{\frac{\kappa T}{2} \int_{t_0}^{t} \mathrm{d}t' \eta(t')^2 e^{-2\kappa \int_{t'}^{t} \mathrm{d}\tau \eta(\tau)}}_{\delta\mathcal{L}(t)}, \tag{5.2}$$

where $\langle . \rangle$ denotes an average over the noise ξ. The first term is an *optimisation* term, which amounts to forgetting the initial condition $w(t_0)$. It is present in absence of noise ($T = 0$) and its decrease is related to the way the dynamics descend in the loss landscape. The second term is a *noise* term, which is proportional to the strength of the noise T, and reflects the impact Langevin noise has on optimisation.

To converge to the solution $w = 0$ as quickly as possible, one is faced with a dilemma: reducing the learning rate suppresses the effect of the noise term $\delta\mathcal{L}$, but also slows down the dynamics, leading to a larger optimisation term $\bar{\mathcal{L}}$. The ideal tradeoff is found when these two effects are comparable. By taking $\eta(t) = \eta_0/t$ we obtain:

$$\bar{\mathcal{L}}(t) \propto t^{-2\eta_0\kappa}, \qquad \delta\mathcal{L}(t) \propto 1/t. \tag{5.3}$$

Hence, the loss decays to zero as $1/t$ if we take $\eta_0 \geq 1/2\kappa$, as found in many previous works [182, 194]. Note that if we take a slower decay such as $\eta(t) \sim 1/t^\beta$ with $\beta < 1$, $\bar{\mathcal{L}}(t)$ converges to 0 exponentially fast, but $\delta\mathcal{L}(t) \propto \eta(t)$ decays slower and bottlenecks the loss. Conversely, if we take a faster schedule, i.e. $\beta > 1$, then the noise term decays faster, but the dynamics stop before reaching the solution, as $\bar{\mathcal{L}}(t)$ does not converge to zero when $t \to \infty$.

This simple example illustrates the trade-off between the speed of optimisation and the noise suppressing effect, which will be the cornerstone of proper scheduling in the high-dimensional non-convex settings studied below.

5.1.3 Optimal decay rates in random landscapes

In this section, we consider purely non-convex optimisation landscapes, where the loss $\mathcal{L}$ is a Gaussian random function defined on the N-dimensional sphere, with zero mean and covariance:

$$\mathbb{E}[\mathcal{L}(x)\mathcal{L}(x')] = \frac{N}{2}(x \cdot x')^p, \quad p \geq 2.$$

This setup, which has been studied in great detail in the context of statistical physics, can be viewed as a special case of the inference problems of Sec. 5.1.4 where the noise is too strong for the signal to be detectable. The aim is not to retrieve a signal, but simply to decrease the loss as quickly as possible on an arbitrarily large (but finite) time.

Sherrington-Kirkpatrick model

We start by focusing on the case $p = 2$. This can be achieved with the spherical version of the spin glass model introduced by Sherrington and Kirkpatrick [288]. Here, the variables w_i and x_j interact with each other via random symmetric couplings* $J_{ij} \sim \mathcal{N}(0,1)$, and, as throughout the paper, are required to satisfy the spherical constraint $\|w(t)\|^2 = N$. The loss function is given by:

$$\mathcal{L}(w) = -\frac{1}{\sqrt{N}} \sum_{i<j}^{N} J_{ij} w_i w_j. \tag{5.4}$$

In this section, we consider the high-dimensional limit $N \to \infty$; finite-dimensional effects are discussed in Sec. 5.1.4.

Solving the dynamics

To obtain the value of the loss function at all times, we multiply the original Langevin equation by w_i and sum over all components. Using Ito's lemma, and the concentration of $z(t)$ in the $N \to \infty$ limit, leads to the simple relation:

$$\begin{aligned} 0 = \left\langle \frac{\partial \|w\|^2}{\partial t} \right\rangle &= \eta(t) \left[-2\mathcal{L}(t) - N z(t) \right] + N \eta(t)^2 T \\ \Rightarrow \mathcal{L}(t) &= -\frac{N}{2} \left(z(t) - \eta(t) T \right) \end{aligned} \tag{5.5}$$

As in the convex setup, we find a competition between an optimisation term and a noise term. Since the temperature is fixed, the latter decays as $\eta(t)$. To obtain the value of the Lagrange multiplier $z(t)$, we impose the spherical constraint at all times, yielding (see App. E.2):

$$z(t) = 2 - \frac{3(1-\beta)}{4t^{1-\beta}}. \tag{5.6}$$

Hence, the scaled loss $\ell = \mathcal{L}/N$ converges to the ground state (global minimum)

*As discussed in App. E.2, due to the universality typical of random matrix theory distributions, our results hold for a broad class of distributions for the couplings. Note also that the diagonal terms do not matter in the large N limit but for simplicity we take $J_{ii} \sim \mathcal{N}(0,2)$.

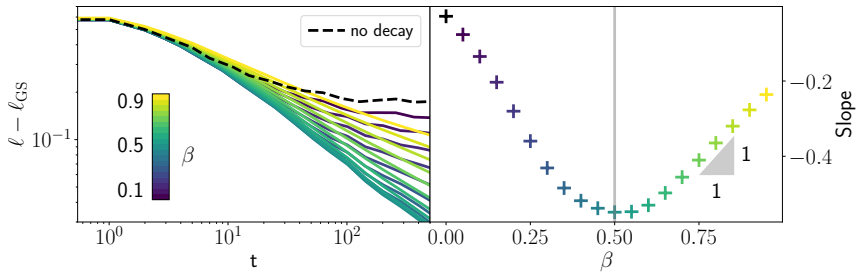


Figure 5.2: In the SK model, the optimal decay rate is $\beta_{opt} = 0.5$. (*Left*) Loss curves of the SK model when decaying the learning rate as $\eta(t) = \eta_0/t^\beta$ for various values of β (colored lines). (*Right*) Decay exponent of $\ell - \ell_{\rm GS}$ at long times as a function of β. We recognize a decay exponent of $\min(\beta, 1-\beta)$ as predicted by Eq. 5.7, which is fastest for $\beta_{\rm opt} = 1/2$. *Parameters:* $N = 3000$, $T = 1$, $\eta_0 = 0.1$.

$\ell_{GS} = -1$ as a sum of power-laws:

$$\ell(t) - \ell_{GS} = \frac{\eta_0 T}{2t^\beta} + \begin{cases} \frac{3(1-\beta)}{8\eta_0 t^{1-\beta}}, & \beta < 1 \\ \frac{3}{8\eta_0 \log t}, & \beta = 1 \end{cases} . \tag{5.7}$$

Optimal decay rate At long times, Eq. 5.7 implies a power-law decay of the loss with an exponent $\min(\beta, 1-\beta)$ due to the speed-noise tradeoff. Hence, the optimal decay rate at long times is $\beta_{\rm opt} = 1/2$. This is confirmed by numerical simulations at finite size, see Fig. 5.2. Note that this decay rate is empirically chosen to train many state-of-the-art neural networks such as the original Transformer [307], but, to the best of our knowledge, has never been justified from a theoretical point-of-view in a non-convex high-dimensional setting.

Curvature analysis To gain better understanding, it is informative to study the local curvature of the effective landscape the dynamics take place in. To do so, one needs to compute the spectrum of the effective Hessian taking into account the spherical constraint of Eq. 5.1, defined as:

$$\text{Hess} = \frac{1}{\sqrt{N}} J + z(t) I. \tag{5.8}$$

In the the $N \to \infty$ limit, the spectral density of the first term, defined as $\rho(\mu) = \sum_{i=1}^N \delta(\mu - \mu_i)$, converges to a semi-circle law [313]:

$$\rho_{sc}(\mu) = \frac{1}{2\pi}\sqrt{4 - \mu^2}, \quad \forall \mu \in [-2, 2]. \tag{5.9}$$

The spectral density of Hess is shifted to the right during the dynamics by the Lagrange multiplier $z(t)$, reflecting the way in which the local curvature changes

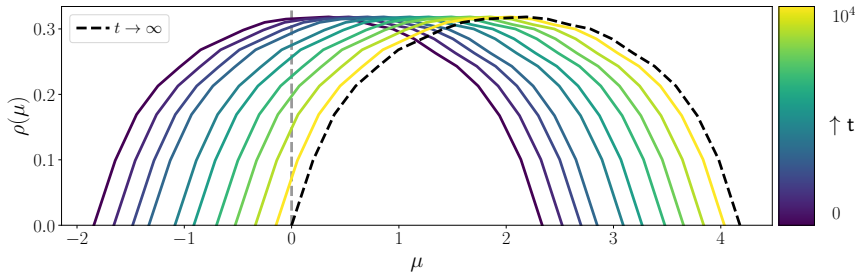


Figure 5.3: **In the SK model, the dynamics never reach a convex region.** During training, the local curvature, i.e. the spectral density of the Hessian (Eq. 5.8) shifts to the right. The left hand side of the spectrum only reaches 0 at $t \to \infty$, signalling that there remains negative eigenvalues at any finite time. *Parameters:* $N = 3000$, $T = 1$, $\eta_0 = 0.1$, $\beta = 0.8$.

with t. As show in Fig 5.3 and known from previous works [71], there remains negative eigenvalues at any finite time: the right edge of the spectrum only reaches 0 asymptotically as $t \to \infty$, since $z(t) \to 2$.

Hence, the dynamics never completely escape the saddles of the landscape at $N \to \infty$. This ruggedness of the landscape entails slow "glassy" dynamics, characterized by a power-law decay of the optimisation term for any $\beta < 1$, contrary to the exponential decay obtained in the convex setup (Sec. 5.1.2).

The p-spin model

We now turn to the analysis of the p-spin model which has been extensively studied in physics as a model of structural glasses, see e.g. [36]. To us, it is an ideal candidate as it corresponds to a random Gaussian landscape (with $p > 2$) for which the *Kac-Rice* approach rigorously shows the existence of a number of critical points growing exponentially with the dimension [16]. It is thus intrinsically harder, i.e. more strongly non-convex than the SK model above. The loss of the p-spin model (for $p > 2$) is written as:

$$\mathcal{L} = -\sqrt{\frac{(p-1)!}{N^{p-1}}} \sum_{i_1 < \ldots < i_p} J_{i_1 \ldots i_p} w_{i_1} \ldots w_{i_p}. \tag{5.10}$$

Solving the dynamics In the high-dimensional limit $N \to \infty$, the Langevin dynamics of the system can be reduced to a closed set of PDEs for a set of "macroscopic" quantities, which concentrate with respect to the randomness in the couplings J and the thermal noise in the dynamics ξ, as shown rigorously in [33]. These quantities are the two-point correlation $C(t, t')$ of the system at times t, t' and the response $R(t, t')$ of the system at time t to a perturbation in the loss function at an

earlier time t':

$$C(t,t') = \lim_{N\to\infty} \frac{1}{N} \mathop{\mathbb{E}}_{\xi,J} \sum_{i=1}^{N} w_i(t) w_i(t'), \tag{5.11}$$

$$R(t,t') = \lim_{N\to\infty} \frac{1}{N} \mathop{\mathbb{E}}_{\xi,J} \sum_{i=1}^{N} \frac{\delta w_i(t)}{\delta \xi_i(t')}. \tag{5.12}$$

Their dynamics is described by a closed set of integro-differential equations, dubbed the Crisanti-Horner-Sommers-Cugliandolo-Kurchan (CHSCK) equations [33, 70, 72]. We extend these equations to the non-constant learning rate case using the methods reviewed in [55]:

$$\frac{\partial R\,(t_1,t_2)}{\partial t_1} = F_R^p(z,R,C,\eta), \tag{5.13}$$

$$\frac{\partial C\,(t_1,t_2)}{\partial t_1} = F_C^p(z,R,C,\eta), \tag{5.14}$$

$$z(t) = T\eta(t) + p\int dt_2 \eta(t_2) R(t_2,t) C^{p-1}(t_2,t), \tag{5.15}$$

where we deferred the full expression of the update functions F_R^p and F_C^p as well as their derivation to App. E.4.

Imposing the the spherical constraint $C(t,t) = 1$ allows to find the value of the spherical constraint $z(t)$. To compute the loss, we follow the same procedure as in the SK model and obtain:

$$\ell(t) \equiv \frac{\mathcal{L}}{N} = -\frac{1}{p}\left(z(t) - T\eta(t)\right). \tag{5.16}$$

Optimal decay rate Here again we find that two competing terms contribute to the loss, the first related to optimisation and the second to noise. By choosing a learning rate $\eta(t) = \eta_0/t^\beta$, the later decays as $t^{-\beta}$. The decay of the former is more complex due to the high complexity of the landscape. It can be shown [72] that the system never reaches the ground state, instead remaining trapped in so-called threshold states where the Hessian has many zero eigenvalues (the density of eigenvalues is a Wigner semicircle whose left edge is zero as in the SK model). The loss is then given by:

$$\ell_{\text{th}} = -\frac{\sqrt{4(p-1)}}{p} > \ell_{GS}. \tag{5.17}$$

The relaxation towards the threshold states is characterised by a power-law due to the rough energy landscape, but with a different exponent this time: $z_{\text{th}} - z(t) \propto t^{-\gamma}$, with $\gamma = 2/3$ at $T = 0$ [302]. Using the CHSCK equations (5.13), we analytically show in App. E.3 that with decaying learning rate the exponent

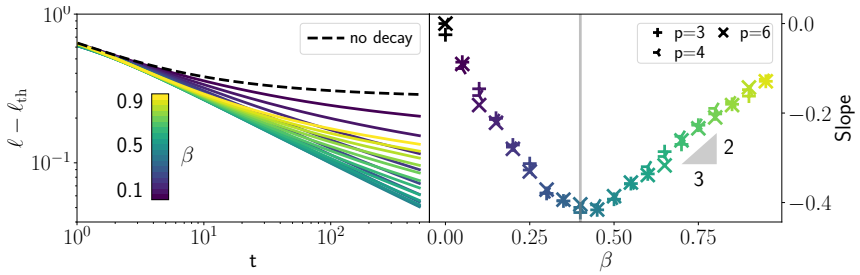


Figure 5.4: **In the p-spin model, the optimal decay rate is $\beta_{opt} = 0.4$.** *(Left)* Loss curves of the 3-spin model at $T = 1$ when decaying the learning rate as $\eta(t) = \eta_0/t^\beta$ for various values of β (colored lines). *(Right)* Decay exponent of $\ell - \ell_{\text{th}}$ at long times, for various p, as a function of β. We recognize a decay exponent of $\min(\beta, \gamma(1-\beta))$, as predicted by Eq. 5.18, which is fastest for $\beta_{\text{opt}} = 2/5$. *Parameters:* $dt = 10^{-2}$, $\eta_0 = 0.5$, $T = 1$.

becomes $\gamma(1-\beta)$. Hence, similarly to the SK model, the decay of the loss is controlled by a competition between two power-laws:

$$\ell(t) - \ell_{\text{th}} \sim t^{-\min(\beta, \gamma(1-\beta))}$$
$$\Rightarrow \beta_{\text{opt}} = \frac{\gamma}{1+\gamma} = \frac{2}{5}. \tag{5.18}$$

In Fig. 5.4, we numerically integrate Eqs. 5.13 for $p = 3, 4, 6$, confirming that the optimal decay rate to balance the noise and the optimisation terms is $\beta_{opt} = 2/5$. The numerical integration is non-trivial and we implement it using the tools developed in [207].

Relation with annealing in physics

The results found in this section can be put in a very general framework that was developed in physics of out of equilibrium systems. As shown in App. E.3, using a learning rate schedule is equivalent to annealing the temperature of the physical system as a power-law $t^{-\beta/1-\beta}$. Thus, finding the optimal learning rate schedule to minimize the loss is equivalent to determining the optimal annealing protocol to decrease the energy. A key ingredient in the solution is how fast the dynamics descend in the loss landscape in absence of noise. In physical systems, this optimisation term generally follows a power-law decay with exponent γ [40, 45, 47].

At finite temperature, the speed-noise tradeoff requires this decay rate to be equal to that of the temperature, $\beta/1-\beta$, leading to $\beta_{\text{opt}} = \gamma/1+\gamma$. The exponent γ has been determined in many statistical physics problems, corresponding to different high-dimensional non-convex landscapes, and typically ranges from zero

(logarithmic relaxation) to one. Our results extend to all these problems and, and predict optimal annealing exponents varying between 0 and $1/2$.

5.1.4 Recovering a signal: the two phases of learning

We now move to the setup where there is a signal $w^\star$ in the problem, which the algorithm aims to retrieve in the shortest time possible. In addition to the random Gaussian function, the loss now contains a deterministic term forming an attraction basin in the landscape, as sketched in the right panel of Fig. 5.1.

Spiked Sherrington-Kirkpatrick model

We first consider the so-called planted SK model, where the objective is to retrieve a ground truth $w^\star$ such that $\|w^\star\|^2 = N$, i.e. maximize the overlap with the signal $m = \sum_i w_i \cdot w_i^\star / N$. We enforce as before the spherical constraint $\|w\|^2 = N$ which induces $m \in [-1, 1]$, and sample randomly the initial configuration of w, such that the initial overlap is of order $1/\sqrt{N}$. The loss function takes the form:

$$\mathcal{L}(w) = -\frac{N}{2}m^2 - \frac{\Delta}{\sqrt{N}}\sum_{i<j}^{N} J_{ij} w_i w_j = \frac{1}{2} w H w^\top, \tag{5.19}$$

with $H = -\frac{\Delta}{\sqrt{N}}J - \frac{1}{N}w^\star w^{\star\top}$.

Decreasing Δ makes the signal easier to detect, leading to an easier problem. For $\Delta < 1/2$, an eigenvalue of H pops out of the semicircle law (5.9) as a BBP transition takes place [21], leading to the follow spectrum:

$$\rho(\mu) = \left(1 - \frac{1}{N}\right)\rho_{sc}(\mu/\Delta) + \frac{1}{N}\rho(\mu - 1) \tag{5.20}$$

This is the regime in which the signal overcomes the noise, i.e. the global minimum of the loss has a finite overlap with the signal, which can then be retrieved by gradient flow (or gradient descent).

In the following, we assume that $\Delta < 1/2$, and define the gap between the largest and second largest eigenvalue as $\kappa \equiv 1 - 2\Delta$. In App. E.2, we analytically show the emergence of a crossover time,

$$t_{\text{cross}} = \left(\frac{\log N}{2\eta_0 \kappa}\right)^{\frac{1}{1-\beta}}. \tag{5.21}$$

Before t_{cross}, the system behaves as if the signal was absent, i.e. as in Sec. 5.1.3: this is the *search* phase. After t_{cross}, the signal is detected: this is the *convergence* phase.

The loss becomes:

$$\ell(t) - \ell_{GS} = \frac{\eta_0 T}{2t^\beta} + \begin{cases} \mathcal{O}(e^{-2\eta_0 \kappa t^{1-\beta}}), & \beta < 1 \\ \mathcal{O}(t^{-2\eta_0 \kappa}), & \beta = 1 \end{cases} \tag{5.22}$$

with $\ell_{GS} = -1$. We recognize here the exact same result as obtained in the convex setup of Eq. 5.3: as long as $\eta_0 > 1/2\kappa$, the optimal learning rate schedule is $\eta = \eta_0/t$. This indicates that the dynamics has entered a convex basin of curvature κ.

Optimal learning rate schedule To speed up the initial phase where the signal hasn't yet aligned with the signal, one needs to reduce t_{cross}, which is achieved by using a large learning rate η_0 without any decay ($\beta = 0$). Passed this crossover, the system enters a convex basin, and the optimal exponent becomes $\beta = 1$. Ergo, the best schedule is to keep the learning rate constant up to t_{cross}, then to decay it with $\beta = 1$, in contrast with the case without signal where $\beta = 1/2$ was optimal, see Sec. 5.1.3. This is confirmed by the numerical experiments of Fig. 5.5, where we start decaying the learning rate as $\eta_0/(t-t_s)^\beta$ for different "switch" times t_s. Decaying too early, with $t_s < t_{\text{cross}}$, slows down the dynamics, whereas $t_s > t_{\text{cross}}$ enables the system to reach the ground state at a rate $t^{-\beta}$.

Finite-dimensional effects The two phases in the dynamics are a general feature when there is a finite gap κ between the largest and second largest eigenvalue of H. In the $N \to \infty$ limit, this only occurs when $\Delta < 1/2$. However, when $\Delta > 1/2$, there is a finite gap at finite N due to the discrete nature of the spectrum, which scales as $\kappa \sim N^{-2/3}$ [305]. This induces a crossover time $t_{\text{cross}} \sim N^{2/3}$. Hence, decaying the learning rate as β_{opt} remains optimal for any finite time budget $t < t_{\text{cross}}$, but for a large budget $t > t_{\text{cross}}$, using the two-step schedule described in this section becomes optimal.

Spiked Matrix-Tensor model

We finally move to the analysis of the SMT model for which the loss function is [274]:

$$\begin{aligned} \mathcal{L}(w) = &- \frac{N}{2\Delta_2} m^2 - \sqrt{\frac{1}{\Delta_2 N}} \sum_{i<j} J_{i,j} w_i w_j \\ &- \frac{N}{p\Delta_p} m^p - \sqrt{\frac{(p-1)!}{\Delta_p N^{p-1}}} \sum_{i_1 < \ldots < i_p} J_{i_1,\ldots,i_p} w_{i_1} .. w_{i_p}, \end{aligned} \tag{5.23}$$

where both J_{ij} and $J_{i_1,..,i_p}$ sampled i.i.d. from $\mathcal{N}(0,1)$. As understood from the loss function, the signal is observed through its contraction with a matrix and a tensor

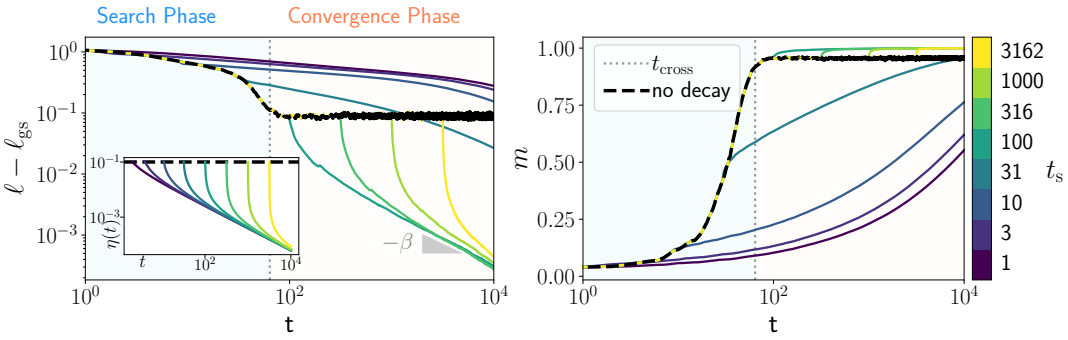


Figure 5.5: **Emergence of a crossover time in the planted SK model.** The dashed black show the loss **(left)** and overlap with the signal **(right)** at constant learning rate $\eta_0 = 0.1$. The colored lines, show the result of keeping the learning rate constant until t_s and then decaying as $\eta(t) = \eta_0/(t - t_s)^\beta$ (as shown in the inset). The dashed vertical line marks the theoretical crossover time $t_{\text{cross}} = \frac{\log N}{2\eta_0\kappa}$, it matches with the time at which the loss, at constant learning rate, saturates. Before the crossover, decaying the learning rate is detrimental. After the crossover, it allows the model to converge to zero loss as $t^{-\beta}$ and to perfectly recover the signal. We set $N = 3000, T = 1., \eta_0 = 0.1, \beta = 0.8, \kappa = 0.5$.

of order p. This model is a natural next step for our analysis: its loss landscape is extremely non-convex, but its dynamics are exactly solvable in the $N \to \infty$ limit. They can be described by a closed set of PDEs describing the dynamical evolution of the quantities $m(t)$, $C(t, t')$, $R(t, t')$ and $z(t)$ described in Sec. 5.1.3. The derivation of these equations is deferred to the appendix E.4.

The difficulty of the problem is controlled by the values of Δ_2 and Δ_p. Here, we focus on the *Langevin easy* phase, defined in [274], where a randomly initialized system recovers the signal and the overlap converges to a value of order one.* The dynamics in this setting have been well understood at constant learning rate in [207], and are shown as a black line in Fig. 5.6 for $\eta_0 = 1$: the system remains trapped in the exponentially many *threshold states* until a time t_{cross}. At t_{cross}, the system finally detects the signal and the overlap jumps to a value m_{gs} of order one. This behavior is reminiscent of the *grokking* phenomenon observed for neural networks [251].

The colored lines of Fig. 5.6 show that decaying the learning rate from a time t_s affects optimisation in two different ways. (i) If we choose $t_s < t_{\text{cross}}$, the loss actually starts by dropping, in contrast with what was observed in Fig. 5.5. However, this drop in the loss does not yield an increase of the overlap with the signal, and the system rapidly gets stuck, remaining in a state of low overlap even

*We must start from a very small initial overlap $m_0 = 10^{-10}$ as explained in [207], since $m_0 = 0$ would cause the system to remain stuck in the $N \to \infty$ limit considered here [14].

after t_{cross}. (ii) If we choose $t_s > t_{\text{cross}}$, once the signal is detected, the noise is suppressed, allowing the system to converge to the ground state and the overlap to increase. Hence, the optimal schedule is again to keep a constant large learning rate during the search phase (i.e. until t_{cross}) then decay with $\beta = 1$. We provide further theoretical justification for this behavior in App. E.4.

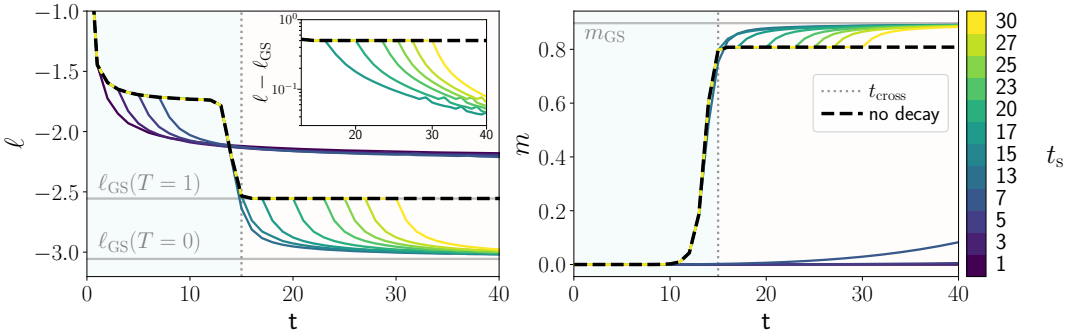


Figure 5.6: **Emergence of a crossover time in the SMT model.** By fixing β from start, or anytime before t_{cross}, a randomly initialised system will remain stuck at *threshold* states at high loss until t_{cross} which is minimal for constant learning rate $\beta = 0$. In contrast, by decaying the learning rate at long times allows to reach lower loss solutions (*left*) with higher overlap with the signal (*right*). The optimal schedule is to keep η constant until t_{cross} and then set $\beta = 1$. By doing so, we get the best of both worlds: the first phase minimises t_{cross} while the second allows to reach more informative solutions. *Parameter:* $\beta = 0.8$, $\Delta_2 = 0.2$, $\Delta_p = 6$, $\eta_0 = 1$, $T = 1$, $dt = 10^{-2}$, $m_0 = 10^{-10}$.

5.1.5 Turning to SGD : teacher-student regression

Our work has demonstrated the emergence of a crossover time in a class of inference problems, before which one should keep the learning rate constant and after which it becomes useful to decay the learning rate.

We now investigate these findings in a setup that is more realistic but simple enough to be amenable to analytical treatment in the near future. We consider a *teacher-student* regression problem in which a student network is trained to mimick the ouputs of a teacher by minimising the mean-squared error (mse) over a dataset of N input-outputs observations $\{x_\mu, y^*_\mu\} \in \{\mathrm{R}^D, \mathrm{R}\}$. Here both the student S and the teacher T are two-layer networks:

$$S(x) = \sum_{k=1}^{K} v^k g\left(\frac{w^k \cdot x}{\sqrt{D}}\right) \qquad T(x) = \sum_{m=1}^{M} \tilde{v}^m \tilde{g}\left(\frac{\tilde{w}^m \cdot x}{\sqrt{D}}\right).$$

We train on i.i.d. gaussian inputs $x_i \sim \mathcal{N}(0,1)$ via SGD, by minimising the mse

over mini-batches of size B:

$$\text{mse} = \frac{1}{B}\sum_{\mu=1}^{B}\left(S(x_\mu) - T(x_\mu)\right)^2, \tag{5.24}$$

The optimisation noise is controlled by the batch size B and is absent for full batch SGD. To study the effect of learning rate scheduling, we focus on a mini-batch of size 1 for which optimisation noise is high.

Fig. 5.7 shows the mse (calculated over the whole training set) of a student with $K = 2$ hidden units learning from a teacher with $M = 2$ hidden units (results with different sizes are presented in App. E.5). As before, we keep the learning rate constant η_0 until a time t_s then decay it as $\eta_0/(t-t_s)^{-\beta}$. The phenomenology is remarkably similar to that of Sec. 5.1.4: there exists a cross-over time t_{cross} such that if the learning rate is decayed before t_{cross}, optimisation remains stuck at high mse. In contrast, decaying the learning rate after after t_{cross} enables to tame the noise associated with optimisation and converge to lower loss solutions.

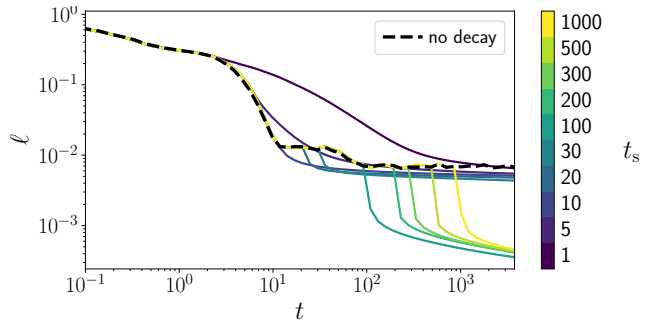


Figure 5.7: **The crossover time is also reflected in a regression task with SGD.** A K hidden nodes 2 layer neural network student is trained to reproduce the output of her M hidden nodes teacher on gaussian inputs in D dimensions. As before, we find that decaying the learning rate before the loss plateaus performance, but decaying as $\eta(t)\sim t^{-1}$ once the plateau is reached allows to reach zero loss. *Parameters:* $D = 500$, $N = 10^4$, $\eta_0 = 10^{-1}$, $M = K = 2$, $\beta = 0.8$.

5.1.6 Recap

In this section, we have analysed learning scheduling in a variety of high-dimensional non-convex optimisation problems. First, we focused on purely non-convex problems (without any basins of attraction), and showed that the optimal learning rate decay in the high-dimensional limit has an exponent smaller than one, which varies according to the degree of non-convexity of the problem at hand (ranging from 0.4 to 0.5 in the problems considered here). Then, we studied models where a

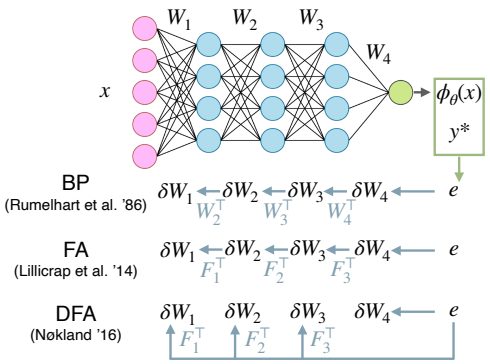


Figure 5.8: **Three approaches to the credit assignment problem in deep neural networks.** In *back-propagation* (BP), the weight updates $\mathrm{d}w^h$ are computed sequentially by transmitting the error Δ from layer to layer using the transpose of the network's weights $w^{h\top}$. In *feedback alignment*(FA) [190], $w^{h\top}$ are replaced by fixed random feedback matrices F_h. In *direct feedback alignment* (DFA) [237], the error is directly injected to each layer using random feedback matrices F_h, enabling parallelized training.

signal must be recovered in presence of noise. In this case, what is important is not *how fast* we decay the learning rate, but *when* we start decaying it. It is better to keep a large learning rate in the *search* phase to find the convex basin as quickly as possible, and only then start decaying the learning rate.

These theoretical findings are remarkably reminiscent of learning rate schedules used in practice. Establishing a tighter connection is an important direction for future work: could the $1/\sqrt{t}$ decay commonly used to train transformers reflect the properties of the landscape the dynamics take place in? Conversely, could one predict the optimal decay rate by inspecting the properties of the landscape? Establishing such connections in simple settings such as that of Sec. 5.1.5 is certainly within reach thanks to the recent analytical tools developed in [7, 56, 116, 221, 259].

5.2 Alternative training algorithm to go beyond BP

5.2.1 Overview

Up to now, we were only concerned with *back-propagation* (BP) [267], the algorithm most widely used to train deep neural networks. Despite its wide-use and its jaw-dropping achievements of recent years, BP is not problems free. Notably, Sec. 5.1 demonstrates that the specific choice of hyper-parameters can have drastic effect on learning and requires careful and tedious tuning.

BP also suffers from more fundamental shortcomings. As described in Sec. 1.4,

it solves the the credit assignment problem of how weights deep in the network should be updated, given only the output of the network and the target label of the input, by using the transpose of the network's weight matrices to transmit the error signal across the network from one layer to the next, see Fig. 5.8. Thus, it enforces a symmetry between the weights in the forward and the backward pass, which makes it a biologically implausible algorithm [69, 123]. It also requires the sequential update of weights in the backward pass preventing efficient parallelisation.

To overcome these shortcomings, several algorithms which only approximate the gradient are recently attracting increasing attention. These include *feedback alignment* of Lillicrap et al. [190], presented in in Eq. 1.20, and *dirrect feedback allignment* of Nøkland [237], in Eq. 1.21. As a reminder, consider a for a fully-connected neural network of depth H, activation function g and a single output unit. On on an input $x \in \mathbb{R}^D$, we denote the pre-activations and the post-activations at the hth layer of width K_h as in Eq. 1.3:

$$x^0 \equiv x \qquad g_H(a^H) \equiv y = \phi_\theta(x) \tag{5.25}$$

$$x^h = g(a^h + b^h) \qquad a^h \equiv \sum_{k=1}^{K_h} w_k^h x_k^{h-1} \in \mathbb{R}^{K_h}, \tag{5.26}$$

where we defined the output activation g_H and set the bias to 0 for simplicity. Then, the three algorithms update the weights of the network as:

$$\begin{aligned} \mathrm{d}w_{ij}^h &\equiv -\eta \delta_j^h x_i^{h-1} \\ \text{with } \delta_j^h &= \begin{cases} \delta_{(BP)j}^h = g'(a_j^h) \sum_k^{K_{h+1}} \delta_k^{h+1} w_{jk}^{h+1} & \text{for BP} \\ \delta_{(FA)j}^h = g'(a_j^h) \sum_k \delta_k^{h+1} F_{jk}^h & \text{for FA} \\ \delta_{(DFA)j}^h = g'(a_j^h) \Delta F_j^h & \text{for DFA} \end{cases} \end{aligned} \tag{5.27}$$

where, given a loss $\mathcal{L}_{\text{emp}}$, we defined as before the *error* $\Delta = \frac{\partial \mathcal{L}_{\text{emp}}}{\partial y}$. The update of the last layer of weights w^H is unchanged in FA and DFA:

$$\mathrm{d}w_{ij}^H = -\eta \Delta g_H^{'}(a_i^H) x_j^{H-1}. \tag{5.28}$$

By replacing the weights w^{h+1} by a random *feedback* matrix F in the updates, FA dispenses with the need of biologically unrealistic symmetric forward and backward weights [69, 123]. DFA propagates the error directly from the output layer to each hidden layer of the network through random feedback connections F^h and thus allows to update different layers in parallel. Fig. 5.8 shows the information flow of all three algorithms.

Despite doubts on whether DFA could scale up to solve practical problems [29, 112], Launay et al. [167] demonstrated on a number of state-of-the-art architectures and benchmark datasets that networks trained with DFA reached performances comparable to fine-tuned BP. Yet, both FA and DFA notoriously fail to train convolutional networks [29, 128, 166, 227], a common architecture for solving vision tasks. Understanding these varied results is the primary motivation for this section which develops theoretical insights into how ad when feedback alignment works.

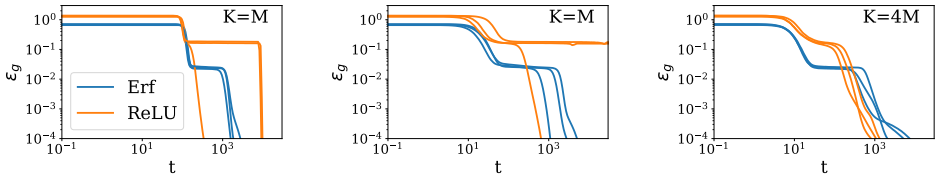
In more details, we start with an analytical description of DFA dynamics in the setting of Sec. 4.1 of shallow non-linear networks. We show that in this setup, DFA proceeds in two steps: an *alignment phase*, where the forward weights adapt to the feedback weights to improve the approximation of the gradient, is followed by a *memorisation phase*, where the network sacrifices some alignment to minimise the loss. Out of the same-loss-solutions in the landscape, DFA converges to the one that maximises gradient alignment, an effect we term "degeneracy breaking". We then focus on the alignment phase in the setup of deep linear networks, and uncover a key quantity underlying gradient alignment (GA): the conditioning of the alignment matrices. Our framework allows us to analyse the impact of data structure on DFA, and suggests an explanation for the failure of DFA to train convolutional layers. Throughout the section, we validate our theoretical results with experiments that demonstrate the occurence of (i) the Align-then-Memorise phases of learning, (ii) degeneracy breaking and (iii) layer-wise alignment in deep neural networks trained on standard vision datasets.

Related Work Lillicrap et al. [190] gave a first theoretical characterisation of feedback alignment by arguing that for two-layer linear networks, FA works because the transpose of the second layer of weights w_2 tends to align with the random feedback matrix F_1 during training. This *weight alignment* (WA) leads the weight updates of FA to align with those of BP, leading to *gradient alignment* (GA) and thus to successful learning. Frenkel et al. [95] extended this analysis to the deep linear case for a variant of DFA called "Direct Random Target Projection" (DRTP) under the restrictive assumption of training on a single data point. Nøkland [237] also introduced a layerwise alignment criterion to describe DFA in the deep nonlinear setup, under the assumption of constant update directions for each data point.

Reproducibility We host all the code to reproduce our experiments online at `https://github.com/sdascoli/dfa-dynamics`.

5.2.2 A two-phase learning process

We begin with an exact description of DFA dynamics in the setting of shallow non-linear networks described in Sec. 4.1. The inputs $x \in \mathbb{R}^D$ are sampled i.i.d. from



(a) BP, matched (b) DFA, matched (c) DFA, overparametrized

Figure 5.9: **Learning dynamics of back-propagation and feedback alignment for sigmoidal and ReLU neural networks learning a target function.** Each plot shows three runs from different initial conditions for every setting, where a shallow neural network with K hidden nodes tries to learn a teacher network with M hidden nodes. *(a)* All networks trained using BP in the matched case $K = M$ achieve perfect test error. *(b)* Sigmoidal networks achieve perfect test error with DFA, but the algorithm fails in some instances to train ReLU networks ($K = M$) *(c)* In the over-parametrised case ($K > M$), both sigmoidal and ReLU networks achieve perfect generalisation when trained with DFA. *Parameters:* $D = 500, H = 2, M = 2, \eta = 0.1, \sigma_0 = 10^{-2}$.

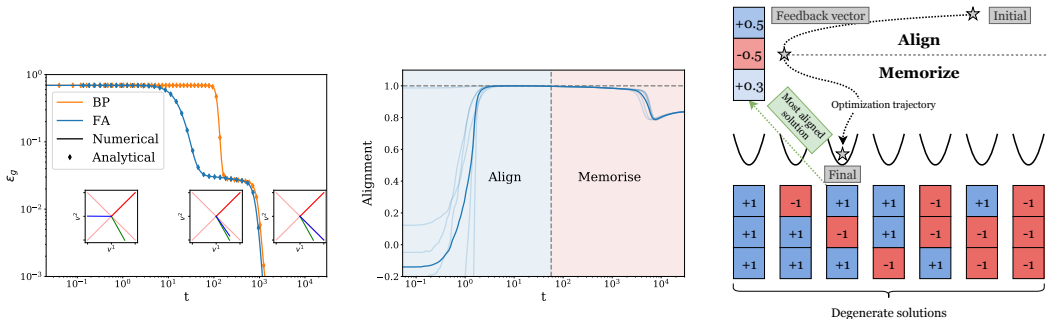
the standard normal distribution and we take the high dimensional limit $D \to \infty$. As before, we work in the teacher-student setup where the labels y^* are given by a teacher networks with random weights [93, 98, 285, 311, 327] and both teacher and student are two-layer networks with $K, M \sim O(1)$ hidden nodes. We consider sigmoidal, $g(x) = \text{erf}\,(x/\sqrt{2})$, and ReLU activation functions, $g(x) = \max(0, x)$. We asses the student's performance on the task through its pmse or test error:

$$\text{pmse}(x) \equiv \frac{1}{2}\,\mathop{\mathbb{E}}_{x}\,(\phi(x) - y^*)^2 \equiv \frac{1}{2}\,\mathop{\mathbb{E}}_{x}\,\Delta^2, \tag{5.29}$$

The expectation $\mathbb{E}$ is taken over the inputs for a given teacher and student networks. In this shallow setup, FA and DFA are equivalent, and only involve one feedback matrix, $F_1 \in \mathbb{R}^K$ which back-propagates the error signal Δ to the first layer weights w_1. The updates of the second layer of weights w_2 are the same as for BP.

Performance of BP vs. DFA We show the evolution of the test error (5.29) of sigmoidal and ReLU students trained via vanilla BP in the "matched" case $K = M$ in Fig. 5.9 a, for three random choices of the initial weights with standard deviation $\sigma_0 = 10^{-2}$. In all cases, learning proceeds in three phases: an initial exponential decay; a phase where the error stays constant, the "plateau" [93, 270, 321]; and finally another exponential decay towards zero test error.

Sigmoidal students trained by DFA always achieve perfect generalisation when started from different initial weights with a different feedback vector each time (blue in Fig. 5.9 b) raising a first question: if the student has to align its second-layer weights with the random feedback vector in order to retrieve the BP gradient [190], i.e. $w_2 \propto F_1$, how can it recover the teacher weights perfectly, i.e. $w_2 = \tilde{w}_2$?



(a) Generalization dynamics (b) Alignment dynamics (c) Degeneracy breaking

Figure 5.10: *(a)* **Theory gives exact prediction for the learning dynamics.** We plot learning curves for BP and DFA obtained from (i) a single simulation (solid lines), (ii) integration of the ODEs for BP dynamics [38, 270] (orange dots), (iii) integration of the ODEs for DFA derived here (blue dots). *Insets*: Teacher second-layer weights (red) as well as the degenerate solutions (light red) together with the feedback vector F_1 (green) and the student second-layer weights v (blue) at three different times during training with DFA. *Parameters:* $D = 500, K = M = 2, \eta = 0.1, \sigma_0 = 10^{-2}$.
(b) **Align-then-Memorise process.** Alignment (cosine similarity) between the student's second layer weights and the feedback vector. In the align phase, the alignment increases, and reaches its maximal value when the test loss reaches the plateau. Then it decreases in the memorization phase, as the student recovers the teacher weights.
(c) **The degeneracy breaking mechanism.** There are multiple degenerate global minima in the optimisation landscape: they are related through a discrete symmetry transformation of the weights that leaves the student's output unchanged. DFA chooses the solution which maximises the alignment with the feedback vector.

For ReLU networks, over-parametrisation is key to the consistent success of DFA: while some students with $K = M$ fail to reach zero test error (orange in Fig. 5.9 b), almost every ReLU student having more parameters than her teacher learns perfectly ($K = 4M$ in Fig. 5.9 c). A second question follows: how does over-parameterisation help ReLU students achieve zero test error?

An analytical theory for DFA dynamics To answer these two questions, we study the dynamics of DFA in the limit of infinite training data as in Sec. 4.1 where the DFA weight updates, Eq. 5.27, are computed at every step on a previously unseen sample (x, y^*). For BP, the theoretical analysis is the one described in Sec. 4.1. For DFA, the rationale is the same and the only change in the equations is the replacements of the student's second layer weight by the feedback vector in the update of the first layer Eq. 4.5 i.e.:

$$w^k_{\mu+1} - w^k_\mu \equiv \mathrm{d}w^k = \frac{\eta_w}{\sqrt{D}} f^k g'\left(a^k_\mu\right) \Delta_\mu x_\mu. \tag{5.30}$$

with $f^k \equiv v_\mu^k$ for BP and to the kth element if the feedback vector F_1^K for DFA. The first contribution of Refinetti et al. [259] is the extension of the ODEs to training with DFA, thus predicting the test error of a student at all times (see SM F.1 for the details). The accuracy of the predictions from the ODEs is demonstrated in Fig. 5.10 a, where the comparison between a single simulation of training a two-layer net with BP (orange) and DFA (blue) and theoretical predictions yield perfect agreement.

Sigmoidal networks learn through "degeneracy breaking"

The test loss of a sigmoidal student trained on a teacher with the same number of neurones as herself ($K = M$) contains several global minima, which all correspond to fixed points of the ODEs (F.7). Among these is a student with exactly the same weights as her teacher. The symmetry $\mathrm{erf}(z) = -\mathrm{erf}(-z)$ induces a student with weights $\{\tilde{w}_1, \tilde{w}_2\}$ to have the same test error as a sigmoidal student with weights $\{-\tilde{w}_1, -\tilde{w}_2\}$. Thus, as illustrated in Fig. 5.10 c, the problem of learning a teacher has various degenerate solutions. A student trained with vanilla BP converges to any one of these solutions, depending on the initial conditions.

Alignment phase A student trained using DFA has to fulfil the same objective (zero test error), with an additional constraint: her second-layer weights w_2 need to align with the feedback vector F_1 to ensure the first-layer weights are updated in the direction that minimises the test error.

And indeed, an analysis of the ODEs (cf. Sec. F.2) reveals that in the early phase of training, $\dot{w}_2 \sim F$ and so w_2 grows in the direction of the feedback vector F_1 resulting in an increasing overlap between w_2 and F_1. In this *alignment phase* of learning, shown in Fig. 5.10 b, w_2 becomes perfectly aligned with F_1. DFA has perfectly recovered the weight updates for w_1 of BP, but the second layer has lost its expressivity (it is simply aligned to the random feedback vector).

Memorisation phase The expressivity of the student is restored in the *memorisation* phase of learning, where the second layer weights move away from F_1 and towards the global minimum of the test error that maintains the highest overlap with the feedback vector. In other words, students solve this constrained optimisation problem by consistently converging to the global minimum of the test loss that simultaneously maximises the overlap between w_2 and F_1, and thus between the DFA gradient and the BP gradient. For DFA, the global minima of the test loss are not equivalent, this "degeneracy breaking' is illustrated in Fig. 5.10 c.

Degeneracy breaking requires over-parametrisation for ReLU networks

The ReLU activation function possesses the continuous symmetry $\max(0, x) =$

$\gamma \max(0, x/\gamma)$ for any $\gamma > 0$ preventing ReLU networks to compensate a change of sign of w_2^k with a change of sign of w_1^k. Consequently, a ReLU student can only simultaneously align to the feedback vector F_1 and recover the teacher's second layer $\tilde{w}_2$ if at least M elements of F_1 have the same sign as $\tilde{w}_2$. The inset of Fig. 5.11 shows that a student trained on a teacher with $M = 2$ second-layer weights $\tilde{w}_2^m = 1$ only converges to zero test error if the feedback vector has 2 positive elements (green). If instead the feedback vector has only 0 (blue) or 1 (orange) positive entry, the student will settle at a finite test error. More generally, the probability of perfect recovery for a student with $K \geq M$ nodes sampled randomly is given analytically as:

$$P(\text{learn}) = \frac{1}{2^K} \sum_{k=0}^{M} \binom{K}{k}. \tag{5.31}$$

As shown in Fig. 5.11, this formula matches with simulations. Note that the importance of the "correct" sign for the feedback matrices was also observed in deep neural networks by Liao et al. [187].

Degeneracy breaking in deep networks

We explore to what extent degeneracy breaking occurs in deep nonlinear networks by training 4-layer multi-layer perceptrons (MLPs) with 100 nodes per layer for 1000 epochs with both BP and DFA, on the MNIST and CIFAR10 datasets, with Tanh and ReLU nonlinearities (cf. App. F.5 for further experimental details). The dynamics of the training loss, shown in the left of Fig. 5.12, are very similar for BP and DFA.

From degeneracy breaking, one expects DFA to drive the optimisation path towards a special region of the loss landscape determined by the feedback matrices. We test this hypothesis by measuring whether networks trained with the same feedback matrices from different initial weights converge towards the same region of the landscape. The cosine similarity between the vectors obtained by stacking the weights of two networks trained independently using BP reaches at most 10^{-2} (right of Fig. 5.12), sig-

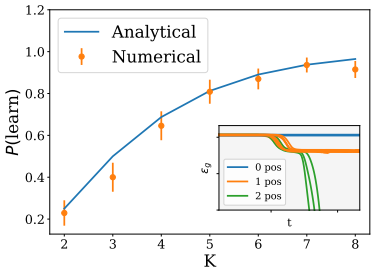


Figure 5.11: Over-parameterisation improves performance of shallow ReLU networks. We show the learning dynamics of a student with $K = 3$ hidden nodes trained on a teacher with $M = 2$ nodes and $\tilde{w}_2^m = 1$ if the feedback vector has 0, 1, or 2 positive entries. *Inset*: Probability of achieving zero test error (Eq. 5.31, line) compared to the fraction of simulations that converged to zero test error (out of 50, crosses). *Other parameters*: $D = 500, \eta = 0.1, \sigma_0 = 10^{-2}$.

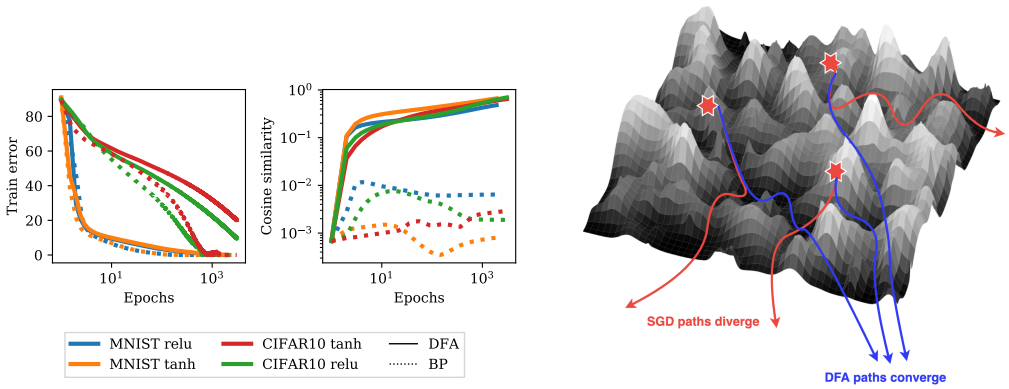


Figure 5.12: **Degeneracy breaking also occurs in deep neural networks.** (*Left*) We plot the training accuracy and the cosine similarity between the weights of four-layer fully-connected neural networks with sigmoidal and ReLU activations during training on MNIST and CIFAR10. Averages taken over 10 runs; for exp. details see Sec. 5.2.2. (*Right*) Cartoon of the degeneracy breaking process in the loss landscape of a deep network: while the optimisation paths of models trained with SGD diverge in the loss landscape, with DFA they converge to a region of the landscape determined by the feedback matrices.

nalling that they reach very distinct minima. In contrast, when trained with DFA, networks reach a cosine similarity between 0.5 and 1 at convergence, thereby confirming that DFA breaks the degeneracy between the solutions in the landscape and biases towards a special region of the loss landscape, both for sigmoidal and ReLU activation functions.

This result suggests that heavily over-parametrised neural networks used in practice can be trained successfully with DFA because they have a large number of degenerate solutions. We leave a more detailed exploration of the interplay between DFA and the loss landscape for future work. As we discuss in Sec. 5.2.4 the Align-then-Memorise mechanism sketched in Fig. 5.10 c also occurs in deep non-linear networks.

5.2.3 How do gradients align in deep networks?

This section focuses on the alignment phase of learning. In the two-layer setup there is a single feedback vector F_1, of same dimensions as the second layer w_2, and to which w_2 must align in order for the first layer to recover the true gradient.

In deep networks, as each layer w^h has a distinct feedback matrix F_h of different size of w^h, it is not obvious how the weights must align to ensure gradient alignment. We study how the alignment occurs by considering deep linear networks with H layers without bias, without any assumption on the training data. While the expressivity of linear networks is naturally limited, their learning dynamics is

non-linear and rich enough to give insights that carry over to the non-linear case both for BP [5, 24, 162, 169, 275] and for DFA [95, 190, 237].

Weight alignment as a natural structure

In the following, we assume that the weights are initialised to zero. With BP, they would stay zero at all times, but for DFA the layers become nonzero sequentially, from the bottom to the top layer. In the linear setup, the updates of the first two layers at time t can be written in terms of the corresponding input and error vectors using Eq. (5.27)*:

$$\mathrm{d}w_1^t = -\eta(F_1\Delta_t)x_t^T, \quad \mathrm{d}w_2^t = -\eta(F_2\Delta_t)(w_1x_t)^\top \tag{5.32}$$

Summing these updates shows that the first layer performs Hebbian learning modulated by the feedback matrix F_1:

$$w_1^t = -\eta\sum_{t'=0}^{t-1} F_1\Delta_{t'}x_{t'}^\top = F_1A_1^t, \tag{5.33}$$

$$A_1^t = -\eta\sum_{t'=0}^{t-1} \Delta_{t'}x_{t'}^\top \tag{5.34}$$

Plugging this result into $\mathrm{d}w_2^t$, we obtain:

$$w_2^t = -\eta\sum_{t'=0}^{t-1} F_2\Delta_t(A_1^{t'}x_{t'})^\top F_1^\top = F_2A_2^tF_1^\top, \tag{5.35}$$

$$A_2^t = \eta^2\sum_{t'=0}^{t-1}\sum_{t''=0}^{t'-1}(x_{t'}\cdot x_{t''})\Delta_{t'}e_{t''}^\top. \tag{5.36}$$

When iterated, the procedure above reveals that DFA naturally leads to *weak weight alignment* of the network weights to the feedback matrices:

$$\textbf{Weak WA: } w_{1<l<H}^t = F_hA_h^tF_{h-1}^\top, \quad w_H^t = A_H^tF_{H-1}^\top, \tag{5.37}$$

where we defined the *alignment matrices* $A_{h\geq 2}^t \in \mathbb{R}^{K_H\times K_H}$:

$$A_{h\geq 2}^t = \eta^2\sum_{t'=0}^{t-1}\sum_{t''=0}^{t'-1}(B_h^{t'}x_{t'})\cdot(B_h^{t''}x_{t''})\Delta_{t'}\Delta_{t''}^\top. \tag{5.38}$$

$B_h \in \mathbb{R}^{K_h\times K_h}$ is defined recursively as a function of the feedback matrices only and its expression together with the full derivation is deferred to App. F.3. These results can be adapted both to DRTP [95], another variant of feedback alignment where

*We implicitly assume a minibatch size of 1 for notational simplicity, but conclusions are unchanged in the finite-batch setup.

$\Delta_t = -y_t$ and to FA by performing the replacement $F_h \to F_h F_{h+1} \dots F_{H-1}$.

Weight alignment leads to gradient alignment

Weak WA builds throughout training, but does not directly imply GA. However, if the alignment matrices become proportional to the identity, we obtain *strong weight alignment*:

$$\textbf{Strong WA:} \quad w^t_{1<h<H} \propto F_h F^\top_{h-1}, \quad w^t_h \propto F^\top_{H-1}. \tag{5.39}$$

Additionally, since GA requires $F_h e \propto w^\top_{h+1} \delta a_{h+1}$ (Eqs. 5.27 and 5.27), strong WA directly implies GA if the feedback matrices $F_{h\geq 2}$ are assumed left-orthogonal, i.e. $F_h^\top F_H = \mathbb{I}_{K_H}$. Strong WA of (5.39) induces the weights, by the orthogonality condition, to cancel out by pairs of two:

$$w^\top_{h+1} \delta a_{h+1} \propto F_h F^\top_{h+1} F_{h+1} \dots F^\top_{H-1} F_{H-1} e = F_h e. \tag{5.40}$$

The above suggests that taking the feedback matrices left-orthogonal is favourable for GA. If the feedback matrices elements are sampled i.i.d. from a Gaussian distribution, GA still holds in expectation since $\mathbb{E}\left[F_h^\top F_h\right] \propto \mathbb{I}_{K_H}$.

Quantifying gradient alignment Our analysis shows that key to GA are the alignment matrices: the closer they are to identity, i.e. the better their conditioning, the stronger the GA. This comes at the price of restricted expressivity, since layers are encouraged to align to a product of (random) feedback matrices. In the extreme case of strong WA, the freedom of layers $h \geq 2$ is entirely sacrificed to allow learning in the first layer! This is not harmful for the linear networks as the first layer alone is enough to maintain full expressivity*. Nonlinear networks, as argued in Sec. 5.2.2, rely on the Degeneracy Breaking mechanism to recover expressivity.

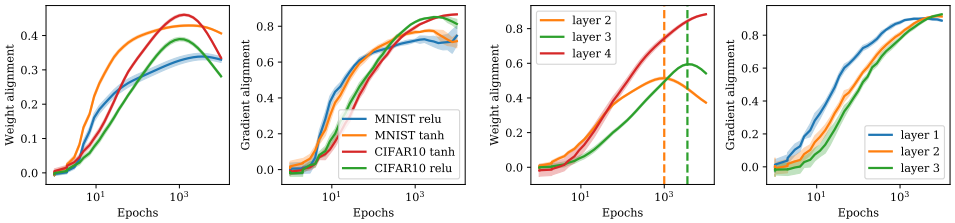
5.2.4 The case of deep nonlinear networks

In this section, we show that the theoretical predictions of the previous two sections hold remarkably well in deep nonlinear networks trained on standard vision datasets.

Weight Alignment occurs like in the linear setup

To determine whether WA described in Sec. 5.2.3 holds in the deep nonlinear setup

*such an alignment was indeed already observed in the linear setup for BP [149].



(a) **Global alignment dynamics of deep nonlinear networks exhibits Align-then-Memorise.** Global weight and gradient alignments, as defined in (5.41), varying the activation function and the dataset. Shaded regions represent the (small) variability over 10 runs.

(b) **Layerwise alignment dynamics reveal sequential Align-then-Memorise.** Layerwise weight and gradient alignments as defined in (5.42), for a ReLU network trained on CIFAR10 with 10% label corruption. Shaded regions represent the (small) variability over 10 runs.

of Sec. 5.2.2, we introduce the global and layerwise alignment observables:

$$\text{WA} = \measuredangle\left(\mathbf{F}, \mathbf{W}\right), \qquad \text{GA} = \measuredangle\left(\mathbf{G}^{\text{DFA}}, \mathbf{G}^{\text{BP}}\right) \tag{5.41}$$

$$\text{WA}_{h\geq 2} = \measuredangle(\mathbf{F}_h \mathbf{W}_h), \qquad \text{GA}_{h\geq 2} = \measuredangle\left(\mathbf{G}_h^{\text{DFA}}, \mathbf{G}_h^{\text{BP}}\right), \tag{5.42}$$

where $\measuredangle(\mathbf{A}, \mathbf{B}) = \text{Vec}(\mathbf{A}) \cdot \text{Vec}(\mathbf{B}) / \|\mathbf{A}\| \|\mathbf{B}\|$ and

$$\begin{aligned} \mathbf{F} &= \left(F_2 F_1^\top, \dots, F_{H-1} F_{H-2}^\top, F_{H-1}^\top\right), \\ \mathbf{W}(t) &= \left(w_2^t, \dots, w_{H-1}^t, w_H^t\right), \\ \mathbf{G}(t) &= \left(\delta_1 t, \dots, \delta_{H-1}^t\right). \end{aligned}$$

Note that the layer-wise alignment of w^h with $F_h F_{h-1}^\top$ was never measured before: it differs from the alignment of F_h with $w_{h+1} \dots w_H$ observed in [68], which is more akin to GA.

If $\mathbf{W}$ and $\mathbf{F}$ were uncorrelated, the WA defined in (5.41) would be vanishing as the width of the layer grows large. Remarkably, WA becomes of order one after a few epochs as shown in Fig. 5.13a (left), and strongly correlates with GA (right). This suggests that the layer-wise WA uncovered for linear networks with weights initialised to zero also drives GA in the general case.

Align-then-Memorise occurs from bottom layers to top

As can be seen in Fig. 5.13a, WA clearly reaches a maximum then decreases, as expected from the Align-then-Memorise process. Notice that the decrease is stronger for CIFAR10 than it is for MNIST, since CIFAR-10 is much harder to fit than MNIST: more WA needs to be sacrificed. Increasing label corruption similarly makes the datasets harder to fit, and decreases the final WA, as detailed in SM F.5.

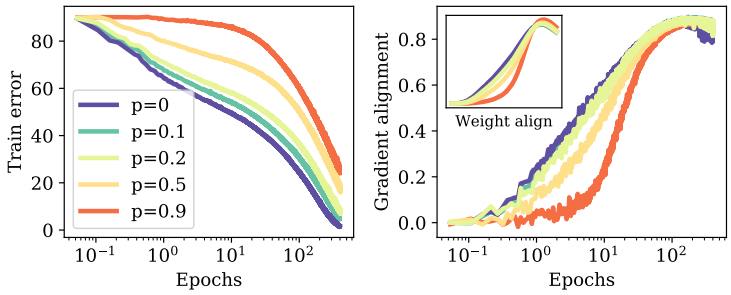


Figure 5.15: **Label corruption hampers alignment in the early stages of training.** We see that the higher the label corruption, the more time WA and GA take to start increasing, since the network initially predicts equal probabilities over the output classes.

However, another question arises: why does the GA keep increasing in this case, in spite of the decreasing WA?

To answer this question, we need to disentangle the dynamics of the layers of the network, as in Eq. (5.42). In Fig. 5.13b, we focus on the ReLU network applied to CIFAR10, and shuffle 10% of the labels in the training set to make the Align-then-Memorise procedure more easily visible. Although the network contains 4 layers of weights, we only have 3 curves for WA and GA: WA is only defined for layers 2 to 4 according to Eq. (5.42), whereas GA of the last layer is not represented here since it is always equal to one.

As can be seen, the second layer is the first to start aligning: it reaches its maximal WA around 1000 epochs (orange dashed line), then decreases. The third layer starts aligning later and reaches its maximal WA around 2000 epochs (green dashed line), then decreases. As for the last layer, the WA is monotonically increasing. Hence, the Align-then-Memorise mechanism operates in a layerwise fashion, starting from the bottom layers to the top layers.

Note that the WA of the last layers is the most crucial, since it affects the GA of all the layers below, whereas the WA of the second layer only affects the GA of the first layer. It therefore makes sense to keep the WA of the last layers high, and let the bottom layers perform the memorization first. This is reminiscent of the linear setup, where all the layers align except for the first, which does all the learning. In fact, this strategy enables the GA of each individual layer to keep increasing until late times: the diminishing WA of the bottom layers is compensated by the increasing WA of the top layers.

5.2.5 What can hamper alignment?

We demonstrated that GA is enabled by the WA mechanism, both theoretically for linear networks and numerically for nonlinear networks. In this section, we leverage our analysis of WA to identify situations in which GA fails.

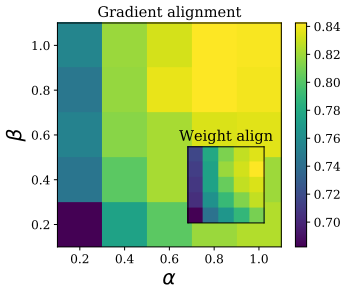


Figure 5.14: **Badly conditioned output statistics can hamper alignment.** WA and GA at the final point of training decrease when the output classes are correlated ($\beta < 1$) or of different variances ($\alpha < 1$).

Alignment is data-dependent

In the linear case, GA occurs if the alignment matrices presented in Sec. 5.2.3 are well conditioned. Note that if the output size K_H is equal to one, e.g. for scalar regression or binary classification tasks, then the alignment matrices are simply scalars, and GA is guaranteed. When this is not the case, one can obtain the deviation from GA by studying the expression of the alignment matrices (5.38). They are formed by summing outer products of the error vectors $\Delta_{t'}\Delta_{t''}^{\top}$, where $\Delta_t = y_t - y_t^*$. Therefore, good conditioning requires the different components of the errors to be uncorrelated and of similar variances. This can be violated by (i) the targets y^*, or (ii) the predictions $y \equiv \phi_\theta(x)$.

(i) Structure of data The first scenario can be demonstrated in a simple regression task on i.i.d. Gaussian inputs $x \sim \mathbb{R}^{10}$. The targets $y^* \in \mathbb{R}^2$ are randomly sampled from the following distribution:

$$y^* \sim \mathcal{N}(0, \Sigma), \quad \Sigma = \begin{pmatrix} 1 & \alpha(1-\beta) \\ \alpha(1-\beta) & \alpha^2 \end{pmatrix}, \quad \alpha, \beta \leq 1. \tag{5.43}$$

In Fig. 5.14, we show the final WA and GA of a 3-layer ReLU network trained for 10^3 epochs on 10^3 examples sampled from this distribution (further details in SM F.5). As predicted, imbalanced ($\alpha < 1$) or correlated ($\beta < 1$) target statistics hamper WA and GA. Note that the inputs also come into play in Eq. (5.38): a more detailed theoretical analysis of the impact of input and target statistics on alignment is deferred to SM F.4.

(ii) Effect of noise For classification tasks, the targets y^* are one-hot encodings whose statistics are naturally well conditioned. However, alignment can be degraded if the statistics of the predictions y become correlated.

One can enforce such a correlation in CIFAR10 by shuffling a fraction p of the labels. The WA and GA dynamics of a 3-layer ReLU network are shown in Fig. 5.15. At high p, the network can only perform random guessing during the first few epochs, and assigns equal probabilities to the 10 classes. The correlated structure of

the predictions prevents alignment until the network starts to fit the random labels: the predictions of the different classes then decouple and WA takes off, leading to GA.

Alignment is impossible for convolutional layers

A convolutional layer with filters H_h can be represented by a large fully-connected layer whose weights are represented by a block Toeplitz matrix $\phi(H_h)$ [75]. This matrix has repeated blocks due to weight sharing, and most of its weights are equal to zero due to locality. In order to verify WA and therefore GA, the following condition must hold: $\phi(H_h) \propto F_h F_{h-1}^\top$. Yet, due to the very constrained structure of $\phi(H_h)$, this is impossible for a general choice of F_h. Therefore, the WA mechanism suggests a simple explanation for why GA doesn't occur in vanilla CNNs, and confirms the previously stated hypothesis that CNNs don't have enough flexibility to align [166].

In the case of convolutional layers, this lack of alignment makes learning near to impossible, and has lead practitioners to design alternatives [128, 227]. However, the extent to which alignment correlates with good performance in the general setup (both in terms of fitting and generalisation) is a complex question which we leave for future work. Indeed, nothing prevents DFA from finding a good optimisation path, different from the one followed by BP. Conversely, obtaining high gradient alignment at the end of training is not a sufficient condition for DFA to retrieve the results of BP, e.g. if the initial trajectory leads to a wrong direction.

Chapter 6

Looking back and beyond

This manuscript reviews several contributions brought by my thesis to the theory of deep leaning. Each of these can be place in the space defined by the three players of machine learning: the architecture, the data and the algorithm, see Fig. 6.1. To conclude, we summarise how they widen our theoretical knowledge in machine learning and propose several axis of future research to expand it further.

In Chap. 3 and [76, 201], we made progress on the architecture side by studying why overparametrisation benefits generalisation in neural networks at odds with the classical variance-bias trade off. By pinning down how various sources of randomness contribute to the test error, we concluded that overparametrisation improves generalisation by taming the fluctuations stemming from the noise in the labels and from the choice of initial weights. These are at the root of the "double-descent" curve displayed by the test error.

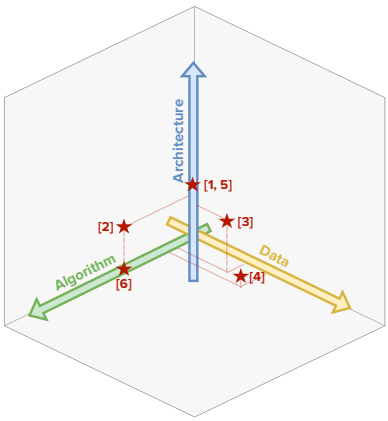


Figure 6.1: The works detailed in this thesis can be placed in the 3D space defined in Fig. 1.2.

Theoretical results like these, however, consider mainly one hidden layer networks, and even then, only in some specific scalings (the ODE limit [38, 269, 270], the lazy regime [145] or the mean-field limit [63, 104, 316]). A major challenge is to develop a theory for deeper networks which is missing except in some limits [145, 175]. A first step to advance this theory is to study the *rectangular* limit of one hidden layer networks in which the hidden layer dimension K scales linearly with input dimension D i.e. $K, D \to \infty$ with $K/D \sim O(1)$. This limit allows for various layers of the same type to be stacked on top of each

other, in contrast to the mean-field limit, where the hidden layer is sent to infinity before the input dimensions, or the ODE limit where the hidden layer is of order 1. The technical difficulty in treating the rectangular case is that the number of dynamical parameters scales as D^2, reminiscent of the *dictionary learning* problem [204, 205, 280, 304] for which Barbier and Macris [27] and Troiani et al. [306] recently made significant contributions. The analysis could build on the tools developed in Sec. 4.1, and focus on one-pass SGD with vanilla i.i.d. Gaussian inputs. One-pass SGD removes correlations between training samples and allows to describe the training dynamics as a Markov process. Gaussian i.i.d. inputs simplify the analysis as they have identity covariance matrix. Finding a theoretical description for the training dynamics of rectangular networks might bring insights into how to treat deep fully connected networks. It is an interesting challenge which I hope to address in future.

Fully connected networks, however, are rarely used by practitioners who most often exploit the power of more complex architectures for which theoretical results are scarce. Describing different types of layers, what they learn and how, is a key step to develop a theory of machine learning. Again, by working in the online learning limit, one might identify the relevant order parameters and understand learning in these architecture models.

Architecture is deeply intertwined with data structure and theoreticians aim at understanding how the two aspects are linked. We established in Sec. 4.2 that two layer neural networks, with few hidden nodes successfully classify a mixture of Gaussian at much lower signal-to-noise ratio than lazy-methods. This emphasises the importance, for structured tasks, of choosing architectures which extract the features in the data. Another manifestation of the interplay between data structure and architecture is the recurring observation that simple models only exploit the first two moments of their inputs. Indeed, we saw in Chap. 4, that for two layer networks and shallow auto-encoders in the limit of small hidden layer and large input dimension, training on gaussian data is indistinguishable from training on real data, a result summarised in the *gaussian equivalence theorem* (GET) [118]. However, in applications neural networks exploit much more than the mean and covariance of their inputs. Pinning down the conditions under which they do so requires understanding the limits of the GET: when and how are networks able to exploit higher order moments? what in the architecture and/or data allows them to do so and why? Answering these questions requires to carefully design the models we work with, simple enough to be analysed theoretically yet complex enough to discriminate between gaussian and non-gaussian data. Progress was made by Ingrosso and Goldt [142] but crisp theoretical answers are still missing.

These might reveal insights into the intrinsic relations between the architecture, the task and the data.

Another way to understand the role of data structure is to focus on feature detection in unsupervised learning which remains much less studied than its supervised counterpart. In Sec. 4.7, we extended the current description of linear shallow autoencoders to encompass non-linear models. We identified which input features the network picks up and showed that these features are learned sequentially in order of importance. We pined-down architectural constraints required for the network to successfully reconstruct its input and finally proposed a variant to vanilla SGD which enables the autoencoder to learn a one to one mapping between its weights and input features. Nonetheless, shallow non-linear autoencoders are simple models which, unlike realistic networks, only exploit the mean and covariance of the inputs. A compelling research topic to further explore representation learning is to consider generative models, such as variational autoencoders [154] which construct "fake" images from Gaussian inputs.

The description of the learning dynamics of ANNs carried out in both [260] and [258] is intrinsically linked to the algorithm used for training. Nowadays, BP is the most widely used algorithm although its properties remain, for the most part, obscure. For instance, we saw in Sec. 4.2 [260], that shallow fully-connected networks with 4 hidden nodes can successfully learn the XOR-like mixture. However, for some initial conditions, the learning dynamics remain stuck in "bad weight configurations" from where they struggle to escape 4.4. This occurrs less frequently for over-parametrised models. A complete description of how performance depends on the choice of initial conditions is fundamental and still missing both in this toy model and in deep neural networks. Performance also depends on the various hyper-parameters chosen for training and on the multiple add-ons to BP. Indeed, we demonstrated in Sec. 5.1 [77] that a good choice of learning rate schedule can speed-up and improve optimisation in non-convex problems. Extending the analysis to realistic settings and nailing down how they impact performance in deep neural networks is an interesting future work direction.

More fundamentally, we saw in Sec. 5.2 that the BP backward pass can be simplified into DFA without hampering performance thanks to a two stage learning process. First networks reduce their expressivity to ensure alignment between the DFA and the BP gradient, then they sacrifice some alignment and converge to the solution. This leads to a "degeneracy breaking" effect: networks consistently converge to the solution maximising alignment between the BP gradient and the DFA gradient. Crucially, this suggests that, when training fully connected networks with BP, part of the information carried by the weights in the backward pass is

unnecessary. These results make progress towards pining down how much and under which conditions one can simplify the BP updates while still performing well. Indeed, architectural constrains, as in convolutional neural networks, and structure in the data can hamper alignment in the first phase thus preventing networks to learn. This highlights once more, the profound connection between architecture, data and algorithm.

Broadening the theory of deep learning is essential as technical innovations occur at remarkable speed, and so does the complexity of networks used in practice. State of the art neural networks have varied architectures; each of them most suited for a different kind of tasks. These include convolutional neural networks [96, 172, 279] for image related tasks, transformers [307], recurrent neural networks [108, 267] , residual networks [133, 140, 295] and many mix between these. In addition, to train a given architecture, there are many add-ons to vanilla BP such as weight-decay [161], different types of optimisers [87, 127, 153], momentum [267, 298, 299], learning rate schedules [147, 222, 245, 292] and many more. Data preprocessing is also very involved, with techniques including data augmentation [289] among others. To grasp how these varied tools impact performance, it is essential to underpin the underlying mechanism driving the success of modern machine learning. We proposed some directions of future research to do so notably by describing the role and interplay between data structure, architecture and algorithm.

We conclude this manuscript, by emphasising that, as long as we do not understand the "black box of AI", we have to address a number of questions on when and how to use deep learning and on its possible ethical consequences [157, 247]. What is learned by the network? How does it depend on the training data used? How *different* can inputs be from those in the training dataset in order for networks to produce a sensible output? Can we trust these technologies enough to integrate them in war drones [226, 242], targeted advertising [78, 130], self-driving cars [82, 216], the judiciary systems [88, 132] and other high impact fields?

Bibliography

[1] S. Abbasi, M. Hajabdollahi, N. Karimi, and S. Samavi. Modeling teacher-student techniques in deep neural networks for knowledge distillation. In *2020 International Conference on Machine Vision and Image Processing (MVIP)*, pages 1–6. IEEE, 2020.

[2] B. Adlam and J. Pennington. Understanding double descent requires a fine-grained bias-variance decomposition. *Advances in neural information processing systems*, 33:11022–11032, 2020.

[3] M. Advani, S. Lahiri, and S. Ganguli. Statistical mechanics of complex neural systems and high dimensional data. *Journal of Statistical Mechanics: Theory and Experiment*, 2013(03):P03014, 2013.

[4] M.S. Advani and A.M. Saxe. High-dimensional dynamics of generalization error in neural networks. *arXiv:1710.03667*, 2017.

[5] M.S. Advani, A.M. Saxe, and H. Sompolinsky. High-dimensional dynamics of generalization error in neural networks. *Neural Networks*, 132:428–446, 2020.

[6] A. Aggarwal, P. Lopatto, and H.T. Yau. Goe statistics for levy matrices. *arXiv:1806.07363*, 2018.

[7] E. Agoritsas, G. Biroli, P. Urbani, and F. Zamponi. Out-of-equilibrium dynamical mean-field equations for the perceptron model. *Journal of Physics A: Mathematical and Theoretical*, 51(8):085002, 2018.

[8] M.A. Aizerman, B. M., and L.I. Rozonoér. Probability problem of pattern recognition learning and potential functions method. *Avtomat. i Telemekh*, 25: 1307–1323, 1964.

[9] Z. Allen-Zhu, Y. Li, and Z. Song. A convergence theory for deep learning via over-parameterization, 2018.

[10] Z. Allen-Zhu, Y. Li, and Z. Song. A convergence theory for deep learning via over-parameterization. In *International Conference on Machine Learning*, pages 242–252. PMLR, 2019.

[11] S. Arora, S. Du, W. Hu, Z. Li, R. Salakhutdinov, and R. Wang. On exact computation with an infinitely wide neural net. In *Advances in Neural Information Processing Systems*, pages 8141–8150, 2019.

[12] S. Arora, N. Cohen, and E. Hazan. On the optimization of deep networks: Implicit acceleration by overparameterization. In *International Conference on Machine Learning*, pages 244–253. PMLR, 2018.

[13] G.B. Arous, R. Gheissari, and A. Jagannath. Algorithmic thresholds for tensor pca. *The Annals of Probability*, 48(4):2052–2087, 2020.

[14] G.B. Arous, R. Gheissari, and A. Jagannath. A classification for the performance of online sgd for high-dimensional inference. *arXiv:2003.10409*, 2020.

[15] B. Aubin, A. Maillard, J. Barbier, F. Krzakala, N. Macris, and L. Zdeborová. The committee machine: Computational to statistical gaps in learning a two-layers neural network. In *Advances in Neural Information Processing Systems 31*, pages 3227–3238, 2018.

[16] A. Auffinger, G. Ben Arous, and J. Černỳ. Random matrices and complexity of spin glasses. *Communications on Pure and Applied Mathematics*, 66(2):165–201, 2013.

[17] J.L. Ba, J.R. Kiros, and G.E. Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[18] F. Bach. Breaking the curse of dimensionality with convex neural networks. *The Journal of Machine Learning Research*, 18(1):629–681, 2017.

[19] F. Bach. Learning theory from first principles, 2021.

[20] Y. Bahri, J. Kadmon, J. Pennington, S. Schoenholz, J. Sohl-Dickstein, and S. Ganguli. Statistical Mechanics of Deep Learning. *Annual Review of Condensed Matter Physics*, 11(1):501–528, 2020.

[21] J. Baik, G. Ben Arous, and S. Péché. Phase transition of the largest eigenvalue for nonnull complex sample covariance matrices. *The Annals of Probability*, 33 (5):1643–1697, 2005.

[22] M. Baity-Jesi, L. Sagun, M. Geiger, S. Spigler, G. Arous, C. Cammarota, Y. LeCun, M. Wyart, and G. Biroli. Comparing Dynamics: Deep Neural Networks versus Glassy Systems. In *Proceedings of the 35th International Conference on Machine Learning*, 2018.

[23] A. Baker, I. Biazzo, A. Braunstein, G. Catania, L. Dall'Asta, A. Ingrosso, F. Krzakala, F. Mazza, M. Mézard, A.P. Muntoni, M. Refinetti, S. Sarao Mannelli, and L. Zdeborová. Epidemic mitigation by statistical inference from contact tracing data. *Proceedings of the National Academy of Sciences*, 118(32): e2106548118, Jul 2021.

[24] P. Baldi and K. Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, 2(1):53–58, 1989.

[25] X. Bao, J. Lucas, S. Sachdeva, and R.B. Grosse. Regularized linear autoencoders recover the principal components, eventually. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 6971–6981. 2020.

[26] D. Barbier, P.H. Pimenta, L.F. Cugliandolo, and D.A. Stariolo. Finite size effects and loss of self-averageness in the relaxational dynamics of the spherical sherrington-kirkpatrick model. *arXiv preprint arXiv:2103.12654*, 2021.

[27] J. Barbier and N. Macris. Statistical limits of dictionary learning: random matrix theory and the spectral replica method. *arXiv preprint arXiv:2109.06610*, 2021.

[28] A.R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, 39(3):930–945, 1993.

[29] S. Bartunov, A. Santoro, B. Richards, L. Marris, G.E. Hinton, and T. Lillicrap. Assessing the scalability of biologically-motivated deep learning algorithms and architectures. In *Advances in Neural Information Processing Systems*, pages 9368–9378, 2018.

[30] M. Bayati and A. Montanari. The dynamics of message passing on dense graphs, with applications to compressed sensing. *IEEE Transactions on Information Theory*, 57(2):764–785, 2011.

[31] M. Belkin, D. Hsu, S. Ma, and S. Mandal. Reconciling modern machine learning and the bias-variance trade-off. *arXiv preprint arXiv:1812.11118*, 2018.

[32] M. Belkin, D. Hsu, and J. Xu. Two models of double descent for weak features. *arXiv preprint arXiv:1903.07571*, 2019.

[33] G. Ben Arous, A. Dembo, and A. Guionnet. Cugliandolo-kurchan equations for dynamics of spin-glasses. *Probability theory and related fields*, 136(4):619–660, 2006.

[34] G. Ben Arous, S. Mei, A. Montanari, and M. Nica. The landscape of the spiked tensor model. *Communications on Pure and Applied Mathematics*, 72(11): 2282–2330, 2019.

[35] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. *Advances in neural information processing systems*, 19, 2006.

[36] L. Berthier and G. Biroli. Theoretical perspective on the glass transition and amorphous materials. *Reviews of modern physics*, 83(2):587, 2011.

[37] M. Biehl and E. Schlösser. The dynamics of on-line principal component analysis. *Journal of Physics A: Mathematical and General*, 31(5):L97–L103, 1998.

[38] M. Biehl and H. Schwarze. Learning by on-line gradient descent. *J. Phys. A. Math. Gen.*, 28(3):643–656, 1995.

[39] M. Biehl, P. Riegler, and C. Wöhler. Transient dynamics of on-line learning in two-layered neural networks. *Journal of Physics A: Mathematical and General*, 29(16), 1996.

[40] G. Biroli. A crash course on ageing. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(05):P05014, 2005.

[41] C. Bishop. *Pattern recognition and machine learning*. Springer, New York, 2006.

[42] B.E. Boser, I.M. Guyon, and V.N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, page 144–152, New York, NY, USA, 1992.

[43] L. Bottou. Stochastic learning. In *Summer School on Machine Learning*, pages 146–168. Springer, 2003.

[44] L. Bottou and O. Bousquet. The Tradeoffs of Large Scale Learning. In *Advances in Neural Information Processing Systems 20*, pages 161–168, 2008.

[45] J.P. Bouchaud, L.F. Cugliandolo, J. Kurchan, and M. Mezard. Out of equilibrium dynamics in spin-glasses and other glassy systems. *Spin glasses and random fields*, 12:161, 1998.

[46] H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4):291–294, 1988.

[47] A.J. Bray. Theory of phase-ordering kinetics. *Advances in Physics*, 51(2): 481–587, 2002.

[48] J. Brea, B. Simsek, B. Illing, and W. Gerstner. Weight-space symmetry in deep networks gives rise to permutation saddles, connected by equal-loss valleys across the loss landscape. *arXiv preprint arXiv:1907.02911*, 2019.

[49] L. Breiman. Reflections after refereeing papers for nips. *The Mathematics of Generalization*, pages 11–15, 1995.

[50] J. Brownlee. Using learning rate schedules for deep learning models in python with keras. *Machine learning mastery, June*, 21, 2016.

[51] J. Bun, J.P. Bouchaud, and M. Potters. Cleaning correlation matrices. *Risk magazine*, 2015, 2016.

[52] Y. Cao and Q. Gu. Generalization bounds of stochastic gradient descent for wide and deep neural networks. In *Advances in Neural Information Processing Systems*, pages 10836–10846, 2019.

[53] A. Caponnetto and E. De Vito. Optimal rates for the regularized least-squares algorithm. *Foundations of Computational Mathematics*, 7(3):331–368, 2007.

[54] A. Cappelli, J. Launay, L. Meunier, R. Ohana, and I. Poli. Ropust: Improving robustness through fine-tuning with photonic processors and synthetic gradients. *arXiv preprint arXiv:2108.04217*, 2021.

[55] T. Castellani and A. Cavagna. Spin-glass theory for pedestrians. *Journal of Statistical Mechanics: Theory and Experiment*, 2005(05):P05012, 2005.

[56] M. Celentano, C. Cheng, and A. Montanari. The high-dimensional asymptotics of first order methods with random data. *arXiv preprint arXiv:2112.07572*, 2021.

[57] N.V. Chawla, T.E. Moore, L.O. Hall, K.W. Bowyer, W.P. Kegelmeyer, and C. Springer. Distributed learning with bagging-like performance. *Pattern recognition letters*, 24(1-3):455–471, 2003.

[58] J. Chen, D. Zhou, Y. Tang, Z. Yang, Y. Cao, and Q. Gu. Closing the generalization gap of adaptive gradient methods in training deep neural networks. *arXiv preprint arXiv:1806.06763*, 2018.

[59] X. Cheng, D. Yin, P. Bartlett, and M. Jordan. Stochastic gradient and langevin processes. In *International Conference on Machine Learning*, pages 1810–1819. PMLR, 2020.

[60] L. Chizat and F. Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. In *Advances in Neural Information Processing Systems 31*, pages 3040–3050, 2018.

[61] L. Chizat, E. Oyallon, and F. Bach. On lazy training in differentiable programming. In *Advances in Neural Information Processing Systems*, pages 2937–2947, 2019.

[62] L. Chizat and F. Bach. Implicit bias of gradient descent for wide two-layer neural networks trained with the logistic loss. In *Conference on Learning Theory*, pages 1305–1338. PMLR, 2020.

[63] L. Chizat, E. Oyallon, and F. Bach. On lazy training in differentiable programming. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 2933–2943. Curran Associates, Inc., 2019.

[64] A. Choromanska, M. Henaff, M. Mathieu, G. Ben Arous, and Y. LeCun. The loss surfaces of multilayer networks. In *Artificial intelligence and statistics*, pages 192–204. PMLR, 2015.

[65] G. Cohen, S. Afshar, J. Tapson, and A. Van Schaik. Emnist: Extending mnist to handwritten letters. In *2017 International Joint Conference on Neural Networks (IJCNN)*, pages 2921–2926. IEEE, 2017.

[66] U. Cohen, S. Chung, D. Lee, and H. Sompolinsky. Separability and geometry of object manifolds in deep neural networks. *Nature communications*, 11(1): 1–13, 2020.

[67] R. Couillet. High dimensional robust classification: A random matrix analysis. In *2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 420–424, 2019.

[68] B. Crafton, A. Parihar, E. Gebhardt, and A. Raychowdhury. Direct feedback alignment with sparse connections for local learning. *Frontiers in neuroscience*, 13:525, 2019.

[69] F. Crick. The recent excitement about neural networks. *Nature*, 337(6203): 129–132, 1989.

[70] A. Crisanti and H.J. Sommers. The spherical p-spin interaction spin glass model: the statics. *Zeitschrift für Physik B Condensed Matter*, 87(3):341–354, 1992.

[71] L.F. Cugliandolo and D.S. Dean. Full dynamical solution for a spherical spin-glass model. *Journal of Physics A: Mathematical and General*, 28(15):4213, 1995.

[72] L.F. Cugliandolo and J. Kurchan. Analytical solution of the off-equilibrium dynamics of a long-range spin-glass model. *Physical Review Letters*, 71(1):173, 1993.

[73] A. Daniely and E. Malach. Learning parities with neural networks. In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[74] A. Daniely, R. Frostig, and Y. Singer. Toward deeper understanding of neural networks: The power of initialization and a dual view on expressivity. In *Advances In Neural Information Processing Systems*, pages 2253–2261, 2016.

[75] S. d'Ascoli, L. Sagun, G. Biroli, and J. Bruna. Finding the needle in the haystack with convolutions: on the benefits of architectural bias. In *Advances in Neural Information Processing Systems*, pages 9334–9345, 2019.

[76] S. d'Ascoli, M. Refinetti, G. Biroli, and F. Krzakala. Double trouble in double descent: Bias and variance (s) in the lazy regime. In *International Conference on Machine Learning*, pages 2280–2290. PMLR, 2020.

[77] S. d'Ascoli, M. Refinetti, and G. Biroli. Optimal learning rate schedules in high-dimensional non-convex optimization problems, 2022.

[78] A. De Bruyn, V. Viswanathan, Y.S. Beh, J.K.U. Brock, and F. von Wangenheim. Artificial intelligence and marketing: Pitfalls and opportunities. *Journal of Interactive Marketing*, 51:91–105, 2020.

[79] A. Dembo and E. Subag. Dynamics for spherical spin glasses: disorder dependent initial conditions. *Journal of Statistical Physics*, 181(2):465–514, 2020.

[80] Z. Deng, A. Kammoun, and C. Thrampoulidis. A model of double descent for high-dimensional binary linear classification. *arXiv:1911.05822*, 2019.

[81] O. Dhifallah and Y.M. Lu. A precise performance analysis of learning with random features, 2020.

[82] L. Ding, D. Li, B. Liu, W. Lan, B. Bai, Q. Hao, W. Cao, and K. Pei. Capture uncertainties in deep neural networks for safe operation of autonomous driving vehicles. In *2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom)*, pages 826–835. IEEE, 2021.

[83] D. Donoho and A. Montanari. High dimensional robust m-estimation: Asymptotic variance via approximate message passing. *Probability Theory and Related Fields*, 166(3):935–969, 2016.

[84] H. Drucker, C. Cortes, L.D. Jackel, Y. LeCun, and V. Vapnik. Boosting and other ensemble methods. *Neural Computation*, 6(6):1289–1301, 1994.

[85] S. Du, J. Lee, Y. Tian, A. Singh, and B. Poczos. Gradient descent learns one-hidden-layer CNN: Don't be afraid of spurious local minima. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1339–1348, 2018.

[86] S. Du, X. Zhai, B. Poczos, and A. Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019.

[87] J. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12 (7), 2011.

[88] M. Dymitruk. Ethical artificial intelligence in judiciary. 02 2019.

[89] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[90] N. El Karoui. The spectrum of kernel random matrices. *Ann. Statist.*, 38(1): 1–50, 02 2010.

[91] S. El-Showk, M.F. Paulos, D. Poland, S. Rychkov, D. Simmons-Duffin, and A. Vichi. Solving the 3d ising model with the conformal bootstrap. *Physical Review D*, 86(2):025022, 2012.

[92] S. El-Showk, M.F. Paulos, D. Poland, S. Rychkov, D. Simmons-Duffin, and A. Vichi. Solving the 3d ising model with the conformal bootstrap ii.

c

c-minimization and precise critical exponents. *Journal of Statistical Physics*, 157(4):869–914, 2014.

[93] A. Engel and C. Van den Broeck. *Statistical mechanics of learning*. Cambridge University Press, 2001.

[94] S. Franz and G. Parisi. The simplest model of jamming. *Journal of Physics A: Mathematical and Theoretical*, 49(14):145001, 2016.

[95] C. Frenkel, M. Lefebvre, and D. Bol. Learning without feedback: Direct random target projection as a feedback-alignment algorithm with layerwise feedforward training, 2019.

[96] K. Fukushima and S. Miyake. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern recognition*, 15(6):455–469, 1982.

[97] M. Gabrié. Mean-field inference methods for neural networks. *Journal of Physics A: Mathematical and Theoretical*, 53(22):223002, 2020.

[98] E. Gardner and B. Derrida. Three unfinished works on the optimal storage capacity of networks. *Journal of Physics A: Mathematical and General*, 22(12): 1983–1994, 1989.

[99] A. Garriga-Alonso, C. Rasmussen, and L. Aitchison. Deep convolutional networks as shallow gaussian processes. In *International Conference on Learning Representations*, 2019.

[100] R. Ge, S.M. Kakade, R. Kidambi, and P. Netrapalli. The step decay schedule: A near optimal, geometrically decaying learning rate procedure for least squares. *arXiv preprint arXiv:1904.12838*, 2019.

[101] M. Geiger, S. Spigler, A. Jacot, and M. Wyart. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(11):113301, 2020.

[102] M. Geiger, A. Jacot, S. Spigler, F. Gabriel, L. Sagun, S. d'Ascoli, G. Biroli, C. Hongler, and M. Wyart. Scaling description of generalization with number of parameters in deep learning. *arXiv preprint arXiv:1901.01608*, 2019.

[103] M. Geiger, S. Spigler, S. d'Ascoli, L. Sagun, M. Baity-Jesi, G. Biroli, and M. Wyart. Jamming transition as a paradigm to understand the loss landscape of deep neural networks. *Physical Review E*, 100(1):012115, 2019.

[104] M. Geiger, S. Spigler, A. Jacot, and M. Wyart. Disentangling feature and lazy learning in deep neural networks: an empirical study. *arXiv preprint arXiv:1906.08034*, 2019.

[105] M. Geiger, A. Jacot, S. Spigler, F. Gabriel, L. Sagun, S. d'Ascoli, G. Biroli, C. Hongler, and M. Wyart. Scaling description of generalization with number of parameters in deep learning. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(2):023401, 2020.

[106] F. Gerace, B. Loureiro, F. Krzakala, M. Mézard, and L. Zdeborová. Generalisation error in learning with random features and the hidden manifold model. In *37th International Conference on Machine Learning*, 2020. arXiv:2002.09339.

[107] C. Gerbelot and R. Berthier. Graph-based approximate message passing iterations. *arXiv preprint arXiv:2109.11905*, 2021.

[108] F. Gers, J. Schmidhuber, and F. Cummins. Learning to forget: continual prediction with lstm. In *1999 Ninth International Conference on Artificial Neural Networks ICANN 99. (Conf. Publ. No. 470)*, volume 2, pages 850–855 vol.2, 1999.

[109] B. Ghorbani, S. Mei, T. Misiakiewicz, and A. Montanari. Limitations of lazy training of two-layers neural network. In *Advances in Neural Information Processing Systems*, volume 32, pages 9111–9121, 2019.

[110] B. Ghorbani, S. Mei, T. Misiakiewicz, and A. Montanari. When do neural networks outperform kernel methods? In *Advances in Neural Information Processing Systems*, volume 33, 2020.

[111] G. Gidel, F. Bach, and S. Lacoste-Julien. Implicit regularization of discrete gradient dynamics in linear neural networks. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. 2019.

[112] J. Gilmer, C. Raffel, S.S. Schoenholz, M. Raghu, and J. Sohl-Dickstein. Explaining the learning dynamics of direct feedback alignment. In *ICLR workshop track*, 2017.

[113] J. Gilmer, B. Ghorbani, A. Garg, S. Kudugunta, B. Neyshabur, D. Cardoze, G. Dahl, Z. Nado, and O. Firat. A loss curvature perspective on training instability in deep learning. *arXiv preprint arXiv:2110.04369*, 2021.

[114] S. Goldt, M. Advani, A. Saxe, F. Krzakala, and L. Zdeborová. Dynamics of stochastic gradient descent for two-layer neural networks in the teacher-student setup. In *Advances in Neural Information Processing Systems 32*, 2019.

[115] S. Goldt, M. Mézard, F. Krzakala, and L. Zdeborová. Modelling the influence of data structure on learning in neural networks. arXiv:1909.11500, 2019.

[116] S. Goldt, M.S. Advani, A.M. Saxe, F. Krzakala, and L. Zdeborová. Dynamics of stochastic gradient descent for two-layer neural networks in the teacher–student setup. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(12): 124010, 2020.

[117] S. Goldt, M. Mézard, F. Krzakala, and L. Zdeborová. Modeling the influence of data structure on learning in neural networks: The hidden manifold model. *Physical Review X*, 10(4):041044, 2020.

[118] S. Goldt, B. Loureiro, G. Reeves, F. Krzakala, M. Mézard, and L. Zdeborová. The gaussian equivalence of generative models for learning with shallow neural networks. *Proceedings of Machine Learning Research*, 145:1–46, 2021.

[119] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[120] A. Gotmare, N.S. Keskar, C. Xiong, and R. Socher. A closer look at deep learning heuristics: Learning rate restarts, warmup and distillation. *arXiv preprint arXiv:1810.13243*, 2018.

[121] P. Goyal, P. Dollár, R. Girshick, P. Noordhuis, L. Wesolowski, A. Kyrola, A. Tulloch, Y. Jia, and K. He. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

[122] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu. A survey of deep learning techniques for autonomous driving. *Journal of Field Robotics*, 37(3): 362–386, 2020.

[123] S. Grossberg. Competitive learning: From interactive activation to adaptive resonance. *Cognitive science*, 11(1):23–63, 1987.

[124] I. Gühring, M. Raslan, and G. Kutyniok. Expressivity of deep neural networks. *arXiv preprint arXiv:2007.04759*, 2020.

[125] S. Gunasekar, B.E. Woodworth, S. Bhojanapalli, B. Neyshabur, and N. Srebro. Implicit regularization in matrix factorization. In *Advances in Neural Information Processing Systems*, pages 6151–6159, 2017.

[126] A. Gupta, A. Anpalagan, L. Guan, and A.S. Khwaja. Deep learning for object detection and scene perception in self-driving cars: Survey, challenges, and open issues. *Array*, 10:100057, 2021.

[127] M.R. Gupta, S. Bengio, and J. Weston. Training highly multiclass classifiers. *The Journal of Machine Learning Research*, 15(1):1461–1492, 2014.

[128] D. Han and H.j. Yoo. Direct feedback alignment based convolutional neural network training for low-power online learning processor. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2019.

[129] L.K. Hansen and P. Salamon. Neural network ensembles. *IEEE transactions on pattern analysis and machine intelligence*, 12(10):993–1001, 1990.

[130] J. Haryanto and L. Moutinho. *Analyzing Children's Consumption Behavior: Ethics, Methodologies, and Future Considerations: Ethics, Methodologies, and Future Considerations*. IGI Global, 2016.

[131] T. Hastie, A. Montanari, S. Rosset, and R.J. Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *arXiv preprint arXiv:1903.08560*, 2019.

[132] Y. Hayashi and K. Wakabayashi. Influence of robophobia on decision making in a court scenario. In *Companion of the 2018 ACM/IEEE International Conference on Human-Robot Interaction*, pages 121–122, 2018.

[133] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[134] G.E. Hinton, S. Osindero, and Y.W. Teh. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7):1527–1554, 2006.

[135] G.E. Hinton, A. Krizhevsky, and S.D. Wang. Transforming auto-encoders. In *International conference on artificial neural networks*, pages 44–51. Springer, 2011.

[136] G.E. Hinton et al. What kind of graphical model is the brain? In *IJCAI*, volume 5, pages 1765–1775, 2005.

[137] R.A. Horn and C.R. Johnson. *Matrix analysis*. Cambridge university press, 2012.

[138] H. Hu and Y.M. Lu. Universality laws for high-dimensional learning with random features. *arXiv:2009.07669*, 2020.

[139] W. Hu, C.J. Li, L. Li, and J.G. Liu. On the diffusion approximation of nonconvex stochastic gradient descent. *arXiv preprint arXiv:1705.07562*, 2017.

[140] G. Huang, Z. Liu, L. Van Der Maaten, and K.Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[141] J. Huang, J. Chai, and S. Cho. Deep learning in finance and banking: A literature review and classification. *Frontiers of Business Research in China*, 14 (1):1–24, 2020.

[142] A. Ingrosso and S. Goldt. Data-driven emergence of convolutional structure in neural networks. *arXiv preprint arXiv:2202.00565*, 2022.

[143] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 448–456, 2015.

[144] A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 8571–8580. 2018.

[145] A. Jacot, F. Gabriel, and C. Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems 32*, pages 8571–8580, 2018.

[146] A. Jacot, B. Şimşek, F. Spadaro, C. Hongler, and F. Gabriel. Implicit regularization of random feature models. *arXiv preprint arXiv:2002.08404*, 2020.

[147] S. Jastrzkbski, Z. Kenton, D. Arpit, N. Ballas, A. Fischer, Y. Bengio, and A. Storkey. Three factors influencing minima in sgd. *arXiv preprint arXiv:1711.04623*, 2017.

[148] S. Jean, K. Cho, R. Memisevic, and Y. Bengio. On using very large target vocabulary for neural machine translation. *arXiv preprint arXiv:1412.2007*, 2014.

[149] Z. Ji and M. Telgarsky. Gradient descent aligns the layers of deep linear networks. In *International Conference on Learning Representations (ICLR)*, 2019.

[150] D. Kalimeris, G. Kaplun, P. Nakkiran, B. Edelman, T. Yang, B. Barak, and H. Zhang. Sgd on neural networks learns functions of increasing complexity. In *Advances in Neural Information Processing Systems 32*, pages 3496–3506, 2019.

[151] N.S. Keskar and R. Socher. Improving generalization performance by switching from adam to sgd. *arXiv preprint arXiv:1712.07628*, 2017.

[152] G. Kimeldorf and G. Wahba. Some results on tchebycheffian spline functions. *Journal of mathematical analysis and applications*, 33(1):82–95, 1971.

[153] D.P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[154] D.P. Kingma and M. Welling. An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, 12(4):307–392, 2019.

[155] G.R. Kini and C. Thrampoulidis. Analytic study of double descent in binary classification: The impact of loss. In *2020 IEEE International Symposium on Information Theory (ISIT)*, pages 2527–2532. IEEE, 2020.

[156] J. Kivinen, A.J. Smola, and R.C. Williamson. Online learning with kernels. *IEEE transactions on signal processing*, 52(8):2165–2176, 2004.

[157] J. Kleinberg, H. Lakkaraju, J. Leskovec, J. Ludwig, and S. Mullainathan. Human decisions and machine predictions. *The quarterly journal of economics*, 133(1):237–293, 2018.

[158] F. Kos, D. Poland, D. Simmons-Duffin, and A. Vichi. Precision islands in the ising and o (n) models. *Journal of High Energy Physics*, 2016(8):1–16, 2016.

[159] M.A. Kramer. Nonlinear principal component analysis using autoassociative neural networks. *AIChE journal*, 37(2):233–243, 1991.

[160] A. Krizhevsky, G. Hinton, et al. Learning multiple layers of features from tiny images. arXiV preprint, 2009.

[161] A. Krogh and J. Hertz. A simple weight decay can improve generalization. *Advances in neural information processing systems*, 4, 1991.

[162] A. Krogh and J.A. Hertz. Generalization in a linear perceptron in the presence of noise. *Journal of Physics A: Mathematical and General*, 25(5):1135, 1992.

[163] A. Krogh and J. Vedelsby. Neural network ensembles, cross validation, and active learning. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7. 1995.

[164] F. Krzakala and J. Kurchan. Landscape analysis of constraint satisfaction problems. *Physical Review E*, 76(2):021122, 2007.

[165] D. Kunin, J. Bloom, A. Goeva, and C. Seed. Loss landscapes of regularized linear autoencoders. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 3560–3569. 2019.

[166] J. Launay, I. Poli, and F. Krzakala. Principled training of neural networks with direct feedback alignment. *arXiv:1906.04554*, 2019.

[167] J. Launay, I. Poli, F. Boniface, and F. Krzakala. Direct feedback alignment scales to modern deep learning tasks and architectures. In *Advances in neural information processing systems*, 2020.

[168] Y. Le Cun. Learning process in an asymmetric threshold network. In *Disordered systems and biological organization*, pages 233–240. Springer, 1986.

[169] Y. Le Cun, I. Kanter, and S.A. Solla. Eigenvalues of covariance matrices: Application to neural-network learning. *Physical Review Letters*, 66(18):2396, 1991.

[170] Y. LeCun and C. Cortes. The MNIST database of handwritten digits, 1998.

[171] Y. LeCun, B. Boser, J. Denker, D. Henderson, R. Howard, W. Hubbard, and L. Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in neural information processing systems*, pages 396–404, 1990.

[172] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[173] Y. LeCun, B. Boser, J.S. Denker, D. Henderson, R.E. Howard, W. Hubbard, and L.D. Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[174] J. Lee, J. Sohl-Dickstein, J. Pennington, R. Novak, S. Schoenholz, and Y. Bahri. Deep neural networks as gaussian processes. In *International Conference on Learning Representations*, 2018.

[175] J. Lee, Y. Bahri, R. Novak, S.S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

[176] D. LeJeune, H. Javadi, and R. Baraniuk. The implicit regularization of ordinary least squares ensembles. In *International Conference on Artificial Intelligence and Statistics*, pages 3525–3535. PMLR, 2020.

[177] M. Lelarge and L. Miolane. Asymptotic bayes risk for gaussian mixture in a semi-supervised setting. In *2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 639–643. IEEE, 2019.

[178] C. Leondes. *Expert Systems: The Technology of Knowledge Management for the 21st Century*. Expert Systems: The Technology of Knowledge Management and Decision Making for the 21st Century. Academic Press, 2002.

[179] M.K. Leung, H.Y. Xiong, L.J. Lee, and B.J. Frey. Deep learning of the tissue-regulated splicing code. *Bioinformatics*, 30(12):i121–i129, 2014.

[180] A. Lewkowycz. How to decay your learning rate. *arXiv preprint arXiv:2103.12682*, 2021.

[181] C. Li, H. Farkhoor, R. Liu, and J. Yosinski. Measuring the intrinsic dimension of objective landscapes. *arXiv preprint arXiv:1804.08838*, 2018.

[182] Q. Li, C. Tai, and E. Weinan. Stochastic modified equations and adaptive stochastic gradient algorithms. In *International Conference on Machine Learning*, pages 2101–2110. PMLR, 2017.

[183] Y. Li and Y. Liang. Learning Overparameterized Neural Networks via Stochastic Gradient Descent on Structured Data. In *Advances in Neural Information Processing Systems 31*, 2018.

[184] Y. Li, C. Wei, and T. Ma. Towards explaining the regularization effect of initial large learning rate in training neural networks. *arXiv preprint arXiv:1907.04595*, 2019.

[185] Y. Li, T. Ma, and H.R. Zhang. Learning over-parametrized two-layer neural networks beyond ntk. In J. Abernethy and S. Agarwal, editors, *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 2613–2682. 2020.

[186] Z. Li, R. Wang, D. Yu, S.S. Du, W. Hu, R. Salakhutdinov, and S. Arora. Enhanced convolutional neural tangent kernels. *arXiv:1911.00809*, 2019.

[187] Q. Liao, J.Z. Leibo, and T. Poggio. How important is weight symmetry in backpropagation? In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1837–1844, 2016.

[188] Z. Liao and R. Couillet. On inner-product kernels of high dimensional data. In *2019 IEEE 8th International Workshop on Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP)*, pages 579–583, 2019.

[189] Z. Liao and R. Couillet. On the spectrum of random features maps of high dimensional data. In *International Conference on Machine Learning*, pages 3063–3071. PMLR, 2018.

[190] T. Lillicrap, D. Cownden, D. Tweed, and C. Akerman. Random synaptic feedback weights support error backpropagation for deep learning. *Nature Communications*, 7:1–10, 2016.

[191] L. Lin and E. Dobriban. What causes the test error? going beyond bias-variance via anova. *Journal of Machine Learning Research*, 22(155):1–82, 2021.

[192] C.Y. Liou, J.C. Huang, and W.C. Yang. Modeling word perception using the elman network. *Neurocomputing*, 71(16-18):3150–3157, 2008.

[193] A.J. Liu and S.R. Nagel. Jamming is not just cool any more. *Nature*, 396(6706): 21–22, 1998.

[194] G.H. Liu and E.A. Theodorou. Deep learning theory review: An optimal control and dynamical systems perspective. *arXiv preprint arXiv:1908.10920*, 2019.

[195] G. Livan, M. Novaes, and P. Vivo. *Introduction to random matrices: theory and practice*, volume 26. Springer, 2018.

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

[197] I. Loshchilov and F. Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[198] C. Louart, Z. Liao, and R. Couillet. A random matrix approach to neural networks. *The Annals of Applied Probability*, 28(2):1190–1248, 2018.

[199] B. Loureiro, C. Gerbelot, H. Cui, S. Goldt, F. Krzakala, M. Mézard, and L. Zdeborová. Learning curves of generic features maps for realistic datasets with a teacher-student model. In *Advances in Neural Information Processing Systems*, volume 34, 2021.

[200] B. Loureiro, G. Sicuro, C. Gerbelot, A. Pacco, F. Krzakala, and L. Zdeborová. Learning gaussian mixtures with generalized linear models: Precise asymptotics in high-dimensions. *Advances in Neural Information Processing Systems*, 34, 2021.

[201] B. Loureiro, C. Gerbelot, M. Refinetti, G. Sicuro, and F. Krzakala. Fluctuations, bias, variance & ensemble of learners: Exact asymptotics for convex losses in high-dimension, 2022.

[202] J. Ma, R.P. Sheridan, A. Liaw, G.E. Dahl, and V. Svetnik. Deep neural nets as a method for quantitative structure–activity relationships. *Journal of chemical information and modeling*, 55(2):263–274, 2015.

[203] X. Mai and Z. Liao. High dimensional classification via empirical risk minimization: Improvements and optimality. *arXiv preprint arXiv:1905.13742*, 2019.

[204] A. Maillard, F. Krzakala, M. Mézard, and L. Zdeborová. Perturbative construction of mean-field equations in extensive-rank matrix factorization and denoising. *arXiv preprint arXiv:2110.08775*, 2021.

[205] J. Mairal, F. Bach, J. Ponce, and G. Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, page 689–696, New York, NY, USA, 2009.

[206] S.S. Mannelli and L. Zdeborová. Thresholds of descending algorithms in inference problems. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(3):034004, 2020.

[207] S.S. Mannelli, G. Biroli, C. Cammarota, F. Krzakala, P. Urbani, and L. Zdeborová. Marvels and pitfalls of the langevin algorithm in noisy high-dimensional inference. *Physical Review X*, 10(1):011057, 2020.

[208] V. Manohar, P. Ghahremani, D. Povey, and S. Khudanpur. A teacher-student learning approach for unsupervised domain adaptation of sequence-trained asr models. In *2018 IEEE Spoken Language Technology Workshop (SLT)*, pages 250–257. IEEE, 2018.

[209] A.G.d.G. Matthews, J. Hron, M. Rowland, R. Turner, and Z. Ghahramani. Gaussian process behaviour in wide deep neural networks. In *International Conference on Learning Representations*, 2018.

[210] J. McCarthy. Recursive functions of symbolic expressions and their computation by machine, part i. *Communications of the ACM*, 3(4):184–195, 1960.

[211] J. McCarthy et al. *Programs with common sense*. RLE and MIT computation center Cambridge, MA, USA, 1960.

[212] S. Mei, T. Misiakiewicz, and A. Montanari. Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. arXiv:1902.06015, 2019.

[213] S. Mei and A. Montanari. The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*, 06 2021.

[214] S. Mei, A. Montanari, and P.M. Nguyen. A mean field view of the landscape of two-layer neural networks. *Proceedings of the National Academy of Sciences*, 115(33):E7665–E7671, 2018.

[215] S. Mei, T. Misiakiewicz, and A. Montanari. Generalization error of random features and kernel methods: hypercontractivity and kernel matrix concentration. *arXiv preprint arXiv:2101.10588*, 2021.

[216] C. Metz. The costly pursuit of self-driving cars continues on. and on. and on. *The New York Times*, 2021.

[217] M. Mézard, G. Parisi, and M. Virasoro. *Spin glass theory and beyond: An Introduction to the Replica Method and Its Applications*, volume 9. World Scientific Publishing Company, 1987.

[218] V. M.henko and L. Pastur. Distribution of eigenvalues for some sets of random matrices. *Matematicheskii Sbornik*, 114(4):507–536, 1967.

[219] P. Mianjy, R. Arora, and R. Vidal. On the implicit bias of dropout. In J. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 3540–3548. 2018.

[220] F. Mignacco, F. Krzakala, Y.M. Lu, and L. Zdeborová. The role of regularization in classification of high-dimensional noisy gaussian mixture. In *37th International Conference on Machine Learning*, 2020.

[221] F. Mignacco, F. Krzakala, P. Urbani, and L. Zdeborová. Dynamical mean-field theory for stochastic gradient descent in gaussian mixture classification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[222] F. Mignacco and P. Urbani. The effective noise of stochastic gradient descent. *arXiv preprint arXiv:2112.10852*, 2021.

[223] C. Mingard, G. Valle-Pérez, J. Skalse, and A.A. Louis. Is sgd a bayesian sampler? well, almost. *Journal of Machine Learning Research*, 22(79):1–64, 2021.

[224] M.L. Minsky. *Computation: Finite and Infinite Machines*. Prentice-Hall, Inc., USA, 1967.

[225] G. Montavon, G. Orr, and K.R. Müller. *Neural networks: tricks of the trade*, volume 7700. springer, 2012.

[226] F.E. Morgan, B. Boudreaux, A.J. Lohn, M. Ashby, C. Curriden, K. Klima, and D. Grossman. *Military Applications of Artificial Intelligence: Ethical Concerns in an Uncertain World*. RAND Corporation, Santa Monica, CA, 2020.

[227] T.H. Moskovitz, A. Litwin-Kumar, and L. Abbott. Feedback alignment in deep convolutional networks. *arXiv preprint arXiv:1812.06488*, 2018.

[228] E. Moulines and F. Bach. Non-asymptotic analysis of stochastic approximation algorithms for machine learning. *Advances in neural information processing systems*, 24:451–459, 2011.

[229] P. Nakkiran. More data can hurt for linear regression: Sample-wise double descent. *arXiv preprint arXiv:1912.07242*, 2019.

[230] P. Nakkiran, G. Kaplun, Y. Bansal, T. Yang, B. Barak, and I. Sutskever. Deep double descent: Where bigger models and more data hurt. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(12):124003, 2021.

[231] P. Nakkiran, B. Neyshabur, and H. Sedghi. The bootstrap framework: Generalization through the lens of online optimization. In *International Conference on Learning Representations*, 2021.

[232] B. Neal, S. Mittal, A. Baratin, V. Tantia, M. Scicluna, S. Lacoste-Julien, and I. Mitliagkas. A modern take on the bias-variance tradeoff in neural networks. *arXiv preprint arXiv:1810.08591*, 2018.

[233] B. Neyshabur, R. Tomioka, and N. Srebro. In search of the real inductive bias: On the role of implicit regularization in deep learning. In *ICLR*, 2015.

[234] B. Neyshabur, R. Tomioka, R. Salakhutdinov, and N. Srebro. Geometry of optimization and implicit regularization in deep learning. *arXiv preprint arXiv:1705.03071*, 2017.

[235] P.M. Nguyen. Analysis of feature learning in weight-tied autoencoders via the mean field lens. *arXiv preprint arXiv:2102.08373*, 2021.

[236] T.V. Nguyen, R.K. Wong, and C. Hegde. On the dynamics of gradient descent for autoencoders. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pages 2858–2867. PMLR, 2019.

[237] A. Nøkland. Direct Feedback Alignment Provides Learning in Deep Neural Networks. In *Advances in Neural Information Processing Systems 29*, 2016.

[238] nvidia. Nvidia launches the world's first graphics processing unit: Geforce 256, 1999. URL https://web.archive.org/web/20160408122443/http://www.nvidia.com/object/gpu.html.

[239] R. Oftadeh, J. Shen, Z. Wang, and D. Shell. Eliminating the invariance on the loss landscape of linear autoencoders. In H.D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 7405–7413. 07 2020.

[240] E. Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.

[241] M. Opper and W. Kinzel. Statistical mechanics of generalization. In *Models of neural networks III*, pages 151–209. Springer, 1996.

[242] S. O'Sullivan, N. Nevejans, C. Allen, A. Blyth, S. Leonard, U. Pagallo, K. Holzinger, A. Holzinger, M.I. Sajid, and H. Ashrafian. Legal, regulatory, and ethical frameworks for development of standards in artificial intelligence (ai) and autonomous robotic surgery. *The international journal of medical robotics and computer assisted surgery*, 15(1):e1968, 2019.

[243] J. Paccolat, L. Petrini, M. Geiger, K. Tyloo, and M. Wyart. Geometric compression of invariant manifolds in neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2021(4):044001, 2021.

[244] M. Pak and S. Kim. A review of deep learning in image recognition. In *2017 4th international conference on computer applications and information processing technology (CAIPT)*, pages 1–3. IEEE, 2017.

[245] D. Park, J. Sohl-Dickstein, Q. Le, and S. Smith. The effect of network width on stochastic gradient descent and generalization: an empirical study. In *International Conference on Machine Learning*, pages 5042–5051. PMLR, 2019.

[246] J. Pennington and P. Worah. Nonlinear random matrix theory for deep learning. In *Advances in Neural Information Processing Systems*, pages 2637–2646, 2017.

[247] E. Pierson, D.M. Cutler, J. Leskovec, S. Mullainathan, and Z. Obermeyer. An algorithmic approach to reducing unexplained pain disparities in underserved populations. *Nature Medicine*, 27(1):136–140, 2021.

[248] E. Plaut. From principal subspaces to principal components with linear autoencoders. *arXiv preprint arXiv:1804.10253*, 2018.

[249] D. Poland, S. Rychkov, and A. Vichi. The conformal bootstrap: Theory, numerical techniques, and applications. *Reviews of Modern Physics*, 91(1): 015002, 2019.

[250] M. Potters and J.P. Bouchaud. *A First Course in Random Matrix Theory: For Physicists, Engineers and Data Scientists*. Cambridge University Press, 2020.

[251] A. Power, Y. Burda, H. Edwards, I. Babuschkin, and V. Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. In *ICLR MATH-AI Workshop*, 2021.

[252] A. Pretorius, S. Kroon, and H. Kamper. Learning dynamics of linear denoising autoencoders. In *International Conference on Machine Learning*, pages 4141–4150. PMLR, 2018.

[253] A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Advances in neural information processing systems*, pages 1177–1184, 2008.

[254] A. Rahimi and B. Recht. Weighted sums of random kitchen sinks: Replacing minimization with randomization in learning. In *Advances in neural information processing systems*, pages 1313–1320, 2009.

[255] M. Ranzato, C. Poultney, S. Chopra, and Y. Cun. Efficient learning of sparse representations with an energy-based model. *Advances in neural information processing systems*, 19, 2006.

[256] R. Rattazzi, V.S. Rychkov, E. Tonni, and A. Vichi. Bounding scalar operator dimensions in 4d cft. *Journal of High Energy Physics*, 2008(12):031, 2008.

[257] M. Reehorst, M. Refinetti, and A. Vichi. Bootstrapping traceless symmetric $o(n)$ scalars, 2020.

[258] M. Refinetti and S. Goldt. The dynamics of representation learning in shallow, non-linear autoencoders, 2022.

[259] M. Refinetti, S. d'Ascoli, R. Ohana, and S. Goldt. Align, then memorise: the dynamics of learning with feedback alignment. In M. Meila and T. Zhang, editors, *International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8925–8935. PMLR, 2021.

[260] M. Refinetti, S. Goldt, F. Krzakala, and L. Zdeborova. Classifying high-dimensional gaussian mixtures: Where kernel methods fail and neural networks succeed. In M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8936–8947. 2021.

[261] D.A. Reynolds. Gaussian mixture models. *Encyclopedia of biometrics*, 741: 659–663, 2009.

[262] J.F. Ritchie Ng. Deep learning wizard. *Zenodo*, Apr 2019.

[263] V. Ros, G. Ben Arous, G. Biroli, and C. Cammarota. Complex energy landscapes in spiked-tensor and simple glassy models: Ruggedness, arrangements of local minima, and phase transitions. *Physical Review X*, 9(1):011003, 2019.

[264] S. Rosset, J. Zhu, and T.J. Hastie. Margin maximizing loss functions. In *Advances in neural information processing systems*, pages 1237–1244, 2004.

[265] G. Rotskoff and E. Vanden-Eijnden. Parameters as interacting particles: long time convergence and asymptotic error scaling of neural networks. In *Advances in Neural Information Processing Systems 31*, pages 7146–7155, 2018.

[266] S. Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[267] D.E. Rumelhart, G.E. Hinton, and R.J. Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

[268] D. Saad. *On-line learning in neural networks*, volume 17. Cambridge University Press, 2009.

[269] D. Saad and S. Solla. Exact Solution for On-Line Learning in Multilayer Neural Networks. *Phys. Rev. Lett.*, 74(21):4337–4340, 1995.

[270] D. Saad and S. Solla. On-line learning in soft committee machines. *Phys. Rev. E*, 52(4):4225–4243, 1995.

[271] D. Saad and S. Solla. Learning with Noise and Regularizers Multilayer Neural Networks. In *Advances in Neural Information Processing Systems 9*, pages 260–266, 1997.

[272] I. Safran and O. Shamir. Spurious local minima are common in two-layer relu neural networks. In *International Conference on Machine Learning*, pages 4433–4441. PMLR, 2018.

[273] T.D. Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural networks*, 2(6):459–473, 1989.

[274] S. Sarao Mannelli, G. Biroli, C. Cammarota, F. Krzakala, and L. Zdeborová. Who is afraid of big bad minima? analysis of gradient-flow in spiked matrix-tensor models. *Advances in Neural Information Processing Systems*, 32:8679–8689, 2019.

[275] A. Saxe, J. McClelland, and S. Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. In *ICLR*, 2014.

[276] A. Saxe, Y. Bansal, J. Dapello, M. Advani, A. Kolchinsky, B. Tracey, and D. Cox. On the information bottleneck theory of deep learning. In *ICLR*, 2018.

[277] A. Saxe, J. McClelland, and S. Ganguli. A mathematical theory of semantic development in deep neural networks. *Proceedings of the National Academy of Sciences*, 116(23):11537–11546, 2019.

[278] E. Schlösser, D. Saad, and M. Biehl. Optimization of on-line principal component analysis. *Journal of Physics A: Mathematical and General*, 32(22):4061–4067, 1999.

[279] J. Schmidhuber, U. Meier, and D. Ciresan. Multi-column deep neural networks for image classification. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649. IEEE Computer Society, 2012.

[280] H.C. Schmidt. *Statistical Physics of Sparse and Dense Models in Optimization and Inference*. PhD thesis, Université Paris Saclay (COmUE), 2018.

[281] B. Scholkopf and A. Smola. *Learning with kernels: support vector machines, regularization, optimization, and beyond*. Adaptive Computation and Machine Learning series, 2018.

[282] B. Schölkopf, K. Tsuda, and J.P. Vert. *Kernel methods in computational biology*. MIT press, 2004.

[283] H. Schütze, C.D. Manning, and P. Raghavan. *Introduction to information retrieval*, volume 39. Cambridge University Press Cambridge, 2008.

[284] M. Seddik, M. Tamaazousti, and R. Couillet. Kernel random matrices of large concentrated data: the example of gan-generated images. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7480–7484. IEEE, 2019.

[285] H.S. Seung, H. Sompolinsky, and N. Tishby. Statistical mechanics of learning from examples. *Physical review A*, 45(8):6056, 1992.

[286] V. Shankar, A. Fang, W. Guo, S. Fridovich-Keil, J. Ragan-Kelley, L. Schmidt, and B. Recht. Neural kernels without tangents. In H.D. III and A. Singh, editors, *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 8614–8623. 2020.

[287] J. Shawe-Taylor, N. Cristianini, et al. *Kernel methods for pattern analysis*. Cambridge university press, 2004.

[288] D. Sherrington and S. Kirkpatrick. Solvable model of a spin-glass. *Physical review letters*, 35(26):1792, 1975.

[289] C. Shorten and T.M. Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[290] J. Sirignano and K. Spiliopoulos. Mean field analysis of neural networks: A central limit theorem. *Stochastic Processes and their Applications*, 2019.

[291] L.N. Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017.

[292] S.L. Smith, P.J. Kindermans, C. Ying, and Q.V. Le. Don't decay the learning rate, increase the batch size. *arXiv preprint arXiv:1711.00489*, 2017.

[293] M. Soltanolkotabi, A. Javanmard, and J. Lee. Theoretical insights into the optimization landscape of over-parameterized shallow neural networks. *IEEE Transactions on Information Theory*, 65(2):742–769, 2018.

[294] S. Spigler, M. Geiger, S. d'Ascoli, L. Sagun, G. Biroli, and M. Wyart. A jamming transition from under-to over-parametrization affects generalization

in deep learning. *Journal of Physics A: Mathematical and Theoretical*, 52(47): 474001, 2019.

[295] R.K. Srivastava, K. Greff, and J. Schmidhuber. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

[296] I. Steinwart, D.R. Hush, C. Scovel, et al. Optimal rates for regularized least squares regression. In *COLT*, pages 79–93, 2009.

[297] I. Sutskever, O. Vinyals, and Q. Le. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N.D. Lawrence, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. 2014.

[298] I. Sutskever. *Training recurrent neural networks*. University of Toronto Toronto, ON, Canada, 2013.

[299] I. Sutskever, J. Martens, G. Dahl, and G. Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147. PMLR, 2013.

[300] T. Suzuki and S. Akiyama. Benefit of deep learning with non-convex noisy gradient descent: Provable excess risk bound and superiority to kernel methods. *arXiv preprint arXiv:2012.03224*, 2020.

[301] E. Tarquini, G. Biroli, and M. Tarzia. Level statistics and localization transitions of levy matrices. *Physical review letters*, 116(1):010601, 2016.

[302] F. Thalmann. Geometrical approach for the mean-field dynamics of a particle in a short range correlated random potential. *The European Physical Journal B-Condensed Matter and Complex Systems*, 19(1):49–63, 2001.

[303] Y. Tian. An analytical formula of population gradient for two-layered relu network and its applications in convergence and critical point analysis. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, page 3404–3413, 2017.

[304] I. Tosic and P. Frossard. Dictionary learning. *IEEE Signal Processing Magazine*, 28:27–38, 2011.

[305] C.A. Tracy and H. Widom. On orthogonal and symplectic matrix ensembles. *Communications in Mathematical Physics*, 177(3):727–754, 1996.

[306] E. Troiani, V. Erba, F. Krzakala, A. Maillard, and L. Zdeborová. Optimal denoising of rotationally invariant rectangular matrices. *arXiv preprint arXiv:2203.07752*, 2022.

[307] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[308] C. Wang, H. Hu, and Y. Lu. A solvable high-dimensional model of gan. In *Advances in Neural Information Processing Systems*, pages 13759–13768, 2019.

[309] J. Wang, A. Pun, J. Tu, S. Manivasagam, A. Sadat, S. Casas, M. Ren, and R. Urtasun. Advsim: Generating safety-critical scenarios for self-driving vehicles. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9909–9918, 2021.

[310] Y. Wang and G. Yan. Survey on the application of deep learning in algorithmic trading. *Data Science in Finance and Economics*, 1(4):345–361, 2021.

[311] T. Watkin, A. Rau, and M. Biehl. The statistical mechanics of learning a rule. *Reviews of Modern Physics*, 65(2):499–556, 1993.

[312] C. Wei, J.D. Lee, Q. Liu, and T. Ma. Regularization matters: Generalization and optimization of neural nets v.s. their induced kernel. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. 2019.

[313] E.P. Wigner. On the distribution of the roots of certain symmetric matrices. *Annals of Mathematics*, pages 325–327, 1958.

[314] A.C. Wilson, R. Roelofs, M. Stern, N. Srebro, and B. Recht. The marginal value of adaptive gradient methods in machine learning. *arXiv preprint arXiv:1705.08292*, 2017.

[315] J. Wishart. The generalised product moment distribution in samples from a normal multivariate population. *Biometrika*, 20(1/2):32–52, 1928.

[316] B. Woodworth, S. Gunasekar, J. Lee, D. Soudry, and N. Srebro. Kernel and deep regimes in overparametrized models. *arXiv preprint arXiv:1906.05827*, 2019.

[317] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiV preprint, 2017.

[318] H.Y. Xiong, B. Alipanahi, L.J. Lee, H. Bretschneider, D. Merico, R.K. Yuen, Y. Hua, S. Gueroussov, H.S. Najafabadi, T.R. Hughes, et al. The human splicing code reveals new insights into the genetic determinants of disease. *Science*, 347(6218):1254806, 2015.

[319] W. Xu. Towards optimal one pass large scale learning with averaged stochastic gradient descent. *arXiv preprint arXiv:1107.2490*, 2011.

[320] G. Yehudai and O. Shamir. On the power and limitations of random features for understanding neural networks. In *Advances in Neural Information Processing Systems*, volume 32, pages 6598–6608, 2019.

[321] Y. Yoshida and M. Okada. Data-dependence of plateau phenomenon in learning with neural network — statistical mechanical analysis. In *Advances in Neural Information Processing Systems 32*, pages 1720–1728, 2019.

[322] Y. Yoshida, R. Karakida, M. Okada, and S.I. Amari. Statistical mechanical analysis of learning dynamics of two-layer perceptron with multiple output units. *Journal of Physics A: Mathematical and Theoretical*, 52(18):184002, 2019.

[323] K. You, M. Long, J. Wang, and M.I. Jordan. How does learning rate decay help modern neural networks? *arXiv preprint arXiv:1908.01878*, 2019.

[324] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda. A survey of autonomous driving: Common practices and emerging technologies. *IEEE access*, 8:58443–58469, 2020.

[325] L. Zdeborová. Understanding deep learning is also a job for physicists. *Nature Physics*, 2020.

[326] L. Zdeborová and F. Krzakala. Phase transitions in the coloring of random graphs. *Physical Review E*, 76(3):031131, 2007.

[327] L. Zdeborová and F. Krzakala. Statistical physics of inference: Thresholds and algorithms. *Advances in Physics*, 65(5):453–552, 2016.

[328] M.D. Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

[329] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017.

[330] S. Zhang, L. Yao, A. Sun, and Y. Tay. Deep learning based recommender system: A survey and new perspectives. *ACM Computing Surveys (CSUR)*, 52 (1):1–38, 2019.

[331] Y. Zhang, J. Duchi, and M. Wainwright. Divide and conquer kernel ridge regression. In *Conference on Learning Theory*, pages 592–617, 2013.

[332] Y. Zhang, J. Duchi, and M. Wainwright. Divide and conquer kernel ridge regression: A distributed algorithm with minimax optimal rates. *The Journal of Machine Learning Research*, 16(1):3299–3340, 2015.

[333] K. Zhong, Z. Song, P. Jain, P. Bartlett, and I. Dhillon. Recovery guarantees for one-hidden-layer neural networks. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pages 4140–4149. JMLR. org, 2017.

Appendix A

Reconciling the bias-variance trade-off with modern deep learning

A.1 Further analytical results

Asymptotic scaling laws

Figure A.1 (left) shows that the various terms entering the decomposition of the

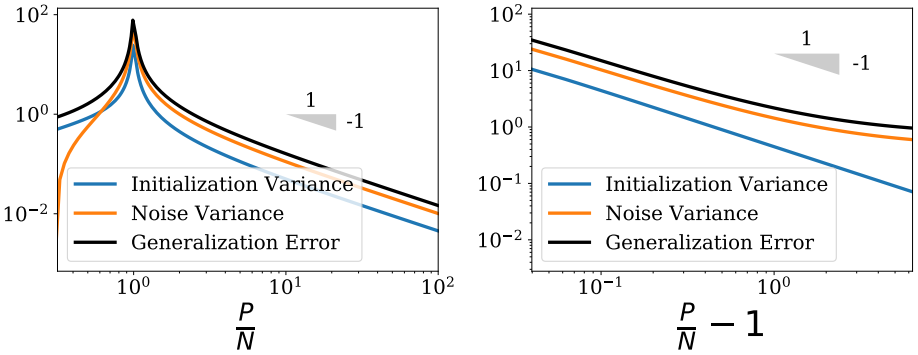


Figure A.1: Log-log behaviour of the quantities of interest at **Left:** $P/N \to \infty$ and **Right**: $P \to N$ with $\lambda = 10^{-5}$, $N/D = 1$ and $\tau = 1$. In both cases, one observes an inverse scaling law.

test error approach their asymptotic values at a rate $(P/N)^{-1}$. This scaling law is consistent with that found in [102] for real neural networks, where P is replaced by the width of the layers of the network. As for the divergence of the noise and initialization variances observed at the interpolation threshold, figure A.1 (right) shows that they also follow an inverse power law $(P/N - 1)^{-1}$ at vanishing

regularization.

Divide and Conquer approach

As mentioned in the main text, another way to average the predictions of differently initialized learners is the *divide and conquer* approach [84]. In this framework, the data set is divided into K splits of size N/K. Each of the K differently initalized learner is trained on a distinct split. This approach is extremely useful for kernel learning [332], where the computational burden is in the inversion of the Gram matrix which is of size $N \times N$. In the random projection approach considered here, it does not offer any computational gain, however it is interesting how it affects the test error.

Within our framework, the test error can easily be calculated as:

$$\mathbb{E}_{\{B\Theta^{(k)}\},BX,\varepsilon}\left[\mathcal{R}_{\mathrm{RF}}\right] = F^2\left(1-2\Psi_1^v\right)+\frac{1}{K}\left(F^2\Psi_2^v+\tau^2\Psi_3^v\right)+\left(1-\frac{1}{K}\right)F^2\Psi_2^d, \quad \text{(A.1)}$$

where the effective number of data points which enters this formula is $N_{\mathrm{eff}} = N/K$ due to the splitting of the training set.

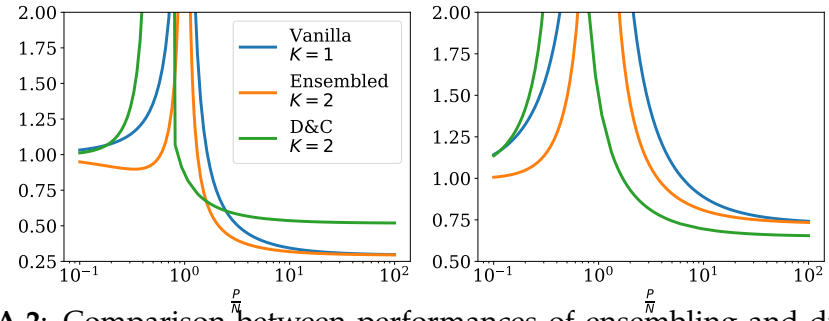


Figure A.2: Comparison between performances of ensembling and divide and conquer for $K = 2$ at different SNR. **Left:** SNR=$\frac{F}{\tau} = 10$. **Right:** SNR=$\frac{F}{\tau} = 1$ Computations performed with fixed $N/D = 1$, $F = 1$ and $\lambda = 10^{-5}$.

Comparing the previous expression with that obtained for ensembling A.30 is instructive: here, increasing K replaces the *vanilla* terms Ψ_2^v, Ψ_3^v by the *divide and conquer* term Ψ_2^d. This shows that divide and conquer has a *denoising* effect: at $K \to \infty$, the effect of the additive noise on the labels is completely suppressed. This was not the case for ensembling. The price to pay is that N_{eff} decreases, hence one is shifted to the underparametrized regime.

In Figure A.2, we see that the kernel limit error of the divide and conquer approach, i.e. the asymptotic value of the error at $P/N \to \infty$, is different from the usual kernel limit error, since the effective dataset is two times smaller at $K = 2$. The denoising effect of the divide and conquer approach is illustrated by the fact

that its kernel limit error is higher at high SNR, but lower at low SNR. This is of practical relevance, and is much related to the beneficial effect of *bagging* in noisy dataset scenarios. The divide and conquer approach, which only differs from bagging by the fact that the different partitions of the dataset are disjoint, was shown to reach bagging-like performance in various setups such as decision trees and neural networks [57].

Is it always better to be overparametrized ?

A common thought is that the double descent curve always reaches its minimum in the over-parametrized regime, leading to the idea that the corresponding model "cannot overfit". In this section, we show that this is not always the case. Three factors tend to shift the optimal generalization to the underparametrized regime: (i) increasing the numbers of learners from which we average the predictions, K, (ii) decreasing the signal-to-noise ratio (SNR), F/τ, and (iii) decreasing the size of the dataset, N/D. In other words, when ensembling on a small, noisy dataset, one is better off using an underparametrized model.

These three effects are shown in Fig. A.3. In the left panel, we see that as we increase K, the minimum of test error jumps to the underparametrized regime $P < N$ for a high enough value of K. In the central/right panels, a similar effect occurs when decreasing the SNR or decreasing N/D.

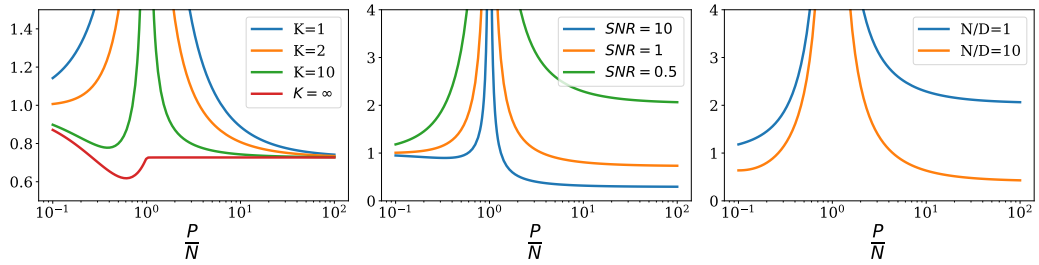


Figure A.3: Generalisation error as a function of P/N: depending on the values of K, F/τ and N/D, optimal generalization can be reached in the underparametrized regime or the overparametrized regime. **Left:** $F/\tau = 1$, $N/D = 1$ and we vary K. This is the same as figure 5 in the main text, except that the higher noise causes the ensembling curve at $K \to \infty$ to exhibit a *dip* in the underparametrized regime. **Center:** $K = 2$, $N/D = 1$ and we vary F/τ. **Right:** $F/\tau = 1$, $K = 2$ and we vary N/D.

A.2 Statement of the Main Result

Assumptions

First, we state precisely the assumptions under which our main result is valid. Note, that these are the same as in [213].

Assumption 1: $\sigma : \mathbb{R} \to \mathbb{R}$ is a weakly differential function with derivative σ'. Assume there exists $c_0, c_1 < \infty \in \mathbb{R}$ such that for all $u \in \mathbb{R}$ $|\sigma(u)|, |\sigma'(u)| \leq c_0 \, \mathrm{e}^{c_1 |u|}$. Then define:

$$\mu_0 = \mathbb{E}\left[\sigma(u)\right] \quad \mu_1 = \mathbb{E}\left[u\sigma(u)\right] \quad \mu_\star^2 = \mathbb{E}\left[\sigma^2(u)\right] - \mu_0^2 - \mu_1^2, \tag{A.2}$$

where the expectation is over $u \sim \mathcal{N}(0,1)$. To facilitate readability, we specialize to the case $\mu_0 = 0$. This simply amounts to a shift ctivation function $\tilde{\sigma}$ of the network, $\tilde{\sigma}(x) = \sigma(x) - \mu_0$.

Assumption 2: We work in the high-dimensional limit, i.e. in the limit where the input dimension D, the hidden layer dimension P and the number of training points N go to infinity with their ratios fixed. That is:

$$N, P, D \to \infty, \qquad \frac{P}{D} \equiv \psi_1 = \mathcal{O}(1), \quad \frac{N}{D} \equiv \psi_2 = \mathcal{O}(1). \tag{A.3}$$

This condition implies that, in the computation of the risk $\mathcal{R}$, we can neglect all the terms of order $\mathcal{O}(1)$ in favour of the terms of order $\mathcal{O}(D)$.

Assumption 3: The labels are given by a linear ground truth, or teacher function:

$$y_\mu = f_d(BX_\mu) + \epsilon_\mu, \quad f_d(Bx) = \langle \boldsymbol{\beta}, Bx \rangle, \quad ||\boldsymbol{\beta}|| = F, \quad \epsilon_\mu \sim \mathcal{N}(0, \tau)\,. \tag{A.4}$$

Note that as explained in [213], it is easy to add a non linear component to the teacher, but the latter would not be captured by the model (the student) in the regime $N/D = \mathcal{O}(1)$, and would simply amount to an extra noise term.

Results

Here we give the explicit form of the quantities appearing in our main result. In these expressions, the index $a \in \{v, e, d\}$ distinguishes the *vanilla*, *ensembling* and

divide and conquer terms.

$$\begin{aligned}
\Psi_1 &= \frac{1}{D}\,\mathrm{Tr}\left[H\left[S^v\right]^{-1} H\left[P_{\Psi_1}\right]\right],\\
\Psi_2^v &= \frac{1}{D}\,\mathrm{Tr}\left[H\left[S^v\right]^{-1} H\left[P_{\Psi_2^v}\right]\right],\\
\Psi_3^v &= \frac{1}{D}\,\mathrm{Tr}\left[H\left[S^v\right]^{-1} H\left[P_{\Psi_3^v}\right]\right],\\
\Psi_2^e &= \frac{1}{D}\,\mathrm{Tr}\left[H\left[S^e\right]^{-1} H\left[P_{\Psi_2^e}\right]\right],\\
\Psi_3^e &= \frac{1}{D}\,\mathrm{Tr}\left[H\left[S^e\right]^{-1} H\left[P_{\Psi_3^e}\right]\right],\\
\Psi_2^d &= \frac{1}{D}\,\mathrm{Tr}\left[H\left[S^d\right]^{-1} H\left[P_{\Psi_2^d}\right]\right],
\end{aligned} \tag{A.5}$$

where the Hessian matrix $H[F]$, for a given function $F : (q, r, \tilde{q}, \tilde{r}) \mapsto \mathbb{R}$ is defined as:

$$H\left[F\right] = @\begin{bmatrix} \frac{\partial F}{\partial q \partial q} & \frac{\partial F}{\partial q \partial r} & \frac{\partial F}{\partial q \partial \tilde{q}} & \frac{\partial F}{\partial q \partial \tilde{r}} \\ \frac{\partial F}{\partial q \partial r} & \frac{\partial F}{\partial r \partial r} & \frac{\partial F}{\partial r \partial \tilde{q}} & \frac{\partial F}{\partial r \partial \tilde{r}} \\ \frac{\partial F}{\partial q \partial \tilde{q}} & \frac{\partial F}{\partial r \partial \tilde{q}} & \frac{\partial F}{\partial \tilde{q} \partial \tilde{q}} & \frac{\partial F}{\partial \tilde{q} \partial \tilde{r}} \\ \frac{\partial F}{\partial q \partial \tilde{r}} & \frac{\partial F}{\partial r \partial \tilde{r}} & \frac{\partial F}{\partial \tilde{q} \partial \tilde{r}} & \frac{\partial F}{\partial \tilde{r} \partial \tilde{r}} \end{bmatrix}_{\substack{q=q^* \\ r=r^* \\ \tilde{r}=0 \\ \tilde{q}=0}},$$

with $q^\star$ and $r^\star$ being the solutions of the fixed point equation for the function $S_0 : (q, r) \mapsto \mathbb{R}$ defined below:

$$\begin{cases} \frac{\partial S_0(q,r)}{\partial q} = 0 \\ \frac{\partial S_0(q,r)}{\partial r} = 0. \end{cases}$$

$$S_0(q,r) = \lambda \psi_1^2 \psi_2 q + \psi_2 \log\left(\frac{\mu_\star^2 \psi_1 q}{\mu_1^2 \psi_1 r + 1} + 1\right) + \frac{r}{q} + (1-\psi_1)\log(q) + \psi_2 \log\left(\mu_1^2 \psi_1 r + 1\right) - \log(r). \tag{A.6}$$

The explicit expression of the above quantities in terms of $(q, r, \tilde{q}, \tilde{r})$ is given below.

Explicit expression of S^v, S^e, S^d

Here we present the explicit formulas for S^v, S^e, S^d, which are defined as the functions $(q, r, \tilde{q}, \tilde{r}) \mapsto \mathbb{R}$ such that:

$$\begin{aligned}
S^v(q,r,\tilde{q},\tilde{r}) &= 2\left(S_0(q,r) + \tilde{q} f^v(q,r) + \tilde{r} g^v(q,r)\right)\\
S^e(q,r,\tilde{q},\tilde{r}) &= S_0(q,r) + \tilde{r}^2 f^e(q,r) + \tilde{q}^2 g^e(q,r)\\
S^d(q,r,\tilde{q},\tilde{r}) &= S_0(q,r) + \tilde{r}^2 f^d(q,r) + \tilde{q}^2 g^d(q,r),
\end{aligned} \tag{A.7}$$

where we defined the functions $(q, r) \mapsto \mathbb{R}$,

$$\begin{aligned}
f^v(q,r) &= \lambda\psi_1^2\psi_2 + \frac{\mu_\star^2\psi_1\psi_2}{\mu_\star^2\psi_1 q + \mu_1^2\psi_1 r + 1} + \frac{1-\psi_1}{q} - \frac{r}{q^2},\\
g^v(q,r) &= -\frac{\mu_\star^2\mu_1^2\psi_1^2\psi_2 q}{\left(\mu_1^2\psi_1 r + 1\right)\left(\mu_\star^2\psi_1 q + \mu_1^2\psi_1 r + 1\right)} + \frac{\mu_1^2\psi_1\psi_2}{\mu_1^2\psi_1 r + 1} + \frac{1}{q} - \frac{1}{r},\\
f^e(q,r) &= \frac{2r\mu_1^2\psi_1\left(1 + q\mu_\star^2\psi_1\right) + \left(1 + q\mu_\star^2\psi_1\right)^2 - r^2\mu_1^4\psi_1^2(-1+\psi_2)}{r^2\left(1 + r\mu_1^2\psi_1 + q\mu_\star^2\psi_1\right)^2},\\
g^e(q,r) &= \frac{\psi_1}{q^2}, \quad f^d(q,r) = \frac{1}{r^2}, \quad g^d(q,r) = \frac{\psi_1}{q^2}.
\end{aligned} \tag{A.8}$$

Explicit expression of $P_{\Psi_1}, P^v_{\Psi_2}, P^v_{\Psi_3}, P^e_{\Psi_2}, P^e_{\Psi_3}, P^d_{\Psi_2}$

Here we present the explicit formulas for $P_{\Psi_1}, P^v_{\Psi_2}, P^v_{\Psi_3}, P^e_{\Psi_2}, P^e_{\Psi_3}, P^d_{\Psi_2}$, which are defined as the functions $(q, r, \tilde{q}, \tilde{r}) \mapsto \mathbb{R}$ such that:

$$\begin{aligned}
P_{\Psi_1} &= \psi_1\psi_2\mu_1^2\left(M_X^{11} + \mu_1^2\mu_\star^2\psi_1^2\left(M_X M_W^v M_X\right)^{11} + \mu_\star^2\psi_1\left(M_X M_W^v\right)^{11}\right), \\
P_{\Psi_2^v} &= D\psi_1^2\psi_2(\mu_1^2\tilde{r} + \mu_\star^2\tilde{q})\left[\mu_1^2 P^v_{XX} - 2\mu_1^2\mu_\star^2\psi_1 P^v_{WX} + \mu_\star^2 P^v_{WW}\right], \\
P_{\Psi_3^v} &= D\psi_1^2\psi_2(\mu_1^2\tilde{r} + \mu_\star^2\tilde{q})\left[\mu_1^2\left(M_X^{12} + \mu_1^2\mu_\star^2\psi_1^2\left[M_X M_W^v M_X\right]^{12}\right) - 2\mu_1^2\mu_\star^2\psi_1\left[M_X M_W^v\right]^{12} + \mu_\star^2\left[M_W^v\right]^{12}\right], \\
P_{\Psi_2^e} &= D\psi_1^2\psi_2\mu_1^2\tilde{r}\left[\mu_1^2 P^e_{XX} - 2\mu_1^2\mu_\star^2\psi_1 P^e_{WX} + \mu_\star^2 P^e_{WW}\right], \\
P_{\Psi_3^e} &= D\psi_1^2\psi_2\mu_1^2\tilde{r}\left[\mu_1^2\left(M_X^{12} + \mu_1^2\mu_\star^2\psi_1^2\left[M_X M_W^e M_X\right]^{12}\right) - 2\mu_1^2\mu_\star^2\psi_1\left[M_X M_W^e\right]^{12} + \mu_\star^2[M_W^e]^{12}\right], \\
P_{\Psi_2^d} &= D\mu_1^2\psi_1\psi_2^2\tilde{r}\left[\psi_1\mu_1^2 P_{XX} + 2\mu_\star^2\mu_1^2\psi_1^2 P_{WX} + \mu_\star^2\psi_1 P_{WW}\right],
\end{aligned} \tag{A.9}$$

where we defined the scalars P_{XX}, P_{WX}, P_{WW} as follows:

$$\begin{aligned}
P^v_{XX} &= \psi_2 N_X^{12} + M_X^{12} + 2\psi_2(\mu_1\mu_\star\psi_1)^2\left[M_X N_X M_W^a\right]^{12} + (\mu_1\mu_\star\psi_1)^2\left[M_X M_W^a M_X\right]^{12} \\
&\quad + \psi_2(\mu_1\mu_\star\psi_1)^4\left[M_X M_W N_X M_W^a M_X\right]^{12}, \\
P^v_{WX} &= \psi_2\left[N_X M_W^a\right]^{12} + \left[M_X M_W^a\right]^{12} + \psi_2(\mu_1\mu_\star\psi_1)^2\left[M_X M_W^a N_X M_W^a\right]^{12}, \\
P^v_{WW} &= \left[M_W^a\right]^{12} + \psi_2(\mu_1\mu_\star\psi_1)^2\left[M_W^a N_X M_W^a\right]^{12}, \\
P^e_{XX} &= P^v_{XX}, \\
P^e_{WX} &= P^v_{WX}, \\
P^e_{WW} &= P^v_{WW}, \\
P^d_{XX} &= \left(N_X^{d11} + 2(\mu_1\mu_\star\psi_1)^2\left[N_X^d M_W^d M_X^d\right]^{11} + (\mu_1\mu_\star\psi_1)^4\left[M_X^d M_W^d N_X^d M_W^d M_X^d\right]^{11}\right), \\
P^d_{WX} &= \left[N_X^d M_W^d\right]^{11} + (\mu_1\mu_\star\psi_1)^2\left[M_X^d M_W^d N_X^d M_W^d\right]^{11}, \\
P^d_{WW} &= (\mu_1\mu_\star\psi_1)^2\left[M_W^d N_X^d M_W^d\right]^{11},
\end{aligned}$$

and the 2×2 matrices M_X, M_W, N_X as follows:

$$M_X^v = \begin{bmatrix} \frac{r}{1+\mu_1^2\psi_1 r} & \frac{\tilde{r}}{\left(1+\mu_1^2\psi_1 r\right)^2} \\ \frac{\tilde{r}}{\left(1+\mu_1^2\psi_1 r\right)^2} & \frac{r}{1+\mu_1^2\psi_1 r} \end{bmatrix}, /; M_W^v = \begin{bmatrix} \frac{q\left(1+\mu_1^2\psi_1 r\right)}{1+2\mu_1^2\psi_1 r+\mu_\star^2\psi_1 q} & \frac{q^2\mu_1^2\mu_\star^2\psi_1^2\tilde{r}}{\left(1+\mu_1^2\psi_1 r+\mu_\star^2\psi_1 q\right)^2} \\ \frac{q^2\mu_1^2\mu_\star^2\psi_1^2\tilde{r}}{\left(1+\mu_1^2\psi_1 r+\mu_\star^2\psi_1 q\right)^2} & \frac{q\left(1+\mu_1^2\psi_1 r\right)}{1+2\mu_1^2\psi_1 r+\mu_\star^2\psi_1 q} \end{bmatrix},$$

$$N_X^v = \frac{1}{\left(1+\mu_1^2\psi_1 r\right)^2}\begin{bmatrix} r & \tilde{r} \\ \tilde{r} & r \end{bmatrix}, \tag{A.10}$$

$$M_X^e = M_X^v, /; M_W^e = M_W^v + \frac{(1+r\mu_1^2\psi_1)^2\tilde{q}}{\left(1+\mu_1^2\psi_1 r+\mu_\star^2\psi_1 q\right)^2}\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, /; N_X^e = \frac{1}{\left(1+\mu_1^2\psi_1 r\right)^2}\begin{bmatrix} r & \tilde{r} \\ \tilde{r} & r \end{bmatrix},$$

$$M_X^d = \frac{r}{1+\mu_1^2\psi_1 r^2}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, /; M_W^d = \frac{q(1+\mu_1^2\psi_1 r)}{1+\mu_1^2\psi_1 r+\mu_\star^2\psi_1 q}\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}, /; N_X^d = \frac{1}{(1+\mu_1^2\psi_1 r)^2}\begin{bmatrix} \tilde{r} & 0 \\ 0 & \tilde{r} \end{bmatrix}.$$

A.3 Replica Computation

Toolkit

Gaussian integrals In order to obtain the main result for the generalisation error, we perform the averages over all the sources of randomness in the system in the following order: over the dataset X, then over the noise W, and finally over the random feature layers Θ. Here are some useful formulaes used throughout the computations:

$$\begin{cases} \int e^{-\frac{1}{2}x_i G_{ij} x_j + J_i x_i} dx & = (\det G)^{-\frac{1}{2}} e^{\frac{1}{2}J_i G_{ij}^{-1} J_j}, \\ \int x_a e^{-\frac{1}{2}x_i G_{ij} x_j + J_i x_i} dx & = P_a^1 (\det G)^{-\frac{1}{2}} e^{\frac{1}{2}J_i G_{ij}^{-1} J_j}, \\ \int x_a x_b e^{-\frac{1}{2}x_i G_{ij} x_j + J_i x_i} dx & = P_{ab}^2 (\det G)^{-\frac{1}{2}} e^{\frac{1}{2}J_i G_{ij}^{-1} J_j}, \\ \int x_a x_b x_c e^{-\frac{1}{2}x_i G_{ij} x_j + J_i x_i} dx & = P_{abc}^3 (\det G)^{-\frac{1}{2}} e^{\frac{1}{2}J_i G_{ij}^{-1} J_j}, \\ \int x_a x_b x_c x_d e^{-\frac{1}{2}x_i G_{ij} x_j + J_i x_i} dx & = P_{abcd}^4 (\det G)^{-\frac{1}{2}} e^{\frac{1}{2}J_i G_{ij}^{-1} J_j}, \end{cases} \tag{A.11}$$

with

$$\begin{aligned} P_a^1 &= [G^{-1}J]_a, \\ P_{ab}^2 &= ((G^{-1})_{ab} + [G^{-1}J]_a[G^{-1}J]_b), \\ P_{abc}^3 &= \sum_{\bar{a},\bar{b},\bar{c} \in perm(abc)} \left((G^{-1})_{\bar{a}\bar{b}}[G^{-1}J]_{\bar{c}} + [G^{-1}J]_{\bar{a}}[G^{-1}J]_{\bar{b}}[G^{-1}J]_{\bar{c}}\right), \\ P_{abcd}^4 &= \sum_{\substack{\bar{a},\bar{b} \\ \bar{c},\bar{d}} \in perm(abcd)} \left((G^{-1})_{\bar{a}\bar{b}}(G^{-1})_{\bar{c}\bar{d}} + [G^{-1}J]_{\bar{a}}[G^{-1}J]_{\bar{b}}[G^{-1}J]_{\bar{c}}[G^{-1}J]_{\bar{d}} + (G^{-1})_{\bar{a}\bar{b}}[G^{-1}J]_{\bar{c}}[G^{-1}J]_{\bar{d})}\right). \end{aligned} \tag{A.12}$$

Replica representation of an inverse matrix To obtain gaussian integrals we will use the

"replica" representation the element (ij) of a matrix M of size D:

$$M_{ij}^{-1} = \lim_{n\to 0} \int \left(\prod_{\alpha=1}^{n} \prod_{i=1}^{D} d\eta_i^\alpha \right) \eta_i^1 \eta_j^1 \exp\left(-\frac{1}{2} \eta_i^\alpha M_{ij} \eta_j^\alpha \right). \tag{A.13}$$

Indeed, using the gaussian integral representation of the inverse of M,

$$\begin{aligned} M_{ij}^{-1} &= \mathcal{B}\mathcal{Z}^{-1} \int \left(\prod_{i=1}^{D} d\eta_i \right) \eta_i \eta_j \exp\left(-\frac{1}{2} \eta_i M_{ij} \eta_j \right), \\ \mathcal{Z} &= \sqrt{\frac{(2\pi)^D}{\det M}} \\ &= \int \left(\prod_{i=1}^{D} d\eta_i \right) \exp\left(-\frac{1}{2} \eta_i M_{ij} \eta_j \right). \end{aligned} \tag{A.14}$$

Using the replica identity, we rewrite this as

$$M_{ij}^{-1} = \lim_{n\to 0} \mathcal{Z}^{n-1} \int \left(\prod_{i=1}^{D} d\eta_i \right) \eta_i \eta_j \exp\left(-\frac{1}{2} \eta_i M_{ij} \eta_j \right). \tag{A.15}$$

Renaming the integration variable of the integral on the left as η^1 and the $n-1$ others as $\eta^\alpha, \alpha \in \{2, n\}$, we obtain expression A.13.

The Random Feature model

In what follows, we will explicitly leave the indices of all the quantities used. We use the notation, called Einstein summation convention in physics, in which all repeated indices are summed but the sum is not explicitly written. Indices $i \in \{1...D\}$ are used to refer to the input dimension, $h \in \{1...P\}$ to refer to the hidden layer dimension and $\mu \in \{1...N\}$ to refer to the number of data points. *With a single learner* In the random features model, the predictor can be computed explicitly:

$$\hat{a} = \frac{1}{\sqrt{D}} B y_\mu^{\mathrm{T}} B Z_{\mu h} \left(B Z^{\mathrm{T}} B Z + \psi_1 \psi_2 \lambda \mathbf{I}_N \right)^{-1}_{hh'} \tag{A.16}$$

$$\begin{aligned} f(Bx) &= \hat{a}_h \sigma\left(\frac{B\Theta_{h'i} B x_i}{\sqrt{D}} \right) \\ &= B y_\mu^{\mathrm{T}} B Z_{\mu h} \left(B Z^{\mathrm{T}} B Z + \psi_1 \psi_2 \lambda \mathbf{I}_N \right)^{-1}_{hh'} \sigma\left(\frac{B\Theta_{h'i} B x_i}{\sqrt{D}} \right) / \sqrt{D}, \end{aligned} \tag{A.17}$$

where

$$By_\mu = f_d\left(BX_\mu\right) + B\epsilon_\mu, \tag{A.18}$$

$$BZ_{\mu h} = \frac{1}{\sqrt{D}}\sigma\left(\frac{1}{\sqrt{D}}B\Theta_{hi}BX_{\mu i}\right). \tag{A.19}$$

Hence the test error can be computed as:

$$\begin{aligned}\mathcal{R}_{\mathrm{RF}} =& \mathbb{E}_{Bx}\left[\left(f_d(Bx) - By_\mu^{\mathrm{T}}BZ_{\mu h}\left(BZ^{\mathrm{T}}BZ + \psi_1\psi_2\lambda\mathbf{I}_N\right)^{-1}_{hh'}\sigma\left(\frac{B\Theta_{h'i}Bx_i}{\sqrt{D}}\right)/\sqrt{D}\right)^2\right] \\ =& \mathbb{E}_{Bx}\left[f_d(Bx)^2\right] - 2By_\mu^{\top}BZ_{\mu h}\left(BZ^{\mathrm{T}}BZ + \psi_1\psi_2\lambda\mathbf{I}_N\right)^{-1}_{hh'}BV_{h'}/\sqrt{D} \\ &+ By_\mu^{\mathrm{T}}BZ_{\mu h}\left(BZ^{\mathrm{T}}BZ + \psi_1\psi_2\lambda\mathbf{I}_N\right)^{-1}_{hh_1}BU_{h_1h_2}\left(BZ^{\mathrm{T}}BZ + \psi_1\psi_2\lambda\mathbf{I}_N\right)^{-1}_{h_2h'}BZ^{\mathrm{T}}_{h'\mu'}By_{\mu'}/D,\end{aligned} \tag{A.20}$$

where

$$BV_h = \mathbb{E}_{Bx}\left[f_d(Bx)\sigma\left(\frac{\langle B\Theta_{hi}Bx_i\rangle}{\sqrt{D}}\right)\right], \tag{A.21}$$

$$BU_{hh'} = \mathbb{E}_{Bx}\left[\sigma\left(\frac{\langle B\Theta_{hi}Bx_i\rangle}{\sqrt{D}}\right)\sigma\left(\frac{B\Theta_{h'i},Bx_i}{\sqrt{D}}\right)\right]. \tag{A.22}$$

Ensembling over K learners When ensembling over K learners with independently sampled random feature vectors, the predictor becomes:

$$f(Bx) = \frac{1}{K\sqrt{D}}\sum_k By_\mu^{\mathrm{T}}BZ^{(k)}_{\mu h}\left(BZ^{\mathrm{T}(k)}BZ^{(k)} + \psi_1\psi_2\lambda\mathbf{I}_N\right)^{-1}_{hh'}\sigma\left(\frac{B\Theta^{(k)}_{h'i}Bx_i}{\sqrt{D}}\right), \tag{A.23}$$

where

$$BZ^{(k)}_{\mu h} = \frac{1}{\sqrt{D}}\sigma\left(\frac{1}{\sqrt{D}}B\Theta^{(k)}_{hi}BX_{\mu i}\right). \tag{A.24}$$

The generalisation error is then given by:

$$\begin{aligned}\mathcal{R}_{\mathrm{RF}} =& \mathbb{E}_{Bx}\left[\left(f_d(Bx) - \frac{1}{K}\sum_k By^{\mathrm{T}}BZ^{(k)}\left(BZ^{\mathrm{T}(k)}BZ^{(k)} + \psi_1\psi_2\lambda\mathbf{I}_N\right)^{-1}\sigma\left(\frac{B\Theta^{(k)}_{h'i}Bx_i}{\sqrt{D}}\right)/\sqrt{D}\right)^2\right] \\ =& \mathbb{E}_{Bx}\left[f_d(Bx)^2\right] - \frac{2}{K}\sum_k By^{\top}BZ^{(k)}\left(BZ^{\mathrm{T}(k)}BZ^{(k)} + \psi_1\psi_2\lambda\mathbf{I}_N\right)^{-1}\frac{BV^{(k)}}{\sqrt{D}} \\ &+ \frac{1}{K^2}\sum_{\substack{k\\ l\neq k}} By^{\mathrm{T}}BZ^{(k)}\left(BZ^{\mathrm{T}(k)}BZ^{(k)} + \psi_1\psi_2\lambda\mathbf{I}_N\right)^{-1}BU^{(kl)} \\ &\left(BZ^{\mathrm{T}(l)}BZ^{(l)} + \psi_1\psi_2\lambda\mathbf{I}_N\right)^{-1}BZ^{(l)}\mathrm{T}By/D,\end{aligned} \tag{A.25}$$

where

$$BV_h^{(k)} = \mathbb{E}_{Bx}\left[f_d(Bx)\sigma\left(\frac{\langle B\Theta_{hi}^{(k)} Bx_i\rangle}{\sqrt{D}}\right)\right], \tag{A.26}$$

$$BU_{hh'}^{(kl)} = \mathbb{E}_{Bx}\left[\sigma\left(\frac{\langle B\Theta_{hi}^{(k)} Bx_i\rangle}{\sqrt{D}}\right)\sigma\left(\frac{\langle B\Theta_{h'i'}^{(l)} Bx_{i'}\rangle}{\sqrt{D}}\right)\right]. \tag{A.27}$$

Equivalent Gaussian Covariate Model It was shown in [213] that the random features model is equivalent, in the high-dimensional limit of Assumption 2, to a Gaussian covariate model in which the activation function σ is replaced as:

$$\sigma\left(\frac{B\Theta_{hi}^{(k)} BX_{\mu i}}{\sqrt{D}}\right) \to \mu_0 + \mu_1 \frac{B\Theta_{hi}^{(k)} BX_{\mu i}}{\sqrt{D}} + \mu_\star BW_{\mu h}^{(k)}, \tag{A.28}$$

with $BW^{(k)} \in \mathbb{R}^{N\times P}$, $W_{\mu h}^{(k)} \sim \mathcal{N}(0,1)$ and μ_0, μ_1 and $\mu_\star$ defined in A.2. To simplify the calculations, we take $\mu_0 = 0$, which amounts to adding a constant term to the activation function σ.

This powerful mapping allows to express the quantities BU, BV. We will not repeat their calculations here: the only difference here is BU^{kl}, which carries extra indices k, l due to the different initialization of the random features $B\Theta^{(k)}$. In our case,

$$\boldsymbol{U}_{hh'}^{(kl)} = \frac{\mu_1^2}{D}\boldsymbol{\Theta}_{hi}^{(k)}\boldsymbol{\Theta}_{h'i}^{(l)} + \mu_\star^2 \delta_{kl}\delta_{hh'}. \tag{A.29}$$

Hence we can rewrite the test error as

$$\mathop{\mathbb{E}}_{\substack{\{B\Theta^{(k)}\}\\ BX,\boldsymbol{\varepsilon}}}[\mathcal{R}_{\mathrm{RF}}] = F^2\left(1 - 2\Psi_1^v\right) + \frac{1}{K}\left(F^2\Psi_2^v + \tau^2\Psi_3^v\right) + \left(1 - \frac{1}{K}\right)\left(F^2\Psi_2^e + \tau^2\Psi_3^e\right), \tag{A.30}$$

where Ψ_1,Ψ_2^v,Ψ_2^e,Ψ_3^v,Ψ_3^e are given by:

$$\begin{aligned}
\Psi_1 &= \frac{1}{D}\,\mathrm{Tr}\left[\left(\frac{\mu_1}{D}\boldsymbol{X}B\Theta^{(1)\top}\right)^\top \boldsymbol{Z^{(1)}}\left(\boldsymbol{Z^{(1)\top}Z^{(1)}}+\psi_1\psi_2\lambda\mathbf{I}_N\right)^{-1}\right],\\
\Psi_2^v &= \frac{1}{D}\,\mathrm{Tr}\left[\left(\boldsymbol{Z^{(1)\top}Z^{(1)}}+\psi_1\psi_2\lambda\mathbf{I}_N\right)^{-1}\left(\frac{\mu_1^2}{D}B\Theta^{(1)}B\Theta^{(1)\top}+\mu_*^2\mathbf{I}_N\right)\right.\\
&\qquad\left.\left(\boldsymbol{Z^{(1)\top}Z^{(1)}}+\psi_1\psi_2\lambda\mathbf{I}_N\right)^{-1}\boldsymbol{Z^{(1)\top}}\left(\frac{1}{D}\boldsymbol{XX}^\top\right)\boldsymbol{Z^{(1)}}\right],\\
\Psi_3^v &= \frac{1}{D}\,\mathrm{Tr}\left[\left(\boldsymbol{Z^{(1)\top}Z^{(1)}}+\psi_1\psi_2\lambda\mathbf{I}_N\right)^{-1}\left(\frac{\mu_1^2}{D}B\Theta^{(1)}B\Theta^{(1)\top}+\mu_\star^2\mathbf{I}_N\right)\right.\\
&\qquad\left.\left(\boldsymbol{Z^{(1)\top}Z^{(1)}}+\psi_1\psi_2\lambda\mathbf{I}_N\right)^{-1}\boldsymbol{Z^{(1)\top}Z^{(1)}}\right],\\
\Psi_2^e &= \frac{1}{D}\,\mathrm{Tr}\left[\left(\boldsymbol{Z^{(1)\top}Z^{(1)}}+\psi_1\psi_2\lambda\mathbf{I}_N\right)^{-1}\left(\frac{\mu_1^2}{D}B\Theta^{(1)}B\Theta^{(2)\top}\right)\right.\\
&\qquad\left.\left(\boldsymbol{Z^{(2)\top}Z^{(2)}}+\psi_1\psi_2\lambda\mathbf{I}_N\right)^{-1}\boldsymbol{Z^{(2)\top}}\left(\frac{1}{D}\boldsymbol{XX}^\top\right)\boldsymbol{Z^{(1)}}\right],\\
\Psi_3^e &= \frac{1}{D}\,\mathrm{Tr}\left[\left(\boldsymbol{Z^{(1)\top}Z^{(1)}}+\psi_1\psi_2\lambda\mathbf{I}_N\right)^{-1}\left(\frac{\mu_1^2}{D}B\Theta^{(1)}B\Theta^{(2)\top}\right)\right.\\
&\qquad\left.\left(\boldsymbol{Z^{(2)\top}Z^{(2)}}+\psi_1\psi_2\lambda\mathbf{I}_N\right)^{-1}\boldsymbol{Z^{(2)\top}Z^{(1)}}\right].
\end{aligned}\tag{A.31}$$

Computation of the *vanilla* terms

To start with, let us compute the vanilla terms (those who carry a superscript v), which involve a single instance of the random feature vectors. Note that these were calculated in [213] by evaluating the Stieljes transform of the random matrices of which we need to calculate the trace. The replica method used here makes the calculation of the vanilla terms carry over easily to the the ensembling terms (superscript e) and the divide and conquer term (superscript d). To illustrate the calculation steps, we will calculate Ψ_3^v, then provide the results for Ψ_2^v and Ψ_1.

In the vanilla terms, the two inverse matrices that appear are the same. Hence we use twice the replica identity A.13, introducing $2n$ replicas which all play the same role:

$$M_{ij}^{-1}M_{kl}^{-1} = \lim_{n\to 0}\int\left(\prod_{\alpha=1}^{2n}d\eta\right)\eta_i^1\eta_j^1\eta_k^2\eta_l^2\exp\left(-\frac{1}{2}\eta^\alpha M_{ij}\eta^\alpha\right).\tag{A.32}$$

The first step is to perform the averages, i.e. the Gaussian integrals, over the dataset X, the deterministic noise W induced by the non-linearity of the activation function and the random features Θ. *Averaging over the dataset* Replacing the activation function by its Gaussian covariate equivalent model and using A.32, the term Ψ_3

can be expanded as:

$$\Psi_3^v = \frac{1}{D}\left[BZ_{\mu h}\left(BZ^\top BZ + \psi_1\psi_2\lambda \mathbf{I}_N\right)^{-1}_{hh_1}\left(\frac{\mu_1^2}{D}B\Theta_{h_1 i}B\Theta_{h_2 i} + \mu_\star^2\delta_{h_1h_2}\right)\left(BZ^\top BZ + \psi_1\psi_2\lambda \mathbf{I}_N\right)^{-1}_{h_2h'}BZ_{h'\mu}\right]$$

(A.33)

$$= \frac{1}{D}\left(\frac{\mu_1^2}{D}B\Theta_{h_1 i}B\Theta_{h_2 i} + \mu_\star^2\delta_{h_1h_2}\right)\left[BZ_{\mu h}BZ_{h'\mu}\right] \tag{A.34}$$
$$\int\left(\prod_{\alpha=1}^{2n}d\eta^\alpha\right)\eta_h^1\eta_{h_1}^1\eta_{h_2}^2\eta_{h'}^2\exp\left(-\frac{1}{2}\eta_h^\alpha\left(BZ^\top BZ + \psi_1\psi_2\lambda\mathbf{I}_N\right)_{hh'}\eta_{h'}^\alpha\right)$$
$$= \frac{1}{D^2}\left(\frac{\mu_1^2}{D}B\Theta_{h_1 i}B\Theta_{h_2 i} + \mu_\star^2\delta_{h_1h_2}\right)\left(\frac{\mu_1}{\sqrt{D}}B\Theta BX + \mu_\star BW\right)_{h\mu}\left(\frac{\mu_1}{\sqrt{D}}B\Theta BX + \mu_\star BW\right)_{h'\mu}$$
$$\int\left(\prod_{\alpha=1}^{2n}d\eta^\alpha\right)\eta_h^1\eta_{h_1}^1\eta_{h_2}^2\eta_{h'}^2\exp\left(-\frac{1}{2}\eta_h^\alpha\left(\frac{1}{D}\left(\frac{\mu_1}{\sqrt{D}}B\Theta BX + \mu_\star BW\right)_{h\mu}\left(\frac{\mu_1}{\sqrt{D}}B\Theta BX + \mu_\star BW\right)_{h'\mu}\right.\right.$$
$$\left.\left.+\psi_1\psi_2\lambda\delta_{hh'}\right)\eta_{h'}^\alpha\right).$$

Now, we introduce $\lambda_i^\alpha := \frac{1}{\sqrt{P}}\eta_h^\alpha B\Theta_{hi}$, and enforce this relation using the Fourier representation of the delta-function:

$$1 = \int d\lambda_i^\alpha d\hat{\lambda}_i^\alpha e^{i\hat{\lambda}_i^\alpha(\sqrt{P}\lambda_i^\alpha - \eta_h^\alpha B\Theta_{hi})}. \tag{A.35}$$

The average over the dataset $BX_{\mu i}$ has the form of A.11 with:

$$(G_X)_{\mu\mu',ii'} = \delta_{\mu\mu'}\left(\delta_{ii'} + \frac{\mu_1^2\psi_1}{D}\lambda_i^\alpha\lambda_{i'}^\alpha\right), \tag{A.36}$$

$$(J_X)_{\mu,i} = \frac{\mu_1\mu_\star\sqrt{\psi_1}}{D}\sum_{\alpha a}\lambda_i^\alpha\eta_h^\alpha BW_{\mu h}. \tag{A.37}$$

Using formulae A.11, we obtain:

$$\begin{aligned}\Psi_3^v =& \frac{N}{D^2}\int\left(\prod_{\alpha=1}^{2n} d\eta^\alpha d\lambda^\alpha d\hat{\lambda}^\alpha\right)\left(\frac{\mu_1^2 P}{D}\lambda_i^1\lambda_i^2+\mu_\star^2\eta_i^1\eta_i^2\right)\\ &\left[\mu_\star^2\eta_h^1 BW_h BW_{h'}\eta_{h'}^2+\mu_1^2\psi_1\lambda_i^1\lambda_{i'}^2\left((G_X^{-1})_{ii'}+(G_X^{-1}J_X)_i(G_X^{-1}J_X)_{i'}\right)\right.\\ &\left.+2\mu_1\mu_\star\sqrt{\psi_1}\lambda_i^1 BW_h\eta_h^2(G_X^{-1}J_X)_i\right]\\ &\exp\left(-\frac{n}{2}\log\det(G_X)-\frac{1}{2}\eta_h^\alpha\left(\frac{\mu_\star^2}{D}BW_h BW_{h'}\delta_{\alpha\beta}-\frac{\mu_1^2\mu_\star^2\psi_1}{D^2}BW_h\lambda_i^\alpha(G_X)^{-1}_{ii'}\lambda_{i'}^\beta BW_{h'}+\psi_1\psi_2\lambda\delta_{hh'}\right)\eta_{h'}^\beta\right.\\ &\left.+i\hat{\lambda}_i^\alpha(\sqrt{P}\lambda_i^\alpha-\eta_h^\alpha B\Theta_{hi})\right)\end{aligned} \tag{A.38}$$

Note that due to with a slight abuse of notation we got rid of indices μ, which all sum up trivially to give a global factor N. *Averaging over the deterministic noise* The expectation over the deterministic noise BW_h is a Gausssian integral of the form A.11 with:

$$[G_W]_{hh'} = \delta_{hh'}+\frac{\mu_\star^2}{D}\eta_h^\alpha A^{\alpha\beta}\eta_{h'}^\beta, \tag{A.39}$$

$$[J_W]_h = 0, \tag{A.40}$$

$$A^{\alpha\beta} = \delta_{\alpha\beta}-\mu_1^2\psi_1\frac{1}{D}\sum_{i,j}\lambda_i^\alpha[G_X^{-1}]_{ij}\lambda_j^\beta. \tag{A.41}$$

Note that the prefactor involves, constant, linear and quadratic terms in BW since:

$$(G_X^{-1}J_X)_i = \frac{\mu_1\mu_\star\sqrt{\psi_1}}{D}\left[\eta^\alpha BW\right]\left[G_X^{-1}\lambda^\alpha\right]_i. \tag{A.42}$$

Thus, one obtains:

$$\begin{aligned}\Psi_3^v =& \frac{\psi_2}{D}\int\left(\prod_{\alpha=1}^{2n} d\eta^\alpha d\lambda^\alpha d\hat{\lambda}^\alpha\right)\left(\mu_1^2\psi_1\lambda_i^1\lambda_i^2+\mu_\star^2\eta_i^1\eta_i^2\right)\\ &\left[\mu_\star^2\left[\eta^1(G_W^{-1})\eta^2\right]+\mu_1^2\psi_1\left[\lambda^1 H_W\lambda^2\right]+2\mu_1^2\mu_\star^2\psi_1[\lambda^1 S_W\eta^2]\right]\\ &\exp\left(-\frac{n}{2}\log\det(G_X)-\frac{n}{2}\log\det(G_W)-\frac{1}{2}\psi_1\psi_2\lambda\sum(\eta_h^\alpha)^2+i\hat{\lambda}_i^\alpha(\sqrt{P}\lambda_i^\alpha-\eta_h^\alpha B\Theta_{hi})\right),\end{aligned}$$

with

$$(H_W)_{ij} = (G_X^{-1})_{ij}+\frac{\mu_1^2\mu_\star^2\psi_1}{D^2}\left[\eta^\alpha(G_W^{-1})\eta^\beta\right]\left[G_X^{-1}\lambda^\alpha\right]_i\left[G_X^{-1}\lambda^\beta\right]_j, \tag{A.43}$$

$$(S_W)_{ih} = \frac{1}{D}\left[G_X^{-1}\lambda^\alpha\right]_i\left[G_W^{-1}\eta^\alpha\right]_h. \tag{A.44}$$

Averaging over the random feature vectors The expectation over the random feature vectors $B\Theta_{hi}$ is a Gausssian integral of the form A.11 with:

$$[G_\Theta]_{hh',ii'} = \delta_{hh',ii'}, \tag{A.45}$$
$$[J_\Theta]_{hi} = -i\hat{\lambda}_i^\alpha \eta_h^\alpha. \tag{A.46}$$

Preforming this integration results in:

$$\begin{aligned}\Psi_3^v = \frac{\psi_2}{D}\int \left(\prod_{\alpha=1}^{2n} d\eta^\alpha d\lambda^\alpha d\hat{\lambda}^\alpha\right)\left(\mu_1^2\psi_1\lambda_i^1\lambda_i^2 + \mu_\star^2\eta_i^1\eta_i^2\right)\\ \left[\mu_\star^2\left[\eta^1(G_W^{-1})\eta^2\right] + \mu_1^2\psi_1\left[\lambda^1 H_W\lambda^2\right] + 2\mu_1^2\mu_\star^2\psi_1[\lambda^1 S_W\eta^2]\right]\\ \exp\left(-\frac{n}{2}\log\det(G_X) - \frac{n}{2}\log\det(G_W) - \frac{1}{2}\psi_1\psi_2\lambda\sum(\eta_h^\alpha)^2 - \frac{1}{2}\eta_h^\alpha\eta_h^\beta\hat{\lambda}_i^\alpha\hat{\lambda}_i^\beta + i\sqrt{P}\hat{\lambda}_i^\alpha\lambda_i^\alpha\right).\end{aligned} \tag{A.47}$$

Expression of the action and the prefactor To complete the computation we integrate with respect to $\hat{\lambda}_i^\alpha$, using again formulae A.11:

$$\left[G_{\hat{\lambda}}\right]_{ii'}^{\alpha\beta} = \delta^{ii'}\eta_h^\alpha\eta_h^\beta, \tag{A.48}$$
$$\left[J_{\hat{\lambda}}\right]_i^\alpha = i\sqrt{P}\lambda_i^\alpha. \tag{A.49}$$

This yields the final expression of the term:

$$\begin{aligned}\Psi_3^v = \frac{\psi_2}{D}\int \left(\prod_{\alpha=1}^{2n} d\eta^\alpha\right)\left(\prod_{\alpha=1}^{2n} d\lambda^\alpha\right)\left(\mu_1^2\psi_1\lambda_i^1\lambda_i^2 + \mu_\star^2\eta_i^1\eta_i^2\right)\left[\mu_\star^2\left[\eta^1(G_W^{-1})\eta^2\right] + \mu_1^2\psi_1\right.\\ \left.\left[\lambda^1 H_W\lambda^2\right] + 2\mu_1^2\mu_\star^2\psi_1[\lambda^1 S_W\eta^2]\right]\\ \exp\left(-\frac{n}{2}\log\det(G_X) - \frac{n}{2}\log\det(G_W) - \frac{D}{2}\log\det\left(G_{\hat{\lambda}}\right) - \frac{1}{2}\psi_1\psi_2\lambda\sum(\eta_h^\alpha)^2 - \frac{P}{2}\lambda_i^\alpha(G_{\hat{\lambda}}^{-1})_{ii'}^{\alpha\beta}\lambda_{i'}^\beta\right).\end{aligned}$$

The above may be written as

$$\Psi_3^v = \int \left(\prod d\eta\right)\left(\prod d\lambda\right) P_{\Psi_3^v}[\eta,\lambda]\exp\left(-\frac{D}{2}S^v[\eta,\lambda]\right), \tag{A.50}$$

with the *prefactor* $P_{\Psi_3^v}$ and the *action* S^v defined as:

$$\begin{aligned}P_{\Psi_3^v}[\eta,\lambda] :=& \frac{\psi_2}{D}\left(\mu_1^2\psi_1\lambda_i^1\lambda_i^2 + \mu_\star^2\eta_i^1\eta_i^2\right)\left[\mu_\star^2\left[\eta^1(G_W^{-1})\eta^2\right] + \mu_1^2\psi_1\left[\lambda^1 H_W\lambda^2\right] + 2\mu_1^2\mu_\star^2\psi_1[\lambda^1 S_W\eta^2]\right],\\ S^v[\eta,\lambda] :=& \psi_2\log\det(G_X) + \psi_2\log\det(G_W) + \log\det\left(G_{\hat{\lambda}}\right) + \frac{1}{D}\psi_1\psi_2\lambda\sum(\eta_h^\alpha)^2 + \frac{P}{D}\left(\lambda_i^\alpha(G_{\hat{\lambda}}^{-1})_{ii'}^{\alpha\beta}\lambda_{i'}^\beta\right).\end{aligned}$$

Expression of the action and the prefactor in terms of order parameters Here we see that

we have a factor $D \to \infty$ in the exponential part, which can be estimated using the saddle point method. Before doing so, we introduce the following order parameters using the Fourier representation of the delta-function:

$$1 = \int dQ_{\alpha\beta} d\hat{Q}_{\alpha\beta} e^{\hat{Q}_{\alpha\beta}(PQ_{\alpha\beta} - \eta_h^\alpha \eta_h^\beta)}, \tag{A.51}$$

$$1 = \int dR_{\alpha\beta} d\hat{R}_{\alpha\beta} e^{\hat{R}_{\alpha\beta}(DR_{\alpha\beta} - \lambda_i^\alpha \lambda_i^\beta)}. \tag{A.52}$$

This allows to rewrite the prefactor only in terms of Q, R: for example,

$$\mu_1^2 \psi_1 \lambda_i^1 \lambda_i^2 + \mu_\star^2 \eta_i^1 \eta_i^2 = \psi_1 D(\mu_1^2 R^{12} + \mu_\star^2 Q^{12}).$$

To do this, there are two key quantities we need to calculate: $\lambda G_X^{-1} \lambda$ and $\eta G_W^{-1} \eta$. To calculate both, we note that G_X ang G_W are both of the form $\mathbf{I} + \mathbf{X}$, therefore there inverse may be calculated using their series representation. The result is:

$$[M_X]^{\alpha\beta} := \frac{1}{D} \lambda^\alpha G_X^{-1} \lambda^\beta = \left[R(I + \mu_1^2 \psi_1 R)^{-1} \right]^{\alpha\beta}, \tag{A.53}$$

$$[M_W]^{\alpha\beta} := \frac{1}{P} \eta^\alpha (G_W^{-1}) \eta^\beta = \left[Q(I + \mu_\star^2 \psi_1 A Q)^{-1} \right]^{\alpha\beta}. \tag{A.54}$$

Using the above, we deduce:

$$\lambda^1 H_W \lambda^2 = D M_X^{12} + P \mu_1^2 \mu_\star^2 \psi_1 \left[M_X M_W M_X \right]^{12}, \tag{A.55}$$

$$\lambda^1 S_W \eta^2 = P \left[M_X M_W \right]^{12}. \tag{A.56}$$

The integrals over η, λ become simple Gaussian integrals with covariance matrices given by $\hat{Q}, \hat{R}$, yielding:

$$1 = \int dQ_{\alpha\beta} d\hat{Q}_{\alpha\beta} e^{-\frac{\psi_1 D}{2}\left(\log\det \hat{Q} - 2\mathrm{Tr} Q\hat{Q}\right)}, \tag{A.57}$$

$$1 = \int dR_{\alpha\beta} d\hat{R}_{\alpha\beta} e^{-\frac{D}{2}\left(\log\det \hat{R} - 2\mathrm{Tr} R\hat{R}\right)}. \tag{A.58}$$

The next step is to take the saddle point with respect to the auxiliary variables $\hat{Q}$ and $\hat{R}$ in order to eliminate them:

$$\frac{\partial S^v}{\partial \hat{Q}_{\alpha\beta}} = \psi_1 \left(\hat{Q}^{-1} - 2Q \right) = 0 \Rightarrow \hat{Q} = \frac{1}{2} Q^{-1}, \tag{A.59}$$

$$\frac{\partial S^v}{\partial \hat{R}_{\alpha\beta}} = \left(\hat{R}^{-1} - 2R \right) = 0 \Rightarrow \hat{R} = \frac{1}{2} R^{-1}. \tag{A.60}$$

One finally obtains that:

$$\Psi_3^v = \int \left(\prod dQ\right) \left(\prod dR\right) P_{\Psi_3^v}[Q,R] \exp\left(-\frac{D}{2} S^v[Q,R]\right), \tag{A.61}$$

With:

$$\begin{aligned} P_{\Psi_3^v}[Q,R] =& D\psi_1^2\psi_2(\mu_1^2 R^{12} + \mu_\star^2 Q^{12}) \left[\mu_\star^2 [M_W^v]^{12} + \mu_1^2 \left(M_X^{12} + \mu_1^2\mu_\star^2\psi_1^2 [M_X M_W^v M_X]^{12}\right)\right. \\ &\left. + 2\mu_1^2\mu_\star^2\psi_1 [M_X M_W^v]^{12}\right], \end{aligned} \tag{A.62}$$

$$S^v[Q,R] = \psi_2 \log\det(G_X) + \psi_2 \log\det(G_W) + \psi_1^2\psi_2\lambda \mathrm{Tr} Q + \mathrm{Tr}\left(RQ^{-1}\right) + (1-\psi_1)\log\det Q - \log\det R.$$

Saddle point equations The aim is now to use the saddle point method in order to evaluate the integrals over the order parameters. Thus, one looks for R and Q solutions to the equations:

$$\frac{\partial S^v}{\partial Q_{\alpha\beta}} = 0, \quad \frac{\partial S^v}{\partial R_{\alpha\beta}} = 0 \ \ \forall \alpha, \beta = 1, \cdots, 2n.$$

To solve the above, it is common to make a *replica symmetric ansatz*. In this case, we assume that the solutions to the saddle points equations take the form:

$$Q = \begin{bmatrix} q & \tilde{q} & \cdots & \tilde{q} \\ \tilde{q} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \tilde{q} \\ \tilde{q} & \cdots & \tilde{q} & q \end{bmatrix}, \quad R = \begin{bmatrix} r & \tilde{r} & \cdots & \tilde{r} \\ \tilde{r} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \tilde{r} \\ \tilde{r} & \cdots & \tilde{r} & r \end{bmatrix} \tag{A.63}$$

The action takes the following form:

$$\begin{aligned} S^v(q,r,\tilde{q},\tilde{r}) &= 2n\left(S_0(q,r) + S_1^v(q,r,\tilde{q},\tilde{r})\right) \\ S_0(q,r) &= \lambda\psi_1^2\psi_2 q + \psi_2 \log\left(\frac{\mu_\star^2\psi_1 q}{\mu_1^2\psi_1 r + 1} + 1\right) + \frac{r}{q} + (1-\psi_1)\log(q) \\ &\quad + \psi_2 \log\left(\mu_1^2\psi_1 r + 1\right) - \log(r) \\ S_1^v(q,r,\tilde{q},\tilde{r}) &= f^v(q,r)\tilde{q} + g^v(q,r)\tilde{r} \\ f^v(q,r) &= \lambda\psi_1^2\psi_2 + \frac{\mu_\star^2\psi_1\psi_2}{\mu_\star^2\psi_1 q + \mu_1^2\psi_1 r + 1} + \frac{1-\psi_1}{q} - \frac{r}{q^2} \\ g^v(q,r) &= -\frac{\mu_\star^2\mu_1^2\psi_1^2\psi_2 q}{\left(\mu_1^2\psi_1 r + 1\right)\left(\mu_\star^2\psi_1 q + \mu_1^2\psi_1 r + 1\right)} + \frac{\mu_1^2\psi_1\psi_2}{\mu_1^2\psi_1 r + 1} + \frac{1}{q} - \frac{1}{r}. \end{aligned} \tag{A.64}$$

Fluctuations around the saddle point

We introduce the following notations:

$$\begin{aligned}
\left[\nabla_{BT} F(BT_\star)\right]_{\alpha\beta} &= \frac{\partial F}{\partial BT_{\alpha\beta}}|_{BT_\star}, \\
\left[H_{BT} F(BT_\star)\right]_{\alpha\beta,\gamma\delta} &= @\begin{bmatrix} \frac{\partial^2 F}{\partial Q_{\alpha\beta}\partial Q_{\gamma\delta}} & \frac{\partial^2 F}{\partial Q_{\alpha\beta}\partial R_{\gamma\delta}} \\ \frac{\partial^2 F}{\partial R_{\alpha\beta}\partial Q_{\gamma\delta}} & \frac{\partial^2 F}{\partial R_{\alpha\beta}\partial R_{\gamma\delta}} \end{bmatrix} BT_\star, \\
H\left[F\right] &= @\begin{bmatrix} \frac{\partial F}{\partial q\partial q} & \frac{\partial F}{\partial q\partial r} & \frac{\partial F}{\partial q\partial \tilde{q}} & \frac{\partial F}{\partial q\partial \tilde{r}} \\ \frac{\partial F}{\partial q\partial r} & \frac{\partial F}{\partial r\partial r} & \frac{\partial F}{\partial r\partial \tilde{q}} & \frac{\partial F}{\partial r\partial \tilde{r}} \\ \frac{\partial F}{\partial q\partial \tilde{q}} & \frac{\partial F}{\partial r\partial \tilde{q}} & \frac{\partial F}{\partial \tilde{q}\partial \tilde{q}} & \frac{\partial F}{\partial \tilde{q}\partial \tilde{r}} \\ \frac{\partial F}{\partial q\partial \tilde{r}} & \frac{\partial F}{\partial r\partial \tilde{r}} & \frac{\partial F}{\partial \tilde{q}\partial \tilde{r}} & \frac{\partial F}{\partial \tilde{r}\partial \tilde{r}} \end{bmatrix}_{\substack{q=q^* \\ r=r^* \\ \tilde{r}=0 \\ \tilde{q}=0}}.
\end{aligned} \tag{A.65}$$

Proposition Let $q^\star$ and $r^\star$ be the solutions of the fixed point equation for the function $S_0 : (q, r) \mapsto \mathbb{R}$ defined in A.64:

$$\begin{cases} \frac{\partial S_0(q,r)}{\partial q} = 0 \\ \frac{\partial S_0(q,r)}{\partial r} = 0. \end{cases} \tag{A.66}$$

Then we have that

$$\Psi_3^v = \frac{1}{D} \operatorname{Tr}\left[H\left[S^v\right]^{-1} H\left[P_{\Psi_3^v}\right]\right]. \tag{A.67}$$

Sketch of proof Solving the saddle point equations:

$$\begin{cases} \frac{\partial S^v(q,r,\tilde{q},\tilde{r})}{\partial q} = 0 \\ \frac{\partial S^v(q,r,\tilde{q},\tilde{r})}{\partial q} = 0 \\ \frac{\partial S^v(q,r,\tilde{q},\tilde{r})}{\partial \tilde{q}} = 0 \\ \frac{\partial S^v(q,r,\tilde{q},\tilde{r})}{\partial \tilde{r}} = 0 \end{cases}, \tag{A.68}$$

one finds $\tilde{q} = \tilde{r} = 0$, which is problematic because the prefactor vanishes: $P_{\Psi_3} \propto \mu_1^2\tilde{q} + \mu_\star^2\tilde{r}$. Therefore we must go beyond the saddle point contribution to obtain a non zero result, i.e. we have to examine the quadratic fluctuations around the saddle point. To do so we preform a second-order expansion of the action A.63 as a function of Q and R:

$$\begin{aligned}
P_{\Psi_3^v}(BT) &\approx P_{\Psi_3^v}(BT_\star) + B(BT - BT_\star)^\top \nabla P_{\Psi_3^v}(BT_\star) + \frac{1}{2}(BT - BT_\star)^\top H_{BT}\left[P_{\Psi_3^v}(BT_\star)\right](BT - BT_\star), \\
S^v(BT) &\approx S_0(BT_\star) + \frac{1}{2}(BT - BT_\star)^\top H_{BT}\left[S^v(BT_\star)\right](BT - BT_\star).
\end{aligned}$$

Computing the second derivative of A.63, it is easy to show that:

$$\begin{aligned}\left[H_{BT}\left[S^v(BT)\right]\right]_{\alpha\beta,\gamma\delta} &= \left[H_{BT}\left[S^v(BT)\right]\right]_{\alpha\beta}\left(\delta_{\alpha\gamma}\delta_{\beta\delta}+\delta_{\alpha\delta}\delta_{\beta\gamma}\right),\\ \left[H_{BT}\left[P_{\Psi_3^v}(BT)\right]\right]_{\alpha\beta,\gamma\delta} &= \left[H_{BT}\left[P_{\Psi_3^v}(BT)\right]\right]_{\alpha\beta}\left(\delta_{\alpha\gamma}\delta_{\beta\delta}+\delta_{\alpha\delta}\delta_{\beta\gamma}\right),\end{aligned} \tag{A.69}$$

where

$$\begin{aligned}H_{BT}\left[F\right]_{\alpha\beta} &= \begin{bmatrix}\frac{\partial^2 F}{\partial Q_{\alpha\beta}\partial Q_{\alpha\beta}} & \frac{\partial^2 F}{\partial Q_{\alpha\beta}\partial R_{\alpha\beta}}\\ \frac{\partial^2 F}{\partial R_{\alpha\beta}\partial Q_{\alpha\beta}} & \frac{\partial^2 F}{\partial R_{\alpha\beta}\partial R_{\alpha\beta}}\end{bmatrix}\\ &= \frac{1}{2n}\delta_{\alpha\beta}\begin{bmatrix}\frac{\partial^2 F}{\partial q\partial q} & \frac{\partial^2 F}{\partial q\partial r}\\ \frac{\partial^2 F}{\partial r\partial q} & \frac{\partial^2 F}{\partial r\partial r}\end{bmatrix}+\frac{2}{2n(2n-1)}(1-\delta_{\alpha\beta})\begin{bmatrix}\frac{\partial^2 F}{\partial\tilde{q}\partial\tilde{q}} & \frac{\partial^2 F}{\partial\tilde{q}\partial\tilde{r}}\\ \frac{\partial^2 F}{\partial\tilde{r}\partial\tilde{q}} & \frac{\partial^2 F}{\partial\tilde{r}\partial\tilde{r}}\end{bmatrix}.\end{aligned} \tag{A.70}$$

Hence,

$$\begin{aligned}\Psi_3^v &= \lim_{n\to 0}\int dBT P_{\Psi_3^v}(BT)\exp^{-\frac{D}{2}S^v(BT)},\\ &= \lim_{n\to 0}\underbrace{P_{\Psi_3^v}(BT_\star)}_{0}+\nabla P_{\Psi_3^v}(BT_\star)_{\alpha\beta}\underbrace{\int dBT(BT-BT_\star)_{\alpha\beta}\exp\left(-\frac{D}{2}S^v(BT)\right)}_{0}\\ &\quad+\frac{1}{2}H_{BT}\left[P_{\Psi_3^v}(BT_\star)\right]_{\alpha\beta}\int dBT(BT-BT_\star)^2_{\alpha\beta}\exp\left(-\frac{D}{2}S^v(BT)\right)\\ &= \lim_{n\to 0}\frac{1}{2}H_{BT}\left[P_{\Psi_3^v}(BT_\star)\right]_{\alpha\beta}e^{-\frac{D}{2}S^v(BT_\star)}\int dBT(BT-BT_\star)^2_{\alpha\beta}e^{-\frac{D}{2}\sum_{\alpha\beta}B(BT-BT_\star)^2_{\alpha\beta}H_{BT}[S^v(BT_\star)]_{\alpha\beta}}\\ &= \lim_{n\to 0}\frac{1}{2}\frac{(2\pi)^n}{\det H_{BT}\left[S^v(BT_\star)\right]}e^{-\frac{D}{2}S^v(BT_\star)}\frac{2}{D}H_{BT}\left[P_{\Psi_3^v}(BT_\star)\right]_{\alpha\beta}H_{BT}\left[S^v(BT_\star)\right]^{-1}_{\alpha\beta}\\ &= \frac{1}{D}\operatorname{Tr}\left[H\left[S^v\right]^{-1}H\left[P_{\Psi_3^v}\right]\right]\end{aligned}$$

In the last step, we used the fact that:

$$\begin{aligned}\lim_{n\to 0}e^{-\frac{D}{2}S^v(BT_\star)} &= \lim_{n\to 0}e^{-nDS_0(q_\star,r_\star)}=1,\\ \lim_{n\to 0}\det H_{BT}\left[S^v(BT_\star)\right] &= 1.\end{aligned} \tag{A.71}$$

The last equality follows from the fact that for a matrix of size $n\times n$ of the form $M_{\alpha\beta}=a\delta_{\alpha\beta}+b(1-\delta_{\alpha\beta})$, we have

$$\det M=(a-b)^n\left(1+\frac{nb}{a-b}\right)\xrightarrow[n\to 0]{}1.$$

Expression of the vanilla terms Using the above procedure, one can compute the terms Ψ_1,Ψ_2^v,Ψ_3^v of A.30: for each of these terms, the action is the same as in A.63, and

the prefactors can be obtained as:

$$
\begin{aligned}
P_{\Psi_1}[Q,R] &= \mu_1^2\psi_1\psi_2\left(M_X^{11}+\mu_1^2\mu_\star^2\psi_1^2\left(M_XM_WM_X\right)^{11}+\mu_\star^2\mu_1\psi_1\left(M_XM_W\right)^{11}\right),\\
P_{\Psi_2^v}[Q,R] &= D\psi_1^2\psi_2\left(\mu_1^2R^{12}+\mu_\star^2Q^{12}\right)\left(\mu_1^2\psi_2P_{XX}-2\mu_1^2\mu_\star^2\psi_1\psi_2P_{XW}+\mu_\star^2P_{WW}\right),\\
P_{\Psi_3^v}[Q,R] &= D\psi_1^2\psi_2(\mu_1^2R^{12}+\mu_\star^2Q^{12})\\
&\qquad\left[\mu_\star^2\left[M_W^v\right]^{12}+\mu_1^2\left(M_X^{12}+\mu_1^2\mu_\star^2\psi_1^2\left[M_XM_W^vM_X\right]^{12}\right)+2\mu_1^2\mu_\star^2\psi_1\left[M_XM_W^v\right]^{12}\right],\\
P_{XX} &= N_X^{12}+\frac{1}{\psi_2}M_X^{12}+2\left(\mu_1\mu_\star\psi_1\right)^2\left[N_XM_WM_X\right]^{12}+\frac{\left(\mu_1\mu_\star\psi_1\right)^2}{\psi_2}\left[M_XM_WM_X\right]^{12}\\
&\quad+\left(\mu_1\mu_\star\psi_1\right)^4\left[M_XM_WN_XM_WM_X\right]^{12},\\
P_{XW} &= \left[N_XM_W\right]^{12}+\frac{1}{\psi_2}\left[M_XM_W\right]^{12}+\left(\mu_1\mu_\star\psi_1\right)^2\left[M_XM_WN_XM_W\right]^{12},\\
P_{WW} &= M_W^{12}+\psi_2\left(\mu_1\mu_\star\psi_1\right)^2\left[M_WN_XM_W\right]^{12}.
\end{aligned}
$$

Where a new term appears:

$$
\left[N_X\right]^{\alpha\beta}=\left[R(I+\mu_1^2\psi_1R)^{-2}\right]^{\alpha\beta}. \tag{A.72}
$$

Computation of the *ensembling* terms

Expression of the action and the prefactor In the *ensembling* terms, the two inverse matrices are different, hence one has to introduce two distinct replica variables. We distinguish them by the use of an extra index $a\in\{1,2\}$, denoted in brackets in order not to be confused with the replica indices α.

$$
\left[M^{(1)}\right]^{-1}_{ij}\left[M^{(2)}\right]^{-1}_{kl}=\lim_{n\to0}\int\left(\prod_{\alpha=1}^{n}\prod_{a=1}^{2}d\eta^{\alpha(a)}\right)\eta_i^{1(1)}\eta_j^{1(1)}\eta_k^{1(2)}\eta_l^{1(2)}\exp\left(-\frac{1}{2}\sum_{(a)}\eta^{\alpha(a)}M_{ij}^{(a)}\eta^{\alpha(a)}\right). \tag{A.73}
$$

Calculations of the Gaussian integrals follow through in a very similar way as for the *vanilla* terms. The matrices appearing in the process are:

$$(G_X^e)_{ii'} = \delta_{ii'} + \frac{\mu_1^2\psi_1}{D}\sum_{(a)\alpha}\lambda_i^{\alpha(a)}\lambda_{i'}^{\alpha(a)}, \tag{A.74}$$

$$(J_X^e)_i = \frac{\mu_1\mu_\star\sqrt{\psi_1}}{D^2}\sum_{\alpha a}\lambda_i^{\alpha(a)}\eta_h^{\alpha(a)}W_h^{(a)}, \tag{A.75}$$

$$(G_W^e)_{hh'}^{(ab)} = \delta_{hh'}^{ab} + \frac{\mu_\star^2}{D}\sum_{\alpha\beta}\eta_h^{\alpha(a)}A_e^{\alpha\beta,ab}\eta_{h'}^{\beta(b)}, \tag{A.76}$$

$$(J_W^e)_h = 0 \tag{A.77}$$

$$(G_\Theta^e)_{hh',ii'}^{(ab)} = \delta_{hh',ii'}^{ab} \tag{A.78}$$

$$(J_\Theta^e)_{hi}^{(a)} = -i\sum_{\alpha}\hat{\lambda}_i^{\alpha(a)}\eta_h^{\alpha(a)}, \tag{A.79}$$

$$(G_{\hat{\lambda}}^e)_{ii'}^{\alpha\beta,(ab)} = 2\delta^{ab}\delta^{ii'}\eta_h^{\alpha(a)}\eta_h^{\beta(b)}, \tag{A.80}$$

$$(J_{\hat{\lambda}}^e)_i^{\alpha(a)} = i\sqrt{P}\lambda_i^{\alpha(a)}. \tag{A.81}$$

$$\tag{A.82}$$

with

$$A_e^{\alpha\beta,(ab)} = \delta_{ab}\delta_{\alpha\beta} - \mu_1^2\psi_1\frac{1}{D}\sum_{i,j}\lambda_i^{\alpha(a)}[G_X^{e-1}]_{ij}\lambda_j^{\beta(b)},$$

$$[H_W^e]_{ii'} = \left[G_X^{e-1}\right]_{ii'} + \sum_{\alpha\beta,ab}\left[G_X^{e-1}\lambda^{\alpha(a)}\right]_i\left[G_X^{e-1}\lambda^{\beta(b)}\right]_{i'}\left[\eta^{\alpha(a)}\left[G_W^{e-1}\right]^{(ab)}\eta^{\beta(b)}\right],$$

$$[S_W^e]_{ih} = \frac{1}{D}\sum_{\alpha,a}\left[G_X^{e-1}\lambda^{\alpha(a)}\right]_i\left[G_W^{e-1}\eta^{\alpha(a)}\right]_h.$$

Starting with the computation of Ψ_3^e in order to illustrate the method used, the prefactor $P_{\Psi_3^e}$ and the action are S^e are given by:

$$\begin{aligned}P_{\Psi_3^e}[\eta,\lambda] :=& \frac{\mu_1^2\psi_1\psi_2}{D}\lambda_i^{1(1)}\lambda_i^{1(2)}\left[\mu_\star^2\left[\eta^{1(1)}(G_W^{e-1})^{(12)}\eta^{1(2)}\right] + \mu_1^2\psi_1\left[\lambda^{1(1)}H_W^e\lambda^{1(2)}\right]\right.\\ &\left. + 2\mu_1^2\mu_\star^2\psi_1[\lambda^{1(1)}S_W^e\eta^{1(2)}]\right],\\ S^e[\eta,\lambda] :=& \psi_2\log\det(G_X^e) + \psi_2\log\det(G_W^e) + \log\det\left(G_{\hat{\lambda}}^e\right) + \frac{1}{D}\psi_1\psi_2\lambda\sum(\eta_h^{\alpha(a)})^2\\ &+ \frac{1}{2D}\left(\lambda_i^{\alpha(a)}(G_{\hat{\lambda}}^{e-1})_{ii'}^{\alpha\beta,ab}\lambda_{i'}^{\beta(b)}\right).\end{aligned}$$

Expression of the action and the prefactor in terms of order parameters

This time, because of the two different systems, the order parameters carry an

additional index a, which turns them into 2×2 block matrices:

$$1 = \int dQ_{\alpha\beta}^{(ab)} d\hat{Q}_{\alpha\beta}^{(ab)} e^{\hat{Q}_{\alpha\beta}^{(ab)}(PQ_{\alpha\beta}^{(ab)} - \eta_h^{\alpha(a)}\eta_h^{\beta(b)})}, \tag{A.83}$$

$$1 = \int dR_{\alpha\beta}^{(ab)} d\hat{R}_{\alpha\beta}^{(ab)} e^{\hat{R}_{\alpha\beta}^{(ab)}(dR_{\alpha\beta}^{(ab)} - \lambda_i^{\alpha(a)}\lambda_i^{\beta(b)})}. \tag{A.84}$$

The systems being decoupled, we make the following ansatz for the order parameters:

$$Q = \left[\begin{array}{ccc|ccc} q & \cdots & 0 & \tilde{q} & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & q & 0 & \cdots & \tilde{q} \\ \hline \tilde{q} & \cdots & 0 & q & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \tilde{q} & 0 & \cdots & q \end{array}\right], \; R = \left[\begin{array}{ccc|ccc} r & \cdots & 0 & \tilde{r} & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & r & 0 & \cdots & \tilde{r} \\ \hline \tilde{r} & \cdots & 0 & r & \cdots & 0 \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & \tilde{r} & 0 & \cdots & r \end{array}\right]. \tag{A.85}$$

In virtue of the simple structure of the above matrices, the replica indices α trivialize and we may replace the matrices Q and R by the 2×2 matrices:

$$Q = \begin{bmatrix} q & \tilde{q} \\ \tilde{q} & q \end{bmatrix}, \; R = \begin{bmatrix} r & \tilde{r} \\ \tilde{r} & r \end{bmatrix}.$$

Define:

$$[M_X^e]_{(ab)} \equiv \frac{1}{D}\lambda_i^{(a)} \left[G_X^{e-1}\right]_{ij} \lambda_j^{(b)} = \left[R(I + \mu_1^2\psi_1 R)^{-1}\right]_{(ab)},$$
$$[M_W^e]_{(ab)} \equiv \frac{1}{P}\eta^{(a)} \left[G_W^{e-1}\right]^{(ab)} \eta^{(b)} = \left[Q(I + \psi_1 A^e Q)^{-1}\right]_{(ab)},$$

where products are now over 2×2 matrices. Then, one has:

$$\begin{aligned} P_{\Upsilon_3^e}[Q,R] :=& D\mu_1^2\psi_1^2\psi_2 R^{(12)} \left[\mu_\star^2 M_W^{e(12)} + \mu_1^2 \left(M_X^{e(12)} + \mu_1^2\mu_\star^2\psi_1^2 \left[M_X^e M_W^e M_X^e\right]^{(12)}\right)\right. \\ &\left. + 2\mu_1^2\mu_\star^2\psi_1 \left[M_X^e M_W^e\right]^{(12)}\right], \\ S^e[Q,R] :=& \psi_2 \log\det(G_X^e) + \psi_2 \log\det(G_W^e) + \sum_a \left[(\psi_1^2\psi_2\lambda)\mathrm{Tr}Q^{(a)(a)}\right. \\ &\left. + \mathrm{Tr}\left(R^{(a)(a)}(Q^{-1})^{(a)(a)}\right) + \log\det Q^{(a)(a)}\right] \\ &- \psi_1 \log\det Q - \log\det R. \end{aligned} \tag{A.86}$$

Where we have:

$$M_X^e = \frac{1}{(1+\mu_1^2\psi_1 r)^2 - (\mu_1^2\psi_1\tilde{r})^2}\begin{bmatrix} r+\mu_1^2\psi_1(r^2-\tilde{r}^2) & \tilde{r} \\ \tilde{r} & r+\mu_1^2\psi_1(r^2-\tilde{r}^2)\end{bmatrix}, \tag{A.87}$$

$$A^e_{(ab)} = \delta_{(ab)} - \mu_1^2\psi_1\left[M_X^e\right]_{(ab)}, \tag{A.88}$$

$$\left[M_W^e\right]_{(ab)} = q\delta_{(ab)} - \mu_\star^2\psi_1\left[A^e(I+\mu_\star^2\psi_1 q A^e)^{-1}\right]_{(ab)}, \tag{A.89}$$

$$\det(G_X^e) = \det\Big(\delta_{(ab)} + \psi_1\mu_1^2 R_{(ab)}\Big), \tag{A.90}$$

$$\det(G_W^e) = \det\Big(\delta_{ab} + \psi_1 q A^e_{(ab)}\Big). \tag{A.91}$$

Finally, we are left with:

$$\begin{aligned} S^e(q,r,\tilde{q},\tilde{r}) =& n\left(S_0(q,r) + S_1^e(q,r,\tilde{q},\tilde{r})\right), \\ S_1^e(q,r,\tilde{q},\tilde{r}) =& \tilde{r}^2 f^e(q,r) + \tilde{q}^2 g^e(q,r), \\ f^e(q,r) =& \frac{2r\mu_1^2\psi_1\left(1+q\mu_\star^2\psi_1\right) + \left(1+q\mu_\star^2\psi_1\right)^2 - r^2\mu_1^4\psi_1^2(-1+\psi_2)}{2r^2\left(1+r\mu_1^2\psi_1+q\mu_\star^2\psi_1\right)^2}, \\ g^e(q,r) =& \frac{\psi_1}{2q^2}. \end{aligned} \tag{A.92}$$

Where S_0 was defined in A.64. *Expression of the ensembling terms* Evaluating the fluctuations around the saddle point follows through in the same way as for the vanilla terms, with the following expressions of the prefactors:

$$\begin{aligned} P_{\Psi_2^e}[Q,R] &= D\psi_1^2\psi_2\mu_1^2\tilde{r}\left[\mu_1^2 P^e_{XX} - 2\mu_1^2\mu_\star^2\psi_1 P^e_{WX} + \mu_\star^2 P^e_{WW}\right], \\ P_{\Psi_3^e}[Q,R] &= D\mu_1^2\psi_1^2\psi_2 R^{(12)}\Big[\mu_\star^2 M_W^{e(12)} + \mu_1^2\Big(M_X^{e(12)} + \mu_1^2\mu_\star^2\psi_1^2\left[M_X^e M_W^e M_X^e\right]^{(12)}\Big) \\ &\qquad + 2\mu_1^2\mu_\star^2\psi_1\left[M_X^e M_W^e\right]^{(12)}\Big], \\ P^e_{XX} &= \psi_2 N_X^{e12} + M_X^{e12} + 2\psi_2(\mu_1\mu_\star\psi_1)^2\left[M_X^e N_X^e M_W^e\right]^{12} + (\mu_1\mu_\star\psi_1)^2\left[M_X^e M_W^e M_X^e\right]^{12} \\ &\quad + \psi_2(\mu_1\mu_\star\psi_1)^4\left[M_X^e M_W^e N_X^e M_W^e M_X^e\right]^{12}, \\ P^e_{WX} &= \psi_2\left[N_X^e M_W^e\right]^{12} + \left[M_X^e M_W^e\right]^{12} + \psi_2(\mu_1\mu_\star\psi_1)^2\left[M_X^e M_W^e N_X^e M_W^e\right]^{12}, \\ P^e_{WW} &= \left[M_W^e\right]^{12} + \psi_2(\mu_1\mu_\star\psi_1)^2\left[M_W^e N_X^e M_W^e\right]^{12}. \end{aligned}$$

Computation of the *divide and conquer* term

Here, we are interested in computing the term Ψ_2^d. This term differs from the previous ones in that there are now two independent data matrices $\boldsymbol{X}^{(1)}$ and $\boldsymbol{X}^{(2)}$. The calculations for the action and the prefactor are very similar to calculations performed for the *ensembling* terms Ψ_2^e, Ψ_3^e, with the addition that BX now also

carries an index (a). Firstly let us write Ψ_2^d as a trace over random matrices:

$$\Psi_2^d = \frac{\mu_1^2}{d^2} \operatorname{Tr}\left[\boldsymbol{X}^{(1)}\boldsymbol{Z}^{(1)}\boldsymbol{B}^{(1)-1}\boldsymbol{\Theta}^{(1)}\boldsymbol{\Theta}^{(2)}\boldsymbol{B}^{(2)-1}\boldsymbol{Z}^{(2)}\boldsymbol{X}^{(2)}\right]. \tag{A.93}$$

Calculations follow through in the same way as in the previous sections. Using the replica formula A.73, and performing the integrals over the Gaussian variables the following quantities appear:

$$\begin{aligned}
&(G_X^d)_{ii'}^{(ab)} = \delta_{ii'} + \frac{\mu_1^2\psi_1}{D}\sum_\alpha \lambda_i^{\alpha(a)}\lambda_{i'}^{\alpha(a)},\\
&(J_X^d)_i^{(a)} = \frac{\mu_1\mu_\star\sqrt{\psi_1}}{D^2}\sum_\alpha \lambda_i^{\alpha(a)}\eta_h^{\alpha(a)}W_h^{(a)},\\
&(G_W^d)_{hh'}^{(ab)} = \delta_{hh'}^{ab} + \frac{\mu_\star^2}{D}\sum_{\alpha\beta}\eta_h^{\alpha(a)}A^{\alpha\beta,ab}\eta_{h'}^{\beta(b)},\\
&(J_W^d)_h = 0,\\
&(G_\Theta^d)_{hh',ii'}^{(ab)} = \delta_{hh',ii'}^{ab}, \qquad (J_\Theta^d)_{hi}^{(a)} = -i\sum_\alpha \hat{\lambda}_i^{\alpha(a)}\eta_h^{\alpha(a)},\\
&(G_{\hat{\lambda}}^d)_{ii'}^{\alpha\beta,(ab)} = 2\delta^{ab}\delta^{ii'}\eta_h^{\alpha(a)}\eta_h^{\beta(b)}, \quad (J_{\hat{\lambda}}^d)_i^{\alpha(a)} = i\sqrt{P}\lambda_i^{\alpha(a)}.
\end{aligned}$$

The saddle point ansatz for Q and R is the same as the one for the ensembling terms (see A.85). The procedure to evaluate Ψ_2^d is also the same as the one for Ψ_2^e except the Hessian is taken with respect to S^d. The final result is given below.

$$\begin{aligned}
P_{\Psi_2^d}[Q,R] =& D\mu_1^2\psi_1\psi_2^2\tilde{r}\left[\psi_1\mu_1^2 P_{XX} + 2\mu_\star^2\mu_1^2\psi_1^2 P_{WX} + \mu_\star^2\psi_1 P_{WW}\right],\\
P_{XX} =& \left(N_X^{11} + 2(\mu_1\mu_\star\psi_1)^2\left[N_X M_W M_X\right]^{11} + (\mu_1\mu_\star\psi_1)^4\left[M_X M_W N_X M_W M_X\right]^{11}\right),\\
P_{WX} =& \left[N_X M_W\right]^{11} + (\mu_1\mu_\star\psi_1)^2\left[M_X M_W N_X M_W\right]^{11},\\
P_{WW} =& (\mu_1\mu_\star\psi_1)^2\left[M_W N_X M_W\right]^{11},\\
S^d[q,r,\tilde{q},\tilde{r}] =& n\left(S_0(q,r) + S_1^d(q,r,\tilde{q},\tilde{r})\right),\\
S_1^d(q,r,\tilde{q},\tilde{r}) =& \frac{\tilde{r}^2}{r^2} + \frac{\psi_1\tilde{q}^2}{q^2}.
\end{aligned}$$

With:

$$\begin{aligned}
M_X &= \frac{r}{1+\mu_1^2\psi_1 r}, \quad A = 1 - \mu_1^2\psi_1 M_X,\\
M_W &= \frac{q}{1+\mu_\star^2\psi_1 q A}, \quad N_X = \frac{\tilde{r}}{(1+\mu_1^2\psi_1 r)^2}.
\end{aligned}$$

Appendix B

Online learning with two layer neural networks in the ODE limit - Toolbox

B.1 Moments of functions of weakly correlated variables

Here, we show how to compute expectation of functions of weakly correlated variables with non zero mean. The derivation follows the ones of [117] (see App. A). We extend their computations to include variables with non-zero means.

Consider the random variables x,$y \in \mathbb{R}$, jointly Gaussian with joint probability distribution:

$$P(x,y) = \frac{1}{2\pi\sqrt{\det M_2}} \exp\left[-\frac{1}{2}\begin{pmatrix} x-\bar{x} & y-\bar{y} \end{pmatrix} M_2^{-1} \begin{pmatrix} x-\bar{x} \\ y-\bar{y} \end{pmatrix}\right], \tag{B.1}$$

where we defined the mean of x, respectively y, as $\bar{x}$, respectively $\bar{y}$ and the covariance matrix:

$$M_2 = \begin{pmatrix} C_x & \epsilon M_{12} \\ \epsilon M_{12} & C_y \end{pmatrix}. \tag{B.2}$$

The weak correlation between x and y is encapsulated in the parameter $\epsilon \ll 1$ while $M_{12} \sim O(1)$.

We are interested in computing expectations of the form $\mathop{\mathbb{E}}_{(x,y)}[f(x)g(y)]$ with two real valued functions $f, g : \mathbb{R} \to \mathbb{R}$. Leveraging the weak correlation between x

and y, we can expand the distribution Eq. (B.1) to linear order in ϵ, i.e.:

$$(\text{B.1}) = \frac{1}{2\pi\sqrt{C_x C_y}} \mathrm{e}^{-\frac{1}{2C_x}(x-\bar{x})^2 - \frac{1}{2C_y}(y-\bar{y})^2} \left[1 - \epsilon(x-\bar{x})\left(C_x^{-1} M_{12} C_y^{-1}\right)(y-\bar{y}) + O\left(\epsilon^2\right)\right] \tag{B.3}$$

Using the above, one can compute the expectations:

$$\begin{aligned} \mathop{\mathbb{E}}_{(x,y)}\left[f(x)g(y)\right] = &\mathop{\mathbb{E}}_{x}\left[f(x)\right]\mathop{\mathbb{E}}_{y}\left[g(y)\right] \\ &+ \epsilon \mathop{\mathbb{E}}_{x}\left[f(x)(x-\bar{x})\right]\left(C_x^{-1} M_{12} C_y^{-1}\right)\mathop{\mathbb{E}}_{y}\left[g(y)(y-\bar{y})\right] + O(\epsilon^2). \end{aligned} \tag{B.4}$$

The expectations are now taken over the 1-dimensional distributions of $x \sim \mathcal{N}(0, C_x)$ and $y \sim \mathcal{N}(0, C_x)$. Similarly, consider the case of three weakly correlated real random variables x_i, $i = 1, 2, 3$ with mean $\bar{x}_i$ and covariance matrix M_3 such that

$$(M_3)_{ij} = \begin{cases} C_i, \text{ if } i = j \\ \epsilon M_{ij}, \text{ if } i \neq j \end{cases} . \tag{B.5}$$

One can use an expansion of the joint probability distribution of $\{x_i\}$ to linear order in ϵ to compute three point moments of real valued functions f, g, h as:

$$\begin{aligned} \mathop{\mathbb{E}}_{\{x\}_i}\left[f(x_1)g(x_2)h(x_3)\right] = &\mathop{\mathbb{E}}_{\{x\}_i}\left[f(x_1)\right]\left[g(x_2)\right]\left[h(x_3)\right] \\ &- \epsilon \mathop{\mathbb{E}}_{x_3}\left[h(x_3)\right]\mathop{\mathbb{E}}_{x_1}\left[(x_1-\bar{x}_1)f(x_1)\right]\left(C_x^{-1} M_{12} C_y^{-1}\right)\mathop{\mathbb{E}}_{x_2}\left[(x_2-\bar{x}_2)g(x_2)\right] \\ &- \epsilon \mathop{\mathbb{E}}_{x_2}\left[g(x_2)\right]\mathop{\mathbb{E}}_{x_1}\left[(x_1-\bar{x}_1)f(x_1)\right]\left(C_x^{-1} M_{13} C_y^{-1}\right)\mathop{\mathbb{E}}_{x_3}\left[(x_3-\bar{x}_3)h(x_3)\right] \\ &- \epsilon \mathop{\mathbb{E}}_{x_1}\left[f(x_1)\right]\mathop{\mathbb{E}}_{x_2}\left[(x_2-\bar{x}_2)g(x_2)\right]\left(C_x^{-1} M_{23} C_y^{-1}\right)\mathop{\mathbb{E}}_{x_3}\left[(x_3-\bar{x}_3)h(x_3)\right] + O(\epsilon^2). \end{aligned} \tag{B.6}$$

In the case in which x_1 and x_2 are weakly correlated with x_3 but not between each other, i.e. $\mathrm{Cov}(x_1, x_2) = M_{12} \sim O(1)$, one has:

$$\begin{aligned} \mathbb{E}\left[f(x_1)g(x_2)h(x_3)\right] = &\mathbb{E}\left[f(x_1)g(x_2)\right]\mathbb{E}\left[h(x_3)\right] \\ &+ \epsilon \frac{\mathbb{E}\left[h(x_3)(x_3-\bar{x}_3)\right]}{\left(C_{x_1}C_{x_2} - M_{12}^2\right)C_{x_3}} \{\mathbb{E}\left[f(x_1)g(x_2)(x_1-\bar{x}_1)\right] M_{13} C_{x_2} \\ &\quad + \mathbb{E}\left[f(x_1)g(x_2)(x_2-\bar{x}_2)\right] M_{23} C_{x_1} \\ &\quad - \mathbb{E}\left[f(x_1)g(x_2)(x_1-\bar{x}_1)\right] M_{12} M_{23} \\ &\quad - \mathbb{E}\left[f(x_1)g(x_2)(x_2-\bar{x}_2)\right] M_{13} M_{12}\} + O(\epsilon^2). \end{aligned} \tag{B.7}$$

B.2 Analytical formula for the integrals in the equations of motion

Sigmoidal activation function

$$I_2 \equiv \mathbb{E}\, g(x_1)g(x_2) = \frac{2}{\pi}\left[\frac{c_{12}}{\sqrt{1+c_{11}}\sqrt{1+c_{22}}}\right] \tag{B.8}$$

$$J_2 \equiv \mathbb{E}\, x_1 g(x_2) = \sqrt{\frac{2}{\pi(1+c_{22})}}c_{12} \tag{B.9}$$

$$I_3 \equiv \mathbb{E}\, g'(x_1)x_2 g(x_3) = \frac{2}{\pi}\frac{1}{\sqrt{(1+c_{11})(1+c_{33})-c_{13}^2}}\frac{c_{23}(1+c_{11})-c_{12}c_{13}}{1+c_{11}} \tag{B.10}$$

$$I_{21} \equiv \mathbb{E}\, g'(x_1)x_1x_2 = \sqrt{\frac{2}{\pi}}\frac{c_{12}}{(c_{11}+1)^{3/2}} \tag{B.11}$$

$$I_{22} \equiv \mathbb{E}\, g'(x_1)x_2^2 = \sqrt{\frac{2}{\pi}}\frac{(c_{11}c_{22}-c_{12}^2+c_{22})}{(c_{11}+1)^{3/2}} \tag{B.12}$$

Linear activation function

$$I_2 = J_2 = I_{21} \equiv \mathbb{E}\, x_1x_2 = c_{12} \tag{B.13}$$

$$I_{22} \equiv \mathbb{E}\, x_2^2 = c_{22} \tag{B.14}$$

$$I_3 \equiv \mathbb{E}\, x_2x_3 = c_{23} \tag{B.15}$$

$$\tag{B.16}$$

ReLU activation function

$$\begin{aligned} I_2 &\equiv \mathbb{E}\, g(x_1)g(x_2) \\ &= \frac{1}{8\pi}\left[2\sqrt{c_{11}c_{22}-c_{12}^2}+c_{12}\pi+2c_{12}+\arctan\frac{c_{12}}{\sqrt{c_{11}c_{22}-c_{12}^2}}\right] \end{aligned} \tag{B.17}$$

$$J_2 \equiv \mathbb{E}\, x_1 g(x_2) = \frac{c_{12}}{2} \tag{B.18}$$

Appendix C

Understanding the interplay between data structure and architecture in Gaussian mixture classification

C.1 Summary of Notations

D	input dimensions	P	number of random features
N	number of samples	$\gamma = P/D$	
$t = N/D$	training time, or equivalently rescaled number of training samples	y^*	true label
$\Omega^\alpha \in \mathbb{R}^{D\times D}$	covariance of the normal distribution of cluster α	$\frac{\mu^\alpha}{\sqrt{D}} \in \mathbb{R}^D$	mean of the Gaussian cluster α in the mixture
$x \in \mathbb{R}^D$	input	σ	standard deviation of the Gaussian clusters in a mixture with $\Omega_\alpha = \sigma^2\mathbb{I}$
$\mathrm{SNR} = \frac{\lvert\mu\rvert}{\sigma\sqrt{D}}$	signal to noise ration	η	learning rate
λ	$L2$-regularisation constant	pmse	population mean squared error

$q(x|y^*) = \sum_{\alpha \in \mathcal{S}(y^*)} P_{\alpha^\pm} \mathcal{N}_\alpha(x)$ conditional probability of x given the true label y^*

Two layer neural networks (2LNN)

K	number of hidden nodes of the 2LNN	$\theta_i^{(1)k}$	first layer weights
$\theta^{(2)k}$	second layer weights	$g : \mathbb{R} \to \mathbb{R}$	activation function
$a \equiv \sum_r \theta_r^{(1)k} x_r / \sqrt{D}$	local field/pre-activation of the 2LNN	$\phi_\theta(x) = \sum_{k=1}^{K} \theta^{(2)k} g(a^k)$	output of the network
$Q_\alpha^{kl} = \sum_{rs} \frac{\theta_r^{(1)k} \Omega_{rs}^\alpha \theta_s^{(1)l}}{D}$	order parameter/ covariance of the local fields	$M_\alpha^k = \sum_r \frac{\theta_r^{(1)k} \mu_r^\alpha}{D}$	order parameter/mean of the local fields
σ_0	weights are initialised i.i.d. from $\mathcal{N}(0, \sigma_0^2)$		

Random Features (RF)

$F \in \mathbb{R}^{P\times D}$	projection matrix $F_{ir} \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0,1)$	$z \equiv \psi\left(\sum_{r=1}^{D} F_{ir}x_r/\sqrt{D}\right)$	features
$\psi : \mathbb{R}\to\mathbb{R}$	activation function applied element wise	$\phi_w(x) = \sum_{i=1}^{P} w_i z_i/\sqrt{D}$	output of the network
$\hat{W}$	fix point solution of the SGD update equation of RF		

C.2 Equations of Motion

In this appendix, we derive the dynamical equations that describe the dynamics of two-layer neural networks trained on the Gaussian mixture from Sec. 4.4. We first derive a useful Lemma for the averages of weakly correlated random variables B.1, which we we then use in the derivation of the dynamical equations C.2

Derivation of the ODEs

In this section, we derive the ODEs describing the dynamics of training of a 2LNN trained on inputs sampled from the distribution 4.36 with $L2$-regularisation constant λ. We restrict to the case where all the Gaussian clusters have the same covariance matrix, i.e. $\Omega^\alpha = \Omega \in \mathbb{R}^{D\times D}$.

In order to track the training dynamics, we analyse the evolution of the macroscopic operators defined in Eq. 4.39 allowing to compute the performances of the network at all training times.

At the sth step of training, the SGD update for the networks parameter is given by Eq. 4.37:

$$\mathrm{d}\theta_i^{(1)k} \equiv \left(\theta_i^{(1)k}\right)_{s+1} - \left(\theta_i^{(1)k}\right)_s = -\frac{\eta_{\theta^{(1)}}}{\sqrt{D}}\left(\theta^{(2)k}\Delta g'(a^k)x_i + \lambda\theta_i^{(1)k}\right),\ \mathrm{d}\theta^{(2)k} = -\frac{\eta_{\theta^{(2)}}}{\sqrt{D}}\left(g(a^k) + \lambda\theta^{(2)k}\Delta\right). \tag{C.1}$$

In order to guarantee that the dynamics can be described by a set of ordinary

equations in the $D \to \infty$ limit, we choose different scalings for the first and second layer learning rates:

$$\eta_{\theta^{(1)}} = \eta, \qquad \eta_{\theta^{(2)}} = \eta / D$$

for some constant η.

Update of the first layer weights To make progress, consider the eigen-decomposition of the covariance matrix:

$$\Omega_{rs} = \frac{1}{D} \sum_{\tau=1}^{D} \Gamma_{s\tau} \Gamma_{r\tau} \rho_\tau; \tag{C.2}$$

where we denote the eigenvalues as ρ_τ, their corresponding eigenvector as Γ_τ and the eigenvalue distribution as p_Ω. We further define the projection of the weights into the projected basis as

$$\tilde{\theta}_\tau^{(1)k} \equiv \frac{1}{\sqrt{D}} \sum_{\tau=1}^{D} \Gamma_{s\tau} \theta_s^{(1)k} \tag{C.3}$$

and similarly $\tilde{x}_\tau$ and $\tilde{\mu}_\tau^\alpha$ as the projected inputs and means. In this basis, the SGD update for the first layer weights is:

$$\mathrm{d}\tilde{\theta}_\tau^{(1)k} \equiv \left(\tilde{\theta}_\tau^{(1)k}\right)_{s+1} - \left(\tilde{\theta}_\tau^{(1)k}\right)_s = -\frac{\eta}{\sqrt{D}} \left(\theta^{(2)k} \Delta g'(a^k) \tilde{x}_\tau + \lambda \tilde{\theta}_\tau^{(1)k}\right). \tag{C.4}$$

The expectation of this update over the distribution Eq. 4.36 is given by:

$$\mathbb{E}\, \mathrm{d}\tilde{\theta}_\tau^{(1)k} = \sum_{\alpha \in \mathcal{S}(+)} \mathcal{P}_\alpha \mathrm{d}\tilde{\theta}^{(1)k}(\rho)_{\alpha^+} + \sum_{\alpha \in \mathcal{S}(-)} \mathcal{P}_\alpha \mathrm{d}\tilde{\theta}^{(1)k}(\rho)_{\alpha^-}, \tag{C.5}$$

where we decomposed the expectation into the different clusters and introduced:

$$\mathrm{d}\tilde{\theta}^{(1)k}(\rho)_{\alpha^\pm} = \pm \frac{\eta}{\sqrt{D}} \theta^{(2)k} \mathcal{C}_\tau^k - \frac{\eta}{\sqrt{D}} \sum_{j \neq k} \theta^{(2)k} \theta^{(2)j} \mathcal{A}_\tau^{kj} - \frac{\eta}{\sqrt{D}} \theta^{(2)k} \theta^{(2)k} \mathcal{B}_\tau^k - \frac{\eta \lambda}{\sqrt{D}} \tilde{\theta}^{(1)k}(\rho), \tag{C.6}$$

with the expectations $\mathcal{A}_\tau^{kj}$, $\mathcal{B}_\tau^k$ and $\mathcal{C}_\tau^k$ defined as:

$$\mathcal{A}_\tau^{kj} = \mathop{\mathbb{E}}_{\alpha} g'(a^k) g(a^j) \tilde{x}_\tau, \quad \mathcal{B}_\tau^k = \mathop{\mathbb{E}}_{\alpha} g'(a^k) g(a^k) \tilde{x}_\tau, \quad \mathcal{C}_\tau^k = \mathop{\mathbb{E}}_{\alpha} g'(a^k) \tilde{x}_\tau. \tag{C.7}$$

A crucial observation, is that a^k and the projected input $\tilde{x}_\tau$ are jointly Gaussian and weakly correlated, with a correlation of order $1/\sqrt{D}$:

$$\mathop{\mathbb{E}}_{\alpha} a^k \tilde{x}_\tau = \frac{1}{\sqrt{D}} \rho_\tau \tilde{\theta}^{(1)k}. \tag{C.8}$$

Thus, we can compute the expressions Eq. C.7 using the proposition for weakly correlated variables derived in App. B.1. This gives:

$$\begin{aligned}
\mathcal{A}_\tau^{kj} &= \mathop{\mathbb{E}}_{\alpha}\, g'(a^k) g(a^j) \frac{\tilde{\mu}_\tau^\alpha}{\sqrt{D}} \\
&\quad + \frac{1}{Q^{kk}Q^{jj} - Q^{kj2}} \left\{ \left(\mathop{\mathbb{E}}_{\alpha}\, g'(a^k) a^k g(a^j) - \mathop{\mathbb{E}}_{\alpha}\, g'(a^k) g(a^j) M^{\alpha k} \right) \left(Q^{jj} \frac{\tilde{\theta}_\tau^{(1)k} \rho_\tau}{\sqrt{D}} - Q^{kj} \frac{\tilde{w}_\tau^j \rho_\tau}{\sqrt{D}} \right) \right. \\
&\qquad \left. \left(\mathop{\mathbb{E}}_{\alpha}\, g'(a^k) a^j g(a^j) - \mathop{\mathbb{E}}_{\alpha}\, g'(a^k) g(a^j) M^{\alpha j} \right) \left(Q^{kk} \frac{\tilde{w}_\tau^j \rho_\tau}{\sqrt{D}} - Q^{kj} \frac{\tilde{\theta}_\tau^{(1)k} \rho_\tau}{\sqrt{D}} \right) \right\} \\
\mathcal{B}_\tau^k &= \mathop{\mathbb{E}}_{\alpha}\, g'(a^k) g(a^k) \frac{\tilde{\mu}_\tau^\alpha}{\sqrt{D}} + \frac{1}{Q^{kk}} \left(\mathop{\mathbb{E}}_{\alpha}\, g'(a^k) a^k g(a^k) - \mathop{\mathbb{E}}_{\alpha}\, g'(a^k) g(a^k) M^{\alpha k} \right) \frac{\tilde{\theta}_\tau^{(1)k} \rho_\tau}{\sqrt{D}}, \\
\mathcal{C}_\tau^k &= \mathop{\mathbb{E}}_{\alpha}\, g'(a^k) \frac{\tilde{\mu}_\tau^\alpha}{\sqrt{D}} + \frac{1}{Q^{kk}} \left(\mathop{\mathbb{E}}_{\alpha}\, g'(a^k) a^k - \mathop{\mathbb{E}}_{\alpha}\, g'(a^k) M^{\alpha k} \right) \frac{\tilde{\theta}_\tau^{(1)k} \rho_\tau}{\sqrt{D}},
\end{aligned} \tag{C.9}$$

where we used that the first moments of the local fields are given by the order parameters, $\mathbb{E}_\alpha[a^k] = M^{\alpha k}$ and $\mathrm{Cov}(a^k, a^l) = Q^{kl}$. The multi-dimensional integrals of the activation function only depend on the order parameters at the previous step. We discuss how to obtain them, using monte-carlo methods in Sec 4.4. The averaged update of the first layer weights follows directly from Eq. C.9.

Update of the Order parameters In order to derive the update equations for the order parameters, we introduce the densities $m(\rho, t)$ and $q(\rho, t)$. These depend on ρ and on the normalised number of steps $t = \mu/D$, which we interpret as a continuous time variable.

$$\begin{aligned}
m^{\alpha k}(\rho, t) &= \frac{1}{D\epsilon_\rho} \sum_\tau \tilde{\theta}_\tau^{(1)k} \tilde{\mu}_\tau^\alpha \mathbb{1}_{(\rho_\tau \in [\rho, \rho+\varepsilon_\rho[)}, \\
q^{kl}(\rho, t) &= \frac{1}{D\varepsilon_\rho} \sum_\tau \tilde{\theta}_\tau^{(1)k} \tilde{\theta}_\tau^{(1)l} \mathbb{1}_{(\rho_\tau \in [\rho, \rho+\varepsilon_\rho[)},
\end{aligned} \tag{C.10}$$

where $\mathbb{1}(.)$ is the indicator function and the limit $\epsilon_\rho \to 0$ is taken after the thermodynamic limit. Using these definitions, the order parameters can be written as:

$$Q^{kl}(t) = \int \mathrm{d}\rho\; p_\Omega(\rho)\, \rho q^{kl}(\rho, t), \quad M^{\alpha k}(t) = \int \mathrm{d}\rho\; p_\Omega(\rho) m^{\alpha k}(\rho, t). \tag{C.11}$$

The equation of motion of m can can be easily computed using the update C.6 and is given by:

$$\frac{\partial m^{\beta k}(\rho, t)}{\partial t} = \sum_{\alpha \in \mathcal{S}(+)} \mathcal{P}_\alpha \frac{\partial m_{\alpha^+}^{\beta k}(\rho, t)}{\partial t} + \sum_{\alpha \in \mathcal{S}(-)} \mathcal{P}_\alpha \frac{\partial m_{\alpha^-}^{\beta k}(\rho, t)}{\partial t}, \tag{C.12}$$

with

$$\begin{aligned}\frac{\partial m^{\beta k}_{\alpha^\pm}(\rho,t)}{\partial t} &= \pm\,\eta\theta^{(2)k} I_{31}(k) T^{\alpha\beta} \pm \frac{\eta\rho\theta^{(2)k}}{Q^{kk}}\left(I_{32}(k,k) - I_{31}(k)M^{\alpha k}\right) m^{\beta k}(\rho,t) \\ &+ \sum_{j\neq k}\Big[-\eta\theta^{(2)k}\theta^{(2)j} I_{22}(k,j) T^{\alpha\beta} \\ &\quad + \frac{\eta\rho\theta^{(2)k}\theta^{(2)j}}{Q^{kk}Q^{jj} - Q^{kj2}}\Big\{\left(-I_3(k,k,j) + I_{32}(k,j)M^{\alpha k}\right)\left(Q^{jj}m^{\beta k}(\rho,t) - Q^{kj}m^{\beta j}(\rho,t)\right) \\ &\qquad + \left(-I_3(k,j,j) + I_{32}(k,j)M^{\alpha j}\right)\left(Q^{kk}m^{\beta j}(\rho,t) - Q^{kj}m^{\beta k}(\rho,t)\right)\Big\}\Big] \\ &- \eta\theta^{(2)k}\theta^{(2)k} I_{22}(k,k) T^{\alpha\beta} - \lambda\eta m^{\beta k}(\rho,t) \\ &- \frac{\eta\rho\theta^{(2)k}\theta^{(2)k}}{Q^{kk}}\left(I_3(k,k,k) - I_{22}(k,k)M^{\alpha k}\right) m^{\beta k}(\rho,t). \end{aligned} \tag{C.13}$$

Note how, in order to close the equation, we introduced an additional order parameter $T^{\alpha\beta} = \sum_{r=1}^{D} \frac{\mu_r^\alpha \mu_r^\beta}{D}$, which is entirely defined by the overlap of the means of the mixture under consideration and is therefore a constant of motion. For compactness, we defined the multidimensional integrals I of the activation function over the local fields as:

$$I_3(k,j,l) = \mathop{\mathbb{E}}_{\alpha}\, g'(a^k)a^k(a^j) \tag{C.14a}$$

$$I_{32}(k,j) = \mathop{\mathbb{E}}_{\alpha}\, g'(a^k)a^j \tag{C.14b}$$

$$I_{31}(k) = \mathop{\mathbb{E}}_{\alpha}\, g'(a^k) \tag{C.14c}$$

$$I_{22}(k,j) = \mathop{\mathbb{E}}_{\alpha}\, g'(a^k)g(a^j). \tag{C.14d}$$

The update of q can similarly be decomposed as a sum over the different Gaussian clusters:

$$\frac{\partial q^{kl}(\rho,t)}{\partial t} = \sum_{\alpha\in\mathcal{S}(+)} \mathcal{P}_\alpha \frac{\partial q^{kl}_{\alpha^+}(\rho,t)}{\partial t} + \sum_{\alpha\in\mathcal{S}(-)} \mathcal{P}_\alpha \frac{\partial q^{kl}_{\alpha^-}(\rho,t)}{\partial t} \tag{C.15}$$

The linear contribution to this update is directly computed by using Eq. C.6 and is similar to the one for $m^{\beta k}(\rho,t)$. The quadratic contribution is obtained by using the fact that the projected inputs $\tilde{x}_\tau$ have a correlation of order $1/\sqrt{D}$ with the local fields. Therefore, to leading order, this contribution is given by terms of the form:

$$\frac{\eta^2}{D^2}\sum_\tau \rho_\tau \mathop{\mathbb{E}}_{\alpha}\left[g'(a^k)g'(a^l)g(a^i)g(a^j)\tilde{x}_\tau^2\right] = \frac{\eta^2}{D}\mathop{\mathbb{E}}_{\alpha}\left[g'(a^k)g'(a^l)g(a^i)g(a^j)\right]\left(\sum_\tau \rho_\tau \mathop{\mathbb{E}}_{\alpha}\left[\tilde{x}_\tau^2\right]\right) + O(D^{-3/2}) \tag{C.16}$$

Let us define the constant of motion $\chi^\alpha = \frac{1}{D}\sum_\tau \rho_\tau^2$, then the quadratic term in the update for $q_{\alpha^\pm}$ is given by:

$$\eta^2 \chi^\alpha \theta^{(2)k} v^l \left(I_{42}(k,l) \mp 2\sum_j \theta^{(2)j} I_{43}(k,l,j) + \sum_{jb} \theta^{(2)j}\theta(2)b I_4(k,l,j,b) \right). \tag{C.17}$$

The multidimensional integrals I are given by:

$$I_4(k,l,j,b) = \mathop{\mathbb{E}}_{\alpha} g'(a^k)g'(a^l)g(a^j)g(a^b) \quad I_{43}(k,l,j) = \mathop{\mathbb{E}}_{\alpha} g'(a^k)g'(a^l)g(a^j) \quad I_{42}(k,l) = \mathop{\mathbb{E}}_{\alpha} g'(a^k)g'(a^l).$$

Finally, the full equation of motion of q is written:

$$\begin{aligned}
\frac{\partial q^{kl}(\rho,t)_{\alpha^\pm}}{\partial t} &= \pm \eta\theta^{(2)k} I_{31}(k) m^{\alpha k}(\rho,t) \pm \frac{\eta\rho\theta^{(2)k}}{Q^{kk}} \left(I_{32}(k,k) - I_{31}(k) M^{\alpha k} \right) q^{kl}(\rho,t) \\
&\quad + \sum_{j\neq k} \Big[-\eta\theta^{(2)k}\theta^{(2)j} I_{22}(k,j) m^{\alpha k}(\rho,t) \\
&\qquad + \frac{\eta\rho\theta^{(2)k}\theta^{(2)j}}{Q^{kk}Q^{jj} - Q^{kj2}} \Big\{ \left(-I_3(k,k,j) + I_{32}(k,j) M^{\alpha k} \right) \left(Q^{jj} q^{kl}(\rho,t) - Q^{kj} q^{jl}(\rho,t) \right) \\
&\qquad\qquad + \left(-I_3(k,j,j) + I_{32}(k,j) M^{j\alpha} \right) \left(Q^{kk} q^{jl}(\rho,t) - Q^{kj} q^{kl}(\rho,t) \right) \Big\} \Big] \\
&\quad - \eta\theta^{(2)k}\theta^{(2)k} I_{22}(k,k) m^{\alpha k}(\rho,t) - \frac{\eta\rho\theta^{(2)k}\theta^{(2)k}}{Q^{kk}} \left(I_3(k,k,k) - I_{22}(k,k) M^{\alpha k} \right) q^{kl}(\rho,t) \\
&\quad + (k \leftrightarrow l) \\
&\quad + \eta^2 \chi^\alpha \theta^{(2)k} v^l \left(I_{42}(k,l) \mp 2\sum_j \theta^{(2)j} I_{43}(k,l,j) + \sum_{jb} \theta^{(2)j}\theta^{(2)b} I_4(k,l,j,b) \right) - 2\lambda\eta q^{kl}(\rho,t).
\end{aligned} \tag{C.18}$$

Update for the second layer weights The update of the second layer weights is also decomposed into the contribution of the different Gaussian clusters and follows from taking the expectation of Eq. 4.37b on the GM distribution 4.36:

$$\begin{aligned}
\mathbb{E}\,\frac{\mathrm{d}\theta^{(2)k}(t)}{\mathrm{d}t} &= \sum_{\alpha\in\mathcal{S}(+)} \mathcal{P}_\alpha \frac{\mathrm{d}\theta^{(2)k}_{\alpha^+}(t)}{\mathrm{d}t} + \sum_{\alpha\in\mathcal{S}(-)} \mathcal{P}_\alpha \frac{\mathrm{d}\theta^{(2)k}_{\alpha^-}(t)}{\mathrm{d}t}, \\
\frac{\mathrm{d}\theta^{(2)k}_{\alpha^\pm}(t)}{\mathrm{d}t} &= \pm\eta \mathop{\mathbb{E}}_{\alpha} g(a^k) - \eta\sum_j \theta^{(2)j} \mathop{\mathbb{E}}_{\alpha} g(a^k)g(a^j) - \eta\lambda\theta^{(2)k}
\end{aligned} \tag{C.19}$$

Equations C.13,C.18 and C.19 suffice to fully characterise the training dynamics, in the limit of high dimensions and online-learning, of a 2LNN trained on an arbitrary Gaussian mixture with $O(1)$ clusters each having mean μ^α and same covariance matrix Ω.

Agreement with Numerical Simulations Here, we verify the agreement of the ODEs

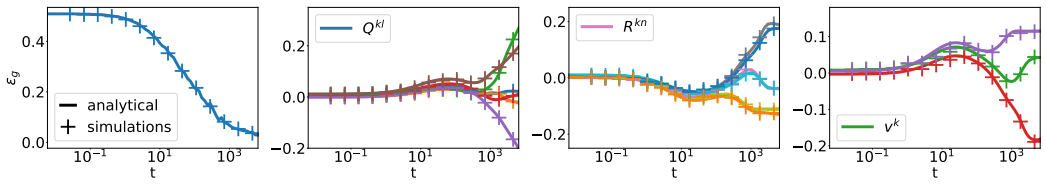


Figure C.1: Agreement between simulations and ODEs when training a $K = 3$ 2LNN on a Gaussian mixture in $D = 800$ dimensions, with four Gaussian clusters with random covariance matrix Ω and random means (i.e. $\mu_r \sim \mathcal{N}(0,1)$) for sigmoidal activation function. We verify that even at finite ($D = 800$) input dimension, the analytical prediction agree well with simulations. $\eta = 0.1$, $g(x) = \text{erf}\left(x/\sqrt{2}\right)$, weights initialised with s.t.d. $\sigma_0 = 1$. Monte-Carlo integration performed with 10^{-4} samples.

derived above with simulation of 2LNN trained via online SGD. To start, Fig. C.1 displays the dynamics of a $K = 3$ network trained on a Gaussian mixture with 4 Gaussian clusters having covariance matrix $\Omega = {}^{F^T F}/\sqrt{D}$ and means μ^α, where the elements of both the matrix $F \in \mathbb{R}^{D\times D}$ and the means are sampled i.i.d. from a standard Gaussian distribution. The agreement between analytical prediction, given by integration of the ODEs, and simulations is very good both in the dynamics of the test error, of the order parameters and of the second layer weights.

Note that the equations of motion describe the evolution of the densities m and q averaged over the input distribution. The agreement between this evolution and simulations justifies, at posteriori, the implicit assumption that the stochastic part of the SGD increment 4.37 can be neglected in the $D\to\infty$ limit. We can thus conjecture that in the $D\to\infty$ limit, the stochastic process defined by the SGD updates converges to a deterministic process parametrised by the continuous time variables $t \equiv {}^N/_D$. We further add that the proof of this conjecture is not a straight-forward extension of the one of Goldt et al. [114] for i.i.d. inputs since here, one must take into account the density of the covariance matrix.

The ODEs are valid for generic covariance matrix and means. Thus, they can be used to analyse the role of data structure in training 2LNNs. Although we leave a detailed analysis for future work, Fig. C.2 gives an example of how this could be done in the case where a $K = 3$ 2LNN is trained on a GM obtained from the FashionMnist dataset. The GM is obtained by computing the means $\bar{x}^\alpha$ and covariance matrix $\text{Cov}_\alpha(x,x)$ of each class in the dataset and assigning a label $+1$ or -1 to the different classes, as is commonly done in binary classification tasks. One could, for example, assign label $y = +1$ to the sneakers, boots, sandals, trousers and shorts categories and $y = -1$ to all others. Extending our analysis to C-class classification is straight forward and follows the analysis of Yoshida

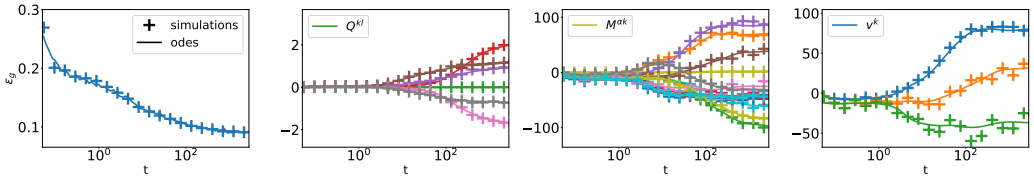


Figure C.2: Agreement between simulations and ODEs when training a $K = 3$ 2LNN with sigmoidal activation function on a Gaussian mixture obtained from the FashionMNIST dataset. The analytical dynamics obtained by integrating the ODEs agrees well with the given by simulations. Thus, the ODEs provide a tool to study the importance of datastructure in training 2LNN. We leave this study for future work. $D = 784$, $\eta = 0.1$, $g(x) = \text{erf}\left(x/\sqrt{2}\right)$, weights initialised with s.t.d. $\sigma_0 = 0.1$. Monte-Carlo integration performed with 10^4 samples.

et al. [322]. The inputs are then sampled from a GM where the cluster's mean are given by $\bar{x}^\alpha$ and the covariance matrix Ω is the mean covariance of all classes: $\Omega = {}^1/_{\text{n}_{\text{classes}}} \sum_\alpha \text{Cov}_\alpha(x, x)$. Note the similarity between this procedure and *linear discriminant analysis* commonly used in statistics. The agreement between simulations and analytical predictions is again very good, both at the level of the test error and of the order parameters.

Simplified ansatz to solve the ODEs for the XOR-like mixture

Here, we detail the procedure, introduced in Sec. 4.4, used to find the long time $t \to \infty$ performance of 2LNN by making an *ansatz* on the form of the order parameters that solve the fix point equations. The motivation for doing so, as argued in of the main text, is that integrating the ODEs is numerically expensive as it requires evaluating various multidimensional integrals and the number of equations to integrate scales as K^2. In order to extract information about the asymptotic performances of the network, one can look for a fix point of the ODEs. However, the number of coupled equations to be solved, also scales quadraticaly with K and is already 26 for a $K = 4$ student. The trick is to make an *ansatz*, with fewer degrees of freedom, on the order parameters that solve the equations. Used in this way, the ODEs have generated a wealth of analytical insights into the dynamics and the performance of 2LNN in the classical teacher-student setup [38, 39, 117, 268–270, 321, 322]. In all these works though, an important simplification occured because the means of the local fields were all zero by construction. This simplification allowed the fixed points to be found analytically in some cases. Here, the means of the local fields evaluated over individual Gaussians in the mixture are *not* zero, so we have to resort to numerical means to find the fixed points of the ODEs.

Consider, for example, a 2LNN trained on the XOR-like mixture of Fig. 4.1. The Gaussian clusters have covariance $\Omega = \sigma^2 \mathbb{I}$ and means chosen as in the left-hand-side diagram of Fig. 4.1, with the remaining $D-2$ components set to 0. This configuration leads to the constrain $\theta^{(1)k} \cdot \mu^{\pm 0} = -\theta^{(1)k} \cdot \mu^{\pm 1}$ that, in terms of overlap matrices, forces $M^{\alpha k} = -M^{\alpha+1 k}$ thus halving the number of free parameters in M. It is also clear that the only components of the weight vectors which contribute to the error are those in the plane spanned by the means of the mixture. The additional $D-2$ components can be taken to 0: i.e. $\theta_r^{(1)k} = 0$ for $r = 2, .., D$. This condition allows to decompose the weight vectors as:

$$w_s^k = \sum_r \frac{\theta_r^{(1)k} \mu_r^{+0}}{\sqrt{D}} \frac{\mu_s^{+0}}{\sqrt{D}} + \sum_r \frac{\theta_r^{(1)k} \mu_r^{-0}}{\sqrt{D}} \frac{\mu_s^{-0}}{\sqrt{D}} \tag{C.20}$$

This decomposition, fully constrains the overlap matrix Q^{kl} in terms of $M^{\alpha k}$:

$$\begin{aligned} Q^{kl} &= \sigma^2 \sum_s \frac{w_s^k w_s^l}{D} \\ &= \sigma^2 \left(\sum_r \frac{\theta^{(1)k} \mu_r^{+0}}{D} \right) \left(\sum_a \frac{w_a^l \mu_a^{+0}}{D} \right) \frac{|\mu^{+0}|^2}{D} + \sigma^2 \left(\sum_r \frac{\theta_r^{(1)k} \mu_r^{-0}}{D} \right) \left(\sum_a \frac{w_a^l \mu_a^{-0}}{D} \right) \frac{|\mu^{-0}|^2}{D} \\ \implies Q^{kl} &= \sigma^2 \left(M^{+0k} M^{+0l} + M^{-0k} M^{-0l} \right), \end{aligned} \tag{C.21}$$

where we used that in the XOR-like mixture, $|\mu^\alpha| = \sqrt{D}$ and $\mu^{0+} \cdot \mu^{0-} = 0$. From the symmetry between the positive and negative sign clusters of the mixture, in the fix point configuration, for every weight having norm $|w|$ and at an angle α with the mean of a positive cluster, there is a corresponding weight of the same norm, at an angle α with a negative mean. I.e. the angles of the weight vectors $\theta^{(1)k}$ to the means μ^α, as well as the norms of the weights, are 2×2 equal (one for the positive sign cluster and the other for the negative sign one). This constrains further half the number of free parameters in the overlap matrix M, which are down to $4K/(2 \times 2) = K$. The second layer weights v are fully constrained by requiring the output of the student to be ± 1 when evaluated on the means. Putting everything together, one is left with K equations to solve for the $K/2$ angles and the $K/2$ norms, or equivalently, for the K free parameters in the overlap matrix M. The agreement between the solution found by solving this reduced set of equations and simulations is displayed both in Fig. 4.3, where we use it to predict the evolution of the test error with the $L2-$regularisation constant.

C.3 Transforming a Gaussian mixture with random features

The distribution of random features is still a mixture

Given an input $x = (x_i) \in \mathbb{R}^D$ sampled from the distribution 4.36, we consider the *feature* vector $z = (z_i) \in \mathbb{R}^N$

$$z_i = \psi\left(\sum_{r=1}^{D} \frac{1}{\sqrt{D}} F_{ir} x_r\right), \tag{C.22}$$

where $F \in \mathbb{R}^{D\times P}$ is a random projection matrix and $\psi : \mathbb{R}\to\mathbb{R}$ is an element-wise non linearity. The distribution of z can be computed as:

$$\begin{aligned} p_z(z) &= \int_{\mathbb{R}^D} dx p_x(x) \delta\left(z - \psi\left(\frac{xF}{\sqrt{D}}\right)\right) \\ &= \int_{\mathbb{R}^D} dx \sum_y q(y) q(x|y) \delta\left(z - \psi\left(\frac{xF}{\sqrt{D}}\right)\right) \\ &\equiv \sum_y q(y) \sum_{\alpha\in\mathcal{S}(y)} \mathcal{P}_\alpha p_z^\alpha(z), \end{aligned} \tag{C.23}$$

with

$$p_z^\alpha(z) = \int_{\mathbb{R}^D} dx \delta\left(z - \psi\left(\frac{xF}{\sqrt{D}}\right)\right) \mathcal{N}_\alpha\left(\frac{\mu^\alpha}{\sqrt{D}}, \Omega^\alpha\right) \tag{C.24}$$

Crucially, the distribution of the features z is still a mixture of distributions. We can thus restrict to studying the transformation of a Gaussian random variable

$$x_r = \frac{\mu_r}{\sqrt{D}} + \sigma w_r \tag{C.25}$$

where w_r is a standard Gaussian. The scaling of μ_r and σ is chosen according to which regime (low or high SNR) one chooses to study. We aim at computing the distribution, in particular the two first moments, of the feature z defined in Eq. C.22. By construction, the random variables u_i are Gaussian with first two moments:

$$\mathbb{E}\, u_i = \frac{1}{\sqrt{D}} \tilde{\mu}_i, \qquad \tilde{\mu}_i \equiv \sum_r \frac{F_{ir}\mu_r}{\sqrt{D}} \tag{C.26}$$

$$\mathbb{E}\, u_i u_j = \frac{1}{D} \tilde{\mu}_i \tilde{\mu}_j + \sigma^2 \sum_r \frac{F_{ir}F_{jr}}{D} \tag{C.27}$$

Low signal-to-noise ratio

Here we compute the statistics of the features, for general activation function, in the low signal to noise regime, for which $|\mu|/\sqrt{D}\sim O(1)$ and $\sigma\sim O(1)$ so that the Gaussian clusters are a distance of order 1 away from the origin.

The mean of z_i can be written as:

$$\mathbb{E}\, z_i = \mathbb{E}\, \psi(u_i) = \mathbb{E}\, \psi\left(\frac{\tilde{\mu}_i}{\sqrt{D}} + \sigma\zeta\right), \tag{C.28}$$

where $\zeta \in \mathbb{R}$ is a standard Gaussian variable. In the scaling we work in, where P and D are send to infinity with their ratio fixed, and $|\mu|/\sqrt{D} \sim O(1)$, $\tilde{\mu}_i/\sqrt{D}$ is of order $O(1/\sqrt{D})$. Thus, the activation function ψ can be expanded around $\sigma\zeta$:

$$\mathbb{E}\, z_i = \mathbb{E}\, \psi\left(\sigma\zeta\right) + \frac{\tilde{\mu}_i}{\sigma\sqrt{D}} \mathbb{E}\, \zeta\psi\left(\sigma\zeta\right) + O\left(\frac{1}{D}\right). \tag{C.29}$$

where we used integration by part to find $\sigma\mathbb{E}\, \partial\psi\left(\sigma\zeta\right) = \mathbb{E}\, \zeta\psi\left(\sigma\zeta\right)$.

For the covariance matrix, we separate the computation of the diagonal from the off-diagonal. Starting with the diagonal elements:

$$\begin{aligned} \mathbb{E}\left[z_i^2\right] =& \mathbb{E}\left[\left(\psi\left(\sigma\zeta\right) + \frac{\tilde{\mu}_i}{\sigma\sqrt{D}}\partial\psi\left(\sigma\zeta\right)\right)^2\right] \\ =& \mathbb{E}\left[\psi\left(\sigma\zeta\right)^2\right] + \frac{\tilde{\mu}_i}{\sigma\sqrt{D}} \mathbb{E}\, \zeta\psi^2\left(\sigma\zeta\right) + O\left(\frac{1}{D}\right) \end{aligned} \tag{C.30}$$

where, once gain, integration by parts was used to obtain $2\sigma\mathbb{E}\left[\psi\left(\sigma\zeta\right)\partial\psi\left(\sigma\zeta\right)\right] = \mathbb{E}\, \zeta\psi^2\left(\sigma\zeta\right)$.

In order to compute the off-diagonal elements, we note that different components of u are weakly correlated since $\mathrm{Cov}(u_i, u_j) = \sigma^2 \sum_r F_{ir}F_{jr}/D \sim O(1/\sqrt{D})$. We can therefore apply formula Eq. B.4 for weakly correlated variables:

$$\begin{aligned} \mathop{\mathbb{E}}_{(z_i,z_j)} z_i z_j =& \mathop{\mathbb{E}}_{(u_i,u_j)} \psi(u_i)\psi(u_j) \\ =& \mathop{\mathbb{E}}_{u_i} \psi(u_i) \mathop{\mathbb{E}}_{u_j} \psi(u_j) + \frac{1}{\sigma^2}\sum_r \frac{F_{ir}F_{jr}}{D} \mathop{\mathbb{E}}_{u_i} u_i\psi(u_i) \mathop{\mathbb{E}}_{u_j} u_j\psi(u_j) + O(\frac{1}{D}) \end{aligned} \tag{C.31}$$

where the averages are now over the one dimensional distributions of $u_i \sim \mathcal{N}\left(\frac{\tilde{\mu}_i}{\sqrt{D}}, \sigma^2\right)$. We can now replace in the above $u_i = \frac{\tilde{\mu}_i}{\sqrt{D}} + \sigma\zeta$ and keep only leading order terms:

$$\begin{aligned} \mathrm{Cov}(z_i, z_j) &= \sum_r \frac{F_{ir}F_{jr}}{\sigma^2 D}\left(\psi(\sigma\zeta + \frac{\mu_i}{\sqrt{D}})(\sigma\zeta + \frac{\mu_i}{\sqrt{D}})\right)\left(\psi(\sigma\zeta + \frac{\mu_j}{\sqrt{D}})(\sigma\zeta + \frac{\mu_j}{\sqrt{D}})\right) \\ &= \sum_r \frac{F_{ir}F_{jr}}{\sigma^2 D}\left(\psi(\sigma\zeta)\sigma\zeta + O(\frac{1}{\sqrt{D}})\right)\left(\psi(\sigma\zeta)\sigma\zeta + O(\frac{1}{\sqrt{D}})\right) \\ &= \mathbb{E}\left[\zeta\psi(\sigma\zeta)\right]^2 \sum_r \frac{F_{ir}F_{jr}}{D} + O\left(\frac{1}{D}\right) \quad \text{, for } i \neq j. \end{aligned} \tag{C.32}$$

Thus yielding the final result:

$$\mathsf{Cov}(z_i, z_j) = \mathbb{E}\left[\zeta\psi(\sigma\zeta)\right]^2 \sum_r \frac{F_{ir}F_{jr}}{D} \quad \text{, for } i \neq j. \tag{C.33}$$

We define the constants a, b and c as in Eq. 4.47 and d as:

$$a = \mathbb{E}\,\psi\left(\sigma\zeta\right), \quad b = \mathbb{E}\,\zeta\psi\left(\sigma\zeta\right), \quad c^2 = \mathbb{E}\left[\psi\left(\sigma\zeta\right)^2\right], \quad d^2 = \mathbb{E}\,\zeta\psi^2\left(\sigma\zeta\right) \tag{C.34}$$

These definitions together with Eq. C.29, Eq. C.30 and Eq. C.33 lead to the statistics of Eq. 4.48 and Eq. 4.49:

$$\begin{aligned} \mathbb{E}\, z_i &= a + \frac{\tilde{\mu}_i}{\sigma\sqrt{D}} b \\ \mathsf{Cov}(z_i, z_j) &= \begin{cases} c^2 - a^2 + \underbrace{\frac{\tilde{\mu}_i}{\sigma\sqrt{D}}\left(d^2 - 2ab\right)}_{O(\frac{1}{\sqrt{D}})\text{: subleading if } c^2 - a^2 > 0} + O(\frac{1}{D}) & \text{, if } i = j \\ b^2 \sum_r \frac{F_{ir}F_{jr}}{D} + O(\frac{1}{D}) & \text{, if } i \neq j \end{cases} \end{aligned} \tag{C.35}$$

The above shows that, the transformation of the means is only linear and in the low SNR regime, the XOR-like mixture of Fig. 4.1, is transformed into a XOR-like mixture in feature space which cannot be learned by linear regression. Note, that the performance of linear regression on the features is equivalent to its performance on inputs $\tilde{z} \in \mathbb{R}^P$ sampled from a Gaussian equivalent model defined as:

$$\tilde{z}_i = \mathbb{E}\, z_i + \sum_j \Omega_{ij}^{1/2} \zeta_j, \tag{C.36}$$

where $\mathsf{Cov}(z_i, z_j) \equiv \sum_k \Omega_{ik}^{1/2}\Omega_{jk}^{1/2}$ and $\zeta \in \mathbb{R}^P$ is a random vector with components sampled i.i.d. from a standard Gaussian distribution.

ReLU features

In the case of Relu activation function i.e. $\psi(x) = \max(0, x)$, the mean and the covariance of the features can be evaluated analytically for all SNR regimes. The distribution of the features within each cluster is given by a modified Gaussian: the probability mass of the Gaussian on the negative real axis is concentrated at the origin while the distribution on the positive axis is unchanged.

In particular, the integral to obtain the mean of z can be computed analytically and is given by:

$$\mathbb{E}\, z_i = \mathbb{E}\ \mathrm{ReLU}(u_i) = \sigma\left[\frac{\tilde{\rho}_i}{2}\left(1 + \mathrm{erf}(\frac{\tilde{\rho}_i}{\sqrt{2}})\right) + \frac{1}{\sqrt{2\pi}} e^{-\tilde{\rho}_i^2/2}\right], \tag{C.37}$$

where we defined:

$$\tilde{\rho}_i \equiv \frac{\tilde{\mu}_i}{\sqrt{D}\sigma} = \frac{\sum_r F_{ir}\mu_r}{D\sigma}.$$

The covariance is once again computed by separating the diagonal terms from the off-diagonal ones. The integral to obtain the diagonal terms has an analytical expression found to be:

$$\mathrm{Cov}(z_i, z_i) = \mathbb{E}\left[\mathrm{ReLU}(u_i)^2\right] - \mathbb{E}\left[\mathrm{ReLU}(u_i)\right]^2 = \sigma^2\left[\frac{\tilde{\rho}_i}{\sqrt{2\pi}}e^{-\tilde{\rho}_i^2/2} + \frac{1}{2}(\tilde{\rho}_i^2+1)(1+\mathrm{erf}(\frac{\tilde{\rho}_i}{\sqrt{2}})) - (\mathbb{E}\, z_i)^2\right] \tag{C.38}$$

For the off-diagonal components, we again use that the covariance of the different components of the u_i is of order $1/\sqrt{D}$ as $\mathrm{Cov}(u_i, u_j) = \sigma^2 \sum_r F_{ir}F_{jr}/D$. Therefore, to evaluate $\mathbb{E}\left[\mathrm{ReLU}(u_i)\,\mathrm{ReLU}(u_j)\right]$, we can use the result Eq. B.4 for weakly correlated variables with $\epsilon M_{12} = \sigma^2 \sum_r F_{ir}F_{jr}/D$. Then to leading order in $\frac{1}{D}$, one finds:

$$\begin{aligned}\mathrm{Cov}(z_i, z_j) &= \mathbb{E}\left[\mathrm{ReLU}(u_i)\,\mathrm{ReLU}(u_j)\right] - \mathbb{E}\left[\mathrm{ReLU}(u_i)\right]\mathbb{E}\left[\mathrm{ReLU}(u_j)\right] \\ &= \sum_r \frac{F_{ir}F_{jr}}{\sigma^2 D}\mathbb{E}\left[u_i\,\mathrm{ReLU}(u_i)\right]\mathbb{E}\left[u_j\,\mathrm{ReLU}(u_j)\right] + O(\frac{1}{D})\end{aligned} \tag{C.39}$$

where the expectations above are over one dimensional distributions $u_i \sim \mathcal{N}\left(\sum_r \frac{F_{ir}\mu_r}{D}, \sigma^2\right)$. The integrals have an analytical closed form expression, which yields the final result for the covariance:

$$\mathrm{Cov}(z_i, z_j) = \begin{cases} \sigma^2\left[\frac{\tilde{\rho}_i}{\sqrt{2\pi}}e^{-\tilde{\rho}_i^2/2} + \frac{1}{2}(\tilde{\rho}_i^2+1)(1+\mathrm{erf}(\frac{\tilde{\rho}_i}{\sqrt{2}})) - (\mathbb{E}\, z_i)^2\right] & \text{if } i = j, \\ \frac{\sigma^2}{4}\frac{\sum_r F_{ir}F_{jr}}{D}\left(1+\mathrm{erf}(\frac{\tilde{\rho}_i}{\sqrt{2}})\right)\left(1+\mathrm{erf}(\frac{\tilde{\rho}_j}{\sqrt{2}})\right) + O(\frac{1}{D}) & \text{if } i \neq j \end{cases} \tag{C.40}$$

Relation with the kernel

As discussed in Sec. 4.5 of the main text , the performances of kernel methods can be studied by using the convergence of RF to a kernel in the $\gamma \to \infty$ limit taken after the $D, P \to \infty$ limit [253]:

$$K(x, y) = \frac{1}{P}\sum_{i=1}^{P}\mathop{\mathbb{E}}_{F}\left[\psi\left(\sum_{r=1}^{D}\frac{x_r F_{ir}}{\sqrt{D}}\right)\psi\left(\sum_{s=1}^{D}\frac{y_s F_{is}}{\sqrt{D}}\right)\right],$$

In the low SNR regime where $|\mu|/\sqrt{D} \sim O(1)$, the action of the kernel is essentially linear. The three constants a, b and c, defined in Eq. 4.47, can be expressed equivalently in terms of the activation function ψ or of the kernel. Consider ω_1, $\omega_2 \in \mathbb{R}$, two i.i.d. standard Gaussian random variables, and denote by angle

brackets $\langle\cdot\rangle$ the expectation over w_1, w_2. Then, by the definition $K(x,y)$ one has:

$$\langle K(\sigma\omega_1, \sigma\omega_1)\rangle = \frac{1}{P}\sum_{i=1}^{P}\langle\psi(u_i)^2\rangle = c^2 \tag{C.41}$$

where $u \in \mathbb{R}^P$ is the random vector whose moments are defined in Eq. C.26 and we used the element wise convergence of $\psi(\sum_r F_{ir}x_r)\psi(\sum_r F_{is}y_s)/P$ to its expected value [253]. Similarly, one has:

$$\langle K(\sigma\omega_1, \sigma\omega_2)\rangle = \frac{1}{P}\sum_{i=1}^{P}\langle\psi(u_i)\rangle^2 = a^2 \tag{C.42}$$

Finally, for b one has to perform a linear expansion of the kernel around the noise variable $\sigma\omega_1$:

$$\begin{aligned}
\langle K(\sigma\omega_1 + \frac{\mu}{\sqrt{D}}, \sigma\omega_2 + \frac{\mu}{\sqrt{D}})\rangle =& \frac{1}{P}\sum_{i=1}^{P}\langle \mathop{\mathbb{E}}_{F}\left[\psi(u_{i1} + \sum_r \frac{\mu_r F_{ir}}{D})\rangle\langle\psi(u_{i2} + \sum_r \frac{\mu_r F_{ir}}{D})\right]\rangle \\
=& \frac{1}{P}\sum_{i=1}^{P}\langle\psi(u_i)\rangle^2 + \sum_{rs}\frac{\mu_r\mu_s}{D^2}\sum_{i=1}^{P}\mathop{\mathbb{E}}_{F}\frac{F_{ir}F_{is}}{P}\langle\psi'(u_i)\rangle^2 \\
=& a^2 + \frac{1}{D\sigma^2}b^2, \\
\implies b^2 =& D\sigma^2\left(-a^2 + \langle K(\sigma\omega_1 + \frac{\mu}{\sqrt{D}}, \sigma\omega_2 + \frac{\mu}{\sqrt{D}})\rangle\right)
\end{aligned} \tag{C.43}$$

These expressions allow to express the statistical properties of the features z, and to asses the performance of RF and kernel methods, directly in terms of the kernel without requiring the explicit form of the activation function.

For completeness, we give the analytical expression of the kernel corresponding to ReLU random features, i.e. $\psi(x) = \max(0, x)$:

$$K_{\text{ReLU}}(x,y) = \frac{|x||y|}{8\pi D}\left\{2|\sin(\theta)| + \cos(\theta)\left(\pi + 2\,\text{Arctan}\left(\frac{\cos(\theta)}{|\sin(\theta)|}\right)\right)\right\}, \tag{C.44}$$

where we defined the angle θ between the two vectors $x, y \in \mathbb{R}^D$ such that $|x|, |y| > 0$:

$$\theta = \frac{x\cdot y}{|x||y|} \tag{C.45}$$

From Eq. C.44, one sees that in case of ReLU activation function, the kernel is an angular kernel, i.e. it depends on the angle between x and y^*.

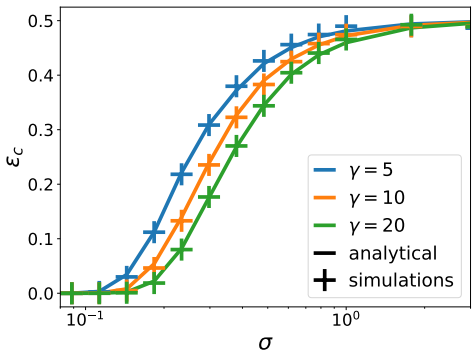


Figure C.3: Agreement between analytical predictions and simulations of the classification error of RF trained on the XOR like mixture of Fig. 4.1 for increasing σ and various γ. The analytical predictions are obtained by computing the moments of the features z using Eq. C.37 and Eq. C.40. These are then used to obtain, first the asymptotic solution of RF via Eq. C.51, which is in turn plugged in Eq. C.54. The final result is then given by Eq. C.53. Simulation results are obtained by training online an RF network with $\eta = 0.1$ until convergence. *Parameters:* $D = 800$, $\eta = 0.1$, $P = \gamma D$, $|\mu|/\sqrt{D} = 1$.

C.4 Final test error of random features

This section details the computations leading to Eq. 4.43 and Eq. 4.45 allowing to obtain the asymptotic performances of RF trained via online SGD on a mixture of Gaussian distribution.

Applying random features on $x \in \mathbb{R}^D$ sampled from the distribution 4.36 is equivalent to performing linear regression on the features $z \in \mathbb{R}^P$ with covariance Ω^z and mean μ_z. In the following, for clarity, we assume the features are centred, so that $\mu_z = 0$, extending the computation to non centred features is straight-forward. The means of the individual clusters are however non zero. The output of the network at a step t in training is given by $\phi(w)_t = \sum_{i=1}^P w_i z_{ti}/\sqrt{P}$. We train the network in the online limit by minimising the squared loss between the output of the network and the label y_t. The pmse is given by:

$$\text{pmse}(w) = \frac{1}{2}\mathbb{E}\left(y - \sum_{i=1}^P \frac{w_i z_i}{\sqrt{P}}\right)^2 = \frac{1}{2} + \sum_{i,j=1}^P \frac{w_i w_j}{2P}\Omega^z_{ij} - \sum_{i=1}^P \frac{w_i}{\sqrt{P}}\Phi_i, \tag{C.46}$$

where we introduced the *input-label* covariance $\Phi_i \equiv \mathbb{E}\left[z_i y\right]$.

The expectation of the SGD update over the distribution of z is thus:

$$\mathbb{E}\, \mathrm{d}w_i = -\frac{\eta}{\sqrt{P}} \mathbb{E}\, z_i \left(\sum_{j=1}^{P} \frac{w_j z_j}{\sqrt{P}} - y \right) = \frac{\eta}{\sqrt{P}} (\Phi_i - \sum_{j=1}^{P} \frac{w_j}{\sqrt{P}} \Omega^z_{ij}) \tag{C.47}$$

Importantly, both the pmse and the average update only depend on the distribution of the features through the covariance matrix of the features Ω^z and the input label covariance Ψ.

To make progress, consider the eigen-decomposition of the covariance matrix Ω^z:

$$\Omega^z_{ij} = \frac{1}{P} \sum_{\tau} \rho_\tau \Gamma_{\tau i} \Gamma_{\tau j}, \tag{C.48}$$

where ρ_τ are the eigenvalues and Γ_τ their corresponding eigenvectors.

Define the rotation of W and Φ into this eigenbasis:

$$\tilde{\theta}^{(1)}_\tau \equiv \frac{1}{\sqrt{P}} \sum_{i=1}^{P} \Gamma_{i\tau} w_i \quad \tilde{\Phi}_\tau \equiv \frac{1}{\sqrt{P}} \sum_{i=1}^{P} \Gamma_{i\tau} \Phi_i \tag{C.49}$$

In this basis, the SGD update for the different components of $\tilde{\theta}^{(1)}$ decouple. One finds a recursive equation in which each mode evolves independently from the others:

$$\mathbb{E}\, \mathrm{d}\tilde{\theta}^{(1)}_\tau = \frac{\eta}{\sqrt{P}} (\tilde{\Phi}_\tau - \frac{1}{\sqrt{P}} \sum_{\tau=1}^{P} \rho_\tau \tilde{\theta}^{(1)}_\tau) \tag{C.50}$$

Thus, the fix point $\tilde{\hat{W}}$ such that $\mathbb{E}\, \mathrm{d}\tilde{w}_\tau = 0$, can be found explicitly:

$$\rho_\tau \tilde{\hat{w}}_\tau = \sqrt{P} \tilde{\Phi}_\tau.$$

Rotating back in the original basis one finds the asymptotic solution for W as:

$$\hat{w}_i = \sum_{\tau; \rho_\tau > 0} \frac{1}{\rho_\tau} \Gamma_{i\tau} \tilde{\Phi}_\tau. \tag{C.51}$$

The asymptotic test error is thus given by:

$$\mathrm{pmse}_{t \to \infty} = \frac{1}{2} \left(1 - \sum_{\tau} \frac{\tilde{\Phi}^2_\tau}{\rho_\tau} \right) \tag{C.52}$$

Asymptotic classification error From the solution of Eq. C.51 for the asymptotic solution found by linear regression, one can obtain the asymptotic classification

error performed by random features as:

$$\epsilon_{ct\to\infty} = \mathbb{E}\,\Theta(-ya) = \sum_{\alpha} \mathcal{P}_\alpha \mathop{\mathbb{E}}_{\alpha}\,\Theta(-ya), \tag{C.53}$$

where $\Theta : \mathbb{R}\to\mathbb{R}$ is the Heaviside step function and we defined $a \equiv \sum_{i=1}^{P} \hat{w}_i z_i/\sqrt{P}$. Introducing the local field a allows to transform the high dimensional integral over the features z into a low-dimensional (in this case one dimensional) expectation over the local field. The Gaussian equivalency theorem of Goldt et al. [118] shows that even though the z are not Gaussian, to leading order in $1/P$, the average Eq. 4.45, only depends on the first two moments a, defined as:

$$M_\alpha = \mathop{\mathbb{E}}_{\alpha}[a] = \sum_{i=1}^{P} \frac{\hat{w}_i\, \mathbb{E}_\alpha[z_i]}{\sqrt{P}} \quad Q_\alpha = \mathop{\mathsf{Cov}}_{\alpha}[a,a] = \sum_{i=1}^{P} \frac{\hat{w}_i\hat{w}_j}{P}\, \mathop{\mathsf{Cov}}_{\alpha}(z,z) \tag{C.54}$$

These moments can be computed analytically from the statistics of the features computed in Sec. C.3 and from the optimal weights W obtained in Eq. C.51. The classification error, Eq. C.53, can thus be evaluated by means of a one dimensional integral over the distribution of a.

$$\epsilon_{ct\to\infty} = \sum_{\alpha} \mathcal{P}_\alpha \int \frac{da}{\sqrt{2\pi Q_\alpha}} \Theta(-y_\alpha) e^{-\frac{1}{2Q_\alpha}(a-M_\alpha)^2} \tag{C.55}$$

$$= \frac{1}{2}\left(1 - \sum_{\alpha} \mathcal{P}_\alpha y_\alpha \,\mathrm{erf}\left(\frac{M_\alpha}{\sqrt{2Q_\alpha}}\right)\right) \tag{C.56}$$

C.5 The three-cluster model

Similar to the analysis of the XOR-like mixture of Fig. 4.6, we analyse a data model with three clusters that was the subject of several recent works [80, 177, 203, 220, 221]. The Gaussian mixture in input space can be seen in the first column of Fig. C.4. The means of both positive clusters are set to 0 while the means of the negative sign clusters have first component $\pm\mu_0$ and all other $D-1$ components 0. The mixture after random feature transformation is displayed in the second column and the third and fourth column show the performance of a 2LNN, respectively, a random feature network, trained via online SGD, on this problem. Here again, we build on the observation that overparametrisation does not impact performances and train a $K = 10$ 2LNN in order to increase the number of runs that converged. The three rows, are as before, three different SNR regimes, they are in order the **low**, **high** and **mixed** SNR regime.

The phenomenology observed in the XOR-like mixture carries through here. In

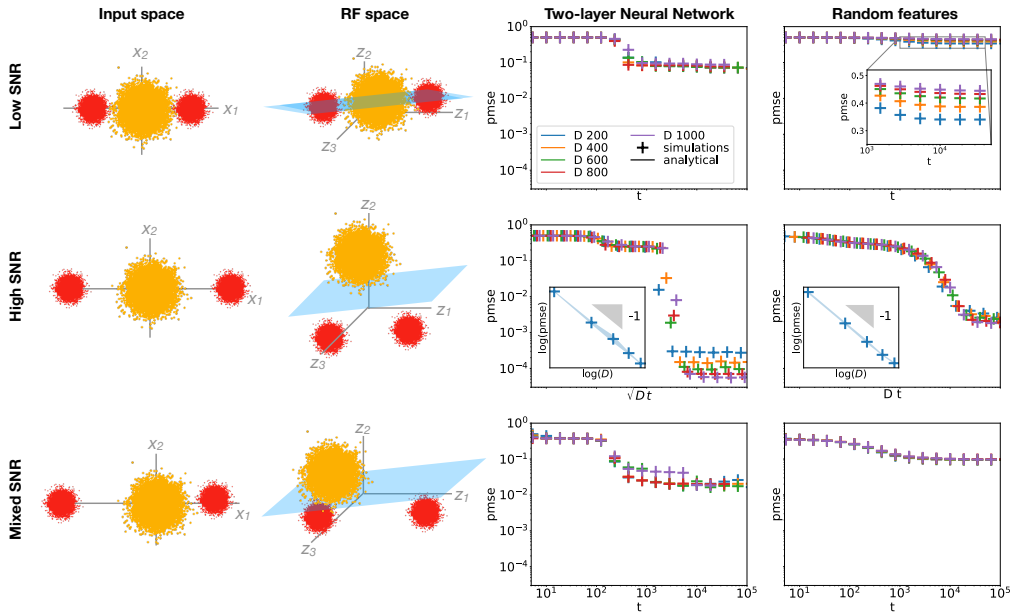


Figure C.4: We compare the performance of 2LNN with $K = 10$ hidden units and ReLU activation function (**third** column) to the ones of random features (**fourth** column) on the three cluster problem with different signal-to-noise ratios. The **right** sketches the input space distribution and the **second** column the transformed distribution in feature space. 2LNN can considerably outperform RF in all three regimes. In the *low* SNR regime (**top**), the action of the random features is essentially linear inducing their performance to be as bad as random guessing in the $D \to \infty$ limit. In the *high* SNR regime (**middle**), instead, both networks manage to learn the task. In the *mixed* SNR regime (**bottom**), the distance between opposite sign clusters remains of orther one in feature space inducing the RF performances to be unchanged with D. We plot the test error as a function of time. *Parameters:* $D = 1000$, $\eta = 0.1$, $\sigma^2 = 0.05$, $P = 2D$ for random features, $K = 10$ for 2LNN.

the low SNR regime, $\mu_0 = \sqrt{D}$ (**top row**), the 2LNN can learn the problem and its performance remains constant with increasing input dimension. On the other hand, in this regime, the transformation performed by the random features is only linear in the large D limit. Consequently, the RF performances degrade with increasing D and are as bad as random chance in the limit of infinite input dimension. In the *high* SNR regime instead (**second row**), where $\mu_0 = D$, the mixtures becomes well separated in feature space allowing RF to perform well. Both the performance of 2LNN and RF improve as the clusters in the mixture become more separated. The *mixed* SNR regime (**bottom row**) is obtained by setting one of the negative sign clusters a distance $\sqrt{D}$ from the origin while maintaining the other one at a distance 1. Here, the random feature perform a non trivial transformation of the far away cluster while its action on the nearby cluster is linear. Hence, in feature space,

one of the negative clusters remains close to the positive clusters while the other is well separated. The RF thus achieve a test error which is better than random but still worse that that of the 2LNN. Its error is constant with increasing D since it is dominated by the "spill-over" of the negative cluster into the positive cluster at the origin.

Lastly, let us comment that in all our work, we did not add a bias to the model. Adding a bias, does not change the conclusion that small 2LNN considerably outperform RF. In fact, the learning curves are only slightly modified. This is due to to our minimisation of the pmse when training the network, which, unlike classification loss that only cares about the *sign* of the estimate, penalises large differences between label $y = \pm 1$ and the output. For simplicity, we thus chose to remove the bias in our analysis, although including it is a straight forward operation.

Appendix D

Autoencoders as a tool to study feature learning

D.1 Online learning algorithms for PCA

We briefly review a number of unsupervised learning algorithms for principal component analysis, leading to Sanger's rule which is the inspiration for the truncated SGD algorithm of 4.7.3.

Setup

We consider inputs $x = (x_i), i = 1, \dots, D$ with first two moments equal to

$$\mathbb{E}\, x_i = 0 \quad \text{and} \quad \mathbb{E}\, x_i x_j = \Omega_{ij}. \tag{D.1}$$

We work in the thermodynamic limit $D \to \infty$.

Hebbian learning

The Hebbian learning rule allows to obtain an estimate of the leading PC by considering a linear linear model $y = a = \sum_i w_i x_i / \sqrt{D}$ with $i = 1, \dots, D$ and the loss function $\mathcal{L}(a) = -a^2$. It updates the weights as:

$$\mathrm{d}w_i^t = -\eta_t \nabla_{w_i} \mathcal{L}(a) = \frac{1}{\sqrt{D}} \eta_t y x_i \tag{D.2}$$

In other words, the Hebbian learning rule tries to maximise the second moment of the pre-activation a. That using this update we obtain an estimate of the first principal component makes intuitive sense. Consider the average over the inputs of the loss:

$$\mathbb{E}\, \mathcal{L}(a) = -\frac{1}{D} w_i \Omega_{ij} w_j. \tag{D.3}$$

If the weight vector e is equal to the τth eigenvector of Ω, then $\mathbb{E}\,\mathcal{L}(a) = -\rho_\tau/D$, and the loss is minimised by converging to the eigenvector with the larges eigenvalue. Also note, that similar to what we find in the main text, this result also suggests that the leading eigenvalue in the large-D limit should scale as $\rho_\tau \sim D$. The Hebbian rule has the well-known deficit that it imposes no bound on the growth of the weight vector. A natural remedy is to introduce some form of weight decay such that

$$\mathrm{d}w_i^t = \frac{\eta}{\sqrt{D}}(yx_i - \kappa w_i). \tag{D.4}$$

Oja's rule

[240] offers a smart choice for the weight decay constant κ. Consider the same linear model $y = a$ trained with a Hebbian rule and weight decay $\kappa = y^2$:

$$\mathrm{d}w_i^t = \frac{\eta}{\sqrt{D}}(yx_i - y^2 w_i) \tag{D.5}$$

The purpose of this choice can be appreciated from substituting in the linear model and setting the update rule to zero, which yields (dropping the constants)

$$\sum_j \Omega_{ij} w_j = \sum_{j,l} w_j \Omega_{j\ell} w_\ell w_i \tag{D.6}$$

which is precisely the eigenvalue equation for Ω.

Then, Oja's update rule is then derived from Hebb's rule by adding normalisation to the Hebbian update (D.2),

$$w_i^{t+1} = \frac{w_i^t + \eta y x_i}{\left(\sum \left[w_i^t + \eta y x_i\right]^p\right)^{1/p}} \tag{D.7}$$

with p integer (in Oja's original paper, $p = 2$.) This learning rule can be seen as a power iteration update. Substituting for y, we see that the numerator corresponds to, on average, repeated multiplication of the weight vector with the average covariance matrix. Expanding D.7 around $\eta = 0$ yields Oja's algorithm (while also assuming that the weight vector is normalised to one). One can show that Oja's rule indeed converges to the PC with the *largest* eigenvalue by allowing the learning rate to vary with time and imposing the mild conditions

$$\lim_{T\to\infty} \sum_{t=0}^{T} \eta_t \to \infty, \qquad \lim_{T\to\infty} \sum_{t=0}^{T} \eta_t^p < \infty \quad \text{for} \quad p > 1. \tag{D.8}$$

Sanger's rule [273] is also known as generalised Hebbian learning in the

literature and extends the idea behind Oja's rule to networks with multiple outputs $y^k = w_i^k x_i$. It allows estimation of the first few leading eigenvectors by using the update rule is given by

$$\mathrm{d}w_i^k = \frac{\eta}{\sqrt{D}} y^k \left(x_i - \sum_{\ell=1}^{k} y^\ell w_i^\ell \right). \tag{D.9}$$

Note that the second sum extends only to the k; the dynamics of the kth input vector hence only depends on weight vectors w^ℓ with $\ell < k$. This dependence is one of the similarities of Sanger's rule with the Gram-Schmidt procedure for orthogonalising a set of vectors (cf. sec. 0.6.4 of Horn and Johnson [137]). The dynamics of Sanger's rule in the online setting were studied by Biehl and Schlösser [37], Schlösser et al. [278]. Sanger's rule reduces to Oja's rule for $K = 1$.

D.2 Online learning in autoencoders

Here we derive the set of equations tracking the dynamics of shallow non-linear autoencoders trained in the one-pass limit of stochastic gradient descent (SGD). These equations are derived for Gaussian inputs $x \in \mathbb{R}^D$ drawn from the generative model 4.53, but as we discuss in 4.7.4, they also capture accurately the dynamics of training on real data.

D.2.1 Statics

The starting point of the analysis is the definition of the test error (4.52) and the identification of order parameters, i.e. low dimensional overlaps of high dimensional vectors.

$$\begin{aligned} \text{pmse} &\equiv \frac{1}{D} \sum_i \mathbb{E}(x_i - \hat{x}_i)^2 \qquad\qquad (D.10) \\ &= \frac{1}{D} \operatorname{Tr} \Omega + \frac{1}{D} \sum_i \sum_{k,\ell} v_i^k v_i^\ell \mathbb{E}\, g\left(\sum_i \frac{w_i^k x_i}{\sqrt{D}} \right) g\left(\sum_i \frac{w_i^\ell x_i}{\sqrt{D}} \right) - 2 \frac{1}{D} \sum_i \sum_k v_i^k x_i g\left(\sum_i \frac{w_i^k x_i}{\sqrt{D}} \right). \end{aligned}$$

First we introduce the local fields corresponding to the encoder and decoder's weights,

$$a^k \equiv \sum_i \frac{w_i^k x_i}{\sqrt{D}} \qquad \nu^k \equiv \sum_i \frac{v_i^k x_i}{D}, \tag{D.11}$$

as well as the order parameter tracking the overlap between the decoder weights:

$$T_0^{k\ell} \equiv \frac{1}{D}\sum_i v_i^k v_i^\ell. \tag{D.12}$$

Note the unusual scaling of v^k with D; the intuition here is that in shallow autoencoders, the second-layer weights will be strongly correlated with the eigenvectors of the input-input covariance, and hence with the inputs, requiring the scaling $1/D$ instead of $1/\sqrt{D}$. This scaling is also the one that yields a set of self-consistent equations in the limit $D\to\infty$. The generalisation error becomes

$$\text{pmse} = \frac{1}{D}\operatorname{Tr}\Omega + \sum_{k,\ell} T_0^{k\ell}\mathbb{E}\, g(a^k)g(a^\ell) - 2\sum_k \mathbb{E}\, \nu^k g(a^k). \tag{D.13}$$

Crucially, the local fields a and ν are jointly Gaussian since the inputs are Gaussian. Since we have $\mathbb{E}\, a^k = \mathbb{E}\, \nu^k = 0$, the pmse can be written in terms of the second moments of these fields only:

$$T_1^{k\ell} \equiv \mathbb{E}\, \nu^k \nu^\ell = \frac{1}{D^2}\sum_{i,j} v_i^k \Omega_{ij} v_j^\ell, \tag{D.14}$$

$$Q_1^{k\ell} \equiv \mathbb{E}\, a^k a^\ell = \frac{1}{\sqrt{D}}\sum_{i,j} w_i^k \Omega_{ij} w_j^\ell, \tag{D.15}$$

$$R_1^{k\ell} \equiv \mathbb{E}\, \nu^k a^\ell = \frac{1}{D^{3/2}}\sum_{i,j} v_i^k \Omega_{ij} w_j^\ell. \tag{D.16}$$

The full expression of $\mathbb{E}\, g(a^k)g(a^\ell)$ and $\mathbb{E}\, \nu^k g(a^k)$ in terms of the order parameters is given, for various activation functions, in App B.2. Note the different scalings of these overlaps with D. These are a direct consequence of the different scaling of the local fields a^k and ν^k. In order to derive the equations of motion, it is useful to introduce the decomposition of Ω in its eigenbasis:

$$\Omega_{rs} = \frac{1}{D}\sum_{\tau=1}^{D} \Gamma_{s\tau}\Gamma_{r\tau}\rho_\tau. \tag{D.17}$$

The eigenvectors $\Gamma_{s\tau}$ are normalised as $\sum_i \Gamma_{\tau i}\Gamma_{\tau' i} = D\delta_{\tau\tau'}$ and $\sum_\tau \Gamma_{\tau i}\Gamma_{\tau j} = D\delta_{ij}$. We further define the rotation of any $z_i \in \{w_i^k, v_i^k, x_i\}$ onto this basis as $z_\tau \equiv {}^1/\sqrt{D}\sum_{s=1}^D \Gamma_{\tau s} z_s$. The order parameters are then written:

$$T_1^{k\ell} = \frac{1}{D^2}\sum_\tau \rho_\tau v_\tau^k v_\tau^\ell, \qquad R_1^{k\ell} = \frac{1}{D^{3/2}}\sum_\tau \rho_\tau w_\tau^k v_\tau^\ell, \qquad Q_1^{k\ell} = \frac{1}{D}\sum_\tau \rho_\tau w_\tau^k w_\tau^\ell. \tag{D.18}$$

D.2.2 Derivation of dynamical equations

At the μth step of training, the SGD update for the rotated weights reads

$$\mathrm{d}w_\tau^k \equiv \left(w_\tau^k\right)_{\mu+1} - \left(w_\tau^k\right)_\mu = -\frac{\eta_W}{D}\frac{1}{\sqrt{D}}\sum_j^D\left(\sum_\ell^K v_j^\ell g(a^\ell) - x_j\right)v_j^k g'(a^k)x_\tau - \frac{\kappa}{D}w_\tau^k$$
$$= -\frac{\eta_W}{\sqrt{D}}\left(\sum_\ell^K T_0^{\ell k} g(a^\ell) - \nu^k\right)g'(a^k)x_\tau - \frac{\kappa}{D}w_\tau^k, \quad \text{(D.19)}$$
$$\mathrm{d}v_\tau^k = -\frac{\eta_V}{D}g(a^k)\left(\sum_m^K v_\tau^m g(a^m) - x_\tau\right) - \frac{\kappa}{D}v_\tau^k. \quad \text{(D.20)}$$

To keep notation concise, we drop the weight decay term in the following analysis. We can now compute the update of the various order parameters by inserting the above into the definition of the order parameters. The stochastic process described by the resulting equations concentrates, in the $D\to\infty$ limit, to its expectation over the input distribution. By performing the average over a fresh sample x, we therefore obtain a closed set of deterministic ODEs tracking the dynamics of training in the high-dimensional limit. We also show in simulations that these are able to capture well the dynamics also at finite dimension $D\sim 500$.

Update of $\mathbf{T_0}$ the overlap of the decoder's weights
Let us start with the update equation for $T_0^{k\ell} = {}^1/_D\sum_\tau v_\tau^k v_\tau^\ell$ which is easily obtained by using the sgd update (D.19).

$$T_{0\mu+1}^{k\ell} - T_{0\mu}^{k\ell} = \frac{1}{D}\sum_\tau dv_\tau^k v_\tau^\ell + v_\tau^k dv_\tau^\ell = -\frac{\eta_V}{D}\left(\sum_a g(a^k)g(a^a)T_0^{a\ell} + \nu^\ell g(a^k)\right) + (k\leftrightarrow\ell) \quad \text{(D.21)}$$

By taking the expectation over a fresh sample x and the $D, N\to\infty$ limit, we obtain a continuous in the normalised number of steps $s = {}^\mu/_D$, which we interpret as a continuous time variable, as:

$$\frac{\partial T_0^{k\ell}}{\partial s} = -\eta_V\left(\sum_a \mathbb{E}\, g(a^k)g(a^a)T_0^{a\ell} + \mathbb{E}\,\nu^\ell g(a^k)\right) + (k\leftrightarrow\ell) \quad \text{(D.22)}$$

This equation requires to evaluate $I_{22}\equiv\mathbb{E}\, g(a^k)g(a^k)$, $I_{21}\equiv\mathbb{E}\,\nu^\ell g(a^k)$, which are two-dimensional Gaussian integrals with covariance matrix given by the order parameters. We give their expression in B.2. We also note that the second order term $\propto \mathrm{d}v^k\mathrm{d}v^\ell$ is sub-leading in the high dimensional limit we work in.

Update of T_1

The update equation for $T_1^{k\ell} = {}^1/_{D^2}\sum_\tau \rho_\tau v_\tau^k v_\tau^\ell$ is obtained similarly as before by using the SGD update (D.19) and taking the high-dimensional limit.

$$\frac{\partial T_1^{k\ell}}{\partial s} = -\eta_V \left(\sum_a \mathbb{E}\, g(a^k) g(a^a) T_1^{al} + \sum_\tau \rho_\tau \frac{v_\tau^\ell}{D^2} \mathbb{E}\, g(a^k) x_\tau \right) + (k \leftrightarrow \ell) \tag{D.23}$$

To make progress, we have to evaluate $\mathbb{E}\, g(a^k) x_\tau$. For this purpose it is crucial to notice that the rotated inputs x_τ are weakly correlated with the local fields:

$$\begin{aligned} \mathbb{E}\, a^k x_\tau &= \frac{1}{D} \sum_{i,j} w_i^k \Omega_{ij} \Gamma_{\tau j} = \frac{1}{\sqrt{D}} \rho_\tau w_\tau^k, \\ \mathbb{E}\, \nu^k x_\tau &= \frac{1}{D^{3/2}} \sum_{i,j} v_i^k \Omega_{ij} \Gamma_{\tau j} = \frac{1}{D} \rho_\tau v_\tau^k. \end{aligned} \tag{D.24}$$

Then, we can compute the expectation $\mathbb{E}\, g(a^k) x_\tau$ using the results of **??** , i.e. Eq. (B.4) with $f(x) = g(x)$ and $h(y) = y$, thus obtaining:

$$\mathbb{E}\, g(a^k) x_\tau = \frac{\mathbb{E}\, a^k g(a^k)}{Q_1^{kk}} \frac{\rho_\tau w_\tau^k}{\sqrt{D}} \tag{D.25}$$

Inserting the above expression in the update of T_1 yields:

$$\frac{\partial T_1^{k\ell}}{\partial s} = \eta_V \left(\frac{\mathbb{E}\, a^k g(a^k)}{Q_1^{kk}} \sum_\tau \rho_\tau^2 \frac{v_\tau^\ell w_\tau^k}{D^{5/2}} - \sum_a \mathbb{E}\, g(a^k) g(a^a) T_1^{al} \right) + (k \leftrightarrow \ell) \tag{D.26}$$

Notice, that in this equation we have the the appearance of $\sum_\tau \rho_\tau^2 \frac{v_\tau^\ell w_\tau^k}{D^{5/2}}$, a term which cannot be simply expressed in terms of order parameters. Similar terms appear in the equation for Q_1 and R_1. To close the equations, we are thus led to introduce order parameter densities in the next step.

Order parameters as integrals over densities

To proceed, we introduce the densities $q(\rho, s), r(\rho, s)$ and $t(\rho, s)$. These depend on ρ and on the normalised number of steps $s = {}^\mu/_D$:

$$\begin{aligned} q^{k\ell}(\rho, s) &= \frac{1}{D\epsilon_\rho} \sum_\tau w_\tau^k w_\tau^\ell \mathbf{1}, \\ r^{k\ell}(\rho, s) &= \frac{1}{D^{3/2}\epsilon_\rho} \sum_\tau v_\tau^k w_\tau^\ell \mathbf{1}, \\ t^{k\ell}(\rho, s) &= \frac{1}{D^2\epsilon_\rho} \sum_\tau v_\tau^k v_\tau^\ell \mathbf{1}, \end{aligned} \tag{D.27}$$

where $\mathbb{1}(.)$ is the indicator function and the limit $\epsilon_\rho \to 0$ is taken after the thermodynamic limit. The order parameters are obtained by integrating these densities over the spectrum of the input-input covariance matrix:

$$Q_1^{k\ell} = \frac{1}{D}\int \mathrm{d}\rho p_\Omega(\rho)\rho q^{k\ell}(\rho,s),\ R_1^{k\ell} = \frac{1}{D}\int \mathrm{d}\rho p_\Omega(\rho)\rho r^{k\ell}(\rho,s),\ T_1^{k\ell} = \frac{1}{D}\int \mathrm{d}\rho p_\Omega(\rho)\rho t^{k\ell}(\rho,s). \tag{D.28}$$

It follows that tracking the dynamics of the functions q, r and t, we obtain the dynamics of the overlaps $\mathrm{Q}_1, \mathrm{R}_1$ and T_1.

Dynamics of $t(s,\rho)$

The dynamics of t is given straightforwardly from D.26 as:

$$\frac{\partial t^{k\ell}(\rho,s)}{\partial s} = \eta_V\left(\frac{\mathbb{E}\, a^k g(a^k)}{Q_1^{kk}}\tilde{\rho} r^{\ell k}(\rho,s) - \sum_a t^{\ell a}(\rho,s)\mathbb{E}\, g(a^k)g(a^a)\right) + (k \leftrightarrow \ell) \tag{D.29}$$

where we defined the *rescalled eigenvalue* $\tilde{\rho} \equiv \rho/D$. Here again, straightforward algebra shows that the second order term is sub-leading in the limit of high input dimensions.

Dynamics of $q(s,\rho)$

In a similar way, we compute the dynamical equation for $q(s,\rho)$ by using the encoder's weight's update in D.19 and taking the expectation over the input distribution.

$$\frac{\partial q^{k\ell}(\rho,s)}{\partial s} = \frac{\eta_W}{\sqrt{D}}\rho\sum_\tau \mathbf{1}\, w_\tau^\ell\left(\mathbb{E}\, g'(a^k)v^k x_\tau - \sum_a T^{ak}\mathbb{E}\, g'(a^k)g(a^a)x_\tau\right) + (k \leftrightarrow \ell) \tag{D.30}$$

In the above, we have the appearance of the expectations:

$$\mathcal{A}_\tau^{ka} = \mathbb{E}\left[g'(a^k)g(a^a)x_\tau\right]/;\mathcal{B}_\tau^k = \mathbb{E}\left[g'(a^k)g(a^k)x_\tau\right]/;\mathcal{C}_\tau^k = \mathbb{E}\left[v^k g'(a^k)x_\tau\right] \tag{D.31}$$

To compute these, we use our results for weakly correlated variables from B.1 and obtain:

$$\begin{aligned}
\mathcal{A}_\tau^{ka} =& \frac{1}{Q_1^{kk}Q_1^{aa} - \left(Q_1^{ka}\right)^2}\left(Q_1^{aa}\mathbb{E}\left[g'\left(a^k\right)a^k g(a^a)\right]\frac{\rho_\tau w_\tau^k}{\sqrt{D}} - Q_1^{ka}\mathbb{E}\left[g'\left(a^k\right)a^a g(a^a)\right]\frac{\rho_\tau w_\tau^k}{\sqrt{D}}\right.\\
&\left. - Q_1^{ka}\mathbb{E}\left[g'\left(a^k\right)a^k g(a^a)\right]\frac{\rho_\tau w_\tau^a}{\sqrt{D}} + Q_1^{kk}\mathbb{E}\left[g'\left(a^k\right)a^a g(a^a)\right]\frac{\rho_\tau w_\tau^a}{\sqrt{D}}\right)\\
\mathcal{B}_\tau^k =& \frac{\mathbb{E}\left[g'\left(a^k\right)a^k g(a^k)\right]}{Q_1^{kk}}\frac{\rho_\tau w_\tau^k}{\sqrt{D}}\\
\mathcal{C}_\tau^k =& \frac{1}{Q_1^{kk}T_1^{kk} - \left(R_1^{kk}\right)^2}\left(T_1^{kk}\mathbb{E}\left[g'\left(a^k\right)a^k\nu^k\right]\frac{\rho_\tau w_\tau^k}{\sqrt{D}} - R_1^{kk}\mathbb{E}\left[g'\left(a^k\right)\nu^{k2}\right]\frac{\rho_\tau w_\tau^k}{\sqrt{D}}\right.\\
&\left. - R_1^{kk}\mathbb{E}\left[g'\left(a^k\right)a^k\nu^k\right]\frac{\rho_\tau v_\tau^k}{D} + Q_1^{kk}\mathbb{E}\left[g'\left(a^k\right)\nu^{k2}\right]\frac{\rho_\tau v_\tau^k}{D}\right)
\end{aligned} \tag{D.32}$$

Using these results in the equations for $q(\rho, s)$ we find:

$$\begin{aligned}
\frac{\partial q^{k\ell}(\rho,s)}{\partial s} =& \eta_W D\tilde{\rho}\left\{\frac{1}{Q_1^{kk}T_1^{kk} - \left(R_1^{kk}\right)^2}\left(T_1^{kk}\mathbb{E}\left[g'\left(a^k\right)a^k\nu^k\right]q^{kl}(\rho,t) - R_1^{kk}\mathbb{E}\left[g'\left(a^k\right)\nu^{k2}\right]q^{kl}(\rho,t)\right.\right.\\
&\left. - R_1^{kk}\mathbb{E}\left[g'\left(a^k\right)a^k\nu^k\right]r(\rho,t)^{kl} + Q_1^{kk}\mathbb{E}\left[g'\left(a^k\right)\nu^{k2}\right]r(\rho)^{kl}\right)\\
&- \sum_{a\neq k} T_0^{ka}\,\frac{1}{Q_1^{kk}Q_1^{aa} - \left(Q_1^{ka}\right)^2}\left(Q_1^{aa}\mathbb{E}\left[g'\left(a^k\right)a^k g(a^a)\right]q^{kl}(\rho,t)\right.\\
&\left. - Q_1^{ka}\mathbb{E}\left[g'\left(a^k\right)a^a g(a^a)\right]q^{kl}(\rho,t) - Q_1^{ka}\mathbb{E}\left[g'\left(a^k\right)a^k g(a^a)\right]q(\rho)^{al}\right.\\
&\left. + Q_1^{kk}\mathbb{E}\left[g'\left(a^k\right)a^a g(a^a)\right]q(\rho,t)^{al}\right)\\
&\left. - T_0^{kk}\,\frac{\mathbb{E}\left[g'\left(a^k\right)a^k g(a^k)\right]}{Q_1^{kk}}q^{kl}(\rho,t)\right\} + (l \leftrightarrow k)
\end{aligned} \tag{D.33}$$

Note, that as explained in the main text, in order to find a well defined $D\to\infty$ limit, we rescale the learning rate of the encoder's weights as $\eta_W = \eta/D$.

Dynamics of $r(s,\rho)$

Lastly, we obtain the equation for $r(\rho, s)$ in the same way as the two previous ones.

We use D.19 and evaluate the expectation over a fresh sample x.

$$\frac{\partial r^{k\ell}(\rho,s)}{\partial s} = \eta\left(\frac{1}{\epsilon_g D^{3/2}}\sum_\tau \mathbf{1} w_\tau^\ell \mathbb{E}\, x_\tau g(a^k) - \sum_a \frac{1}{\epsilon_g D^{3/2}}\sum_\tau \mathbf{1} w_\tau^\ell v_\tau^a \mathbb{E}\, g(a^k)g(a^a)\right) + \eta\left(\frac{1}{\epsilon_g D}\sum_\tau \mathbf{1} v_\tau^k \mathbb{E}\, g'(a^\ell)\nu^\ell x_\tau - \sum_a T_0^{al}\frac{1}{\epsilon_g D}\sum_\tau \mathbf{1} v_\tau^k \mathbb{E}\, g'(a^\ell)g(a^a)x_\tau\right) \tag{D.34}$$

We can evaluate the expectations as before (B.1). Thus, we obtain the equation of motion for $r(\rho,s)$ as:

$$\begin{aligned}\frac{\partial dr^{kl}(\rho,s)}{\partial s} =& \eta\tilde{\rho}\left\{\frac{1}{Q_1^{kk}T_1^{kk} - \left(R_1^{kk}\right)^2}\left(T_1^{kk}\mathbb{E}\left[g'\left(a^k\right)a^k\nu^k\right]r^{lk}(\rho,t) - R_1^{kk}\mathbb{E}\left[g'\left(a^k\right)\nu^{k2}\right]r^{lk}(\rho,t)\right.\right.\\ &\left. - R_1^{kk}\mathbb{E}\left[g'\left(a^k\right)a^k\nu^k\right]t^{kl}(\rho,t) + Q_1^{kk}\mathbb{E}\left[g'\left(a^k\right)\nu^{k2}\right]t^{kl}(\rho,t)\right)\\ &- \sum_{a\neq k} T_0^{ka}\frac{1}{Q_1^{kk}Q_1^{aa} - \left(Q_1^{ka}\right)^2}\left(Q_1^{aa}\mathbb{E}\left[g'\left(a^k\right)a^k g(a^a)\right]r^{lk}(\rho,t)\right.\\ &\quad - Q_1^{ka}\mathbb{E}\left[g'\left(a^k\right)a^a g(a^a)\right]r^{lk}(\rho,t)\\ &\quad - Q_1^{ka}\mathbb{E}\left[g'\left(a^k\right)a^k g(a^a)\right]r^{la}(\rho,t)\\ &\quad \left. + Q_1^{kk}\mathbb{E}\left[g'\left(a^k\right)a^a g(a^a)\right]r^{la}(\rho,t)\right)\\ &- T_0^{kk}\frac{\mathbb{E}\left[g'\left(a^k\right)a^k g(a^k)\right]}{Q_1^{kk}}r^{lk}(\rho,t)\Bigg\}\\ &+ \eta\left\{\frac{\mathbb{E}\, a^\ell g(a^\ell)}{Q_1^{ll}}\tilde{\rho}q(\rho,t) - \sum_a r^{ak}(\rho,t)\mathbb{E}\, g(a^\ell)g(a^a)\right\}\end{aligned} \tag{D.35}$$

D.2.3 Simplification of the equations for spiked covariance matrices

In the case of the synthetic dataset defined as $x = \mathbf{A}c + \xi$, the spectrum of the covariance matrix is decomposed into a *bulk* of small, i.e. $\mathcal{O}(1)$ eigenvalues, of continuous support, and a *few* outlier eigenvalues taking values of order $\mathcal{O}(D)$ (see 4.7). This allows to obtain M equations for controlling the evolution of the spikes modes and one equation controlling the evolution of the bulk for all order parameters. Indeed we can simplify the equations of motion of the bulk eigenvalues (i.e. those for which $\rho{\sim}O(1)$) by neglecting terms proportional to $\rho/D \ll 1$. Doing

so leads to:

$$\begin{aligned}
\frac{\partial t^{k\ell}(\rho,s)}{\partial s} &= -\eta \sum_a t^{\ell k}(\rho,s)\mathbb{E}\, g(a^k)g(a^a) + (k \leftrightarrow \ell) - \kappa t^{k\ell}(\rho,s) \\
\frac{\partial r^{kl}(\rho,s)}{\partial s} &= -\eta \sum_a \mathbb{E}\, g(a^k)g(a^a) r^{ak}(\rho,t) - \kappa r^{k\ell}(\rho,s) \\
\frac{\partial q^{k\ell}(\rho,s)}{\partial s} &= -\kappa q^{k\ell}(\rho,s).
\end{aligned} \tag{D.36}$$

We can define *bulk* order parameters, that take into account the contribution from the bulk modes as:

$$\begin{aligned}
\mathrm{T}_{\mathrm{bulk}} &= \int_{\rho\in\mathrm{bulk}} \mathrm{d}p_\Omega(\rho)\frac{\rho}{D}t(\rho),\ \mathrm{R}_{\mathrm{bulk}} = \int_{\rho\in\mathrm{bulk}} \mathrm{d}p_\Omega(\rho)\frac{\rho}{D}r(\rho), \\
\mathrm{Q}_{\mathrm{bulk}} &= \int_{\rho\in\mathrm{bulk}} \mathrm{d}p_\Omega(\rho)\frac{\rho}{D}q(\rho).
\end{aligned} \tag{D.37}$$

Note that even though the integrals involve $\rho/D \sim O(1/D)$ since we are integrating over a large number ($O(D)$) of modes, the result is of order 1. The equation for these overlaps are thus obtained as the integrated form of Eq. (D.36):

$$\begin{aligned}
\frac{\partial T^{k\ell}_{\mathrm{bulk}}}{\partial s} &= -\eta \sum_a T^{\ell k}_{\mathrm{bulk}}\mathbb{E}\, g(a^k)g(a^a) + (k \leftrightarrow \ell) - \kappa T^{k\ell}_{\mathrm{bulk}} \\
\frac{\partial R^{k\ell}_{\mathrm{bulk}}}{\partial s} &= -\eta \sum_a R^{ak}_{\mathrm{bulk}}(\rho,t)\mathbb{E}\, g(a^k)g(a^a) - \kappa R^{k\ell}_{\mathrm{bulk}} \\
\frac{\partial Q^{k\ell}_{\mathrm{bulk}}}{\partial s} &= -\kappa Q^{k\ell}_{\mathrm{bulk}}.
\end{aligned} \tag{D.38}$$

The order parameters can be decomposed into the contribution from the bulk eigenvalues and those of the spikes as:

$$\mathrm{T}_1 = \sum_{i\in\mathrm{outliers}}^{M} \tilde{\rho}_i t_i + T_{\mathrm{bulk}} \tag{D.39}$$

and similarly for Q_1 and R_1. The M spike modes t_i, r_i and q_i obey the full equations Eq. (D.29) , Eq. (D.34) and Eq. (D.33). In particular, it is clear from Eq. (D.38) that for any non-zero weight decay constant κ, the bulk contribution to Q_1 will decay to 0 in a characteristic time κ^{-1}. Further note that the equations for $\mathrm{T}_{\mathrm{bulk}}$ and $\mathrm{R}_{\mathrm{bulk}}$ result in an exponential decay of the bulk modes towards 0.

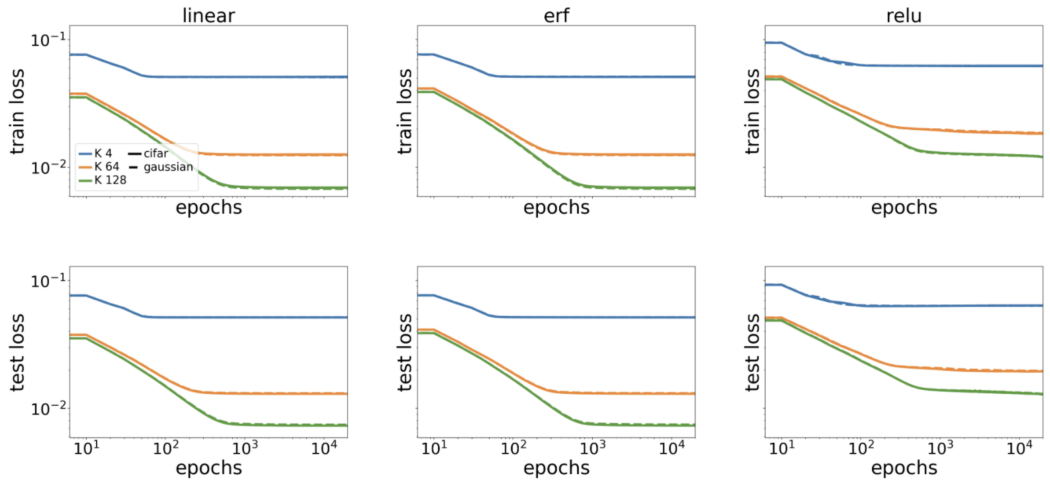


Figure D.1: **Gaussian equivalence for autoencoders** Train and test mean-squared error for linear, sigmoidal and relu autoencoders with different numbers of hidden neurons ($K = 4, 64, 128$ in blue, orange and green, resp.) The autoencoders were trained on CIFAR10 grayscale images (solid lines) or, starting from the same initial conditions, on Gaussian inputs with the same covariance (dashed lines). The agreement is essentially perfect. *Parameters:* 10k training samples, $D = 1024$, mini-batch size=1, $\eta = 1$.

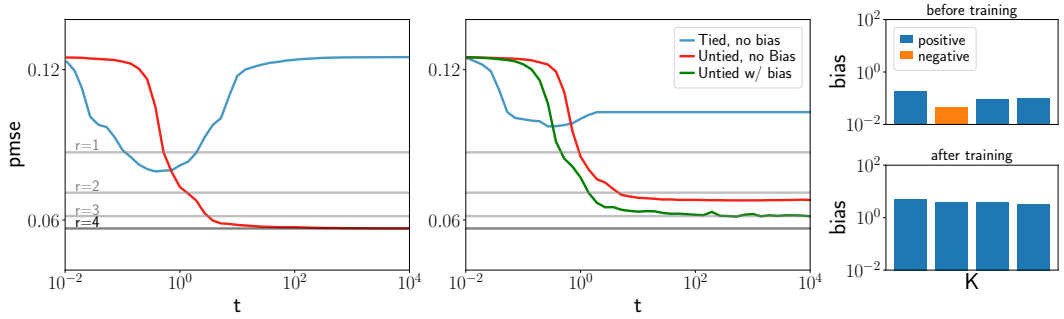


Figure D.2: **Results derived on synthetic data carry over to finite real datasets** **(a)** Also in real dataset, a sigmoidal AE requires untied weights to attain good performances. For an untied student, which asymptotically performs as PCA, the dynamics slow down at intermediary ranks, signalling a sequential learning of different modes. **(b)** In order to learn, a ReLU autoencoder requires training the bias in the encoder's weights. The asymptotic error is given by PCA plus a residual due to the biases. *Parameters:* CIFAR10 and FashionMNIST with 10k images. batch-size=1, $\eta = 1$.

D.3 Reduced equations for long-time dynamics of learning

In this section we illustrate how to use the equations of motion in order to study the long training time dynamics of shallow non-linear autoencoders. At sufficiently long times, we have seen that the weights of the network spam the leading PC

subspace of the covariance matrix. We restrict to the case in which the eigenvectors are recovered directly. The case in which a rotation of them is found follows straightforwardly. We also restrict to the matched case in which $K = M$.

Consider the following ansatz on the weights configuration: $w_i^k = {}^{\alpha_\omega}/\sqrt{D}\Gamma_i^k$ and $v_i^k = \alpha_v^k \Gamma_i^k$. We defined dynamical constants $\alpha_w^k, \alpha_v^k \in \mathbb{R}$ which control the norm of the weights. Using this ansatz, the order parameters take the form:

$$Q_1^{k\ell} = \alpha_w^{k2}\tilde{\rho}_k\delta^{k\ell} \qquad R_1^{k\ell} = \alpha_w^k\alpha_v^k\tilde{\rho}_k\delta^{k\ell} \qquad T_1^{k\ell} = \alpha_v^{k2}\tilde{\rho}_k\delta^{k\ell} \qquad T_0^{k\ell} = \alpha_v^{k2}\delta^{k\ell} \tag{D.40}$$

Using the above, we can rewrite the pmse as:

$$\text{pmse}(\alpha_w^k, \alpha_v^k) = \sum_k \left\{ T_0^{kk}\mathbb{E}\, g(a^k)^2 - 2\mathbb{E}\, \nu^k g(a^k) + \tilde{\rho}_k \right\} + \sum_{i>K} \tilde{\rho}_i \tag{D.41}$$

The first observation is that the generalisation error reaches a minimum when the term in bracket f^k is minimised. Minimising it leads to an equation $\alpha_v^{*k}(\alpha_w)$ at which $f^k(\alpha_v^{*k}(\alpha_w), \alpha_w) = 0$ and the pmse is equal to the PCA reconstruction error.

For a linear activation function, we have

$$\text{pmse}(\alpha_w^k, \alpha_v^k) = \sum_k \underbrace{\left\{ \alpha_v^{k2}\alpha_w^{k2}\tilde{\rho}_k - 2\alpha_w^k\alpha_v^k\tilde{\rho}_k + \tilde{\rho}_k \right\}}_{f_k(\alpha_v^k, \alpha_w^k)} + \underbrace{\sum_{i>K} \tilde{\rho}_i}_{\text{PCA reconstruction error}} \tag{D.42}$$

The pmse is a function only of the product $\alpha^{k2} \equiv \alpha_w^k\alpha_v^k$ and is minimal and equal to the PCA reconstruction error whenever:

$$\frac{\partial f_k(\alpha_v^k, \alpha_w^k)}{\partial \alpha_v^l} = 0 \Longrightarrow \alpha_v^{*k} = \frac{1}{\alpha_w^k}, \quad \forall k \qquad \text{and} \qquad f_k(\frac{1}{\alpha_w^k}, \alpha_w^k) = 0 \tag{D.43}$$

The above is nothing but the intuitive result that for a linear autoencoder, any rescaling of the encoder's weights can be compensated by the decoder's weights.

For a sigmoidal autoencoder, instead, the expressions of the integrals given in App. B.2, give:

$$\text{pmse}(\alpha_w^k, \alpha_v^k) = \sum_k \underbrace{\left\{ \frac{2}{\pi}\alpha_v^{k2}\frac{\alpha_w^{k2}\tilde{\rho}_k}{1+\alpha_w^{k2}\tilde{\rho}_k} - \sqrt{\frac{2}{\pi(1+\alpha_w^{k2}\tilde{\rho}_k)}}2\alpha_w^k\alpha_v^k\tilde{\rho}_k + \tilde{\rho}_k \right\}}_{f_k(\alpha_v^k, \alpha_w^k)} + \sum_{i>K} \tilde{\rho}_i \tag{D.44}$$

Differentiating f_K with respect to α_w^k, and requiring the derivative to be 0, we find

the equation:

$$\begin{aligned} 2\pi\left(1+\tilde{\rho}^k\alpha_\omega^{k2}\right)^2 =& 4\alpha_\omega^{k2}\alpha_v^{k2}\left(1+\tilde{\rho}^k\alpha_\omega^{k2}\right) \\ \implies \alpha_v^{k*2}(\alpha_w) =& \frac{\pi}{2}\left(\frac{1}{\alpha_w^{2k}}+\tilde{\rho}\right) \end{aligned} \tag{D.45}$$

We point out that this solution is independent of the sign of α_ω^k or α_v^k as recovering the eigenvectors or minus the eigenvectors is equivalent. Replacing this solution for α_ω^k in the expression for the pmse, we find $f^k(\alpha_v^{k*}(\alpha_w), \alpha_w^k) = 0$. The autoencoder reaches the same reconstruction error as the one achieved by PCA.

Ansatz in the equations of motion

The ansatz Eq. 4.66 results in order parameters of the form:

$$q^{kl}(\rho_m) = \frac{\alpha_w^{k2}}{D}\delta^{kl}\delta_m^k \qquad r^{kl}(\rho_m) = \frac{\alpha_w^k\alpha_v^k}{D}\delta^{kl}\delta_m^k \qquad t^{kl}(\rho_m) = \frac{\alpha_v^{k2}}{D}\delta^{kl}\delta_m^k \tag{D.46}$$

Inserting these expressions into the full dynamical equations of motion Eq. (D.33), Eq. (D.34) and Eq. (D.29), allows us to finnd a simplified set of $2K$ ODEs for the scaling constants α_v^k and α_w^k:

$$\begin{aligned} \frac{\partial\alpha_v^k}{\partial s} &= \frac{\tilde{\rho}^k\alpha_w^k}{\pi\left(\alpha_w^{k2}\tilde{\rho}+1\right)}\sqrt{2\pi\left(\alpha_w^{k2}\tilde{\rho}+1\right)-2\alpha_v^k\alpha_w^k} \\ \frac{\partial\alpha_w^k}{\partial s} &= \alpha_v^k\tilde{\rho}^k\sqrt{\frac{2}{\pi\left(\alpha_w^{k2}\tilde{\rho}+1\right)^3}}\left(1-\sqrt{\frac{2\left(\alpha_w^{k2}\tilde{\rho}+1\right)}{\pi\left(2\alpha_w^{k2}\tilde{\rho}+1\right)}}\alpha_v^k\alpha_w^k\right) \end{aligned} \tag{D.47}$$

The validity of the equations is verified in Fig. 4.9 where we compare the result of integrating them to simulations. We provide a Mathematica notebook which allows to derive these equations on the Github repository of this paper.

To validate our calculations for autoencoders with finite D, we trained an autoencoder from three distinct initial conditions: A *random* one, an *informed* configuration in which the weights are initialised as in Eq. 4.66 with $\alpha_v^k = \alpha_w^k = 1$ and a *perfect* one, in which additionally, α_v^k is initialised from the minimal pmse solution Eq. D.45. The first observation is that the pmse of the network started with *perfect* initial conditions remains at the PCA reconstruction error, thus validating Eq. D.45. Since we have a noiseless dataset, the PCA error is given by the numerical error, however we note that the same conclusion caries over to noisy datasets with finite PCA reconstruction error. In D.3, we plot the pmse of the randomly initialised and informed autoencoders on the left as well as the evolution of α_v^k and α_w^k, i.e. of the overlap between the weights and the eigenvectors during training, on the right.

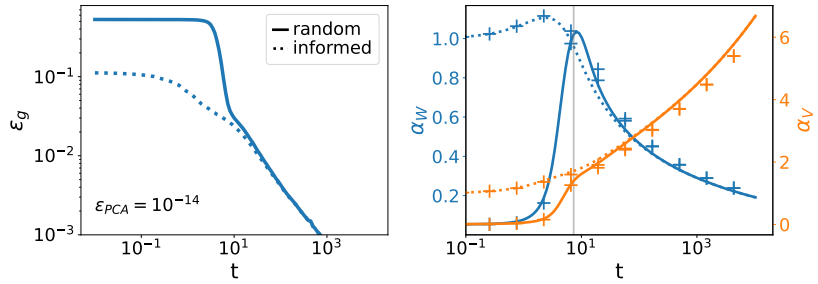


Figure D.3: **Learning in non-linear shallow AE occurs in two phases (a)** The pmse of an auto-encoder trained from random initial conditions first decays exponentially and then as a power law in time. **Dynamics of the scaling constants α_v and α_w** The reduced set of $2K$ ODEs match the results of simulations. The dynamics of an AE initialised randomly with random initial conditions become indistinguishable with those of an AE whose weights are initialised proportional to the PCs. This shows that the network first recovers the leading PCs subspace and then adjusts the norm of the weights in order to recover the linear regime.*Parameters:* $\eta = 1, K = 64, bs = 1$

We can see that the network with random initial weights learns in two phases: first the pmse decays exponentially until it reaches the error of the aligned network. Then, its pmse coincides with that of the informed network and the error decays as a power-law. This shows that during the first phase of exponential learning, the network recovers the leading PC subspace. This occurs early on in training. In the second phase, the networks adjust the norm of the weights to reach PCA performances. As discussed in the main text, this is achieved in the sigmoidal network by shrinking the encoder's weights, so as to recover the linear regime of the activation function. It keeps the norm of the reconstruction similar to the one of the input by and growing the decoder's weights.

Appendix E

Learning rate schedules and how they improve BP performance

E.1 Dynamics of the convex model

Here we give additional details and steps in the computations on the convex model of Sec. 5.1.2. The loss function is given by $\mathcal{L}(w) = \frac{\kappa}{2}w^2$. Integrating the Langevin equation (Eq. 5.1) from t_0 to t for w yields:

$$w(t) = \underbrace{w(t_0)e^{-\kappa\int_{t_0}^{t}\mathrm{d}\tau\eta(\tau)}}_{\bar{w}(t)} + \underbrace{\int_{t_0}^{t}\mathrm{d}t' e^{-\kappa\int_{t_0}^{t'}\mathrm{d}t\tau\eta(\tau)}\eta(t')\xi(t')}_{\delta w(t)}\,. \tag{E.1}$$

In order to obtain a typical realisation of the loss which does not depend on the optimisation noise ξ, we take the expectation over ξ. This gives for the loss $\mathcal{L}$:

$$\langle\mathcal{L}(t)\rangle = \frac{\kappa}{2}\left(\langle\bar{w}(t)^2\rangle + \langle\delta w(t)^2\rangle + 2\underbrace{\langle\bar{w}(t)\delta w(t)\rangle}_{0}\right) \tag{E.2}$$

$$= \frac{\kappa}{2}\left(\underbrace{w(t_0)^2e^{-2\kappa\int_{t_0}^{t}\mathrm{d}\tau\eta(\tau)}}_{\bar{\mathcal{L}}(t)} + \underbrace{2T\int_{t_0}^{t}\mathrm{d}t'\eta(t')^2e^{-2\kappa\int_{t'}^{t}\mathrm{d}\tau\eta(\tau)}}_{\delta\mathcal{L}(t)}\right) \tag{E.3}$$

The first term is an optimisation term while the second is the contribution of the noise inherent to the optimisation algorithm. Thus, to converge to the solution as quickly as possible, one has to find the trade-off between decreasing the impact of the noise term while not slowing down optimisation excessively. The ideal schedule is determined by requiring these two effects are comparable. Defining $\eta(t) = \eta_0/t$,

we obtain

$$\bar{\mathcal{L}}(t) \propto e^{-2\eta_0\kappa \log(t)} \propto t^{-2\eta_0\kappa} \tag{E.4}$$

$$\delta\mathcal{L}(t) \int_{t_0}^{t} \mathrm{d}t' \frac{1}{t'^2} \left(\frac{t'}{t}\right)^{2\eta_0\kappa} \propto 1/t. \tag{E.5}$$

If $\eta_0 > 1/2\kappa$, the loss is dominated by the noise term $\delta\mathcal{L}$ and decays as $1/t$. If $\eta_0 < 1/2\kappa$, the loss is dominated by the optimization term $\delta\mathcal{L}$ and decays as $t^{-2\eta_0\kappa}$.

E.2 Dynamics of the Sherrington-Kirkpatrick model

In this section, we provide derivations for the results obtained in the SK model.

Unplanted model

The loss function is given by:

$$\mathcal{L}(w) = -\frac{1}{\sqrt{N}} \sum_{i<j}^{N} J_{ij} w_i w_j. \tag{E.6}$$

Solving the dynamics

Following Cugliandolo and Dean [71], we express the spin configurations in the eigenbasis of J and define $w_\mu = w \cdot J_\mu / \sqrt{N}$ as the projection of w onto the eigenvector J_μ. x_μ evolves as:

$$\frac{\partial w_\mu(t)}{\partial t} = \eta(t) \left[(\mu - z(t)) w_\mu(t) + \xi_\mu(t) \right]. \tag{E.7}$$

Integrating this equation yields again two terms, one related to the optimisation and the second related to the noise:

$$\begin{aligned} w_\mu(t) = w_\mu(0)\, e^{-\int_0^t d\tau \eta(\tau)(\mu - z(\tau))} \\ + \int_0^t dt'' e^{-\int_{t''}^t d\tau' \eta(\tau')(z(\tau') - \mu)} \eta(t'') \xi_\mu(t''). \end{aligned} \tag{E.8}$$

In the $t \to \infty$ limit, a non-exploding $\bar{w}_\mu$ requires $\mu - z(t)$ to be negative for all μ in the support of ρ, implying $z(t) < 2$. We must also impose $z(t) \to_{t\to\infty} 2$, otherwise $w_\mu(t) \to 0\ \forall \mu$, in contradiction with the spherical constraint. To comply with these two requirements we define $z(t) = 2 - f(t)$, with $f(t) \to_{t\to\infty} 0$.

In the constant learning rate setup $\eta(t) = 1$ we know from [71] that $f(t) = 3/(4t)$. With $\eta(t) = \eta_0/t^\beta$, a natural ansatz is $f(t) = c/t^{1-\beta}$. To determine c, we

impose the spherical constraint:

$$\begin{aligned}1 &= \langle \int \frac{\mathrm{d}\mu}{N} \rho(\mu) w_\mu(t)^2 \rangle \\ &= t^{2c\eta_0} \int_{-2}^{2} \mathrm{d}\mu \sqrt{4-\mu^2} e^{2\eta_0(\mu-2)t^{1-\beta}} \\ &= t^{2c\eta_0 - 3(1-\beta)/2} \int_0^\infty \mathrm{d}\epsilon \sqrt{2\epsilon} e^{-2\eta_0 \epsilon} \propto t^{2c\eta_0 - 3(1-\beta)/2} \Rightarrow c = \frac{3(1-\beta)}{4\eta_0}.\end{aligned}$$

For $\beta = 1$, we instead use the ansatz $z(t) = 2 - c/\log(t)$:

$$\begin{aligned}1 &= \int_{-2}^{2} \mathrm{d}\mu \sqrt{4-\mu^2} e^{2\eta_0(\mu-2)\log t} e^{2c\eta_0 \log\log t} \\ &= (\log t)^{2c\eta_0 - 3/2} \int_0^\infty \mathrm{d}\epsilon \sqrt{2\epsilon} e^{-2\eta_0 \epsilon} \propto (\log t)^{2c\eta_0 - 3/2} \Rightarrow c = \frac{3}{4\eta_0}.\end{aligned}$$

Hence, the scaled loss $\ell = \mathcal{L}/N$ converges to the ground state (global minimum) $\ell_{GS} = -1$ as a sum of power-laws:

$$\ell(t) - \ell_{GS} = \frac{\eta_0 T}{2t^\beta} + \begin{cases} \frac{3(1-\beta)}{8\eta_0 t^{1-\beta}}, & \beta < 1 \\ \frac{3}{8\eta_0 \log t}, & \beta = 1 \end{cases}. \tag{E.9}$$

Dependency on the spectrum of J One may naturally ask whether our conclusions are affected by changing the spectrum of the coupling matrix J. Notice that the key to solving the self-consistent equation is the behavior of the spectrum near its right edge. For the semi-circle law considered here, the right edge of the spectrum behaves as a square root. This law applies to a rather wide range of random matrix ensembles. Besides, many other common spectral densities, such as the Marcenko-Pastur law, also exhibit a same square root behavior on their right edge. Hence we expect our results to hold for a wide range of random matrix ensembles.

Planted model

The loss function is given by:

$$\mathcal{L}(w) = -\frac{N}{2} m^2 - \frac{\Delta}{\sqrt{N}} \sum_{i<j}^{N} J_{ij} w_i w_j. \tag{E.10}$$

Solving the dynamics

Again we choose $\eta(t) = \eta_0/t^\beta$ and consider the high signal-to-noise setting,

$\Delta < \frac{1}{2}$. Writing $z(t) = 1 - f(t)$, we obtain:

$$1 = \langle \int \frac{\mathrm{d}\mu}{N} \rho(\mu) w_\mu(t)^2 \rangle \tag{E.11}$$

$$= \frac{N-1}{N} \int_{-2}^{2} \mathrm{d}\mu \rho_{sc}(\mu) e^{2\int_{t_0}^{t} \mathrm{d}\tau \eta(\tau)(\mu-1)} e^{2\int_{t_0}^{t} \mathrm{d}\tau \eta(\tau) f(t)} + \frac{1}{N} e^{2\int_{t_0}^{t} \mathrm{d}\tau \eta(\tau) f(t)} \tag{E.12}$$

$$= e^{2\int_{t_0}^{t} \mathrm{d}\tau \eta(\tau) f(\tau)} \left(\underbrace{e^{-2\eta(t)(1-2\Delta)} \int_{-2}^{2} \mathrm{d}\mu \rho_{sc}(\mu/\Delta) e^{2\eta(t)(\mu-2\Delta)t}}_{A(t)} + \frac{1}{N} \right) \tag{E.13}$$

The expression above involves two terms. The first is of order one but decays exponentially over time; using results above, we obtain that

$$A(t) \sim t^{-3(1-\beta)/2} e^{-2\eta_0 \kappa t^{1-\beta}}. \tag{E.14}$$

Hence, there is a crossover time at which the first term becomes smaller than the second term, given by:

$$A(t) \sim 1 \Rightarrow t_{\text{cross}} = \left(\frac{\log N}{2\eta_0 \kappa} \right)^{\frac{1}{1-\beta}} \tag{E.15}$$

Before t_{cross}, the signal is not detected and we have as before $z(t) = 2\Delta - c/t^{1-\beta}$.

After t_{cross}, we have $A(t) \ll 1/N$. Multiply Eq. E.13 by N and taking the log, we obtain:

$$\log N = 2 \int_{t_0}^{t} \mathrm{d}\tau f(\tau) t^{-\beta} + \log\left(1 + NA(t)\right) \tag{E.16}$$

$$\sim 2 \int_{t_0}^{t} \mathrm{d}\tau f(\tau) t^{-\beta} + NA(t) \tag{E.17}$$

Taking the derivative with respect to t, we find the following asymptotics for late times:

$$f(t) \sim \frac{-NA'(t)t^{\beta}}{2} \sim t^{-5(1-\beta)/2} e^{-2\eta_0 \kappa t^{1-\beta}} \tag{E.18}$$

Hence,

$$\ell(t) - \ell_{GS} = \frac{\eta_0 T}{2t^{\beta}} + \frac{1}{2} f(t) \tag{E.19}$$

with $\ell_{GS} = 1$. As previously, it is straightfoward to extend this to the setup $\beta = 1$, for which we obtain $f(t) \sim t^{-2\eta_0 \kappa}$.

Curvature analysis

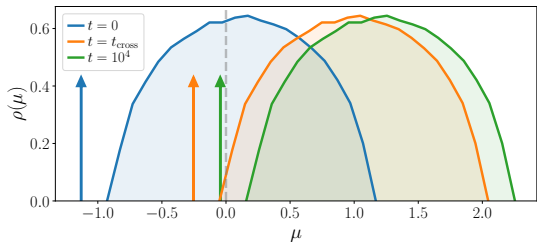


Figure E.1: **The landscape becomes convex at the crossover time.** *Parameters:* $N = 3000$, $\eta_0 = 0.1$.

As before, the spectrum of interest to study the landscape is that of H shifted to the right by the spherical constraint $z(t)$, depicted in Fig. E.1. The crossover time t_{cross} corresponds to the time at which the left edge of the semi-circle reaches 0. Thanks to the presence of the signal, the dynamics do not stop at this point; they continue until the eigenvalue corresponding to the signal reaches zero (which as achieved at $t \to \infty$). After the t_{cross}, the landscape becomes locally convex: the only negative eigenvalue is in the direction of the signal. Due to the spherical constraint, the effective Hessian (of dimension $N-1$) does not feel this negative eigenvalue when w is close to $w^\star$.

E.3 Dynamics of the p-spin model

Rescaling the temperature

Introducing a learning rate schedule is equivalnt to changing the "clock" directly in the Langevin (Eq. 5.1) as $\mathrm{d}\tilde{t} = \eta(t)\mathrm{d}t$. Then, for $\beta < 1$ we have:

$$\delta(\tilde{t})\mathrm{d}\tilde{t} = \delta(t)\mathrm{d}t \Rightarrow \delta(t) = \delta(\tilde{t}) \left(\frac{\eta_0}{1-\beta}\right)^{\frac{\beta}{1-\beta}} \tilde{t}^{\frac{-\beta}{1-\beta}} \tag{E.20}$$

The Langevin equation becomes:

$$\begin{aligned} \frac{\mathrm{d}w_i(t)}{\mathrm{d}t} &= -\left(\frac{\partial \mathcal{L}(w, w^\star)}{\partial w_i} + \xi_i(t) + z(t) w_i(t)\right), \\ \langle \xi(t)\xi(t')\rangle &= 2T\tilde{\eta}_0 \tilde{t}^{-\beta/1-\beta} \delta(t-t'), \end{aligned} \tag{E.21}$$

where we defined $\tilde{\eta}_0 = \left(\frac{\eta_0}{1-\beta}\right)^{\frac{\beta}{1-\beta}}$. This equation reveals that the process optimised with a varying learning rate is equivalent to a process at an effective temperature:

$$\tilde{T} = \tilde{\eta}_0 \tilde{t}^{\frac{-\beta}{1-\beta}} T \tag{E.22}$$

This equation corresponds to the physical protocol in which the temperature $\tilde{T}$ is annealed as a power law. As we show for the p-spin model in the next section, the solution is governed by a speed-noise trade-off.

Application to the p-spin model

For the p-spin model the loss can therefore be written:

$$\mathcal{L}(t;\tilde{T}) = \frac{-N}{p}\left(\tilde{z}(t) - \tilde{T}\right) = \mathcal{L}(t;T=0) + \mathcal{L}_{\text{th}}(\tilde{T}) \tag{E.23}$$

where we assumed that, at all times, the temperature dependent contribution to the loss has time to equilibrate in the threshold states.

To find $\mathcal{L}_{\text{th}}(\tilde{T})$, we assume that, since we are looking at long times, we have $\tilde{T} \ll 1$. We can consider the loss by performing an expansion around the $\tilde{T} = 0$ minimum i.e. considering that the motion is oscillatory around the minimum. At $T = 0$, the threshold overlap is given by $q_{\text{th}} = 1$. At $T \ll 1$, we thus write $q = 1 - \chi T$. Performing a similar matching argument as the one of [207], described in more details in Sec. E.4, we find that the close to the threshold, the loss is given by Eq. E.68:

$$\mathcal{L}_{\text{th}}(T) = -\frac{1}{p}\left[\sqrt{(p-1)q^{p-2}} + \frac{1}{T}\sqrt{\frac{2}{p}}\left(1 - q^{p-1}\right)\right] \tag{E.24}$$

In addition, we can expand the threshold overlap solution around $T = 0$ [274]:

$$\begin{aligned} q_{\text{th}}^{p-2}(1-q_{\text{th}})^2 &= \frac{T^2}{p-1} \\ \Rightarrow \chi &= \sqrt{\frac{1}{(p-1)}} \end{aligned} \tag{E.25}$$

Replacing this solution in Eq. E.24, we find:

$$\mathcal{L}_{\text{th}}(T) = \underbrace{-\frac{\sqrt{4(p-1)}}{p}}_{\mathcal{L}_{\text{th}}(T=0)} + \underbrace{\frac{p-2}{p}T}_{\propto T} \tag{E.26}$$

We see that $\mathcal{L}_{\text{th}}(T)$ is composed of a constant term, which is the same as the threshold loss defined in Eq. 5.17 and a term scaling linearly with T. Thus:

$$\mathcal{L}_{\text{th}}(\tilde{T}) - \mathcal{L}_{\text{th}} \propto \tilde{T} \propto t^{-\beta/1-\beta}. \tag{E.27}$$

We find again the two competing term in the speed of optimisation. On the one hand, the noiseless term, which is the same as the zero temperature loss, decays as $\mathcal{L}(t; T = 0) \propto t^{-\gamma}$. On the other hand, the temperature dependent term, which blocks the dynamics at loss $\mathcal{L}_{\text{th}}(\tilde{T})$, which decays as $t^{-\beta/1-\beta}$. The loss decays as $t^{-\min\left(\gamma, \frac{\beta}{1-\beta}\right)}$. Equaling the two exponents gives the optimal value of $\beta_{\text{opt}} = \frac{2}{5}$.

E.4 Dynamics of the Spiked Matrix-Tensor model

Derivation of the PDE equations

For simplicity, we detail the derivation of the *p-spin model* without the spike as studied in Sec. 5.1.3 in the case $p = 3$. The derivation for the full spiked tensor model is similar and can be found in [274]. The Langevin equation for each spin w is given by:

$$\dot{w}_i(t) = -z(t)\eta(t) - \eta(t)\partial_{w_i}\mathcal{L} + \eta(t)\xi_i(t) \tag{E.28}$$

where $\xi \in \mathrm{R}^N$ is the Langevin noise with distribution $\langle \xi_i(t) \rangle = 0$ and $\langle \xi_i(t)\xi_j(t') \rangle = 2T\delta_{ij}\delta(t - t')$. The solution w to the Langevin equation depends on the realisation of the noise. We can obtain a probability distribution over w given the distribution of ξ by considering the expectation of an observable $A(t)$:

$$\begin{aligned}\langle A(x) \rangle &= \int D\xi P(\xi) A\left(w_\xi\right) = \int dx \left[\int D\xi P(\xi) \delta\left(\dot{x} + \eta(t)\partial_x \mathcal{L} - \eta(t)\xi\right) \right] A(x) \\ &= \int dx P(x) A(x)\end{aligned} \tag{E.29}$$

We are now interested in considering $P(x)$ when averaged over the quenched disorder J. We therefore resort to:

$$\begin{aligned}1 \equiv Z &= \int DxP(x) \\ &= \int DxD\hat{w}D\xi \exp\left[-\frac{1}{2} \int dtdt' \xi(t) D_0^{-1}\left(t, t'\right) \xi\left(t'\right) + i \int dt\hat{w}(t) \left(\partial_t x + \eta(t)\partial_x \mathcal{L}\right) \right. \\ &\qquad \left. -i \int dt\eta(t)\hat{w}(t)\xi(t) \right] \\ &= \int DxD\hat{w} \exp\left[-\frac{1}{2} \int dtdt' \hat{x}(t)\eta(t) D_0\left(t, t'\right) \hat{x}\left(t'\right) \eta(t') + i \int dt\hat{w}(t) \left(\partial_t x + \eta(t)\partial_x \mathcal{L}\right) \right] \\ &= \int DxD\hat{w} \exp\left[S(x, \hat{x})\right]\end{aligned} \tag{E.30}$$

where we defined $D_0(t, t') = 2T\delta(t - t')$. Crucially, $S(x, \hat{x})$ acts as a generating functional and allows to obtain correlation functions by a term $\int dt\hat{x}(t)h(t) +$

$x(t)\hat{h}(t)$. We can thus define

$$\langle x(t)\hat{x}(t')\rangle = \partial_{h(t')}\langle x(t)\rangle \equiv R(t,t') \qquad \langle x(t)x(t')\rangle = \partial_{\hat{h}(t')}\langle x(t)\rangle \equiv C(t,t') \quad \text{(E.31)}$$

We now want to average the partition function over the quenched disorder Z. We note that the only time depend term in the exponent is $i\hat{x}(t)\eta(t)\partial_x\mathcal{L}$. We thus have to compute:

$$\overline{e^{i\hat{x}(t)\eta(t)\partial_x\mathcal{L}}} \equiv e^{\Delta(x,\hat{x})} \quad \text{(E.32)}$$

The average over the disorder will induce corrections both to the propagator D_0 and to the interaction term $\hat{x}(t)x(t')$ i.e. $\Delta(x,\hat{w}) = -\frac{1}{2}\hat{w}D_1(x,\hat{w})\hat{w} + i\hat{w}\mathcal{L}_1(x,\hat{w})$. By performing the average we obtain:

$$\begin{aligned}\overline{i\hat{w}_i(t)\eta\partial_{w_i}\mathcal{L}} &= \int \prod_{i>k>l} dJ_{ikl} \exp\left\{-\frac{1}{2}J_{ikl}^2 - \sqrt{\frac{(p-1)!}{N^{p-1}}}J_{ikl}\int dt\eta(t)\left[i\hat{w}_i w_k w_l + w_i i\hat{w}_k w_l + w_i w_k i\hat{w}_l\right]\right\} \\ &= \exp\left\{\int \frac{dtdt'}{2N^{p-1}}\eta(t)\eta(t')\left[(i\hat{w}\cdot i\hat{w})(x\cdot x)^{p-1} + (p-1)(i\hat{w}\cdot x)(x\cdot i\hat{w})(x\cdot x)^{p-2}\right]\right\}\end{aligned} \quad \text{(E.33)}$$

where we introduced the notation $x\cdot x \equiv \sum_{i=1}^{N} w_i(t)w_i(t')$. We now introduce dynamical overlaps Q_1, Q_2, Q_3 and Q_4 as:

$$\begin{aligned}\overline{e^{i\hat{w}_i(t)\eta\partial_{w_i}\mathcal{L}}} = \int DQ\delta\left(NQ_1 - \sum_k i\hat{w}_k(t)i\hat{w}_k(t')\right)\delta\left(NQ_2 - \sum_k w_k(t)w_k(t')\right) \\ \cdot\delta\left(NQ_3 - \sum_k i\hat{w}_k(t)w_k(t')\right)\delta\left(NQ_4 - \sum_k w_k(t)i\hat{w}_k(t')\right) \\ \cdot\exp\left\{\frac{N}{2}\int dtdt'\eta(t)\eta(t')\left[Q_1(t,t')Q_2(t,t')^{p-1}\right.\right. \\ \left.\left.+(p-1)Q_3(t,t')Q_4(t,t')Q_2(t,t')^{p-2}\right]\right\}\end{aligned} \quad \text{(E.34)}$$

We can easily see that we have the correspondence $Q_1(t,t') = 0$, $Q_2(t,t') = C(t,t')$, $Q_3(t,t') = R(t',t)$ and $Q_4(t,t') = R(t,t')$. By using the exponential form of the delta function and solving the fix point equations for the conjugate fields $\hat{Q}_1$, $\hat{Q}_2$,

$\hat{Q}_3$ and $\hat{Q}_4$ we find:

$$\begin{cases} i\hat{Q}_1 = \frac{1}{2}\eta(t)\eta(t')Q_2^{p-1} \\ i\hat{Q}_2 = \frac{p-1}{2}\eta(t)\eta(t')Q_1Q_2^{p-2} + \frac{(p-1)(p-2)}{2}\eta(t)\eta(t')Q_3Q_4Q_2^{p-3} \equiv 0 \\ i\hat{Q}_3 = \frac{p-1}{2}\eta(t)\eta(t')Q_4Q_2^{p-2} \\ i\hat{Q}_4 = \frac{p-1}{2}\eta(t)\eta(t')Q_3Q_2^{p-2} \end{cases} \tag{E.35}$$

From the definition of the $\hat{Q}$'s we find the new term in the generating functional as:

$$\Delta = \sum_k \int dtdt'\eta(t)\eta(t') \left\{ -\frac{1}{2}C\left(t,t'\right)^{p-1}\hat{w}_k(t)\hat{w}_k\left(t'\right) - (p-1)R\left(t,t'\right)C\left(t,t'\right)^{p-2} i\hat{w}_k(t)w_k\left(t'\right) \right\} \tag{E.36}$$

This allows us to write an effective Langevin equation for a scalar degree of freedom w:

$$\dot{x}(t) = -z(t)\eta(t)x(t) + \eta(t)(p-1)\int dt''\eta(t')R\left(t,t''\right)C\left(t,t''\right)^{p-2}\sigma\left(t''\right) + \eta(t)\tilde{\xi}(t), \tag{E.37}$$

with:

$$\langle\tilde{\xi}(t)\tilde{\xi}(t')\rangle = 2T\delta(t-t') + C^{p-1}(t,t'). \tag{E.38}$$

In order to write down a set of PDE's for R and C, note the useful relations:

$$\begin{aligned} \langle\frac{\partial x(t)}{\partial\xi(t')}\rangle &= -i\langle x(t)\hat{x}(t')\rangle \\ \langle x(t)\xi(t')\rangle &= 2T\eta(t')R(t,t^{'}) \\ \langle\tilde{\xi}(t_1)x(t_2)\rangle &= 2T\eta(t_1)R(t_1,t_2) + \int dt''\eta(t'')R(t'',t_2)C^{p-1}(t'',t_1) \end{aligned} \tag{E.39}$$

We therefore find:

$$\begin{aligned} \frac{\partial R\left(t_1,t_2\right)}{\partial t_1} &= \left\langle\frac{\delta\dot{x}\left(t_1\right)}{\delta\tilde{\xi}\left(t_2\right)}\right\rangle \\ &= -z\left(t_1\right)\eta(t_1)R\left(t_1,t_2\right) + \eta(t_1)\delta\left(t_1,t_2\right) \\ &\quad + (p-1)\eta(t_1)\int_{t_2}^{t_1} dt''\eta(t'')R\left(t_1,t''\right)C^{p-2}\left(t_1,t''\right)R\left(t'',t_2\right) \end{aligned} \tag{E.40}$$

$$\begin{aligned}\frac{\partial C\left(t_1,t_2\right)}{\partial t_1} &= \langle \dot{x}\left(t_1\right) x\left(t_2\right)\rangle \\ &= -\eta(t_1) z\left(t_1\right) C\left(t_1,t_2\right) + 2T\eta(t_1)^2 R(t_1,t_2) \\ &\quad + (p-1)\eta(t_1)\int_{-\infty}^{t_1} dt''\eta(t'')R\left(t_1,t''\right) C^{p-2}\left(t_1,t''\right) C\left(t'',t_2\right) \\ &\quad + \eta(t_1)\int dt''\eta(t'')R(t'',t_2)C^{p-1}(t'',t_1)\end{aligned} \tag{E.41}$$

The equation for $z(t)$ is given by differentiation $C(1,1)=1$, i.e. $\left[\partial_t C\left(t,t'\right)+\partial_{t'}C\left(t,t'\right)\right]_{t,t'=s} = 0$:

$$z(t_1) = T\eta(t_1) + p\int dt_2\eta(t_2)R(t_2,t_1)C^{p-1}(t_2,t_1). \tag{E.42}$$

The loss at all times is found by using the Ito identity:

$$\frac{1}{N}\frac{d}{dt}\sum_i w_i^2(t) = \frac{2}{N}\sum_i w_i(t)\dot{x}_i(t) + 2 \tag{E.43}$$

which yields:

$$\mathcal{L}(t) = \frac{N}{p}\left(T\eta(t) - z(t)\right) \tag{E.44}$$

Spiked matrix-tensor model The derivation of the PDEs describing the dynamics of C, R and z in the spiked matrix-tensor model are similar as the ones for the p-spin. In addition, one also needs to keep track of the evolution of the overlap of the estimate with the signal i.e. the magnetisation $m = {}^{w\cdot w^\star}/_N$. Using the same method as before we find:

$$
\begin{aligned}
\frac{\partial}{\partial t}C\left(t,t'\right) =& -z(t)\eta(t)C\left(t,t'\right)+\eta(t)Q'(m(t))m\left(t'\right)\\
&+\eta(t)\int_0^t \eta(t')R\left(t,t''\right)Q''\left(C\left(t,t''\right)\right)C\left(t',t''\right)dt''\\
&+\eta(t)\int_0^{t'} \eta(t')R\left(t',t''\right)Q'\left(C\left(t,t''\right)\right)dt''+2T\eta(t)^2R(t,t')\\
\frac{\partial}{\partial t}R\left(t,t'\right) =& -z(t)\eta(t)R\left(t,t'\right)+\delta(t-t')\eta(t)\\
&+\eta(t)\int_{t'}^t \eta(t')R\left(t,t''\right)Q''\left(C\left(t,t''\right)\right)R\left(t'',t'\right)dt''\\
\frac{d}{dt}m(t) =& -\eta(t)z(t)m(t)+\eta(t)Q'(m(t))\\
&+\eta(t)\int_0^t \eta(t')R\left(t,t''\right)m\left(t''\right)Q''\left(C\left(t,t''\right)\right)dt''\\
z(t) =& \eta(t)T+Q'(m(t))m(t)\\
&+\int_0^t \eta(t')R\left(t,t''\right)\left[Q'\left(C\left(t,t''\right)\right)+Q''\left(C\left(t,t''\right)\right)C\left(t,t''\right)\right]dt''
\end{aligned}
\tag{E.45}
$$

where we defined $Q(w) = Q_p(w) + Q_2(w) = \frac{w^p}{p\Delta_p} + \frac{w^2}{2\Delta_2}$. The loss is related to $z(t)$ via:

$$
z(t) = T\eta(t) - p\frac{\mathcal{L}_p}{N} - 2\frac{\mathcal{L}_2}{N}, \tag{E.46}
$$

with $\mathcal{L}_2$, respectively $\mathcal{L}_2$ are the loss associated with the matrix, respectively tensor, channel. *The Langevin easy phase* As explained in [207], one finds different phases in the two dimensional space spamed by the noise intensities Δ_2 and Δ_p. In the *Langevin easy* phase, a system initialised with a magnetisation $m{\sim}O(1/\sqrt{N})$ recovers the signal and converges to an overlap of order 1. It is delimited by $\Delta_2 < \Delta_2^*$, where Δ_2^* is the solution to the implicit equation:

$$
\Delta_2 < \Delta_2^\star = \sqrt{\frac{\Delta_p}{(p-1)(1-\Delta_2^*)^{p-3}}}. \tag{E.47}
$$

In contrast, in the *Langevin hard* and *Langevin impossible* phase, i.e. $\Delta_2 > \Delta_2^*$, the dynamics fail to recover the signal and remain at low magnetisation. More details in [274].

Derivation of the Ground-state Loss

In order to derive the ground state properties of the system, we resort to the replica method, developed in physics as a tool to deal with random systems. Using

these tools, involves performing a mapping between the optimisation problem, an inference problem and a physical system. We can consider the estimator w as a guess on the planted signal $w^\star$ and y be the observations.The, using Bayes formula we can express the posterior probability of the estimator w given the observation y:

$$P[x|y] = \frac{1}{P[y]} P[x]P[y|x] \approx_{\beta=1} \frac{1}{P[y]} P[x]P[y|x]^{-\beta} = \frac{1}{Z(y)} \mathrm{e}^{-\beta \mathcal{L}}. \tag{E.48}$$

We can identify the last terms with a Gibbs distribution at temperature $\beta = 1/T$ and Z is a normalisation constant named the *partition function*. At $\beta = 1$, the posterior E.48 is the exact posterior of the problem. At $\beta \to \infty$, the distribution is dominated by the spin configuration minimising the loss, i.e. the maximum likely hood approximator of the problem. The partition function, and its logarithm the *free energy*:

$$\Phi = \frac{-1}{N} \log Z, \tag{E.49}$$

act as a generating functional. I.e. they encapsulate all the relevant information needed to describe of the system. Notably, all observables can be obtained by taking derivatives of it. In particular, the loss and the overlap with the signal are given by:

$$\begin{aligned} \mathcal{L} &= \frac{1}{N} \frac{1}{Z} \int_{S^{N-1}} \mathcal{L} e^{-\beta \mathcal{L}} = \frac{-1}{N} \frac{\partial \log Z}{\partial \beta} = \frac{\partial \Phi}{\partial \beta} \\ m &= \frac{1}{N} \sum_{i=1}^{N} \frac{1}{Z} \int_{S^{N-1}} w_i w_i^\star e^{-\beta \mathcal{L} + Bh \cdot w} |_{Bh=0} = -w^\star \cdot \boldsymbol{\nabla}_{Bh} \Phi. \end{aligned} \tag{E.50}$$

The spiked tensor model is rendered more complex due to the randomness associated with the couplings. We need to evaluate the averaged logarithm of the partition function $\overline{\log Z}$ which is in general prohibitive. To deal with this problem, physics have developed the heuristic *replica method* based on the equality:

$$\overline{\log Z} = \lim_{n \to 0} \frac{\overline{Z^n} - 1}{n}. \tag{E.51}$$

In practice, one computes $\overline{Z^n}$ for $n \in \mathbb{N}$ and then extends the result to real n. The problem can be viewed as introducing n identical, replicated, copies of the system. As we will see, averaging over the random couplings introduces correlation

between the copies. $\overline{Z^n}$ can easily be evaluated as:

$$\begin{aligned}\overline{\mathcal{Z}^n} &= \mathop{\mathbb{E}}_{\substack{J_{i_1\dots i_p}\\ J_{ij}}} \int \prod_{a=1}^{n} \mathrm{e}^{\beta\sqrt{\frac{1}{p\Delta_p N}}\sum_{i_1,\dots,i_p} J_{i_1,\dots,i_p} x_{i_1}^{(a)}\dots x_{i_p}^{(a)} + \beta\sqrt{\frac{1}{2\Delta_2 N}}\sum_{i,j} J_{i,j} x_i^{(a)} x_j^{(a)} + N\beta\sum_i Q\left(\frac{x_i^{(a)} x_i^*}{N}\right)} \prod_{a=1}^{n} dx^{(a)} \\ &= \int_{\mathbb{S}^{n(N-1)}(\sqrt{N})} \mathrm{e}^{N\beta\sum_i Q\left(\frac{x_i^{(a)} x_i^*}{N}\right) + \frac{N\beta^2}{2} Q\left(\sum_{a,b=1}^{n}\sum_i \frac{x_i^{(a)} x_i^{(b)}}{N}\right)} \prod_{a=1}^{n} dx^{(a)}.\end{aligned} \tag{E.52}$$

where we introduced $Q(x) = x^2/2\Delta_2 + x^p/p\Delta_p$. The second term in the exponent carries the interaction between the different copies obtained after averaging out the random couplings. It depends on the overlap Q having entries $\mathcal{Q}_{ab} = \sum_i \frac{x_i^{(a)} x_i^{(b)}}{N}$. We associate the index $a = 0$ with the ground truth signal $w^\star$. Using the exponential representation of the Dirac delta function, we introduce the overlap matrix into the partition function. After some manipulation we obtain:

$$\overline{Z^n} = \int \mathrm{e}^{N\beta S(\mathrm{Q})} \tag{E.53}$$

$$\beta S(\mathrm{Q}) = \frac{1}{2}\log\det \mathrm{Q} + \beta^2 \sum_{a,b=1}^{n} Q(\mathcal{Q}_{ab}) + \beta \sum_{a=1}^{n} Q(\mathcal{Q}_{a0}). \tag{E.54}$$

The factor N in the exponential in the integrand, implies that in the $N\to\infty$ limit, the integral is dominated by the matrix $\mathcal{Q}$ maximising the action S. In order to progress, we make a replica symmetric ansatz*: i.e. we assume the different systems have overlaps q between each other and m with the ground truth. This imposes a matrix $\mathcal{Q}$ has the form:

$$\mathcal{Q} = \begin{pmatrix} 1 & m & m & m \\ m & 1 & q & q \\ m & q & 1 & q \\ m & q & q & 1 \end{pmatrix}. \tag{E.55}$$

Replacing this overlap matrix in E.54 and taking $n\to 0$, we obtain:

$$\beta S_{\mathrm{RS}}(q,m) = n\left\{\frac{1}{2}\frac{q-m^2}{1-q} + \frac{1}{2}\log(1-q) + \frac{\beta^2}{2}Q(1) - \frac{\beta^2}{2}Q(q) + \beta Q(m)\right\} \tag{E.56}$$

*Since we only consider the Langevin easy phase, where there is no ergodicity breaking, we do not need to consider a 1RSB ansatz.

We now maximise S with respect to m and q and obtain the saddle point equations:

$$\frac{S_{\text{RS}}(q,m)}{\partial m} = \frac{-m}{1-q} + \beta Q'(m) \tag{E.57}$$

$$\frac{S_{\text{RS}}(q,m)}{\partial q} = \frac{q-m^2}{(q-1)^2} + \beta^2 Q'(q) \tag{E.58}$$

The expression of the loss as a function of the overlaps m and q is given by using Eq. E.50:

$$\mathcal{L}(m,q) = -\beta(Q(1) - Q(q)) + Q(m) \tag{E.59}$$

By evaluating the above at the solutions Eqs. E.58, we obtain the ground state loss at a given temperature. $T = 1$ *solution* At $T = 1$ (i.e. $\beta = 1$) the posterior Eq. E.48 is exact and we can use the Nishimori identity stating that the distribution of the estimator is the same as the one of the signal implying $m = q$. Replacing the identity in Eq. E.58 and in Eq. E.59 we have:

$$m = (1-m)Q'(m), \mathcal{L}_{\text{gs}}^{T=1} = -Q(1). \tag{E.60}$$

$T = 0$ *solution* We can think of the 0 temperature system (i.e. $\beta \to \infty$) as physical system coupled to a thermal bath. As the temperature goes to 0, all particles collapse to a point at the minimum of the loss. Thus, the overlap tends to 1. However, we check that Eqs. E.58 are singular at $q = 1$. To properly take the limit, we perform a linear expansion in the temperature by replacing $q = 1 - \chi T$ in the equations and linearising in T. We then obtain the equation for m:

$$\begin{aligned} \chi &= \sqrt{\frac{1-m^2}{Q'(1)}} \\ m &= \chi Q'(m) \end{aligned} \tag{E.61}$$

and the ground state loss:

$$\mathcal{L}_{\text{gs}}^{T\to 0} = (-Q(m) - \chi Q'(1)). \tag{E.62}$$

Additional results on the optimal learning rate schedule in the SMT model

In this section, we give additional results confirming the optimal decay of the learning rate in the spiked-matrix tensor model. We have seen in the main text, that there is a crossover time t_{cross} before which the learning rate should be kept fixed as the system is in the search phase. After t_{cross}, the dynamics enter a convex basin

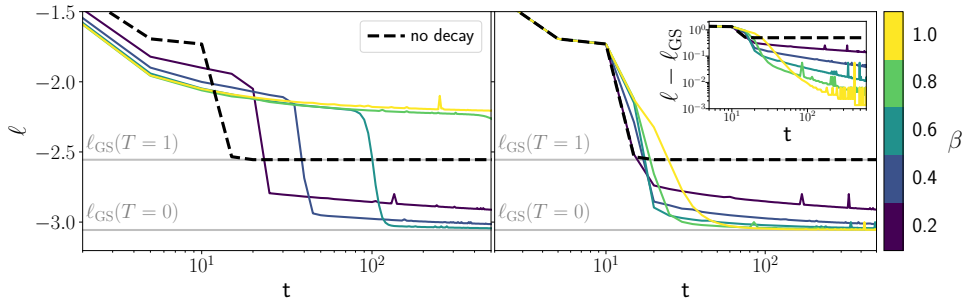


Figure E.2: **Emergence of a crossover time in the SMT model.** By fixing β from start, a randomly initialised system will remain stuck at *threshold* states of 0 overlap until t_{cross} which is minimal for $\beta = 0$. Higher β, allow to reach lower loss solutions but require much longer to converge. *(Right)* The optimal schedule is to keep η constant until t_{cross} and then setting $\beta = 1$. By doing so, we get the best of both worlds: the first phase minimises t_{cross} while the second allows to reach more informative solutions. *(Inset)* $m - m_{\text{gs}}$ shows that choosing higher β after t_{cross} allows to reach more informative minima. *Parameter:* $\beta = 0.8$, $\Delta_2 = 0.2$, $\Delta_p = 6$, $\eta_0 = 1$, $T = 1$, $dt = 10^{-2}$, $m_0 = 10^{-10}$.

and one should decay the learning rate as $\eta(t) \sim t^{-\beta}$. To verify that $\beta = 1$ leads to the lowest loss, in the right panel of Fig. E.2, we keep a constant learning rate until t_s after which we vary the exponent with which the learning rate is decayed. We check that $\beta = -1$ allows to reach the best solutions. However, the left panel shows that if the learning rate is decayed from start, the dynamics take much longer to converge towards the signal and remain stuck at high loss for very long.

Separation of time scales and matching solution

The long time dynamics, i.e. $t \to \infty$ of the p-spin model can be separated into two regimes:

- For all times $t, t \to \infty$ with $\frac{t-t'}{t} \to 0$ the system is stationary. Here, the dynamics are time-translation invariant (TTI) and the fluctuation-dissipation theorem (FDT) holds. The two time functions $C(t, t')$ and $R(t, t')$ are thus only a function of the time difference $\tau = t - t'$. In this regime, we define $C_{\text{TTI}}(\tau) \equiv C(t - t', 0)$ and $R_{\text{TTI}}(\tau) \equiv R(t - t', 0)$. The FDT gives $R_{\text{TTI}}(\tau) = -\frac{1}{T}\frac{dC_{\text{TTI}}(\tau)}{d\tau}$. As a consequence, the equations for R and C collapse into a single equation.

- For all times $t, t \to \infty$ with $\frac{t-t'}{t} = O(1)$ the system *ages* i.e. the dynamics remain trapped in metastable states and does not lose memory of its history. The relevant variable to consider in this regime is $\lambda = t'/t$. The correlation and response functions can be rescaled as $\mathcal{R}(\lambda) = tR(t, t')$ and $q\mathcal{C}(\lambda) = C(t, t')$

with $q = \lim_{\tau\to\infty} C_{\text{TTI}}(\tau)$. In this aging regime, a generalised form of the FDT holds and $\mathcal{R}(\lambda) = \frac{x}{T} q \frac{\mathrm{d}\mathcal{C}(\lambda)}{\mathrm{d}\lambda}$. The *violation parameter* w is found by *matching* i.e. considering the equations for the response and the correlation separetly. q is found by imposing $q = lim_{\tau\to\infty} C_{\text{TTI}}(\tau)$ in the equation of the TTI regime.

In order to derive analytical results, we use the hypothesis of these two times regimes to split the time integrals in Eqs. E.40- E.42. For compactness we also define $Q(x) = {}^{x^p}/_2$. As noted in the main text, we can re-scale time according to $\mathrm{d}\tilde{t} = \eta(t)\mathrm{d}t$ and obtain a system at an effective temperature $\tilde{T} = (1-\beta)^{\frac{1}{1-\beta}} \frac{T}{t^{\beta/1-\beta}}$. We are ultimately interested in determining the threshold loss, a static quantity, and can hence perform its derivation using a constant learning rate. This analysis is a special case of the more general on performed in [274]. Here, we show it for the special case of the p-spin model with no signal. In particular, we skip all the computations and refer the reader to [274] (Appendix B) for additional details.

Lagrange multiplier in the long time-limit Let us start to illustrate how to proceed by computing the long time limit of the loss $z_\infty = \lim_{t\to\infty} z(t)$ using Eqs. E.40 E.41 E.42 :

$$\begin{aligned} z_\infty(T) - T &= p \underbrace{\int_0^t dt'' R(t'',t) Q'(C(t'',t))}_{\int_{\text{TTI}} + \int_{\text{aging}}} \\ &= -\int_0^\infty \frac{1}{T}\frac{d}{d\tilde{t}} Q\left(C_{\text{TTI}}(\tilde{t})\right) d\tilde{t} + \int_0^1 \mathcal{R}(\lambda) Q'(q\mathcal{C}(\lambda)) d\lambda \\ \Leftrightarrow z_\infty &= \frac{1-q^p}{2T} + \int_0^1 \mathcal{R}(\lambda) Q'(q\mathcal{C}(\lambda)) d\lambda, \end{aligned} \tag{E.63}$$

where we used the fact that by definition $C_{\text{TTI}}(\infty) = q$ and $C_{\text{TTI}}(0) = 1$. Also note that we neglected all the finite time contribution to the integrals. We are going to determine z_∞ using this equation.

Stationary regime In order to find the dynamical equations in the stationary regime, we proceed as before and separate the contributions of the TTI regime from those of the aging regime in the integrals. Since both equations for the response and the correlation collapse into a single equation, we consider only the evolution of the correlation C_{TTI}. Using Eqs. E.40 we have:

$$(z_\infty + \partial_\tau) C_{\text{TTI}}(\tau) = \int_0^{t_1} dt'' R\left(t_1,t''\right) Q''\left(C\left(t_1,t''\right)\right) C\left(t'',t_2\right) + \int_0^{t_2} dt'' R(t'',t_2) Q'\left(C(t'',t_1)\right) \tag{E.64}$$

Using Eqs. 62 of [274], we have:

$$\partial_\tau C_{\mathrm{TTI}}(\tau) + \left(\frac{1}{T}Q'(1) - \mu_\infty\right)\left[1 - C_{\mathrm{TTI}}(\tau)\right] + T = -\frac{1}{T}\int_0^\tau Q'\left(C_{\mathrm{TTI}}\left(\tau - \tau''\right)\right)\frac{d}{d\tau''}C_{\mathrm{TTI}}\left(\tau''\right)d\tau'' \tag{E.65}$$

When $\tau \to \infty$, the time variations of $C_{\mathrm{TTI}}(\tau)$ are negligible. Taking this limit in the above equation gives:

$$z_\infty = \sqrt{Q''(q)} + \frac{Q'(1) - Q'(q)}{T} \tag{E.66}$$

This equation allows to determine the threshold loss, i.e. the loss at the plateau reached by the system before the recovery of the signal. We notice that the equality above holds for all Δ_2, Δ_p and hence also if one of the two is sent to infinity. Therefore, we have:

$$\ell_{\mathrm{th}} = \ell_p + \ell_2, \tag{E.67}$$

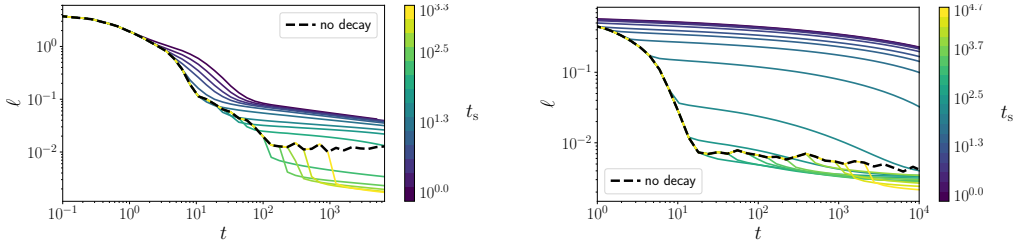
with $\ell_2 = \frac{1}{2}(\eta(t) - z_{\infty;\Delta_p \to \infty})$ and similarly for ℓ_p. Thus, by defining $Q_k(x) = {}^{x^k}/_{k\Delta_k}$, we obtain:

$$\ell_k = \frac{1}{k}\left(\eta(t) - \sqrt{Q_k''(q)} - \frac{Q_k'(1) - Q_k'(q)}{T}\right) \tag{E.68}$$

Using this equation, and performing an expansion around 0 for T and 1 for q, we can determine that at low temperatures, the threshold energy scales linearly with T.

E.5 Additional results for the Teacher-Student Regression Task

In this section we give additional results on the teacher-student regression task discussed in Sec. 5.1.5. The setting is the same as in the main text: a K hidden nodes 2 layer neural network student is trained to reproduce the output of her 2 layer neural network teacher of M nodes on gaussian inputs. We train the model with on a finite dataset of P examples using a mini-batch size $B = 1$. Fig. E.3 verifies that the conclusions drawn in the main text hold for different values of K and M. The optimal schedule is to keep the learning rate constant until t_{cross} and to then decay it as $1/t$. If the learning rate is decayed too soon, i.e. at $t_s < t_{\text{cross}}$, learning remains stuck at high loss values. Decaying after t_{cross} instead allows to reduce the noise in optimisation and reach lower loss solutions. We verify that in both these cases, t_{cross} matches the end of the "specialisation" transition, where the loss achieved student trained at constant learning rate plateaus.



(a) $K = M = 5$ (b) $K = M = 20$

Figure E.3: **The crossover time is also reflected in a regression task with SGD.** A student with K hidden nodes is trained to reproduce the output of her M hidden nodes. *(Left)* $K = M = 5$. *(Right)* $K = M = 20$. As in the main text.before, we find that decaying the learning rate before the loss plateaus performance, but decaying as $\eta(t) \sim t^{-1}$ once the plateau is reached allows to reach zero loss. *Parameters:* $N = 500$, $P = 10^4$, $\eta_0 = 10^{-1}$, $\beta = 0.8$.

Appendix F

Alternative training algorithm to go beyond BP

F.1 Derivation of the ODE

The derivation of the ODE's that describe the dynamics of the test error for shallow networks closely follows the one of Saad and Solla [270] and Biehl and Schwarze [38] for back-propagation. Here, we give the main steps to obtain the analytical curves of the main text and refer the reader to their paper for further details.

As we discuss in Sec. 5.2.2, student and teacher are both two-layer networks with K and M hidden nodes, respectively. For an input $x \in \mathbb{R}^D$, their outputs $\hat{y}$ and y^* can be written as

$$\begin{aligned} \hat{y} = \phi_\theta(x) &= \sum_{k=1}^{K} w_2^k \, g\left(a^k\right), \\ y = \phi_{\tilde{\theta}}(x) &= \sum_{m=1}^{M} \tilde{w}_2^m \, g\left(\tilde{a}^m\right), \end{aligned} \tag{F.1}$$

where we have introduced the pre-activations $a^k \equiv w_1^k x/\sqrt{D}$ and $\tilde{a}^m \equiv \tilde{w}_1^m x/\sqrt{D}$. Evaluating the test error of a student with respect to the teacher under the squared loss leads us to compute the average

$$\epsilon_g\left(\theta, \tilde{\theta}\right) = \frac{1}{2}\mathbb{E}_x\left[\sum_{k=1}^{K} w_2^k \, g\left(a^k\right) - \sum_{m=1}^{M} \tilde{w}_2^m \, g\left(\tilde{a}^m\right)\right]^2, \tag{F.2}$$

where the expectation is taken over inputs x for a fixed student and teacher. Since x only enters Eq. (F.2) via the pre-activations $a = (a^k)$ and $\tilde{a} = (\tilde{a}^m)$, we can replace the high-dimensional average over x by a low-dimensional average over the $K + M$

variables $(a, \tilde{a})$. The pre-activations are jointly Gaussian since the inputs are drawn element-wise i.i.d. from the Gaussian distribution. The mean of $(a, \tilde{a})$ is zero since $\mathbb{E}\, x_i = 0$, so the distribution of $(a, \tilde{a})$ is fully described by the second moments

$$Q^{kl} = \mathbb{E}\, a^k a^l = w_1^k \cdot w_1^l / D, \tag{F.3}$$

$$R^{km} = \mathbb{E}\, a^k \tilde{a}^m = w_1^k \cdot \tilde{w}_1^m / D, \tag{F.4}$$

$$T^{mn} = \mathbb{E}\, \tilde{a}^m \tilde{a}^n = \tilde{w}_1^m \cdot \tilde{w}_1^n / D. \tag{F.5}$$

which are the "order parameters" that we introduced in the main text. We can thus rewrite the generalisation error (5.29) as a function of only the order parameters and the second-layer weights,

$$\lim_{D\to\infty} \epsilon_g(\theta, \tilde{\theta}) = \epsilon_g(Q, R, T, w_2, \tilde{w}_2) \tag{F.6}$$

As we update the weights using SGD, the time-dependent order parameters Q, R, and w_2 evolve in time. By choosing different scalings for the learning rates in the SGD updates (5.27), namely

$$\eta_{w_1} = \eta, \qquad \eta_{w_2} = \eta / D$$

for some constant η, we guarantee that the dynamics of the order parameters can be described by a set of ordinary differential equations, called their "equations of motion". We can obtain these equations in a heuristic manner by squaring the weight update (5.27) and taking inner products with $\tilde{w}_1^m$, to yield the equations of motion for Q and R respectively:

$$\frac{\mathrm{d}R^{km}}{\mathrm{d}\alpha} = -\eta F_1^k \mathbb{E}\left[g'(a^k)\tilde{a}^m e\right] \tag{F.7a}$$

$$\frac{\mathrm{d}Q^{k\ell}}{\mathrm{d}\alpha} = -\eta F_1^k \mathbb{E}\left[g'(a^k)a^\ell e\right] - \eta F_1^\ell \mathbb{E}\left[g'(a^\ell)a^k e\right] + \eta^2 F_1^k F_1^\ell \mathbb{E}\left[g'(a^k)g'(a^\ell)e^2\right], \tag{F.7b}$$

$$\frac{\mathrm{d}w_2^k}{\mathrm{d}\alpha} = -\eta \mathbb{E}\left[g(a^k)e\right] \tag{F.7c}$$

where, as in the main text, we introduced the error $e = \phi_\theta(x) - \phi_{\tilde{\theta}}(x)$. In the limit $D\to\infty$, the variable $\alpha = \mu / D$ becomes a continuous time-like variable. The remaining averages over the pre-activations, such as

$$\mathbb{E}\, g'(a^k)a^\ell g(\tilde{a}^m),$$

are simple three-dimensional integral over the Gaussian random variables a^k, a^ℓ and $\tilde{a}^m$ and can be evaluated analytically for the choice of $g(x) = \text{erf}(x/\sqrt{2})$ [38] and for linear networks with $g(x) = x$. Furthermore, these averages can be expressed only in term of the order parameters, and so the equations close. We note that the asymptotic exactness of Eqs. F.7 can be proven using the techniques used recently to prove the equations of motion for BP [114].

We provide an integrator for the full system of ODEs for any K and M in the Github repository.

F.2 Detailed analysis of DFA dynamics

In this section, we present a detailed analysis of the ODE dynamics in the matched case $K = M$ for sigmoidal networks ($g(x) = \text{erf}\,(x/\sqrt{2})$).

The Early Stages and Gradient Alignment We now use Eqs. (F.7) to demonstrate that alignment occurs in the early stages of learning, determining from the start the solution DFA will converge to (see Fig. 5.10 which summarises the dynamical evolution of the student's second layer weights).

Assuming zero initial weights for the student and orthogonal first layer weights for the teacher (i.e. T^{nm} is the identity matrix), for small times ($t \ll 1$), one can expand the order parameters in t:

$$\begin{aligned} R^{km}(t) &= t\dot{R}^{km}(0) + \mathcal{O}(t^2), \\ Q^{kl}(t) &= t\dot{Q}^{kl}(0) + \mathcal{O}(t^2), \\ w_2^k(t) &= t\dot{W}_2^k(0) + \mathcal{O}(t^2). \end{aligned} \tag{F.8}$$

where, due to the initial conditions, $R(0) = Q(0) = w_2(0) = 0$. Using Eq. F.7, we can obtain the lowest order term of the above updates:

$$\begin{aligned} \dot{R}^{km}(0) &= \frac{\sqrt{2}}{\pi}\eta\tilde{w}_2^m F_1^k, \\ \dot{Q}^{kl}(0) &= \frac{2}{\pi}\eta^2\left((\tilde{w}_2^k)^2 + (\tilde{w}_2^l)^2\right)F_1^l F_1^k, \\ \dot{W}_2^k(0) &= 0 \end{aligned} \tag{F.9}$$

Since both $\dot{R}(0)$ and $\dot{Q}(0)$ are non-zero, this initial condition is not a fixed point of DFA. To analyse initial alignment, we consider the first order term of $\dot{W}_2$. Using Eq. (F.8) with the derivatives at $t = 0$ (F.9), we obtain to linear order in t:

$$\dot{W}_2^k(t) = \frac{2}{\pi^2}\eta^2||\tilde{w}_2||^2 F_1^k t. \tag{F.10}$$

Crucially, this update is in the direction of the feedback vector F_1. DFA training thus constrains the student to initially grow in the direction of the feedback vector and align with it. This implies gradient alignment between BP and DFA and dictates into which of the many degenerate solutions in the energy landscape the student converges.

Plateau phase After the initial phase of learning with DFA where the test error decreases exponentially, similarly to BP, the student falls into a symmetric fixed point of the Eqs. (F.7) where the weights of a single student node are correlated to the weights of all the teacher nodes ([38, 93, 270]). The test error stays constant while the student is trapped in this fixed point. We can obtain an analytic expression for the order parameters under the assumption that the teacher first-layer weights are orthogonal ($T^{nm} = \delta_{nm}$). We set the teacher's second-layer weights to unity for notational simplicity ($\tilde{w}_2^m = 1$) and restrict to linear order in the learning rate η, since this is the dominant contribution to the learning dynamics at early times and on the plateau [269]. In the case where all components of the feedback vector are positive, the order parameters are of the form $Q^{kl} = q, R^{km} = r, w_2^k = w_2$ with:

$$q = \frac{1}{2K-1}, \quad r = \sqrt{\frac{q}{2}}, \quad w_2 = \sqrt{\frac{1+2q}{q(4+3q)}}. \tag{F.11}$$

If the components of the feedback vector are not all positive, we instead obtain $R^{km} = \mathrm{sgn}(F^k)r$, $w_2^k = \mathrm{sgn}(F^k)w_2$ and $Q^{kl} = \mathrm{sgn}(F^k)\,\mathrm{sgn}(F^l)q$. This shows that on the plateau the student is already in the configuration that maximises its alignment with F_1. Note that in all cases, the value of the test error reached at the plateau is the same for DFA and BP.

Memorisation phase and Asymptotic Fixed Point At the end of the plateau phase, the student converges to its final solution, which is often referred to as the *specialised* phase [38, 93, 270]. The configuration of the order parameters is such that the student reproduces her teacher up to sign changes that guarantee the alignment between w_2 and F_1 is maximal, i.e. $\mathrm{sgn}(w_2^k) = \mathrm{sgn}(F_1^k)$. The final value of the test error of a student trained with DFA is the same as that of a student trained with BP on the same teacher. *Choice of the feedback vector* In the main text, we saw how a wrong choice of feedback vector F_1 can prevent a ReLU student from learning a task. Here, we show that also for sigmoidal student, a *wrong* choice of feedback vector F_1 is possible. As Fig. F.1 shows, in the case where the F_1 is taken orthogonal to the teacher second layer weights, a student whose weights are initialised to zero remains stuck on the plateau and is unable to learn. In contrast, when the F_1 is chosen with random i.i.d. components drawn from the standard normal distribution, perfect recovery is achieved.

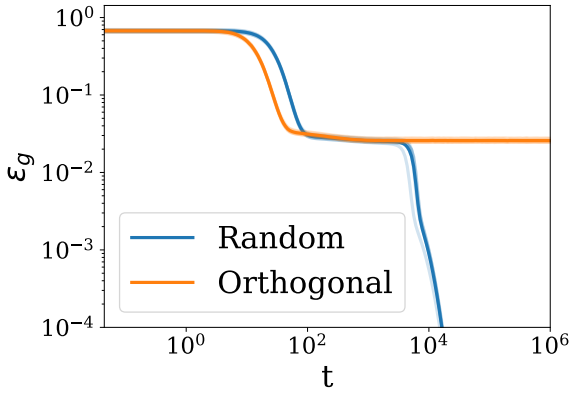


Figure F.1: Test error of a sigmoidal student started with zero initial weights. The feedback vector F_1 is chosen random (blue) and orthogonal to the teacher's second layer weights $\tilde{w}_2$ (orange). *Parameters:* $\eta = 0.1, K = M = 2$.

F.3 Derivation of weight alignment

Since the network is linear, the update equations are (consider the first three layers only):

$$\mathrm{d}w_1 = -\eta(F_1 e)x^T, \tag{F.12}$$
$$\mathrm{d}w_2 = -\eta(F_2 e)(w_1 x)^\top, \tag{F.13}$$
$$\mathrm{d}w_3 = -\eta(F_3 e)(w_2 w_1 x)^\top \tag{F.14}$$

First, it is straightforward to see that

$$w_1^t = -\eta \sum_{t'=0}^{t-1} F_1 e_{t'} x_{t'}^\top = F_1 A_1^t \tag{F.15}$$
$$A_1^t = -\eta \sum_{t'=0}^{t-1} e_{t'} x_{t'}^\top \tag{F.16}$$

This allows to calculate the dynamics of w_2^t:

$$\mathrm{d}w_2^t = -\eta F_2 e_t (A_1^t x_t)^\top F_1^\top \tag{F.17}$$
$$w_2^t = -\eta \sum_{t'=0}^{t-1} F_2 e_t (A_1^{t'} x_{t'})^\top F_1^\top = F_2 A_2^t F_1^\top \tag{F.18}$$
$$A_2^t = -\eta \sum_{t'=0}^{t-1} e_{t'} (A_1^{t'} x_{t'})^\top = \eta^2 \sum_{t'=0}^{t-1} \sum_{t''=0}^{t'-1} (x_{t'} \cdot x_{t''}) e_{t'} e_{t''}^\top. \tag{F.19}$$

Which in turns allows to calculate the dynamics of w_3^t:

$$\mathrm{d}w_3^t = -\eta F_3 e_t (F_2 A_2^{t'} F_1^\top F_1 A_1^{t'} x_t)^\top \tag{F.20}$$

$$w_3^t = -\eta \sum_{t'=0}^{t-1} F_3 e_{t'} (F_2 A_2^{t'} F_1^\top F_1 A_1^{t'} x_t)^\top = F_3 A_3^t F_2^\top \tag{F.21}$$

$$A_3^t = -\eta \sum_{t'=0}^{t-1} F_3 e_{t'} (A_2^{t'} F_1^\top F_1 A_1^{t'} x_{t'})^\top \tag{F.22}$$

$$= \eta^2 \sum_{t'=0}^{t-1} \sum_{t''=0}^{t'-1} (A_1^{t'} x_{t'}) \cdot (A_1^{t''} x_{t''}) e_{t'} e_{t''}^\top. \tag{F.23}$$

By induction it is easy to show the general expression:

$$A_1^t = -\eta \sum_{t'=0}^{t-1} e_{t'} x_{t'}^\top \tag{F.24}$$

$$A_2^t = \eta^2 \sum_{t'=0}^{t-1} \sum_{t''=0}^{t'-1} (x_{t'} \cdot x_{t''}) e_{t'} e_{t''}^\top \tag{F.25}$$

$$A_{h\geq 3}^t = \eta^2 \sum_{t,t'=0} (A_{h-2}^{t'} \dots A_1^{t'} x_{t'}) \cdot (A_{h-2}^{t''} \dots A_1^{t''} x_{t''}) e_{t'} e_{t''}^\top \tag{F.26}$$

Defining $A_0 \equiv \mathbb{I}_{n_0}$, one can rewrite this as in Eq. 5.38

$$A_{h\geq 2}^t = \eta^2 \sum_{t'=0}^{t-1} \sum_{t''=0}^{t'-1} (B_h^{t'} x_{t'}) \cdot (B_h^{t''} x_{t''}) e_{t'} e_{t''}^\top, \tag{F.27}$$

$$B_h = A_{h-2} \cdots A_0. \tag{F.28}$$

F.4 Impact of data structure

To study the impact of data structure on the alignment, the simplest setup to consider is that of Direct Random Target Projection [95]. Indeed, in this case the error vector $e_t = -y_t$ does not depend on the prediction of the network: the dynamics become explicitly solvable in the linear case.

For concreteness, we consider the setup of [190] where the targets are given by a linear teacher, $y = Tx$, and the inputs are i.i.d Gaussian. We denote the input and target correlation matrices as follows:

$$\mathbb{E}\left[xx^\top\right] \equiv \Sigma_x \in \mathbb{R}^{n_0 \times n_0}, \tag{F.29}$$

$$\mathbb{E}\left[TT^\top\right] \equiv \Sigma_y \in \mathbb{R}^{K_H \times K_H} \tag{F.30}$$

If the batch size is large enough, one can write $x_t x_t^\top = \mathbb{E}\left[xx^\top\right] = \Sigma_x$. Hence the

dynamics of Eq. 5.32 become:

$$\mathrm{d}w_1^t = -\eta(F_1 e_t)x_t^T = \eta F_1 T x_t x_t^\top = \eta F_1 T \Sigma_x \tag{F.31}$$

$$\mathrm{d}w_2^t = -\eta(F_2 e_t)(w_1 x_t)^\top = \eta F_2 T \Sigma_x w_1^\top \tag{F.32}$$

$$= \eta^2 F_2 \left(T\Sigma_x^2 T^\top\right) F_1^\top \tag{F.33}$$

$$\mathrm{d}w_3^t = -\eta(F_3 e_t)(w_2 w_1 x_t)^\top = \eta F_3 T \Sigma_x w_1^\top w_2^\top \tag{F.34}$$

$$= \eta^3 F_3 \left(T\Sigma_x^2 T^\top\right)\left(T\Sigma_x^2 T^\top\right) F_2^\top \tag{F.35}$$

From which we easily deduce $A_1^t = \eta T \Sigma_x t$, and the expression of the alignment matrices at all times:

$$A_{h\geq 2}^t = \eta^l \left(T\Sigma_x^2 T^\top\right)^{h-1} t \tag{F.36}$$

As we saw, GA depends on how well-conditioned the alignement matrices are, i.e. how different it is from the identity. To examine deviation from identity, we write $\Sigma_x = \mathbb{I}_{n_0} + \tilde{\Sigma}_x$ and $\Sigma_y = \mathbb{I}_{K_H} + \tilde{\Sigma}_y$, where the tilde matrices are small perturbations. Then to first order,

$$A_{h\geq 2}^t - I_{K_H} \propto (h-1)\left(\tilde{\Sigma}_y + 2T\tilde{\Sigma}_x T^\top\right) \tag{F.37}$$

Here we see that GA depends on how well-conditioned the input and target correlation matrices Σ_x and Σ_y are. In other words, if the different components of the inputs or the targets are correlated or of different variances, we expect GA to be hampered, observed in Sec. 5.2.5. Note that due to the $h-1$ exponent, we expect poor conditioning to have an even more drastic effect in deeper layers.

Notice that in this DRTP setup, the norm of the weights grows linearly with time, which makes DRTP inapplicable to regression tasks, and over-confident in classification tasks. It is clear in this case the the first layer learns the teacher, and the subsequent layers try to passively transmit the signal.

F.5 Details about the experiments

Direct Feedback Alignment implementation

We build on the Pytorch implementation of DFA implemented in [167], accessible at `https://github.com/lightonai/dfa-scales-to-modern-deep-learning/tree/master/TinyDFA`. Note that we do not use the shared feedback matrix trick introduced in this work. We sample the elements of the feedback matrix F_h from a

centered uniform distribution of scale $1/\sqrt{K_h + 1}$.

Experiments on realistic datasets

We trained 4-layer MLPs with 100 nodes per layer for 1000 epochs using vanilla SGD, with a batch size of 32 and a learning rate of 10^{-4}. The datasets considered are MNIST and CIFAR10, and the activation functions are Tanh and ReLU.

We initialise the networks using the standard Pytorch initialization scheme. We do not use any momentum, weight decay, dropout, batchnorm or any other bells and whistles. We downscale all images to 14×14 pixels to speed up the experiments. Results are averaged over 10 runs.

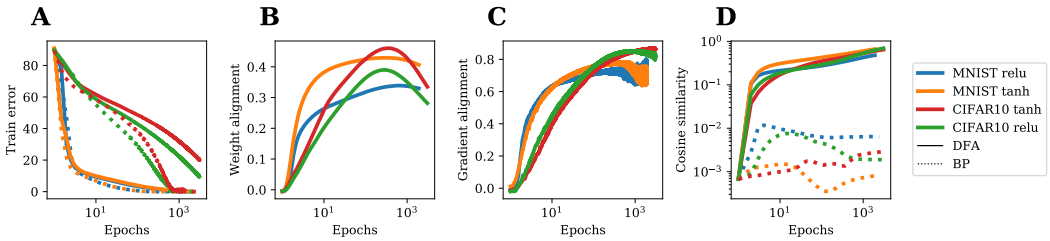
For completeness, we show in Fig. F.2 the results in the main text for 4 different levels of label corruption. The transition from Alignment phase to Memorisation phase can clearly be seen in all cases from the drop in weight alignment. Three important remarks can be made:

- **Alignment phase**: Increasing label corruption slows down the early increase of weight alignment, as noted in Sec. 5.2.5.
- **Memorization phase**: Increasing label corruption makes the datasets harder to fit. As a consequence, the network needs to give up more weight alignment in the memorization phase, as can be seen from the sharper drop in the weight alignment curves.
- **Transition point**: the transition time between the Alignement and Memorization phases coincides with the time at which the training error starts to decrease sharply (particularly at high label corruption), and is hardly affected by the level of label corruption.

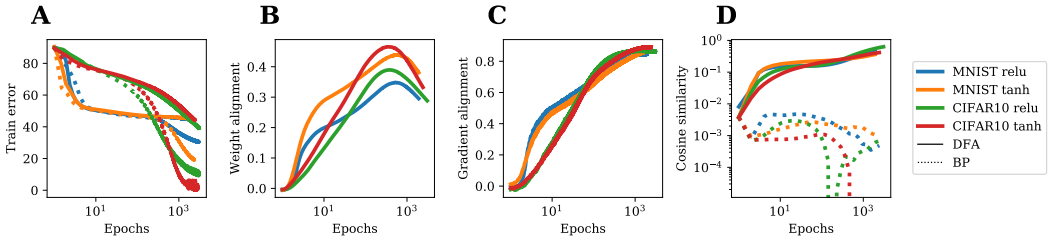
Experiment on the structure of targets

We trained a 3-layer linear MLP of width 100 for 1000 epochs on the synthetic dataset described in the main text, containing 10^4 examples. We used the same hyperparameters as for the experiment on nonlinear networks. We choose 5 values for α and β: 0.2, 0.4, 0.6, 0.8 and 1.

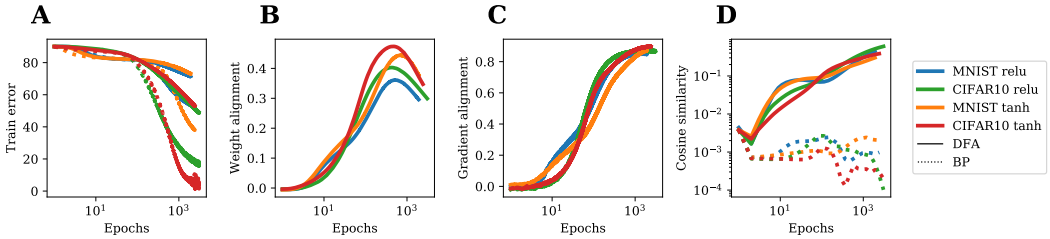
In Fig. F.3, we show the dynamics of weight alignment for both ReLU and Tanh activations. We again see the Align-then-Memorise process distinctly. Notice that decreasing α and β hampers both the mamixmal weight alignment (at the end of the alignment phase) and the final weight alignment (at the end of the memorisation phase).



(a) No label corruption

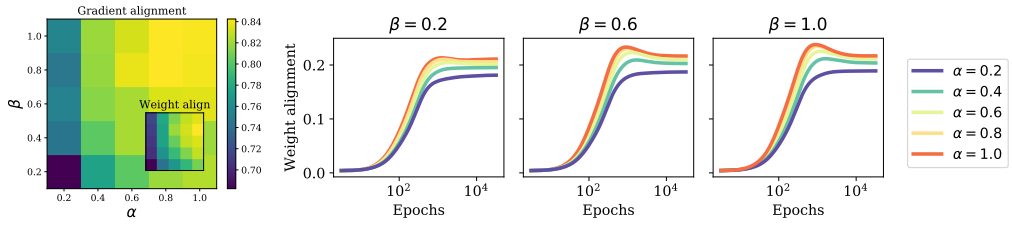


(b) 50% label corruption

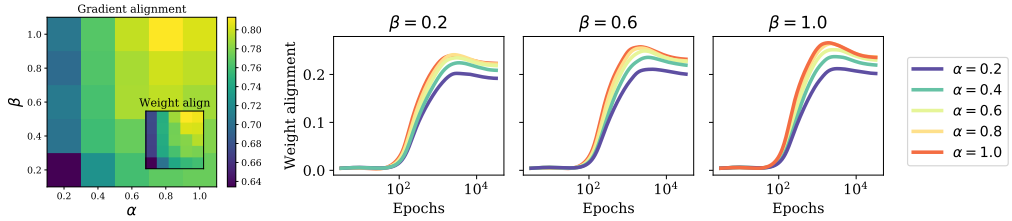


(c) 90% label corruption

Figure F.2: Effect of label corruption on training observables. **A**: Training error. **B** and **C**: Weight and gradient alignment, as defined in the main text. **D**: Cosine similarity of the weight during training.



(a) ReLU



(b) Tanh

Figure F.3: WA is hampered when the output dimensions are correlated ($\beta < 1$) or of different variances ($\alpha < 1$).