Inductive Bias from Layerwise Structure and Backpropagation in Neural Networks: Analysis through Simplified Models and Empirical Frameworks

Yoonsoo Nam

Rudolf Peierls Centre for Theoretical Physics

University of Oxford

This dissertation is submitted for the degree of

Doctor of Philosophy in Theoretical Physics

November 2024

To my wife, Suhyun

Acknowledgements

I would like to express my deepest gratitude to all my collaborators for their dedication and support. First, I am thankful to Prof. Ard Louis for the opportunity to pursue a DPhil at Oxford. Working alongside Nayara has been a truly enriching experience, where I learned how insightful feedback can make research both enjoyable and productive. Seokhyeong, through his elegant equations, always clarified my vague intuitions and sparked new ideas. Chris always kindly shared his vast energy and motivation alongside his insights not only as a talented collaborator but also as a great friend. Ouns, having the rigor of a mathematician but the intuition of a physicist, always opened up my eyes to approaching a problem. Ilja's vast knowledge of theoretical deep learning has always been invaluable. Clementine, being a sociable, kind, and brilliant researcher with vast expertise in linear neural networks, opened my eyes on the field of dynamics.

I would also like to thank other brilliant researchers with whom I've had the pleasure of discussing ideas: Charlie, Zohar, Soufiane, Ching, Nora, Niclas, and Yechan. Although our discussions were brief and often on topics outside their main areas of interest, they generously devoted their time, leading to valuable and insightful conversations. I thank all master students Eric, Minjun, Rhys, and Hadi, for their patience and collaboration in their projects. My heartfelt thanks go to my officemates — Shoufeng, Mehrana, and Hanzi -– for their lively discussions and unwavering support. Finally, I extend my thanks to all past and present members of the Louis group for their warm welcome and camaraderie.

I am deeply grateful to those who supported me through challenging times. Chris Cullen from the Oxford Counselling Service provided warm and professional guidance, while Linda from Physics HR offered her unwavering kindness and assistance. My medic friend Ilhoon has helped lift many of the burdens I carried with his 'prescription' of dragging me to the gym.

Most importantly, I want to express my deepest thanks to my wife, Suhyun, who endured so much throughout my studies. She has been my greatest source of support and love, standing by me through the pandemic, tight deadlines, scorching summers, freezing winters with boiler breakdowns, blackouts, and all the hardships life threw our way. None of this work would have been possible without her.

Abstract

Deep Neural Networks (DNNs) are revolutionizing industries, pushing the boundaries of image and language processing. Despite their remarkable performance, we still have little understanding of how they achieve it. Unlocking their potential and ensuring their safe use requires exploring their inductive bias — the preferences that guide models in selecting solutions from countless possibilities.

Another mystery lies in their dynamics. Trained via gradient descent, DNNs often find generalizable solutions among many that could overfit. Practitioners developed tools to control training dynamics, and a deeper understanding of dynamics helps to interpret decisions, debug models, and impose desired constraints. However, complex architectures, non-linear activations, and finite yet vast parameters make analytical theories challenging.

In this thesis, we adopt a physicist's approach — studying simpler, often solvable models — to explore the **dynamical inductive bias of DNNs**, focusing on gradient descent dynamics within a layerwise structure as the core principle. Understanding the dynamical inductive bias through solvable models allows us to explain phenomena such as neural collapse, scaling laws, emergence, lazy/rich regimes, and grokking observed in practical models. Finally, we revisit the long-standing question of how dynamics correlate with performance, offering insights for future research.

Because inductive biases are often intractable or ill-defined, we begin with a tutorial on linear models trained with gradient descent where the inductive bias is well formalized and understood. We diagonalize features to decouple their learning dynamics and define three properties for each feature: quality, utilization, and intensity. We review the linear model's inductive bias toward features with larger intensity and demonstrate the benign overfitting as an example of the inductive bias. As a tutorial, we leave the formal derivation of the inductive bias to the references and only share intuitive examples.

In the next chapter, we extend the framework (quality, utilization, and intensity) of linear models to neural networks to quantify their feature learning – or how the model evolves the features that better describe the data. We empirically observe that DNNs have an inductive bias toward finding the minimum number of features to express the data. We refer to dynamics with a strong bias as the minimum feature (MF) regime, and its complement as the extended feature (EF)

regime. We conclude by showing that the MF regime implies neural collapse (NC) – a collapse of last layer features into low-rank structure – and discuss how our framework extends beyond NC.

In the third chapter, we propose the dynamical feedback principle or how the layerwise structure amplify the dynamics under gradient descent. Using the principle, we highlight how the dynamics of layerwise linear models (amplifying dynamics) differs from (single layer) linear models and how it leads stage-like training where more important features are learned in sequences. We conclude by showing that a linear neural network (a member of layerwise linear models) or an unconstrained feature model can explain neural collapse observed in the previous chapter.

We then examine the multilinear model — a variant of layerwise linear models with prebuilt features — to study scaling laws, which describe how loss scales with additional resources. We study the scaling laws of a two-layer MLP trained on the multitask sparse parity task, where skills appear with power-law frequency and form orthogonal functions on the dataset measure. Using the multilinear model, which incorporates these skills as prebuilt features, we formally derive the time, data, parameter, and compute scaling laws observed in the MLP.

Extending the multilinear model, we predict the time, data, and parameter emergence – an abrupt change in the model's performance on a skill – observed in a 2-layer MLP trained on the multitask sparse parity task. We discuss stage-like training, driven by layerwise dynamics, that leads to sequential skill learning, explaining why our decoupled model with a fixed basis and no feature learning effectively approximates a 2-layer MLP that learns features and lacks skill decoupling.

In Chapter 6, we explore how mitigating the amplifying dynamics of layerwise structure can lead to the dynamics of linear models or the lazy regime. We also examine grokking — where training accuracy saturates long before test accuracy — as a transition from lazy to rich (non-linear) regimes. Solving a scalar-in, scalar-out linear neural network, we identify key factors governing the lazy/rich regimes, such as initialization scale, target scaling, and input scaling. Empirically, we show that the lazy/rich dichotomy aligns with the EF/MF dichotomy, both reflecting the degree of dynamical feedback among the layers. Finally, we demonstrate that methods to initialize linear neural network in the rich regime also eliminate delayed generalization in practical models.

In Chapter 7, we present our empirical findings within the framework examining how learning rate, weight decay, batch normalization, and architecture influence both amplifying dynamics and performance. While we observe a strong correlation between amplifying dynamics – measured through the tightness of MF regime – and performance, developing a more solvable model to understand this mechanism is left for future work and only discuss our current speculation.

Originality, Publications, and Presentations

This work has not been submitted to any other university or to any other program at the University of Oxford and is wholly original unless otherwise stated in the thesis. We used dataset widely used in machine learning:

- MNIST from [1]
- Fashion MNIST from [2]
- CIFAR10 and CIFAR100 are from [3]

Notes on specific chapters:

1. Chapter 1 is original.
2. Chapter 2 is original containing only Yoonsoo Nam's contribution in [4]
3. Chapter 3 is original containing only Yoonsoo Nam's contribution in [5, 6].
2. Chapter 2 is original containing only Yoonsoo Nam's contribution in [5].
3. Chapter 2 is original containing only Yoonsoo Nam's contribution in [5].
5. Chapter 5 is original containing only Yoonsoo Nam's contribution in [4]
6. Chapter 6 is original containing only Yoonsoo Nam's contribution in [6]

Additional work not included in the thesis: [7, 8]

Contents

1 Preliminary and linear models

Contents

1.1 Introduction

In learning theory, the learning agent returns the hypothesis that best describes the training data. For a less expressive model, often one hypothesis in a model best fits the data. When multiple hypotheses of the model fit the data, however, the learning agent's inductive bias – preference of the learning agent – determines the returned hypothesis.

Overparameterized linear models and deep neural networks (DNNs) trained with gradient descent algorithms are examples of where inductive bias plays a crucial role in their performance. In this chapter, we introduce basic concepts in learning theory and review the inductive bias of overparameterized linear models, especially how it relates to the gradient descent dynamics.

1.2 Random variables

Random variables are variables with an underlying probability distribution or a measure. A familiar example will be a Gaussian random variable:

$$x \sim q(x), \qquad q(x) = \frac{1}{\sqrt{2\pi}} e^{-x^2/2}. \tag{1.1}$$

The tilde $\sim$ means that x is a random variable with an underlying distribution q.

Transformed random variables are also random variables. For a random variable x with underlying distribution $q(x)$ and a function f, $f(x)$ is also a random variable with a distribution determined by the transformation of x through f. For example, we obtain a Chi-squared random variable by squaring a Gaussian random variable.

Linear correlation. The linear correlation between two random variables y and z with underlying joint distribution $q(y, z)$ is

$$\langle y|z\rangle_{q(y,z)} := \mathbf{E}_{q(y,z)}[yz] = \int yzq(y,z)dydz, \tag{1.2}$$

where $\langle\cdot|\cdot\rangle$ the braket notation used in physics. In the thesis, we often assume y and z are both transformed from a random variable x (i.e. $y = f(x)$ and $z = g(x)$). Then the linear correlation between y and z becomes an inner product between two functions f and g over a measure q

$$\langle f(x)|g(x)\rangle_{q(x)} = \mathbf{E}_{q(x)}[f(x)g(x)] = \int f(x)g(x)q(x)dx, \tag{1.3}$$

where $q(x)$ is the underlying distribution for random variable x. We will often discard the random variable x and the underlying distribution $q(x)$ to write $\langle f|g\rangle$ in the braket notation.

Function space. If all random variables of interest are transformed from $x \sim q(x)$ with a measurable map (L2 integrable function), the maps form a **function space** $\mathcal{H}$: to be specific, a Hilbert space. The function space, a set of functions, has the linear correlation as the inner product and a set of orthogonal basis functions. For example, the Hermite polynomials,

$$H_k(x) = (-1)^k e^{x^2/2} \frac{d^k}{dx^k} e^{-x^2/2}, \tag{1.4}$$

form the orthogonal basis [9] for a set of (L2 integrable) functions that satisfy

$$\int_{-\infty}^{\infty} |f(x)|^2 \, e^{-x^2/2} dx < \infty. \tag{1.5}$$

The underlying probability distribution $q(x)$ for this function space is a Gaussian distribution, and we can check by the inner product (Eq. (1.3)) that the Hermite polynomials are orthogonal

$$\langle H_j | H_k \rangle = \int_{-\infty}^{\infty} H_j(x) H_k(x) \frac{1}{\sqrt{2\pi}} e^{-x^2/2} dx = k! \delta_{jk}, \tag{1.6}$$

where δ is Kronecker delta. In short, any function f in the function space $\mathcal{H}$ (Eq. (1.5)) can be spanned by the Hermite polynomials $[H_0, H_1, \cdots, H_\infty]$ or

$$f(x) = \sum_{k=0}^{\infty} \frac{\langle f | H_k \rangle}{\langle H_k | H_k \rangle} H_k(x). \tag{1.7}$$

1.3 Learning theory

Learning is to guess the dataset's target function f^* that maps the inputs to the outputs or labels. A model is trained with independent and identically distributed (i.i.d) samples of the dataset called the training set. After training, the performance or the similarity between the model's learned function $\hat{f}$ and the target function is measured by a loss.

Dataset We define sample space $\mathcal{X}$ as the space of all possible inputs and assume a probability distribution q on $\mathcal{X}$, making our input a random variable. Then we will assume a target function that maps inputs to a C dimensional vector or $f^* : \mathcal{X} \to \mathbb{R}^C$. The output or label $f^*(x)$ becomes a random variable that depends on the input.

For example, $\mathcal{X}$ of MNIST will be a hypothetical set of all possible hand-written digit images in $\mathbb{R}^{28\times 28}$, $q(x)$ the probability that an image x will be sampled, and f^* the function from the digit in the image into a 10 dimensional one-hot vector.

A noise-free training set is n independently identically distributed samples of the inputs from the true distribution q and their output generated by the target function $f^*(x)$:

$$S_{train} := \{(x^{(1)}, f^*(x^{(1)})), \cdots, (x^{(n)}, f^*(x^{(n)}))\}. \tag{1.8}$$

The superscript with parenthesis (e.g. $x^{(i)}$) denote the i^{th} sample. For a noisy setup, a Gaussian noise σ is added to the labels:

$$S_{train} := \{(x^{(1)}, f^*(x^{(1)}) + \sigma), \cdots, (x^{(n)}, f^*(x^{(n)}) + \sigma)\}. \tag{1.9}$$

Linear model For an input random variable $x \sim q$, we can define a p dimensional linear model with a feature map $\Phi : \mathcal{X} \to \mathbb{R}^p$ that maps inputs to a p dimensional vector. We call the p-dimesional random variable $[\Phi_1(x), \Phi_2(x), \cdots, \Phi_p(x)]$ the features and each entry a feature. A linear model is a set of all linear combinations of the features

$$\hat{f}(x; w) = \sum_{k=1}^{p} w_k \Phi_k(x). \tag{1.10}$$

where $w \in \mathbb{R}^p$ are the parameters and we assumed a scalar output function $\hat{f} : \mathcal{X} \to \mathbb{R}$. Note that Φ can be a **non-linear** transformation on x, and the name linear model means linearity between the parameters and the output.

Mean square error (MSE) loss The loss, often called the generalization loss, measures the similarity between the learned function $\hat{f}$ and the target function f^*. In our thesis, unless explicitly specified, we will assume the mean square error loss (MSE) [10]

$$\mathcal{L}(\hat{f}) = \frac{1}{2}\mathbf{E}_{q(x)}\left[|\hat{f}(x) - f^*(x)|^2\right]. \tag{1.11}$$

Because the true distribution $q(x)$ is inaccessible, generalization loss is approximated by the test loss which replaces $q(x)$ with i.i.d. samples unseen during training (i.e. test set). Replacing $q(x)$ with the empirical distribution from the training set gives the empirical loss or the training loss $\mathcal{L}_{emp}$

$$\mathcal{L}_{emp}(\hat{f}) = \frac{1}{2n}\sum_{i=1}^{n}|\hat{f}(x^{(i)}) - f^*(x^{(i)})|^2.$$

Gradient descent algorithm The gradient descent algorithm is a widely used iterative algorithm to minimize the training loss by updating the parameters along the direction of the gradient of the loss. A continuous update is called the gradient flow (GF) and the dynamic equation for a parameter w_k is

$$\frac{dw_k}{dt} = -\eta\frac{d\mathcal{L}_{emp}}{dw_k}, \tag{1.12}$$

where η is a learning rate and $\mathcal{L}_{emp}$ is the (empirical) training loss.

1.4 Additional preliminaries

Hilbert-Schmidt operator - infinite-dimensional symmetric matrix A Gram matrix is a positive semidefinite symmetric matrix of the form AA^T for a real-valued matrix A. It maps from a finite vector space to a finite vector space. Analogously for a function space $\mathcal{H}$, we can define a Hilbert-Schmidt (integral) operator $\mathcal{T} : \mathcal{H} \to \mathcal{H}$ with the feature map $\Phi : \mathcal{X} \to \mathbb{R}^p$.

$$\mathcal{T}[f](x') := \int_{\mathcal{X}}\sum_{k=1}^{p}\Phi_k(x)\Phi_k(x')f(x)q(x)dx, \tag{1.13}$$

where $q(x)$ is the underlying distribution and $\Phi_k(x)$ is the k^{th} entry of $\Phi(x)$. The integral operator maps a function to a function, and in the bra-ket notation, the integral operator $\mathcal{T}$ is

$$\mathcal{T} = \sum_{k=1}^{p} |\Phi_k\rangle \langle \Phi_k| , \tag{1.14}$$

which explicitly shows that it takes in a function $f : \mathcal{X} \to \mathbb{R}$ with right 'bra' $\langle \cdot|$ and returns a function with the left 'ket' $|\cdot\rangle$.

Analogous to how a symmetric real matrix is diagonalized, we can diagonalize the integral operator by Mercer's theorem [11]

$$\mathcal{T} := \sum_{k=1}^{p} \rho_k |e_k\rangle \langle e_k| , \qquad \mathcal{T}[e_k] = \rho_k e_k, \qquad \langle e_k | e_k \rangle = \delta_{kl} \tag{1.15}$$

where ρ_k is non-negative eigenvalue and e_k are the orthonormal eigenfunctions.

Effective dimension A gram matrix or an integral operator may be fully ranked, but their eigenvalues may decay fast (e.g. by a power-law). The small eigendirections are often negligible, and researchers use various effective dimensions to measure the number of **significant** dimensions. In this thesis, we will use the exponent of entropy [12] to measure the effective dimension.

For a matrix or an operator with positive eigenvalues $[\rho_1, \cdots, \rho_p]$, the effective dimension is

$$D_{eff}(\rho) = \exp\left(- \sum_i \frac{\rho_i}{\sum_j \rho_j} \ln \left(\frac{\rho_i}{\sum_j \rho_j} \right) \right). \tag{1.16}$$

Note that the effective dimension, the exponential of Shannon entropy, is d when the eigenvalues have d equal non-zero entries. A slower decay of entries of ρ (in non-increasing order of entries) generally yields a higher effective dimension than a vector with a faster decay.

The effective dimension of a matrix (operator) can be interpreted as the number of linearly independent vectors (functions) needed to effectively describe the matrix (operator). This is similar in spirit to the principal component analysis (PCA) [13] in that only the directions with significant variance are considered.

1.5 Linear models

Despite their simplicity, linear models (often called kernel methods) can approximate infinite-width DNNs [14, 15] and serve as tools for understanding the inductive bias of DNNs [16, 17]. To understand the inductive bias of linear models in GF, we need to quantify features and their metrics because a linear model is a linear combination of features.

Here, we show that GF dynamics naturally defines orthogonal features, with each learned at different speeds. We then define each feature's usefulness, utilization, and intensity to further analyze the inductive bias from the dynamics.

1.5.1 Dynamics of linear model

The dynamics equation of a linear model with feature map Φ on trained with GF (Eq. (1.12)) under MSE loss (Eq. (1.11)) is

$$\frac{df}{dt} = -\eta \mathcal{T}\left[f - f^*\right], \tag{1.17}$$

where the integral operator $\mathcal{T} : \mathcal{H} \to \mathcal{H}$ is

$$\mathcal{T}[f](x') := \int_{\mathcal{X}} \Phi(x)^T \Phi(x') f(x) q(x) dx. \tag{1.18}$$

The derivation of Eq. (1.17) follows by expanding f in the features and plugging in Eq. (1.12) for all parameters

$$\frac{df}{dt}(x') = \sum_k^p \frac{dw_k}{dt} \Phi_k(x') = \sum_k^p -\eta \frac{d\mathcal{L}}{dw_k} \Phi_k(x') \tag{1.19}$$

$$= -\eta \int_{\mathcal{X}} \left(f(x) - f^*(x)\right) \sum_k^p \frac{df}{dw_k}(x) q(x) dx \Phi_k(x') \tag{1.20}$$

$$= -\eta \int_{\mathcal{X}} \left(f(x) - f^*(x)\right) \sum_k^p \Phi_k(x) \Phi_k(x') q(x) dx \tag{1.21}$$

$$= -\eta \mathcal{T}\left[f - f^*\right], \tag{1.22}$$

where in the second line, we used Eq. (1.11) and Eq. (1.12), and assumed infinite datapoints so $\mathcal{L}_{emp} = \mathcal{L}$. In the last line, we used the definition of integral operator (Eq. (1.13)).

Using the eigenvalues ρ_k and eigenfunctions e_k the Hilbert-Schmidt operator $\mathcal{T}$, we obtain the dynamics for each $\langle e_k|f\rangle$ by inner producting e_k on both sides of Eq. (1.17),

$$\frac{d\langle e_k|f\rangle}{dt} = \int_{\mathcal{X}} e_k(x')\frac{df}{dt}(x')q(x')dx' = -\eta\int_{\mathcal{X}} e_k(x')\mathcal{T}\left[f-f^*\right](x')q(x')dx' \tag{1.23}$$

$$= -\eta\langle e_k|\mathcal{T}|f-f^*\rangle \tag{1.24}$$

$$= -\eta\rho_k(\langle e_k|f\rangle - \langle e_k|f^*\rangle). \tag{1.25}$$

Expanding f in the eigenfunction basis and plugging in Eq. (1.23), the function f at time t is a sum of p independent modes that saturate faster for larger eigenvalues:

$$f(x,t) = \sum_{k=1}^{p}\langle e_k|f^*\rangle\left(1-e^{-\eta\rho_k t}\right)e_k(x), \tag{1.26}$$

where we assumed f is a zero function at initialization. Eq. (1.26) shows that GF decouples the dynamics of linear models into p modes, where each mode corresponds to the evolution of $\langle e_k|f\rangle$ having a saturation speed governed by ρ_k [15].

1.5.2 Eigenfunctions - orthonormalized features

The transformation between the eigenfunction basis and the feature basis is

$$\Phi_i(x) = \sum_j O_{ij}\sqrt{\rho_j}e_j(x), \qquad O_{ij} = \frac{1}{\sqrt{\rho_j}}\langle\Phi_i|e_j\rangle\,. \tag{1.27}$$

where $O\in\mathbb{R}^{p\times p}$ is an orthogonal matrix (see Appendix A.1 for derivation). Using Eq. (1.27), the model can be reparameterized in the eigenfunction basis

$$f(x) = \sum_{k=1}^{p} w_k\Phi_k(x) = \sum_{k=1}^{p}\theta_k\sqrt{\rho_k}e_k(x). \tag{1.28}$$

Note that because $\theta = Ow$, the norm of the parameters is conserved (i.e. $\|\theta\| = \|w\|$), indicating that weighted basis $[\sqrt{\rho_1}e_1, \sqrt{\rho_2}e_2, \cdots, \sqrt{\rho_p}e_p]$ instead of normalized basis $[e_1, e_2, \cdots, e_p]$ must be used to represent the 'intensity' or norm of the features in $[\Phi_1, \cdots, \Phi_p]$. See Fig. 1.1 for an illustration of why eigenvalues are necessary when diagonalizing unnormalized, overlapping features.

Features The diagonalized features in the descending order of the eigenvalues $[\sqrt{\rho_1}e_1, \sqrt{\rho_2}e_2, \cdots, \sqrt{\rho_p}e_p]$ decouples the dynamics in GF dynamics, spans the model's function space $\mathcal{H}$, preserves the norm of the parameters (and features), and, unlike $[\Phi_1, \cdots, \Phi_p]$, is orthogonal. Thus, we will call the diagonalized features simply the **features**.

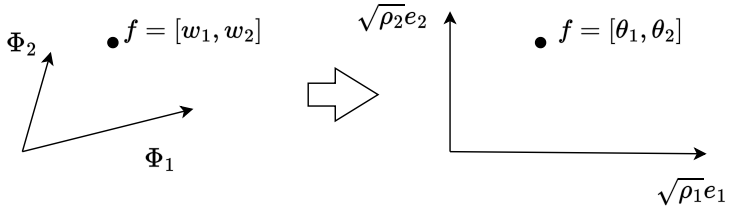


Figure 1.1: Diagonalization of features Since the features $[\Phi_1, \Phi_2, \cdots, \Phi_p]$ span a vector space, each function $\Phi_k : \mathcal{X} \to \mathbb{R}$ can be represented as a vector. The overlap among $\Phi_k's$ (non-zero linear correlation) and differing norms of Φ_k's result in diagonalized features $[\sqrt{\rho_1}e_1, \sqrt{\rho_2}e_2, \cdots, \sqrt{\rho_p}e_p]$ to have distinct intensities $\rho_1, \cdots, \rho_p$.

1.5.3 Metrics of diagonalized features

We quantify the properties of the diagonalized features $[\sqrt{\rho_1}e_1, \sqrt{\rho_2}e_2, \cdots, \sqrt{\rho_p}e_p]$: Intensity, quality, and utilization.

Intensity The intensity, or the eigenvalue ρ_k, of the k^{th} feature measures the projection of Φ along e_k in function space.

$$\rho_k = \langle e_k | \mathcal{T}[e_k] \rangle = \langle e_k | \sum_{j=1}^{p} |\Phi_j\rangle \langle \Phi_j | e_k \rangle = \sum_{j=1}^{p} \langle \Phi_j | e_k \rangle^2 . \tag{1.29}$$

It quantifies the feature's significance: how often the feature (along e_k) is non-zero and how large it is when observed. For example, let Φ is p-dimensional zero-mean Gaussian random variable with

$$\Phi(x_1, x_2) = \mathcal{N}\left(0, \Sigma = \begin{pmatrix} 1.5 & 0.5 \\ 0.5 & 1.5 \end{pmatrix}\right).$$

The eigenvalues of Σ is 2 and 1, which will be ρ_1 and ρ_2 in our formalism. In a vector representation (Fig. 1.1), the feature intensity is the norm of the feature vector along e_k.

Quality Quality of the k^{th} feature Q_k^* is a measure of how **useful** the feature is in describing the target function or how much of the target function can be projected onto e_k:

$$Q_k^* = \langle e_k | f^* \rangle^2 / \langle f^* | f^* \rangle . \tag{1.30}$$

Utilization Utilization of the k^{th} feature $\hat{Q}_k$ is a measure of the current use, or the projection of the learned function $\hat{f}$ along e_k:

$$\hat{Q}_k = \langle e_k | \hat{f} \rangle^2 / \langle \hat{f} | \hat{f} \rangle \tag{1.31}$$

Cumulative quality and utilization To quantify the usefulness of the first k features in describing the target/learned function, we define the cumulative quality Π_k^* and cumulative utilization $\hat{\Pi}_k$:

$$\Pi_k^* = \sum_{i=1}^{k} Q_i^*, \qquad \hat{\Pi}_k = \sum_{i=1}^{k} \hat{Q}_i. \tag{1.32}$$

Note that the cumulative measures are intensity-ordered, and a target function described by k features may not be expressed by the k most significant features; we will justify the intensity-ordering in the coming chapters.

1.5.4 Approximating the metrics in practice

To calculate the eigenvalues and eigenfunctions (and thus the quality, utilization, and intensity) for a feature map Φ, we also need the true input distribution q, which is in principle inaccessible. However, we can use Nyström method [18, 10] for a sufficiently large sample size of n to approximate the eigenfunctions and eigenvalues of interest.

The n samples of $\Phi(x)$ can be represented as $p \times n$ matrix which we call $\phi \in \mathbb{R}^{p \times n}$. The matrix ϕ can be singular value decomposed (SVD) into U, S, V to approximate the eigenvalues $[\sqrt{\rho_1}, \sqrt{\rho_2}, \cdots, \sqrt{\rho_p}]$ and the orthogonal matrix O in Eq. (1.27). The eigenvalues of $\mathcal{T}$ can be approximated by squaring the singular values $s \in \mathbb{R}^p$ of matrix ϕ, and O by the left column eigenvectors $u_k \in \mathbb{R}^p$ of ϕ. The k^{th} eigenfunction then can be approximated as

$$e_k(x) \approx u_k^T \Phi(x) / s_k, \tag{1.33}$$

which can be used on **any** data point x beyond the sample used to obtain ϕ. The algorithm is summarized in Algorithm 1.

Algorithm 1 Empirical eigenfunctions and eigenvalues

1: Φ, S_{tr} (Prepare a feature map Φ and n samples of input S_{tr})
2: $\phi \leftarrow \Phi(S_{tr})$ (forward transform the samples into the feature space)
3: $U, S, V \leftarrow SVD(\phi)$ (find the eigenvalues and eigenvectors via SVD)
4: $[u_1, u_2, \cdots, u_p] \leftarrow U$ (find the column vectors of U)
5: **for** k in $1, 2, \ldots, p$ **do**
6: $\quad \rho_k \leftarrow s_k^2$ (approximate eigenvalues)
7: $\quad e_k(x) \leftarrow u_k^T \Phi(x)/s_k$ (approximate eigenfunctions)
8: **end for**

In Algorithm 1, we implicitly assumed $n > p$ as we can only approximate up to $\min(n, p)$ eigenfunctions and eigenvalues. In practice, the eigenvalues often decay sufficiently rapidly that we only require precision on the first few significant eigenfunctions.

The inner product over the true measure q in calculating the quality Eq. (1.30) and utilization Eq. (1.31) is approximated by a Riemannian sum:

$$\frac{\langle e_k|f\rangle^2}{\|e_k\|_2^2\|f\|_2^2} = \frac{\left(\int_{\mathcal{X}} e_k(x)f(x)q(x)dx\right)^2}{\int_{\mathcal{X}} e_k(x)^2q(x)dx \int_{\mathcal{X}} f(x)^2q(x)dx} \approx \frac{\left(\sum_{x^{(i)}\in S_{te}} e_k(x^{(i)})f(x^{(i)})\right)^2}{\sum_{x^{(i)}\in S_{te}} e_k(x^{(i)})^2 \sum_{x^{(i)}\in S_{te}} f(x^{(i)})^2}, \tag{1.34}$$

where a sample set S_{te} is i.i.d samples of q different from those used to approximate the eigenfunctions.

1.6 Inductive bias of overparameterized linear models

In the infinite data scenario (Eq. (1.26)), linear models learn larger intensity features faster. While the inductive bias does not affect the performance in infinite training samples, it plays a crucial role in overparameterized setups ($p > n$) where multiple linear combinations of the features can fit the training set. Intuitively, if two eigenfunctions can perfectly fit the training samples, the eigenfunction with larger intensity will fit faster and thus **preferred** over the smaller intensity eigenfunction.

The preference or inductive bias toward larger features for overparameterized linear models trained with GF and zero initialization can be explicitly calculated:

$$L_k := \mathbf{E}_{S\sim q^n}\left[\frac{\langle \hat{f}|e_k\rangle}{\langle f^*|e_k\rangle}\right] = \frac{\rho_k}{\rho_k+\kappa}, \qquad \text{where } \sum_{k=1}^{p}\frac{\rho_k}{\rho_k+\kappa} = n. \tag{1.35}$$

The proof of Eq. (1.35) is out of the tutorial's scope and we leave the proof to the references on kernel regression [19, 20, 21, 16, 22, 23]. See [24] for an overview.[1]

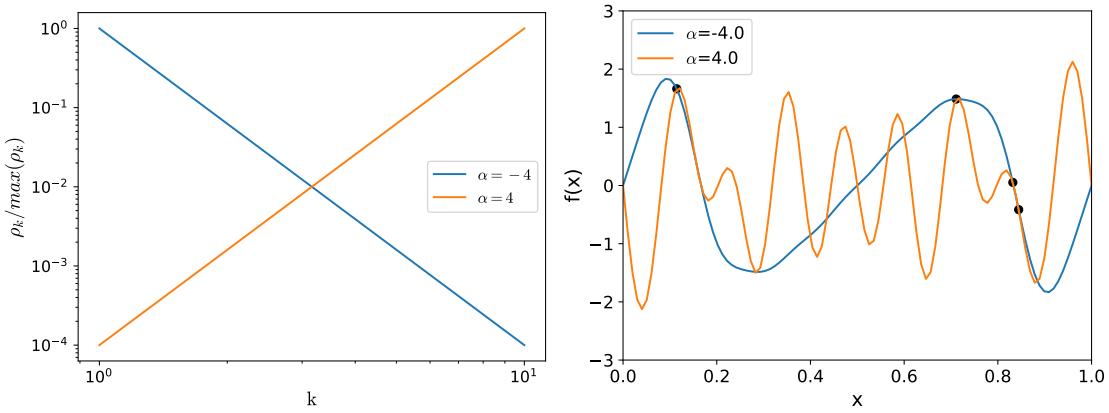
The learnability L_k measures the inductive bias or how likely the model will learn the signal $\langle f^*|e_k\rangle$ along e_k (in expectation) without mistakenly fitting the training sample with other eigenfunction. The constant κ is a positive constant that satisfies the second equation in Eq. (1.35). The larger eigenvalues ρ_k lead to larger learnability L_k, showing the inductive bias toward larger features. A similar value also appears in the analysis of Gaussian processes [10], which requires further research.

In Fig. 1.2(b), we present an example of inductive bias in which we fit the training samples with two overparameterized models with identical eigenfunctions but different eigenvalues (Fig. 1.2(a)). Note that two models, sharing the sample eigenfunctions, span the same function space and differ only in their "preference". The inductive bias toward larger features (eigenvalue), even for finite n, results in distinct fits where more significant features are used to express the training set.

1.6.1 Benign overfitting - good inductive bias

Benign overfitting, first studied by [25], refers to the condition where a heavily overparameterized linear model interpolates training data with noise, yet generalizes. In Fig. 1.3(a), the learned function interpolates all noisy training datapoints, but still closely resembles the target function with discrepancy only at sample points: benign overfitting. We demonstrate how a good inductive bias or the alignment between the task and the model separates the signal from the noise.

[1] For the connection between kernel ridgeless regression and GF, we can show that GF with zero initialization finds the same minimum norm solution as kernel ridgeless regression. see Appendix A.2



(a) Eigenvalues **(b)** Bias by the eigenvalues

Figure 1.2: Different inductive biases by the eigenvalues (intensity). Two 10-parameter linear models are trained on four data points (overparmaterized) with GF. Both Linear models use *sin* basis functions such that $f(x) = \sum_k^{10} w_k k^\alpha \sqrt{2}\sin(2\pi kx)$ – identical function space – but differ in the eigenvalues with $\alpha = -4$ (blue) and $\alpha = 4$ (orange), leading to different inductive biases. **(a):** The blue model has a greater intensity $\rho_k = k^{2\alpha}$ for lower-frequency sin functions, while the orange model shows the opposite. **(b):** The model fits show that the blue model used lower-frequency functions while the orange used higher-frequency functions to express the training samples.

Setup We prepare a p dimensional overparameterized linear model

$$f(x) = \sum_{k=1}^{p} w_k \sqrt{\rho_k} e_k(x),$$

where only the first d out of p eigenvalues are significant. We will assume that $d \ll n \ll p$ such that the learnability is near 1 for $k \leq d$, but negligibly small for $k > d$:

$$\rho_k = \begin{cases} 1, & k <= d \\ n/(p-d), & k > d \end{cases} \qquad L_k \approx \begin{cases} 1, & d <= n \\ (n-d)/(p-d), & d > n \end{cases} \tag{1.36}$$

The model is then heavily biased toward the first d features, learning the signal $\langle f^* | e_k \rangle$ in these features with high probability (Eq. (1.35)).

We set up the target function so the first d features can express it:

$$f^*(x) = \sum_{k=1}^{d} w_k^* e_k(x) \tag{1.37}$$

A Gaussian noise $\mathcal{N}(0, \sigma)$ is added to the labels of the training set:

$$y = f^*(x) + \mathcal{N}(0, \sigma). \tag{1.38}$$

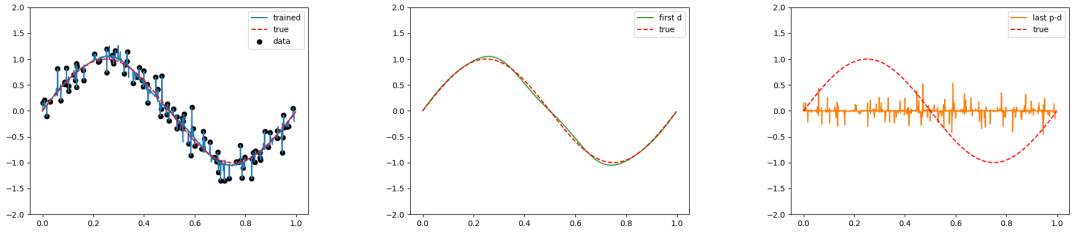
Task-model alignment The first d features are preferred by the model with larger L_k (Eq. (1.36)), and are also useful at expressing the target function (Eq. (1.37)). This alignment between the feature's intensity (model preference) and the quality (target expressiveness) is called a task-model alignment [19, 16].

The equation for the generalization loss in [19, 20, 21, 16, 22, 23] also readily shows that the model generalizes. However, we aim to understand how inductive bias and dynamics play a role in the process.

Signal fitting phase In the time scale of $1 \ll \eta t \ll (p-d)/n$, GF will initially learn only the first d features because of the large intensity difference (Eq. (1.26)). The learning resembles training a d-parameter linear model, capable of expressing the target function, on n noisy samples. Because $n \gg d$, the signal $\langle f^* | e_k \rangle$ in the first d features – the target function – will be learned with a high probability (Fig. 1.3(b)).

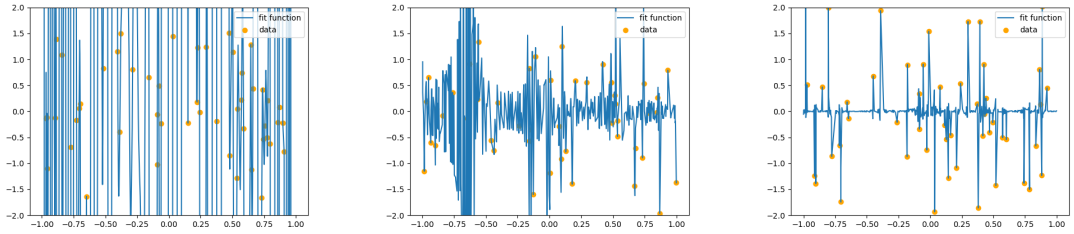
Noise fitting At a significantly later stage, all $p-d$ eigenfunctions fit the reminiscent noise. As seen in Fig. 1.4, a larger parameter to datapoint ratio makes the fitted function sharper, to the point when they become delta functions on the training datapoints. The extremely sharp function becomes the benign noise fitting function in Fig. 1.3(c). We can intuitively understand the sharp functions from the inductive bias toward the minimum norm solution: with more eigenfunctions, the model becomes more expressive, and delta functions minimize the norm of the function by concentrating variance (deviation away from $y = 0$) only near the training datapoints. This phenomenon is often expressed as the spread of the noise over many directions [25, 26].

Because the learner 'prefers' to see the signal of the first d eigenfunctions, the projection of the data into these eigenfunctions are learned, which also does not suffer from variance [16] as $d \ll n$. The leftover discrepency, which are mostly noise if the first d eigenfunctions approximated the true function, is fitted with the delta-like functions, which are benign on points aside from training set.



(a) $\sum_{k=1}^{p} \hat{w}_k e_k(x)$ **(b)** $\sum_{k=1}^{d} \hat{w}_k e_k(x)$ **(c)** $\sum_{k=d+1}^{p} \hat{w}_k e_k(x)$

Figure 1.3: Benign Overfitting A model with sine basis functions as e_k – $f(x) = \sum_k^{10} w_k \sqrt{\rho_k} \sqrt{2} \sin(2\pi k x)$ – were used with $d = 5, n = 100, p = 2000$. The target function used only the first feature: $f^*(x) = \sin(2\pi x)$. **(a):** After training, the learned function interpolates all training samples but closely resembles the target function. **(b):** The first d features fit the target function. **(c):** The latter $p - d$ features fit the reminiscent noise by a delta functions.



(a) 100 eigenfunctions **(b)** 400 eigenfunctions **(c)** 2000 eigenfunctions

Figure 1.4: Noise fitting function with more features 50 random labels were sampled from a unit Gaussian for inputs $x \in [0, 1]$. The data were fit with a uniformly biased linear model with sine basis functions $f(x) = \sum_k^p = w_k sin(2\pi k x)$ for different p: (a) 100, (b) 400, (c) 2000. As the number of features grows, the learned function becomes sharper, to the point in which they become a benign delta-like functions.

1.6.2 Malignant overfitting - bad inductive bias

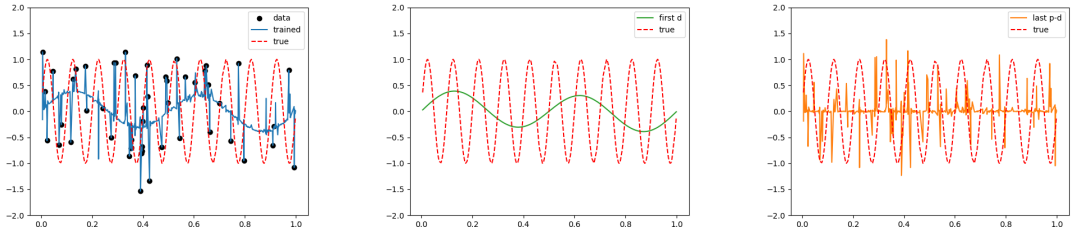
The separation between the noise and signal in benign overfitting was possible because the preferred features (high intensity) aligned with useful features (high quality). We demonstrate another model with a strong inductive bias but without the alignment between intensity and quality: a model with bad inductive bias.

We use the same model with strong inductive bias (Eq. (1.36)), but the target function now is expressed by the non-significant features

$$f^*(x) = \sum_{k=d+1}^{2d} w_k^* e_k(x). \tag{1.39}$$

The alignment is lost as the model no longer prefers the useful features with smaller intensities.

In Fig. 1.5(b), the model uses the first d features to express the training samples, but in contrast to Fig. 1.3, the first d features fail to express the target function. The latter $p-d$ features, similar to Fig. 1.3(c), fits the reminiscent training loss with delta-like functions, ignoring the signal from the target function. The mismatch between the intensity and quality of the features leads to a malignant overfitting Fig. 1.5(a).



(a) $\sum_{k=1}^{p} \hat{w}_k e_k(x)$ **(b)** $\sum_{k=1}^{d} \hat{w}_k e_k(x)$ **(c)** $\sum_{k=d+1}^{p} \hat{w}_k e_k(x)$

Figure 1.5: Malignant Overfitting A model with sine basis functions as e_k – $f(x) = \sum_k^{10} w_k \sqrt{\rho_k}\sqrt{2}\sin(2\pi k x)$ – were used with $d = 5, n = 100, p = 2000$. The target function used only the tenth feature: $f^*(x) = \sin(20\pi x)$. **(a):** After training, the learned function interpolates all training samples but differs from the target function. **(b):** The first d features resembles the learned function but differs from the target function. **(c):** The latter $p - d$ features fits with delta-like functions to interpolate the data, ignoring the signal from the tenth feature.

1.7 Summary

In this chapter, we reviewed the concepts in linear regression and showed that diagonalized features offer a decoupled interpretation for linear models with GF. We quantified three metrics for analyzing the diagonalized features: intensity, quality, and utilization.

Linear models trained with GF have an inductive bias toward higher-intensity features, learning them first and generalizing if these features align with high-quality features. We examined benign and malignant overfitting, where good or bad inductive bias causes dramatic performance differences.

2

Feature learning visualization framework

Contents

2.1 Introduction

Feature learning, described by Bengio et al. [27] as *"learning representations of the data that make it easier to extract useful information when building classifiers or other predictors."*, is recognized as the distinct chararteristic of DNNs [28, 29, 30, 31, 32, 33, 34]. Understanding feature learning – from its inductive bias [35, 36, 16], dynamics [37, 38], to interpretation [39, 40, 41, 42] – is a popular

conundrum of machine learning.

Though feature learning is widely seen as a performance-improving departure from models with fixed features -— often defined as intermediate layer updates [31], non-linear dynamics [43, 44], changes in kernels like NTK [45, 46, 47], NNGP kernel [48, 49, 34], or forward feature kernel [31, 17, 50] – how the features change remains an active area of research.

In this chapter, we define feature learning in DNN as a deviation from last-layer-only training and quantify the features' usefulness, utilization, and effective number by extending the framework for linear models (Section 1.5.3). With our framework, we visualize that generalizing models for vision tasks have an inductive bias toward a minimum number of features and discuss how it relates to Neural Collapse (NC).

2.2 Toy example and key insights

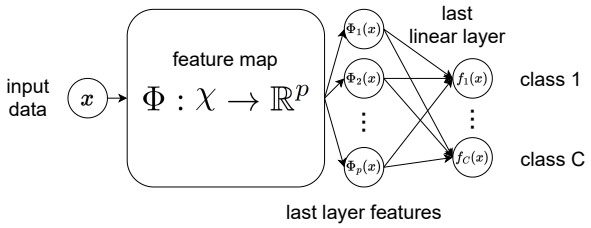


Figure 2.1: A DNN decomposed into a feature map and a linear last layer An abstract diagram depicting a DNN architecture as a combination of the forward feature-map $\Phi : \mathcal{X} \to \mathbb{R}^p$ from the dataspace $\mathcal{X}$ to the p inputs of the final layer, and final linear classifier for C-way classifcation. Most DNNs share this abstract structure and mainly vary in how they create their feature maps. Within this picture, we define feature learning as any change to the feature map upon training, reflected in changes to the p 'features' compared to initialization.

As illustrated in Fig. 2.1, a DNN can be decomposed into a forward feature map $\Phi : \mathcal{X} \to \mathbb{R}^p$ that maps the input (e.g. a 28×28 pixel image of MNIST) to a p-dimensional vector, and a final linear layer of finite width p. Analogous to the setup in Chapter 1, the inputs $x \in \mathcal{X}$ are sampled from a true distribution q (i.e. $x \sim q$) and features are the transformation of x via Φ. In contrast to the linear

model's fixed feature map, DNN's feature map, which has learnable parameters, changes during training or **feature-learns**.

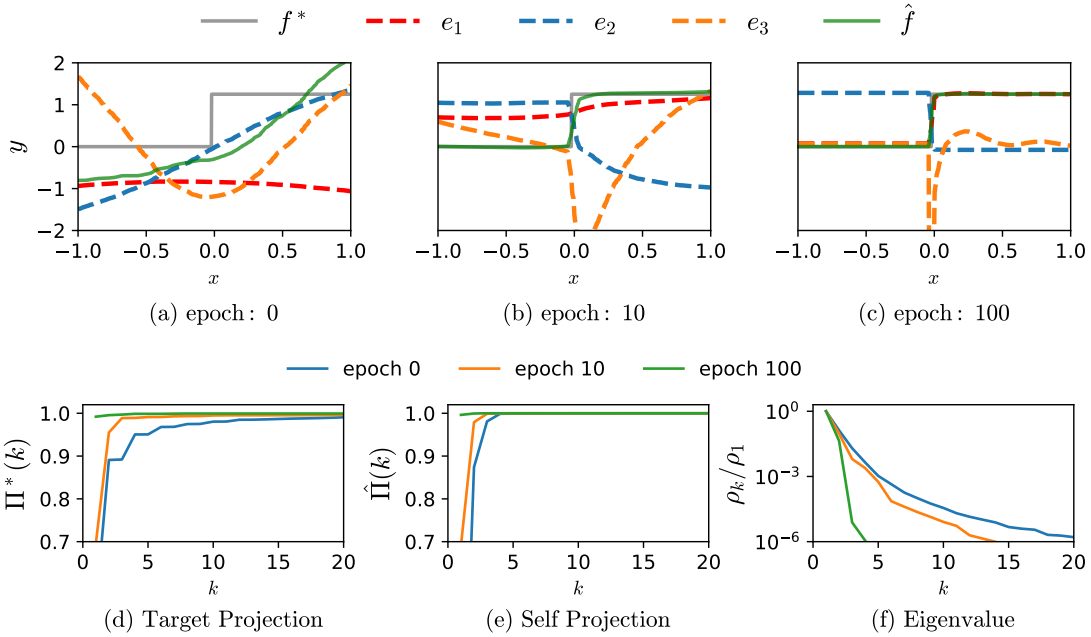


Figure 2.2: Toy model demonstrating feature learning by a DNN A 4-layer fully connected network (FCN) with width 1000 and scalar input and output, is trained to learn the Heaviside step function f^* over the domain $[-1, 1]$. **(a):** At initialization (epoch 0), the first three eigenfunctions e_i (dashed) of the feature map Φ are shown in order of their eigenvalues. If Φ and thus the features are fixed, about 20 features are needed to accurately approximate the target function f^* (solid grey) to an MSE loss less than 10^{-5}. **(b,c):** During training with SGD, DNN learns a single feature (red) that fits both the target and learned functions (f^* and $\hat{f}$). **(d,e,f):** To quantify how the features change, we plot the features metrics defined in Section 1.5.3 at different epochs. (d) and (e) shows that fewer features can express the target/learned function, and (f) shows that only a few features are significant after training.

Because all DNNs share a final **linear** classifier (Fig. 2.1), we use metrics of the diagonalized features in linear models (Section 1.5.3) – defined by the DNN's forward feature map Φ – to track the evolution of the last-layer features.

In Fig. 2.2(a–c), we plot the evolution of the first three features of a 4-layer FCN (width 1000) trained with SGD to fit a Heaviside function. Instead of using a linear combination of initial features, the FCN feature-learns a **single** feature (eigenfunction) that sufficiently describes the target/learned function (Fig. 2.2(c)).

Because the features are often challenging to visualize for practical DNNs with

high dimensional input, we used the feature metrics in Section 1.5.3 to visualize the change of features in Fig. 2.2(d–f): the cumulative quality $\Pi^*(k)$ in (d) and cumulative utilization $\hat{\Pi}(k)$ in (e) illustrate that fewer features can express the target/learned function while the normalized intensity ρ_k/ρ_1 in (f) shows that fewer features are significant after training. The chapter aims to extend the metrics in Fig. 2.2(d–f) and demonstrate DNN's consistent inductive bias toward a minimum number of features (in this case 1).

2.3 Definitions and methodology

For balanced C-way classification vision tasks, the target function is no longer a scalar but a vector function (i.e. $f^*(x) = [f_1^*(x), \cdots, f_C^*(x)]$), necessitating the extension of the metrics in Section 1.5.3. Because the training set partitions into equal subspace of classes A_i, each entry $f_i^* : \chi \to \mathbb{R}$ is

$$f_i^*(x) = \mathbf{1}_{A_i}(x) \qquad (A_i = \{x : x \text{ is in class } i\}) \tag{2.1}$$

where $\mathbf{1}_{A_i}$ is the indicator function for partitioned subspace A_i for each class. The partition makes all entries of the target function mutually exclusive

$$\langle f_i^* | f_j^* \rangle = \int_\chi f_i^*(x) f_j^*(x) q(x) dx = \frac{1}{C} \delta_{ij}, \tag{2.2}$$

where the C^{-1} on the right-hand side comes from the probability measure for each class i. Eq. (2.2) shows that we need at least C orthogonal scalar functions (thus features) to express f^*.

We extend the quality and utilization in Section 1.5.3 because the features must express all entries of vector target functions. While the intensity remains unchanged, we extend the effective dimension for a practical reason discussed in Section 2.5.

Quality To extend the inner product $\langle e_k | f^* \rangle$ for vector function, we define the target function space $\mathcal{H}^*$, spanned by $[f_1^*, \cdots, f_C^*]$, and the a projection operator $P_{\mathcal{H}^*} : L^2(\chi) \to L^2(\chi)$ onto this space. The quality Q_k^* of a feature e_k is

$$P_{\mathcal{H}^*}[g] := \sum_{j=1}^{C} \frac{1}{\|f_i^*\|^2} |f_i^*\rangle \langle f_i^* | g\rangle , \qquad Q_k^* := \frac{\langle e_k | P_{\mathcal{H}^*}[e_k] \rangle}{C} = \sum_{i=1}^{C} \langle e_k | f_i^* \rangle^2 . \tag{2.3}$$

The quality, analogous to Eq. (1.30) in Chapter 1, measures how much e_k overlaps with the C dimensional target function space $\mathcal{H}^*$. The cumulative measure is defined likewise

$$\Pi^*(k) := \sum_{j=1}^{k} Q_k^* = \sum_{j=1}^{k}\sum_{i=1}^{C} \langle e_j | f_i^* \rangle^2 . \tag{2.4}$$

The cumulative quality ranges between $0 \leq \Pi^*(k) \leq 1$ and measures how much the first k features can express the target function. Similar quality measures have been studied in the past to quantify feature learning. See, for exmaple, [51, 52, 16, 17].

utilization Analogously, we define the utilization for the DNN's learned function $\hat{f} : \chi \to \mathbb{R}^C$. The learned function space is given by $\hat{\mathcal{H}} = span\{\hat{f}_1, \cdots, \hat{f}_C\}$ and a projection operator $P_{\hat{\mathcal{H}}}$ onto the learned function space is defined analogous to Eq. (2.3). Then **utilization** $\hat{Q}_k$ and **cumulative utilization** are defined as:

$$\hat{Q}_k = \frac{\langle e_k | P_{\hat{\mathcal{H}}}[e_k] \rangle^2}{\dim(\hat{\mathcal{H}})}, \qquad \hat{\Pi}(k) = \sum_{j=1}^{k} \hat{Q}_k. \tag{2.5}$$

The cumulative utilization also ranges between $0 \leq \hat{\Pi}(k) \leq 1$ with $\hat{\Pi}(p) = 1$. It measures the amount of the learned function expressed by the first k features.

Intensity and number of significant features We can reuse the definition of intensity in Eq. (1.29). However, we adjust the effective dimension on intensity (Eq. (1.16)) as

$$D_{eff}(\rho) := 1 + D_{eff}([\rho_2, \cdots, \rho_p]). \tag{2.6}$$

The first intensity is treated separately because we empirically observe that the first feature is a constant function. Since constant functions are not of interest and can be addressed by the bias term in the final layer, we omit the contribution from its value and add a dimension instead. See Section 2.5.1 for further discussion.

metric	initialization	after training
$D_{eff}(Q^*)$	3.2	1.1
$D_{eff}(\hat{Q})$	1.8	1.0
$D_{eff}(\rho)$	1.6	1.2

Table 2.1: Summary of change of effective dimensions for Fig. 2.2

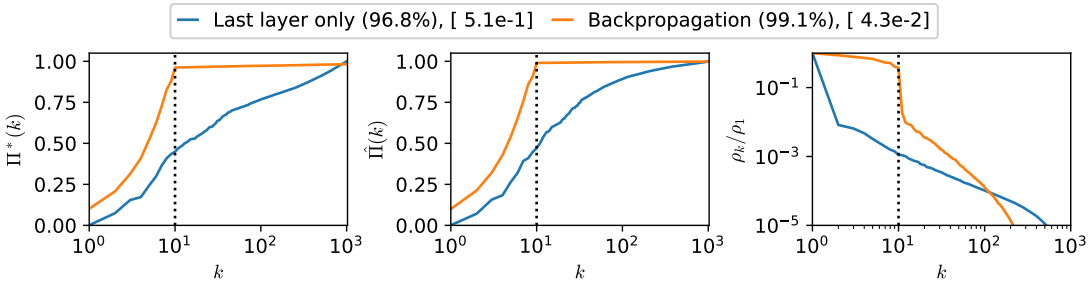
Effective number of useful/used features We can also measure the effective dimension (Eq. (1.16)) of the quality vector $Q^* = [Q_1^*, \cdots, Q_p^*]$ and the utilization vector $\hat{Q} = [\hat{Q}_1, \cdots, \hat{Q}_p]$ to measure the number of features that significantly contribute in describing the target/learned function. For example, the effective dimension measures summarize the change in the number of useful/used/significant features in Fig. 2.2 (Table 2.1).[1]

Approxmiation of the metric A feature map is well-defined for a DNN and we can use the methods described in Section 1.5.4 to approximate all metrics. We use the large training set to approximate the eigenfunctions and eigenvalues and the test set to approximate the inner products. As the limit of the approximation, cumulative projection measures such as $\Pi^*(k)$ suffer from finite size errors when we approximate k larger than the test set size. However, we often work in the limit wher the width of the last layer is much smaller than the number of test samples $p \ll n_{test}$, and typically only the first few eigenfunctions dominate our measures. Nevertheless, due to these finite size effects, we exercise caution and present plots only up to $k \sim \mathcal{O}(10^3) \ll n_{test}$.

2.3.1 The minimum feature (MF) regime and the extended feature (EF) regime

In Fig. 2.3, we train a CNN on MNIST with two dynamics: a linear dynamics with last-layer-only training and a feature learning dynamics with full backpropagation. The two dynamics both achieve above 95% test accuracy after training, but the features differ. Full backpropagation dynamics used only the first 10 features

[1] Note that this is the only case where we do not make exceptions regarding the first eigenvalue when calculating $D_{eff}(\rho)$.



(a) Target projection (b) Self projection (c) Eigenvalue

Figure 2.3: EF and MF regime from different dynamics A CNN with width $p = 1024$ is trained on the full MNIST dataset using two dynamics: last-layer-only training (EF regime) and default backpropagation (MF regime). Parentheses show test accuracy and square brackets indicate the CKA-based scalar measure κ_{CKA} from Definition 1. Cumulative quality (Eq. (2.4)) is shown in (a), cumulative utilization (Eq. (2.5)) in (b), and intensity (eigenvalues) in (c). Although both dynamics achieve test accuracy above 95%, two dynamics lead to dramatically different features: the MF regime concentrates the metrics in the first 10 features, while the EF regime spreads these across all 1024 features.

(Fig. 2.3(b)) that are of high quality (Fig. 2.3(a)) and high-intensity (Fig. 2.3(c)). The last-layer-only training used 10^3 features with poorer qualities and slower decaying intensities.[2]

We can define two regimes to distinguish the dynamics: the **minimum features (MF) regime** and an **extended feature (EF) regime**. For the C-way balanced classification task, the MF regime is when only the first C features are significant and are used by the model, leading to

$$D_{eff}(\hat{Q}) \approx C \qquad D_{eff}(\rho) \approx C. \tag{2.7}$$

Indeed we observe $D_{eff}(\hat{Q}) = 10.5$ and $D_{eff}(\rho) = 11.19$ for the backpropagated model in Fig. 2.3.

The EF regime covers all dynamics when the number of significant or used features deviates from C. The last-layer-only dynamics is an example with $D_{eff}(\hat{Q}) = 121$ and $D_{eff}(\rho) = 119$. To quantify a scalar measure for the MF regime, we first define an ideal operator $\mathcal{T}_{MP}$.

[2]The decay of intensities should be analyzed excluding the first intensity, as the first feature is a constant function.

Definition 1 (Minimum Projection (MP) operator)**.** *For a DNN on a balanced C-way classification task, the integral operator $\mathcal{T}$ is an MP-operator $\mathcal{T}_{MP}$ if it has two nontrivial eigenspaces, one being the span of a constant function $\mathbf{1}$ and the other orthogonal complement of $\mathbf{1}$ inside $\hat{\mathcal{H}}$, where $\hat{\mathcal{H}}$ is C-dimensional function space spanned by entries of the learned function $\hat{f} : \mathcal{X} \to \mathbb{R}^C$ expressed by the DNN. This is equivalent to that $\mathcal{T}_{MP}$ is given as*

$$\mathcal{T}_{MP}[u] = a_1 \langle \mathbf{1}, u \rangle \mathbf{1} + a_2 P_{\hat{\mathcal{H}}}(u) \quad \textit{for all } u \tag{2.8}$$

where a_1, a_2 are positive scalars, and $P_{\hat{\mathcal{H}}}$ denotes the orthogonal projection onto $\hat{\mathcal{H}}$.

Ignoring the constant function and setting $a_1 = 0$, whose discussion is deferred to Section 2.5.1, the MP-operator is a projection operator $P_{\hat{\mathcal{H}}}$ onto the learned function space $\hat{\mathcal{H}}$. The projection operator has C equal eigenvalues ($D_{eff}(\rho) = 10$) and C (degenerate) eigenfunctions ($D_{eff}(\hat{Q}) = 10$).[3]

We then use centered kernel alignment (CKA) [53] – used for comparing features of DNNs [53] and also for studying the evolution of the NTK [54, 55] – to measure the similarity between $\mathcal{T}$ of a DNN is to an ideal operator $\mathcal{T}_{MP}$:

$$CKA(\mathcal{T}, \mathcal{T}_{MP}) = \frac{\mathrm{Tr}(c(\mathcal{T})c(\mathcal{T}_{MP}))}{\|c(\mathcal{T})\|_F \|c(\mathcal{T}_{MP})\|_F}, \tag{2.9}$$

where $c(A)$ is centering operator $(I - |\mathbf{1}\rangle \langle \mathbf{1}|)A(I - |\mathbf{1}\rangle \langle \mathbf{1}|)$, I is identity and $|\mathbf{1}\rangle$ is the constant function 1 in $L^2(\mathcal{X})$, and the norm $\|\cdot\|_F$ is Frobenius norm. The similarity measure Eq. (2.9) classifies the two regimes:

Definition 2 (CKA Minimum Feature Regime Measure κ_{CKA})**.** *For a distribution q and a class-balanced learned function $\hat{f}$, a DNN is in* ***Minimum Feature (MF) regime*** *if $\kappa_{CKA} = 1 - \mathrm{CKA}(\mathcal{T}, \mathcal{T}_{MP}) < \epsilon$, where $\mathcal{T}$ is the feature kernel (operator) of a DNN and $\mathcal{T}_{MP}$ is the MP-operator.*

Although there is some arbitrariness with the choice of ϵ, we find that $\epsilon = 0.1$ works well in practice (square brackets in Fig. 2.3).

[3] We assumed that $[\hat{f}_1, \cdots, \hat{f}_C]$ are C linearly independent, which is typically true for DNN trained on a balanced dataset.

model	dataset	$D_{eff}(\rho)$	$D_{eff}(\hat{Q})$	CKA measure	test accuracy (%)
ResNet18	MNIST	10.1	10.0	4.4×10^{-3}	99.7
VGG16	MNIST	10.0	10.0	1.8×10^{-2}	99.4
ResNet18	CIFAR10	10.4	10.0	3.1×10^{-3}	94.8
VGG16	CIFAR10	10.0	10.0	1.1×10^{-3}	93.3
ResNet18	CIFAR100	99.9	99.7	2.5×10^{-2}	78.3
VGG16	CIFAR100	95.8	99.4	7.4×10^{-2}	71.9

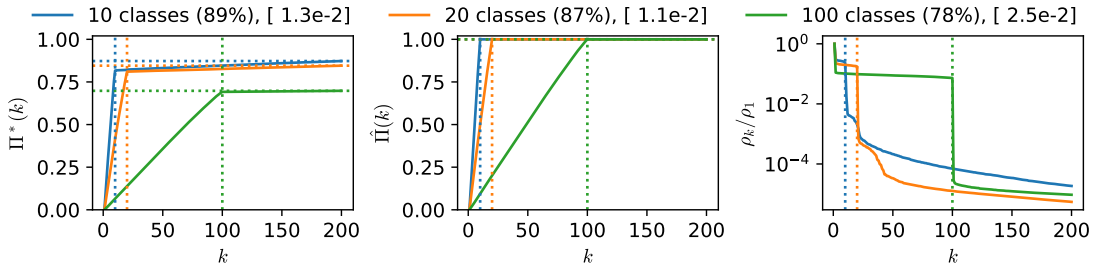
Table 2.2: metrics of ResNet18 and VGG16 trained on image datasets All generalizing models in the table are in the MF regime ($\kappa_{CKA} < 0.1$). The models all use only C features ($(D_{eff}(\hat{Q}) \approx C)$, with only those features being significant ($(D_{eff}(\rho) \approx C)$.

2.4 Experiments

Here, we show empirical results of generalizing DNNs in the MF regime. We trained VGG16 and ResNet18 on MNIST, CIFAR10, and CIFAR100 datasets. All models are trained with SGD with momentum 0.9 and weight decay of 10^{-3}. The learning rate is 0.05 for ResNet18 and 0.01 for VGG16. Models were trained for 200 epochs for MNIST and CIFAR10 with learning rate scheduling that decays by a factor of 0.2 every 60 epochs. When trained on CIFAR100, the models were trained for 600 epochs without learning rate scheduling. CIFAR10 and CIFAR100 used the standard data augmentation (random crops and horizontal flips), and MNIST was padded to the size of CIFAR10 images.

In Fig. 2.4, we trained ResNet18 on CIFAR100 with only 10, 20, and 100 classes (C). All models are in the MF regime (Section 2.3.1) where only the first C features are utilized (Fig. 2.4(b)) and only the first C eigenvalues are significant (Fig. 2.4(c)). The criteria in Definition 1 is also met for $\epsilon = 0.1$ (in square brackets). In addition to the MF regime, we note that only the first C features are significantly useful for describing the target function (Fig. 2.4(a)).

In Table 2.2, we present the metrics for VGG16 and ResNet18 trained on MNIST, CIFAR10, and CIFAR100 datasets. For all studied cases, we confirm that all models are in a clear MF regime from reading the CKA measures or the effective dimensions of the intensity and utilization.



(a) Target projection (b) Self projection (c) Eigenvalue

Figure 2.4: Minimum feature regime for different number of classes ResNet18 is trained on 10, 20, and 100 classes from CIFAR100 for 600 epochs (until convergence). (a) All models show that only the first C features are significantly useful in describing the target function (i.e. $\Pi^*(C) \approx \Pi^*(p)$) where $\Pi^*(p)$ is plotted as dotted horizontal line. (b) All models are in the MF regime with only C features being used to express the learned function (i.e. $\hat{\Pi}(C) \approx 1$). (c) All three cases show typical MF regime intensities (Eq. (2.7)) where only the first C intensities are significant and similar with the first being somewhat larger.

2.5 The relationship between the MP-operator and neural collapse

Neural collapse [56] (NC) refers to a state of a DNN when the feature vectors of the training set and last layer weights form a symmetric and clustered structure at the terminal phase of training (TPT) or when trained past the point where the training error vanishes. The reason for the structure has been studied mainly in the Unconstrained Feature Model (UFM) [57, 58] and has sparked theoretical studies (see e.g. [59] for a review).

NC is defined by the emergence of four interconnected phenomena upon TPT: NC1) collapse of within-class variability, NC2) convergence of features to a rigid simplex equiangular tight frame (ETF) structure, NC3) alignment of the last layer and features, and NC4) simplified decision by nearest class. We provide more formal measures of these phenomena in Appendix D.

In this section, we show how MP-operator $\mathcal{T}_{MP}$ (Definition 1) – a state of the feature **functions** – relates to NC – a state of feature **vector samples** – similar in spirit to that taken in [60]. A key observation is that the first eigenfunction of $\mathcal{T}_{MP}$ is a constant function, and we first discuss its implication for NC. Subsequently,

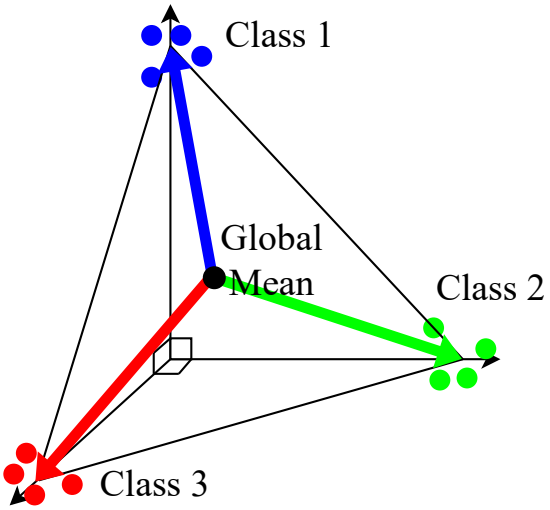


Figure 2.5: Illustration of neural collapse. In NC, the last layer feature vectors (the post-activation of the penultimate layer), illustrated as colored dots, cluster by their class mean vector, illustrated as colored arrow, and form a simplex ETF structure (orthogonal vectors projected at the compliment of the global mean vector).

we present propositions demonstrating that substituting the true data distribution q with the empirical distribution results in the NC conditions.

2.5.1 The first eigenfunction of the MP-operator is a constant function

MP-operators $\mathcal{T}_{MP}$ (Definition 1) include a constant function in addition to a more conventional projection operator onto $\hat{\mathcal{H}}$, the C-dimensional function space spanned by entries of the learned function $\hat{f} : \mathcal{X} \to \mathbb{R}^C$. This difference with simple projection operators means that an MP-operator uniformly rescales all function components in $\hat{\mathcal{H}}$ in an isotropic manner, except the direction of the constant function. It is not surprising that $\hat{\mathcal{H}}$ includes the constant function because a non-zero bias term in the last layer can necessitate it.

However, it is surprising that the constant function is the first eigenfunction of $\mathcal{T}_{MP}$ with the largest eigenvalue. Interestingly, this function is often disregarded in other analyses, for example in principal component analysis (PCA) and in the CKA. We also observe that the first eigenfunction of $\mathcal{T}$, not limited to its limit $\mathcal{T}_{MP}$, often closely resembles the constant function. Across all our experiments, except for the 1-D toy model in Fig. 2.2, we consistently observe $\langle e_1 | \mathbf{1} \rangle > 0.95$. Due

to this observation, we handled ρ_1 as an exception, for example when calculating $D_{eff}(\rho)$ or defining the MF regime.

Analogous to the constant function in $\mathcal{T}_{MP}$, a constant vector in NC explains why the feature vectors form a simplex equiangular tight frame (ETF), instead of a set of orthonormal basis vectors: A $(C-1)$-dimensional simplex ETF can be obtained as the projection of the standard basis onto the orthogonal complement of $[\frac{1}{C}, \frac{1}{C}, \cdots, \frac{1}{C}]$, which is the average vector of all basis vectors in $\mathbb{R}^C$. For example, projecting $[0,0,1]$, $[0,1,0]$, and $[1,0,0]$ onto the orthogonal complement of $[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$ gives three vertices $[-\frac{1}{3}, -\frac{1}{3}, \frac{2}{3}]$, $[-\frac{1}{3}, \frac{2}{3}, -\frac{1}{3}]$, $[\frac{2}{3}, -\frac{1}{3}, -\frac{2}{3}]$ which forms an (ordinary plane) equilateral triangle. We will show in Eq. (2.14) that a constant function maps to a global average vector of features and transforms an orthogonal structure into a simplex ETF structure.

2.5.2 The MP-operator implies NC

The analogy between $\mathcal{T}_{MP}$ and NC is clear: $\mathcal{T}_{MP}$ is an isotropic projection operator on the learned function space $\hat{\mathcal{H}}$ except on the direction of the constant function and NC is a symmetric clustering of the feature vectors in an orthogonal basis except along the constant vector.

If we define $\mathcal{T}_{MP}$ on the empirical distribution q' of the training set with $\hat{f}$ as the empirical output function of the balanced C-way classification task, we can show that $\mathcal{T}_{MP}$ implies NC1 and NC2 conditions (following two propositions). By the Theorem 2 of [56], NC3 and NC4 follow from NC1 and NC2.

Proposition 2.5.1. *Let $\mathcal{T}_{MP}$ be MP-operator, the inputs x are taken from q' the empirical distribution of the training set, and $\hat{f}$ the empirical output function of balanced C-way classification task. The following NC1 condition holds:*

$$Tr\left(\left(\sum_i \sum_{x\in A_i} (h_i(x)-\mu_i)(h_i(x)-\mu_i)^T\right)\left(\sum_j (\mu_j-\mu_g)(\mu_j-\mu_g)^T\right)^T\right) = 0 \quad (2.10)$$

Proposition 2.5.2. *Let $\mathcal{T}_{MP}$ be the MP-operator on data taken from the empirical distribution of the training set, and $\hat{f}$ the empirical output function of balanced C-way classification task; The following NC2 condition holds:*

$$(\mu_i - \mu_g)^T(\mu_j - \mu_g) \propto \delta_{ij} - \frac{1}{C} \tag{2.11}$$

In the two propositions above, the class feature map $h_i : \mathcal{X} \to \mathbb{R}^p$ is a feature map to the last layer features for given class i, and class mean vector $\mu_i \in \mathbb{R}^p$ is the expectation of $h_i(x)$ over a class partition A_i. The global mean vector $\mu_g \in \mathbb{R}^p$ is the average of all class means.

NC1 condition means the covariance within the class is orthogonal to the covariance among class means and NC2 that class mean vectors form a simplex ETF.

To prove these propositions, which are shown in Appendix D, we express class feature vector $h_i(x)$, class mean vector μ_i, and global mean vector μ_g in terms of feature map Φ, class indicator function $\hat{f}_i(x) = \mathbf{1}_{A_i}(x)$, and constant function $\mathbf{1}$.

$$h_i(x) = \hat{f}_i(x)\Phi(x), \tag{2.12}$$

$$\mu_i = \frac{1}{A_i}\int_{\mathcal{X}} h_i(x)q'(x)dx = C\int_{\mathcal{X}} \hat{f}_i(x)\Phi(x)q'(x)dx, \tag{2.13}$$

$$\mu_g = \frac{1}{C}\sum_i^C \mu_i = \int_{\mathcal{X}} \Phi(x)q'(x)dx. \tag{2.14}$$

Using the equations above, the expectation of the inner product between feature/mean vectors can be expressed by $\hat{f}_i$ and $\mathcal{T}$. For example,

$$\mu_i^T\mu_j = C^2\left(\int_{\mathcal{X}} \hat{f}_i(x)\Phi(x)q'(x)dx\right)^T \int_{\mathcal{X}} \hat{f}_j(x')\Phi(x')q(x')dx' \tag{2.15}$$

$$= C^2\int_{\mathcal{X}} \hat{f}_i(x)q'(x)\int_{\mathcal{X}} \Phi(x)^T\Phi(x')\hat{f}_j(x')q(x')dx' \tag{2.16}$$

$$= C^2\int_{\mathcal{X}} \hat{f}_i(x)q'(x)\mathcal{T}[\hat{f}_j](x)dx \tag{2.17}$$

$$= \langle \hat{f}_i|\mathcal{T}[\hat{f}_j]\rangle \tag{2.18}$$

where we used the definition of the integral operator $\mathcal{T}$ (Eq. (1.13)) in the third line.

Finally, we can show Propositions 2.5.1 and 2.5.2 from the following technical proposition, which is derived from the definition of $\mathcal{T}_{MP}$ and the class indicator functions.

Proposition 2.5.3. *If $\mathcal{T}_{MP}$ is MP-operator and $\hat{f}_i(x) = \mathbf{1}_{A_i}(x)$ for a balanced partition $\{A_1, \cdots, A_c\}$ of χ, then*

$$\langle C\hat{f}_i | \mathcal{T}_{MP}[C\hat{f}_j - \mathbf{1}] \rangle = a_2(\delta_{ij} - C^{-1}) \qquad \langle \mathbf{1} | \mathcal{T}_{MP}[C\hat{f}_j - \mathbf{1}] \rangle = 0, \tag{2.19}$$

where C is the number of classes.

Proof. See Appendix D □

2.5.3 Extending beyond NC

Even though NC and the MP-operator are closely related, NC is restricted to the training set, while our framework provides information about the properties of the functions that are defined beyond the training set. Because MP-operator conditions lead to NC conditions, our conditions are stricter. The notable advantage of the function/random variable framework is that we can measure the quality of each feature, and make a natural connection to the generalization loss.

2.6 Summary

In this chapter, we extended the feature metrics of the linear model to the last layer features of DNNs, revealing a bias toward a minimal number of features with significant quality, utilization, and intensity. We categorized dynamics that concentrate utilization and intensity in the first C features (number of classes) as the MF regime, with all other dynamics falling under the EF regime. We proposed an ideal MP-operator to measure the tightness of the MF regime and showed its connection to neural collapse (NC).

3

Dynamics of layerwise linear models

Contents

3.1 Introduction

In the last chapter, we empirically observed DNNs' inductive bias toward a minimum number of features. In an ideal world, we would solve the dynamics of DNN exactly, but the non-linearity (e.g., ReLU) makes the analysis challenging. As physicists often do, we instead focus on simpler layerwise models without non-linearities to study the dynamics of DNNs. The dynamics of layerwise linear models are often

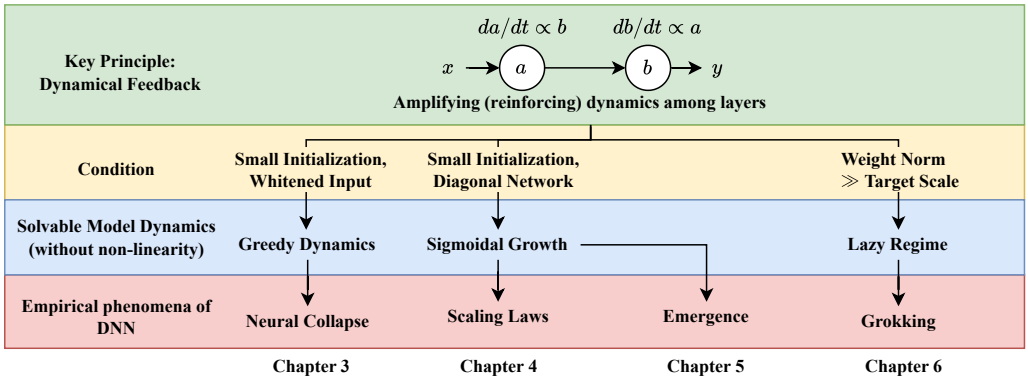


Figure 3.1: Outline of next 4 chapters For a systematic presentation of various works on layerwise linear models, we begin by building intuition on how the key principle (green) behaves under a condition (yellow). We then formalize this intuition as a key property (blue) using a solvable layerwise linear model. Finally, we discuss how this property from the layerwise linear model extends to describe an empirical phenomenon in DNNs (red).

exactly solvable [61] and can explain various dynamical phenomena of DNNs such as emergence [5] observed in large language models [62], neural collapse [57] observed in image classification tasks [56], lazy/rich regimes [63] observed in extreme limits [15, 64], and grokking [65] observed in algorithmic tasks [66]. These topics will be discussed in the following chapters as summarized in Fig. 3.1.

In this chapter, we propose the dynamical feedback principle or how the layers amplify the dynamics under gradient descent. Using the principle, we highlight how the dynamics of layerwise linear models (amplifying dynamics) differs from (single layer) linear models and how it leads to stage-like training where more important features are learned in sequences. We conclude by showing that a linear neural network (a member of layerwise linear models) or unconstrained feature model can explain neural collapse observed in the previous chapter.

3.2 Setup

In this chapter, unless explicitly stated, we assume the input features $x \in \mathbb{R}^d$ to be d dimensional independent zero-mean Gaussian random variable. We only consider gradient flow (GF) and assume that the training dataset follows a certain probability distribution for analytic simplicity.

Target function We assume a realizable target function f^* with *target scales* $S_i \in \mathbb{R}$ for $i = 1, \ldots, d$:

$$f^*(x) = \sum_{i=1}^{d} x_i S_i. \tag{3.1}$$

Loss We consider the mean square error (MSE) loss

$$\mathcal{L} := \frac{1}{2}\mathbf{E}\left[(f^*(x) - f(x))^2\right]. \tag{3.2}$$

Models We define *layerwise linear models* as 2-layer models whose outputs are multilinear with respect to any layers of parameters. Examples include the variants of linear neural networks, illustrated in Fig. 3.2:

$$f_{a,b}^{(diag)}(x) = \sum_{i=1}^{d} x_i a_i b_i, \quad a, b \in \mathbb{R}^d, \tag{3.3}$$

$$f_{W_1,W_2}^{(lnn)}(x) = x^T W_1 W_2, \quad W_1 \in \mathbb{R}^{d\times p}, W_2 \in \mathbb{R}^{p\times c}. \tag{3.4}$$

A linear model with no hidden layer is used as a base model for comparison:

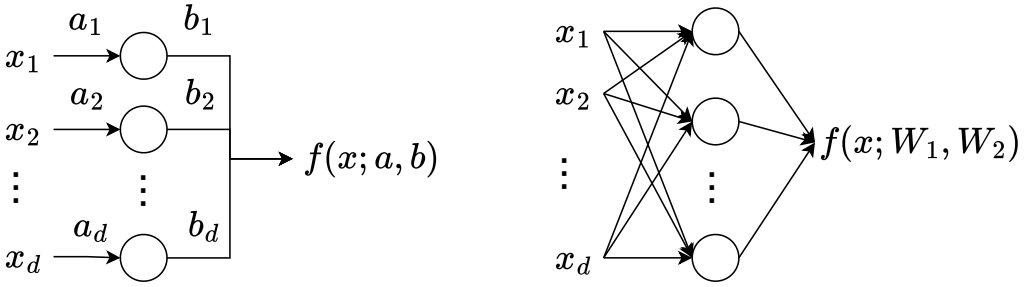
$$f_\theta^{(lin)}(x) = x^T\theta, \quad \theta \in \mathbb{R}^d. \tag{3.5}$$

Like linear models, layerwise linear models do not require linearity with the input features x_i. For example, $f_{a,b}(x) = \sum_{i=1}^{d} \phi_i(x) a_i b_i$ with a non-linear function $\phi : \mathcal{X} \to \mathbb{R}$ are also a layerwise linear model due to the parameters' multilinearity to the output. Although the arguments in this chapter holds for more general feature map ϕ, we use $\phi_i(x) = x_i$ for demonstrative purposes.

3.3 The dynamical feedback principle

The dynamical feedback principle arises from the gradient descent dynamics under layerwise structure and describes how the magnitude of one layer mutually scales the rate of change in other layers (Fig. 3.1). For example, the GF equation of diagonal linear neural network (Eq. (3.3)) is

$$\frac{da_i}{dt} = -b_i\mathbf{E}[x_i^2](a_i b_i - S_i), \quad \frac{db_i}{dt} = -a_i\mathbf{E}[x_i^2](a_i b_i - S_i). \tag{3.6}$$



(a) Diagonal linear neural network (b) Linear neural network

Figure 3.2: Layerwise linear models. Layerwise linear models include, but are not limited to, **(a):** diagonal linear neural networks (Eq. (3.3)) and **(b):** linear neural networks (Eq. (3.4)). The layerwise structure leads to distinct dynamics compared to linear models (Eq. (3.5)).

The product of parameters (i.e., layerwise structure) creates an *amplifying dynamics*: the magnitude of a_i governs the rate of change for b_i, and vice versa, yielding a **dynamical feedback**. To highlight its significance, we compare the gradient equation to that of the linear model (Eq. (3.4)):

$$\frac{d\theta_i}{dt} = -\mathbf{E}[x_i^2](\theta_i - S_i). \tag{3.7}$$

In linear models, parameters evolve depending only on their distances to respective target scales $(\theta_i - S_i)$, lacking the feedback with other parameters. The feedback in Eq. (3.6) generates sigmoidal growth and greedy dynamics distinct from linear models, which we will explore in the following sections.

Conservation of magnitude difference Because of the commutativity of products in Eq. (3.3), we identify a conserved quantity in Eq. (3.6) as

$$a_i \frac{d}{dt} a_i - b_i \frac{d}{dt} b_i = 0 \quad \Rightarrow \quad a_i^2 - b_i^2 = \mathcal{C}_i, \tag{3.8}$$

where $\mathcal{C}_i$ is determined at initialization. We can extend the argument for linear neural networks to show that $W_2 W_2^T - W_1^T W_1$ is conserved (Appendix A.2.2).

3.4 Layerwise dynamics and stage-like training

In Eq. (3.6), when $a_i \approx b_i$, the dynamical feedback exhibits rich-get-richer (poor-get-poorer) property: a larger (smaller) $|a_i|$ results in a faster (slower) evolution

of b_i, which amplifies (attenuates) in positive (negative) feedback. In this section, we formalize this property as *sigmoidal growth* (delayed saturation) of $a_i b_i$ in diagonal linear neural networks.

3.4.1 Sigmoidal growth under small initialization

We consider dynamics of diagonal linear neural networks (Eq. (3.6)) with small and equal initialization; $a_i(0) = b_i(0)$ and $0 < a_i(0) \ll 1$. The dynamics decouples into d independent *modes*, where each mode follows a sigmoidal saturation:

$$a_i(t)b_i(t)/S_i = \frac{1}{1 + \left(\frac{S_i}{a_i(0)b_i(0)} - 1\right) e^{-2S_i \mathbf{E}[x_i^2]t}}. \tag{3.9}$$

This contrasts to the exponential saturation of modes in the linear model with $\theta_i(0) = 0$ (Eq. (3.7)):

$$\theta_i(t)/S_i = 1 - e^{-\mathbf{E}[x_i^2]t}. \tag{3.10}$$

Note the connection between training of eigenfunctions in Chapter 1 and the modes of learning in Eq. (3.7): each feature, up to scaling constant, x_i is an eigenmode because we assumed independence between input features ($\mathbf{E}[x_i x_j] = 0$). See Appendix A for the derivation of both models. For a general derivation when $a_i \neq b_i$, see Saxe et al. ([61]).

Likewise, we can extend the defintion of features in Chapter 2 to check that each mode $a_i b_i x_i$ in Eq. (3.9) has one-to-one correspondence with the eigenfunction of integral operator $\mathcal{T}$ for the diagonal linear neural network:

$$\mathcal{T}[f](x') = \sum_i^d \mathbf{E}[a_i^2 x_i f(x)]x_i', \quad \mathcal{T}[x_i] = \mathbf{E}[x_i^2]a_i^2 x_i'. \tag{3.11}$$

Coming back to the dynamics of the modes, Fig. 3.3 demonstrates the difference between the dynamics of linear and layerwise linear models. While a mode in linear models saturates immediately after initialization (Fig. 3.3(a)), that of diagonal linear neural network saturates **after a delay**. The small value of $a_i b_i$ delays the saturation as small a_i downscales the gradient of b_i and vice versa (feedback principle). After sufficient evolution however, the feedback principle abruptly saturates $a_i b_i$ through reinforcement between layers (Fig. 3.3(b)).

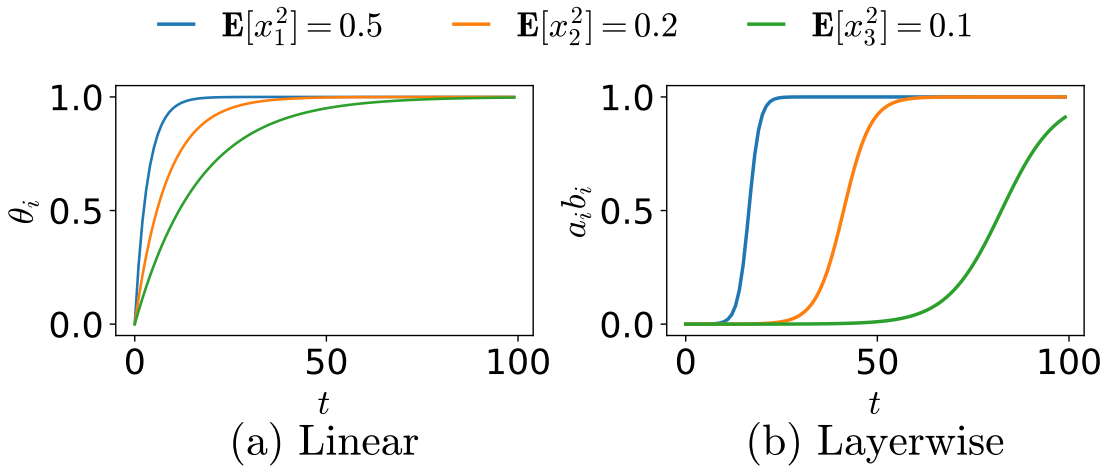


Figure 3.3: Dynamics of the linear model and the diagonal linear neural network. The colored lines show the saturation curves of modes with different variances for **(a):** the linear model (Eq. (3.10)) and **(b):** the diagonal linear neural network (Eq. (3.9)) with $S_i = 1$. For the linear model, all θ_i's saturate from $t = 0$ only differing in the saturation speed. For the layerwise model, $a_i b_i$'s show delayed saturations depending on $\mathbf{E}[x_i^2]$, learning the modes in sequences.

3.4.2 Stage-like training

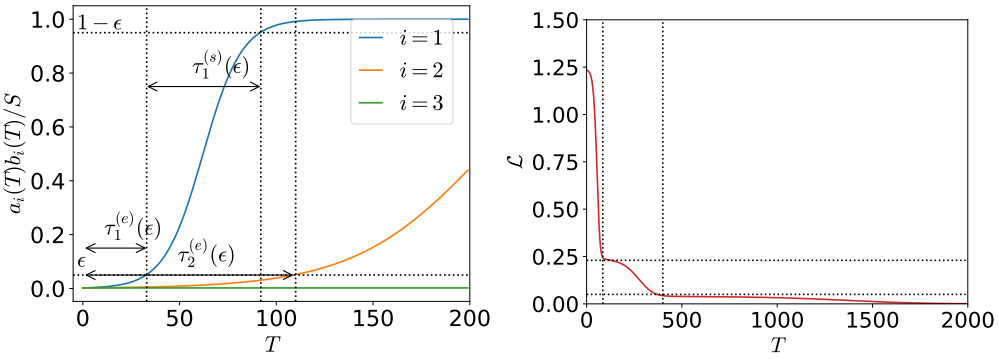
A consequence of sigmoidal dynamics is the stage-like training where modes of different intensity ($\mathbf{E}[x_i^2]$) saturate sequentially. In Fig. 3.3(b), the first mode (blue) saturates fully while the next mode remains negligible, contrasting with the concurrent saturation of modes in linear models (Fig. 3.3(a)). When saturation time scales ($1/(S_i \mathbf{E}[x_i^2])$) differ sufficiently, the sigmoidal dynamics (Eq. (3.9)) in layerwise models delays slower modes long enough for faster modes to fully saturate, leading to the sequential learning of modes or the stage-like training.

To quantify stage-like training, we define two intervals for each mode (Fig. 3.4(a)):

- The emergent time $\tau_i^{(e)}(\epsilon)$: the time for $a_i b_i / S_i$ to reach ϵ;
- The saturation time $\tau_i^{(s)}(\epsilon)$: the time for $a_i b_i / S_i$ to saturate from ϵ to $1 - \epsilon$.

Using the dynamics equation (Eq. (3.9)) for target function Eq. (3.1) with $S_i = S$ for all i, the emergent time and saturation time of the i^{th} mode becomes

$$\tau_i^{(e)}(\epsilon) = \frac{1}{2\mathbf{E}[x_i^2]S} \ln\left(\frac{\frac{S}{a_i(0)b_i(0)} - 1}{\frac{1}{\epsilon} - 1}\right), \qquad \tau_i^{(s)}(\epsilon) = \frac{1}{\mathbf{E}[x_i^2]S} \ln\left(\frac{1}{\epsilon} - 1\right). \tag{3.12}$$



(a) Emergent and saturation time **(b)** Saddle-to-saddle

Figure 3.4: Stage-like training. The multilinear model is trained on the multitask sparse parity problem with $\alpha = 0.6$ and $S = 5$. **(a):** Mode strength of the model as a function of time. The emergent time $\tau_i^{(e)}(\epsilon)$ is the time required for the k^{th} mode to reach $a_i b_i/S = \epsilon$. The saturation time $\tau_i^{(s)}(\epsilon)$ is the time required for $a_i b_i/S$ to saturate from ϵ to $1-\epsilon$. The model shows stage-like training if the emergent time interval $\tau_{k+1}^{(e)}(\epsilon) - \tau_k^{(e)}(\epsilon)$ is larger than the saturation time $\tau_i^{(s)}(\epsilon)$ for sufficiently small ϵ (0.05 in the figure). **(b):** The loss as a function of time for the same system as (a). For stage-like training, the change in the loss for the k^{th} emergence is $\mathbf{E}[x_i^2]\mathcal{L}_i + \mathcal{O}(\epsilon)$ and the interval for the next emergence is $\Delta\tau^{(e)}(\epsilon) = \tau_{k+1}^{(e)}(\epsilon) - \tau_k^{(e)}(\epsilon)$.

We have stage-like training with

$$\epsilon = \frac{1}{1 + \left(\frac{S}{a_i(0)b_i(0)} - 1\right)^{\frac{\mathbf{E}[x_{i+1}^2] - \mathbf{E}[x_i^2]}{\mathbf{E}[x_{i+1}^2] + \mathbf{E}[x_i^2]}}}. \tag{3.13}$$

For sufficiently **small initialization** ($a_i(0)b_i(0) \ll S$), we get a **stage-like** training with minimal overlap (infinitesimal ϵ):

$$\tau_i^{(s)}(\epsilon) < \tau_{i+1}^{(e)}(\epsilon) - \tau_i^{(e)}(\epsilon), \qquad \epsilon \ll 1, \tag{3.14}$$

where the model finishes learning (saturating) the i^{th} mode before starting to learn (emerging) the next mode.

3.4.3 Application 1: Saddle-to-saddle learning

Saddle-to-saddle learning refers to sudden drops and plateaus of the loss during training [67, 68, 69]. We can describe the DNN phenomenon with the stage-like training of diagonal linear neural network with varying $S_i\mathbf{E}[x_i^2]$. A drop maps to the abrupt saturation of a feature, while a plateau to the delay before the saturation of the next feature.

3.5 From greedy (low-rank) dynamics toward salient features to neural collapse

So far, we demonstrated that sigmoidal saturation (Eq. (3.9)) in diagonal linear neural networks, compared to exponential saturation in linear models, significnatly prioritizes learning features that are more correlated (with larger $S_i\mathbf{E}[x_i^2] = \mathbf{E}[x_i y_i]$). In this section, by building on Saxe et al. ([61]), we show how the preference toward correlated features leads to a *low-rank bias* **(greedy dynamics)** in linear neural networks. We then draw on Mixon et al. ([57]) to explain neural collapse observed in the last chapter.

3.5.1 Greedy dynamics

The dynamics of linear neural networks is challenging to solve in general. Assuming small initialization and whitened input ($\mathbf{E}[xx^T] = I$), Saxe et al. ([61]) and following works [70, 71, 72, 73, 74] solved the exact dynamics of linear neural networks. The dynamics **decouples** into c independent modes similar to that of diagonal linear neural network (Appendix A):

$$\frac{\alpha_i^2(t)}{\rho_i} = \frac{1}{1 + \left(\frac{\rho_i}{\alpha_i^2(0)} - 1\right) e^{-2\rho_i t}}, \tag{3.15}$$

where $\alpha_i := u_i^T W_1 W_2 v_i$, and u_i, ρ_i, v_i are the i^{th} left singular vector, singular value, and the right singular vector of the correlation matrix $\mathbf{E}\left[x f^*(x)^T\right] = UPV$, respectively.

Note the similarity between Eq. (3.9) and Eq. (3.15), where $a_i b_i$ and S_i in Eq. (3.9) are replaced by α_i^2 and ρ_i in Eq. (3.15), respectively. Linear neural networks automatically **identify** the linear combination of input features **most correlated** (largest ρ_i) with the target (salient features). As discussed in stage-like training, modes are learned sequentially based on the order of ρ_i or their saliency. This bias toward salient features introduces a preference for minimum rank, as only target-relevant c (the dimension of the output) modes are trained, even though

W_1 can reach a rank of $\min(d, p)$. We will refer to the tendency toward learning the most target-relevant features first as the **greedy dynamics**.

3.5.2 Explaining neural collapse with linear neural network

To study the dynamics of NC, Mixon et al. [57] and similar works [75, 58] studied the unconstrained feature model (UFM). The UFM trains a product of two matrices where one matrix represents the features and the other matrix represents the last layer weights, a special case of linear neural networks. See also relevant works on matrix factorization [76, 77, 78].

Informal derivation of NC We show a non-rigorous derivation of NC and discuss why feature vectors of linear neural network XW_1, where $X \in \mathbb{R}^{n\times d}$ is the input feature matrix for n training datapoins, collapse into a rank c matrix. For rigorous theorems, see [57, 75]. First, we assume sufficiently small initialization such that the conserved quantity (Eq. (3.8)) is $W_1(0)^T W_1(0) - W_2(0)W_2(0)^T \approx 0$. After training,

$$XW_1(\infty)W_2(\infty) = Y, \tag{3.16}$$

where $Y \in \mathbb{R}^{n\times c}$ is the label matrix with one-hot vector as the labels, and $W_i(\infty)$ the trained matrices. Since Y is a rank c matrix, $W_2(\infty)$ is also rank c. The small initialization condition $W_1^T W_1 - W_2 W_2^T \approx 0$ ensures that W_1 and W_2 have the same rank, resulting in the feature matrix XW_1 having rank c instead of $\min(n, p)$. The matrix XW_1 collapses into c orthogonal directions, with each direction trivially mapping to a class to satisfy Eq. (3.16).

Intuition Loosely speaking, $W_1^T W_1 - W_2 W_2^T \approx 0$ is analogous to the small initialization in Saxe et al. ([61]).[1] From the dynamics perspective, the greedy dynamics of layerwise models (Eq. (3.15)) sends gradients toward only the relevant c directions (modes), effectively limiting the rank of the feature matrix (XW_1) to the minimal number.

[1] Large initialization can achieve $W_1^T W_1 - W_2 W_2^T \approx 0$ and similar dynamics; see [79, 63].

Practical DNNs are often initialized with small weights and are expressive enough to fit the training set without using all layers. Because of the layerwise structure, we can expect the greedy dynamics toward minimal rank to approximately hold even with non-linear activations, resulting in NC.

3.5.3 Empirical verification

Here, we verify whether the explanation of NC through layerwise linear model agrees with empirical findings. One common claim is that internal structure in the data creates such phenomenon [56]. Our theory suggests that the phenomenon is, under suitable assumptions, a byproduct of layerwise dynamics.

In Fig. 3.5, ResNet18 models trained on CIFAR10 with various levels of label noise, including completely **random labels**, still reached the MF regime, excluding that internal structure of the data plays a role and supporting our theory.

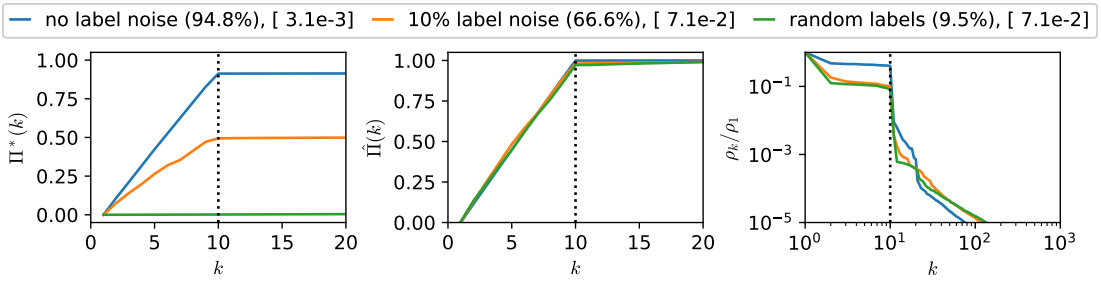
In Fig. 3.6, we train ResNet18 on the MNIST **regression** task, where the digit serves as the **scalar** target function $f^* : \mathcal{X} \to \mathbb{R}$. Unlike the C-way classification task, which uses C features, the regression model reaches the MF regime with just one non-constant feature.[2] This aligns with our theory, which predicts that the feature matrix rank is determined by the rank of the target function—here, one. If the data's internal structure mattered, the effective rank of $\mathcal{T}$ would be 10.

Figs. 3.5 and 3.6 suggest that DNNs' inductive bias for the MF regime is independent of the underlying input signal but depends more on the dimension of the target/learned function, in agreement with our theory. In Chapter 6, we further show more direct evidence where techniques shifting linear neural networks out of the MF regime also push practical DNNs into the EF regime.

3.6 Summary

In this chapter, we propose the dynamical feedback principle or sigmoidal dynamics where layers reinforce their learning. Under small initialization, this will lead to stage-like training and greedy dynamics that differs from the dynamics of limear models.

[2] It uses two, with the first being a constant function. See Section 2.5.1 for discussion.

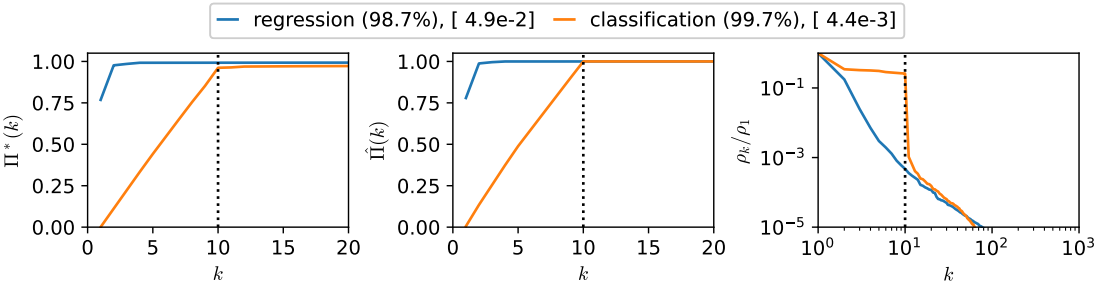


(i) Target projection (ii) Self projection (iii) Eigenvalue

(a) ResNet on CIFAR10 with label randomization

Figure 3.5: MF regime for ResNet18 with randomized labels In (a) we trained a ResNet18 10000 datapoints of CIFAR10 with 0%(blue) 10%(orange) and 100%(green) label noise. All three cases converge to the MF regime, but differ markedly in performance.

.



(a) Target projection (b) Self projection (c) Eigenvalue

Figure 3.6: MF regime for regression problem with scalar output. ResNet18 was trained on MNIST via regression (blue) and classification (orange). For regression, the target/learned functions are scalar output functions, and an image x with digit i is correctly classified if $i - 0.5 < f(x) < i + 0.5$. It can be seen that both models are in the MF regime, using the minimum number of features.

We explained neural collapse sing linear neural network with small initialization and whitened input,

or how the layers amplify the dynamics under gradient descent. Using the principle, we highlight how the dynamics of layerwise linear models (amplifying dynamics) differs from (single layer) linear models and how it leads stage-like training where more important features are learned in sequences. We conclude by showing that a linear neural network (a member of layerwise linear models) or unconstrained feature model can explain neural collapse observed in the previous chapter.

We have introduced the dynamical feedback principle, how they effect

In the last chapter, we empirically observed DNNs' inductive bias toward a

minimum number of features. In an ideal world, we would solve the dynamics of DNN exactly, but the non-linearity (e.g., ReLU) makes the analysis challenging. As physicists often do, we instead focus on simpler layerwise models without non-linearities to study the dynamics of DNNs. The dynamics of layerwise linear models are often exactly solvable [61] and can explain various dynamical phenomena of DNNs such as emergence [5] observed in large language models [62], neural collapse [57] observed in image classification tasks [56], lazy/rich regimes [63] observed in extreme limits [15, 64], and grokking [65] observed in algorithmic tasks [66]. These topics will be discussed in the following chapters as summarized in Fig. 3.1.

4

Scaling laws of MLP in multitask sparse parity problem

Contents

4.1 Introduction

A widely observed property of deep learning models is that the test loss predictably follows a power-law improvement with increasing data, model parameters, or compute. These neural scaling laws [80, 81] have been widely observed across different architectures and datasets [82, 83, 84, 85, 86, 87]. While scaling exponents

vary, the overall phenomenon is robust, raising key questions like: What drives this near-universal scaling behavior?

To study such behavior, Michaud et al. [88] expanded the sparse parity problem [89, 88] and studied the multitask sparse parity problem – a dataset consisting of multiple sparse parity tasks (i.e. skills) where the frequency that each task (skill) appears in the dataset follows a power-law distribution – on 2-layer MLP as a proxy for studying neural scaling laws in large language models (LLMs). For this data set, the authors were able to empirically measure and schematically derive scaling laws as a function of training steps or time (T), parameters (N), and training samples (D). However, no formal derivation or connection to the gradient descent dynamics was studied in their paper.

In this chapter, we formally define each skill as an orthogonal function and propose a model that is expanded in the basis of skill functions. We demonstrate that the learning dynamics of each skill are decoupled and, using this decoupling, analytically derive the scaling laws observed in Michaud et al. [88] and the scaling law for the optimal compute with our model (Table 4.1).

Bottleneck	Time	Data	Parameter	Exponent
Time (T)	T	∞	∞	$-\alpha/(\alpha+1)$
Data (D)	∞	D	∞	$-\alpha/(\alpha+1)$
Parameter (N)	∞	∞	N	$-\alpha$
Compute (C)	$C^{(\alpha+1)/(\alpha+2)}$	∞	$C^{1/(\alpha+2)}$	$-\alpha/(\alpha+2)$

Table 4.1: Summary of the scaling laws. The leftmost column shows the bottleneck of the scaling law. The middle three columns show the resource values in terms of the bottleneck (either taken to infinity or proportional to the bottleneck). The last column shows the scaling exponent for the loss as power-law of the bottleneck where $\alpha + 1$ is the exponent of the skills' distribution (Eq. (4.1)).

4.2 Setup

In this section, we define the multitask sparse parity problem with mean-squared error (MSE) loss, represent skills as orthogonal functions, and measure their strength

Skill idx (I)	Control bits	Skill bits (X)	y	$M(i,x)$	$g_1(i,x)$	$g_2(i,x)$	...	$g_{n_s}(i,x)$
1	1000000	110110000100	S	[1,1,0]	1	0	...	0
1	1000000	100101010001	$-S$	[0,1,0]	-1	0	...	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
2	0100000	001001011011	$-S$	[0,0,1]	0	-1	...	0
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
n_s	0000001	001010100110	$-S$	[1,1,1]	0	0	...	-1

Table 4.2: Multitask sparse parity dataset and skill basis functions. The control bits n_s are one-hot vectors encoding specific parity tasks, indexed in the first column. The frequency of the distinct parity tasks follows a rank-frequency distribution with an inverse power law relation (Eq. (4.1)). The skill bits are binary strings with $m = 3$ relevant sparse bits (highlighted in colors) with their locations varying by skill. The y column shows the target scale S multiplied by the parity computed from the relevant bit set $M(i,x)$. The last columns show the values of the skill basis functions $g_k(i,x)$, defined in Eq. (4.2).

via the linear correlation between model output and skill basis functions. We also introduce the multilinear model used in this and the next chapter.

4.2.1 Multitask sparse parity problem

In the sparse parity problem, n_b skill bits are presented to the model. The target function is a parity function applied to a fixed subset of the input bits. The model must detect the relevant $m < n_b$ sparse bits and return the parity function on this subset ($M(i,x)$, see Table 4.2). Michaud et al. [88] introduced the **multitask** sparse parity problem by introducing n_s unique sparse parity variants – or skills – with different sparse bits (for a representation, see Table 4.2). Each skill is represented in the n_s control bits as a one-hot string, and the model must solve the specific sparse parity task indicated by the control bits.

The n_s skills (random variable $I \in \{1, 2, \ldots, n_s\}$) follow a power law distribution $\mathcal{P}_s$, and the skill bits (random variable $X \in \{0,1\}^{n_b}$) are uniformly distributed. Because $\mathcal{P}_s$ and $\mathcal{P}_b$ are independent, the input distribution $\mathcal{P}(I,X)$ follows a product of two distributions:

$$\mathcal{P}_s(I=i) := \frac{i^{-(\alpha+1)}}{\sum_j^{n_s} j^{-(\alpha+1)}}, \qquad \mathcal{P}_b(X=x) := 2^{-n_b}, \qquad \mathcal{P}(I,X) := \mathcal{P}_s(I)\mathcal{P}_b(X). \tag{4.1}$$

We denote $A = \left(\sum_{j=1}^{n_s} j^{-(\alpha+1)}\right)^{-1}$ so that $\mathcal{P}_s(i) := Ai^{-(\alpha+1)}$.

4.2.2 Skills as orthogonal basis functions

Skill basis functions. We represent the k^{th} skill as a function $g_k : \{0,1\}^{n_s+n_b} \to \{-1,0,1\}$ that returns the parity ($\{-1,1\}$) on the k^{th} skill's sparse bits if $i = k$, but returns 0 if the control bit mismatches that of the k^{th} skill ($i \neq k$):

$$g_k(i,x) := \begin{cases} (-1)^{\sum_j M_j(i,x)} & \text{if } i = k \\ 0 & \text{otherwise} \end{cases}, \tag{4.2}$$

where $M : \{0,1\}^{n_s+n_b} \to \{0,1\}^m$ is the map that selects the relevant sparse bits for the i^{th} skill (Table 4.2) and $M_j(i,x)$ is the j^{th} entry of $M(i,x)$. Note that different skill functions have 0 correlation as the supports of skills functions are **mutually exclusive**:

$$g_k(i,x)g_{k'}(i,x) = \delta_{i,k}\delta_{k,k'}. \tag{4.3}$$

The target function. The target function is a sum over n_s skill functions multiplied by a target scale S:

$$f^*(i,x) := S\sum_{k=1}^{n_s} g_k(i,x). \tag{4.4}$$

The target scale S is the norm of the target function ($\mathbf{E}_{I,X}\left[f^*(I,X)f^*(I,X)\right] = S^2$). Note that the skill functions serve as 'features' or countable basis for describing the target function as in Hutter [90].

MSE loss. We use MSE loss for analytic tractability:

$$\mathcal{L} := \frac{1}{2}\mathbf{E}_{X,I}\left[\left(f^*(I,X) - f(I,X)\right)^2\right], \tag{4.5}$$

where f is the function expressed by a given model. We define the skill loss $\mathcal{L}_k$ as the loss when only the k^{th} skill is given, which can be weighted by their skill frequencies to express the total loss:

$$\mathcal{L}_k := \frac{1}{2}\mathbf{E}_X\left[\left(f^*(I=k,X) - f(I=k,X)\right)^2\right], \qquad \mathcal{L} = \sum_{k=1}^{n_s}\mathcal{P}_s(I=k)\mathcal{L}_k. \tag{4.6}$$

Skill strength. The skill strength or the linear correlation between the k^{th} skill (g_k) and a function expressed by the model at time T (f_T) is

$$\mathcal{R}_k(T) := \mathbf{E}_X \left[g_k(I = k, X) f_T(I = k, X) \right]. \tag{4.7}$$

The skill strength $\mathcal{R}_k$ is the k^{th} coefficient if a model is expanded in the basis of the skill functions (g_k). The skill strength, like the test loss, can be accurately approximated by a sum. The skill loss $\mathcal{L}_k$ (Eq. (4.6)) can be expressed by the skill strength and the norm of the learned function for $I = k$:

$$\mathcal{L}_k(T) = \frac{1}{2} \left(S^2 + \mathbf{E}_X \left[f(I = k, X)^2 \right] - 2S\mathcal{R}_k(f_T) \right). \tag{4.8}$$

The skill loss becomes 0 if and only if $f(I = k, X) = Sg_k(I = k, X)$.

4.2.3 Multilinear model

We propose a simple multilinear model – multilinear with respect to the parameters – with the first N most frequent skill functions $g_k(i, x)$ as the basis functions (features):

$$f(i, x, T; a, b) = \sum_{k=1}^{N} a_k(T) b_k(T) g_k(i, x), \tag{4.9}$$

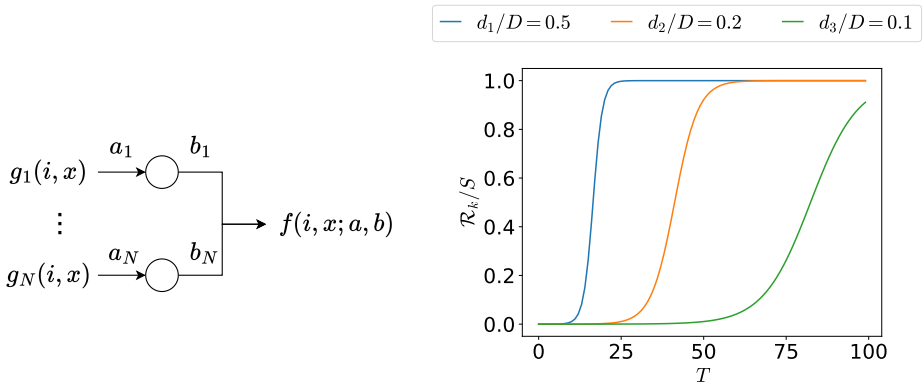
where $a, b \in \mathbb{R}^N$ are the parameters. The model has built-in skill functions g_k – which transform control bits and skill bits into the parity outputs of each skill – so the model only needs to scale the parameters to $a_k b_k = S$.

The multilinear structure (product of a_k, b_k) is analogous to the layered structure of NNs and results in emergent dynamics discussed in the next chapter. A similar model has been studied by Saxe et al. [61] in the context of linear neural networks.

In the multilinear model, because it has the skill functions g_k as the basis, the product $a_k(T)b_k(T)$ is the skill strength $\mathcal{R}_k$ (Eq. (4.7)) and the skill loss (Eq. (4.6)) is a function of S and $\mathcal{R}_k$ only:

$$a_k(T)b_k(T) = \mathcal{R}_k(T), \qquad \mathcal{L}_k(T) = \frac{1}{2}(S - \mathcal{R}_k(T))^2. \tag{4.10}$$

Assuming that we are training the model on D samples from $\mathcal{P}(I, X)$, the empirical loss decomposes into a sum of empirical skill losses because g_k's supports are



(a) Multilinear model illustration **(b)** Decoupled dynamics

Figure 4.1: Multilinear model. (a): An illustration of the multilinear model which is multilinear in terms of parameters, generating a layerwise structure. The model also has the skill functions g_ks as basis functions. **(b):** The dynamics of the multilinear model are decoupled and each skill strength ($\mathcal{R}_k$) shows a sigmoidal growth in time. Note that less frequent skills have a more delayed growth.

mutually exclusive. This **decouples** the dynamics of each skill ($\mathcal{R}_k(T)$), which is analytically solvable under gradient flow (Appendix B.1.1).

$$\boxed{\mathcal{L}^{(D)}(T) = \frac{1}{2D}\sum_{k=1}^{n_s} d_k(S - \mathcal{R}_k(T))^2, \qquad \frac{\mathcal{R}_k(T)}{S} = \frac{1}{1 + \left(\frac{S}{\mathcal{R}_k(0)} - 1\right) e^{-2\eta \frac{d_k}{D} ST}},} \tag{4.11}$$

where d_k is the number of samples of the k^{th} skill (i.e., number of samples (i, x) with $g_k(i, x) \neq 0$), η is the learning rate, and $0 < \mathcal{R}_k(0) < S$ is the skill strength at initialization.

4.2.4 Time scaling law from stage-like training

Assuming our model satisfies the stage-like training for all k of interest, we can derive the time scaling law from the stage-like training. At $\tau_k^{(e)}(\epsilon)$, because of stage-like training, all skills with index up to but not including k have saturated ($\mathcal{R}_{i<k} \approx S$), or equivalently $\mathcal{L}_{i<k} \approx 0$ (Eq. (4.10)). The total loss, the sum of $\mathcal{L}_j$ weighted by $\mathcal{P}_s(j) \propto j^{-(\alpha+1)}$ (Eq. (4.6)), becomes $\sum_{j=k}^{\infty} \mathcal{P}_s(I = j)S^2/2$ (Fig. 3.4(b)). The saturation of

the k^{th} skill results in a loss difference of $\mathcal{P}_s(I=k)S^2/2$. Thus, we obtain

$$\frac{\Delta\mathcal{L}}{\mathcal{L}} \approx \frac{\mathcal{P}_s(I=k)}{\sum_{j=k}^{\infty}\mathcal{P}_s(I=j)} = -\frac{k^{-(\alpha+1)}}{\sum_{j=k}^{\infty} j^{-(\alpha+1)}} \approx -\frac{k^{-(\alpha+1)}}{\int_k^{\infty} j^{-(\alpha+1)}dj} \tag{4.12}$$

$$= -\alpha k^{-1} + \mathcal{O}(k^{-2}). \tag{4.13}$$

Accordingly, the emergent interval between the k and $k+1$ skills relative to the $\tau_k^{(e)}(\epsilon)$ is

$$\frac{\Delta T}{T} = \frac{\tau_{k+1}^{(e)}(\epsilon) - \tau_k^{(e)}(\epsilon)}{\tau_k^{(e)}(\epsilon)} = \frac{(k+1)^{\alpha+1} - k^{\alpha+1}}{k^{\alpha+1}} \tag{4.14}$$

$$= (\alpha+1)k^{-1} + \mathcal{O}(k^{-2}). \tag{4.15}$$

Assuming $k \gg 1$ and combining Eq. (4.13) and Eq. (4.15) to the largest order, we have the equation for the power-law with exponent $-\alpha/(\alpha+1)$ in Fig. 4.2(a):

$$\frac{\Delta\mathcal{L}}{\mathcal{L}} = -\frac{\alpha}{\alpha+1}\frac{\Delta T}{T}. \tag{4.16}$$

If the stage-like training holds for any resource (e.g., time, data, or parameters), the scaling law can be derived using the ratio of change in loss per skill (Eq. (4.13)) and the ratio of change with respect to the resource (given by the emergent time in Eq. (4.15)). The T, D, N scaling laws in the previous chapter can be derived if the stage-like assumption holds and the quanta model [88] is an example where the stage-like training holds for all resources.

4.3 Derivation of the scaling law exponents

We will derive the scaling laws in Table 4.1 with our multilinear model (Section 4.2.3) for time (T), data (D), parameters (N) and optimal compute (C) where we define compute as $C := T \times N$ [91]. Note that we achieve the same exponent as in Hutter [90] for D and in Michaud et al. [88] for T, D, and N. Assuming $0 < \alpha < 1$, the exponents are consistent with the small power-law exponents reported in large-scale experiments, see, e.g., [81, 86, 92].

Using Eqs. (4.6), (4.10) and (4.11), the loss becomes a function of time (T), data (D), parameters (N), and the number of observations for each skill $[d_1, \cdots, d_{n_s}]$:

$$\mathcal{L} = \frac{S^2}{2}\sum_{k=1}^{N}\mathcal{P}_s(k)\frac{1}{\left(1+\left(\frac{S}{\mathcal{R}_k(0)}-1\right)^{-1}e^{2\eta\frac{d_k}{D}ST}\right)^2} + \frac{S^2}{2}\sum_{k=N+1}^{n_s}\mathcal{P}_s(k). \tag{4.17}$$

The equation Eq. (4.17), under suitable assumptions (e.g., for the T scaling law, we take $D, N \to \infty$ and $d_k/D \to \mathcal{P}_s(k)$), results in decoupled evolution of each skill loss, which we use to derive our scaling laws.

4.3.1 Time scaling law

To derive the time scaling law exponent, we assume the time as the bottleneck (finite resource) and take $N, D \to \infty$. By using the decoupled dynamics of each skill loss (Eq. (4.11)),

$$\mathcal{L}_k = \frac{S^2}{2\left(1+\left(\frac{S}{\mathcal{R}_k(0)}-1\right)^{-1}e^{2\eta\frac{d_k}{D}ST}\right)^2}. \tag{4.18}$$

Noting that $d_k/D \to \mathcal{P}_s(k)$ as $D \to \infty$, where $\mathcal{P}_s(k) = Ak^{-(\alpha+1)}$, we have

$$\mathcal{L}_k = \frac{S^2}{2\left(1+\left(\frac{S}{\mathcal{R}_k(0)}-1\right)^{-1}e^{2\eta Ak^{-(\alpha+1)}ST}\right)^2}. \tag{4.19}$$

This is a function of $k^{-(\alpha+1)}T$ only, suggesting the **decoupling** dynamics for each skill. Thus,

$$\frac{d\mathcal{L}_k}{dT} = -\frac{k}{(\alpha+1)T}\frac{d\mathcal{L}_k}{dk}. \tag{4.20}$$

Using Eq. (4.6) and taking $N, n_s \to \infty$ at the same rate,[1] we can approximate the loss as an integral instead of a sum over k:

$$\mathcal{L} = \lim_{N\to\infty}\sum_{k=1}^{N}\mathcal{P}_s(I=k)\mathcal{L}_k \approx \lim_{N\to\infty}\int_1^N Ak^{-(\alpha+1)}\mathcal{L}_k dk, \tag{4.21}$$

[1]We take N and n_s to ∞ at the same rate since we do not want the number of parameters to be a bottleneck in this setup.

where A is the normalization constant for $\mathcal{P}_s$. We can differentiate the loss and use Eq. (4.20) to express the equation in terms of k:

$$\frac{d\mathcal{L}}{dT} = \lim_{N\to\infty} \int_1^N Ak^{-(\alpha+1)} \frac{d\mathcal{L}_k}{dT} dk = -\lim_{N\to\infty} \frac{1}{(\alpha+1)T} \int_1^N Ak^{-\alpha} \frac{d\mathcal{L}_k}{dk} dk. \tag{4.22}$$

Integrating by parts, we obtain

$$\frac{d\mathcal{L}}{dT} = -\lim_{N\to\infty} \frac{1}{(\alpha+1)T} \left[Ak^{-\alpha}\mathcal{L}_k\right]_1^N - \lim_{N\to\infty} \frac{\alpha}{(\alpha+1)T} \int_1^N Ak^{-(\alpha+1)}\mathcal{L}_k dk \tag{4.23}$$

$$= -\lim_{N\to\infty} \mathcal{O}\left(N^{-\alpha}\frac{1}{T}\right) + \mathcal{O}\left(\frac{1}{Te^T}\right) - \frac{\alpha}{(\alpha+1)T}\mathcal{L}. \tag{4.24}$$

The first term goes to 0 as $N \to \infty$ and the second term goes to 0 exponentially faster compared to the last term for $T \gg 1$, which leads to the scaling law with exponent $-\alpha/(\alpha+1)$:

$$\frac{d\mathcal{L}(T)}{\mathcal{L}(T)} = -\frac{\alpha}{\alpha+1}\frac{dT}{T}. \tag{4.25}$$

4.3.2 Data scaling law

The data scaling law assumes $T \to \infty$ and $N \to \infty$ with data as the bottleneck. From the decoupled dynamics of the multilinear model (Eq. (4.11)), we can show that our model is a one-shot learner (Corollary B.1.1):

One shot learner. *Given that $N > k$, $T \to \infty$, and d_k is the number of samples from the training set with $g_k(i, x) \neq 0$, the k^{th} skill loss after training is*

$$\mathcal{L}_k(\infty) = \begin{cases} 0 & : d_k > 0 \\ (S - \mathcal{R}_k(0))^2/2 \approx S^2/2 & : d_k = 0. \end{cases} \tag{4.26}$$

Proof. See Appendix B.1.2. □

Our model requires only one sample from the k^{th} skill to learn such a skill, similar to how language models are few-shot learners at inference.[2] The model can one-shot learn a skill since it has g_k as the basis functions, and the dynamics among

[2]Few-shot learning is typically discussed in the context of models that have undergone pre-training (see, e.g. [62]). We speculate that expanding in the basis g_k in our framework can model aspects of the pre-training process.

different skills are decoupled. A similar one-shot learner has been studied in Hutter [90] where the error depends on a single 'observation' of a feature.

Because the k^{th} skill loss **only depends** on d_k (number of observations for the k^{th} skill), we can calculate the expectation of the skill loss for D data points from $P_{observed}(k|D)$ or the probability that $d_k > 0$:

$$P_{observed}(k|D) = 1 - (1 - \mathcal{P}_s(k))^D . \tag{4.27}$$

Using the one-shot learning property (Eq. (4.26)), the probability of observing the k^{th} skill (Eq. (4.27)), and the decomposition of the loss into skill losses (Eq. (4.6)), the expected loss for D datapoints is

$$\begin{aligned}
\mathbf{E}_D[\mathcal{L}] &= \frac{1}{2}\sum_{k=1}^{\infty} S^2 \mathcal{P}_s(k)(1 - P_{observed}(k)) && (4.28)\\
&= \frac{1}{2} S^2 A \sum_{k=1}^{\infty} k^{-(\alpha+1)} (1 - \mathcal{P}_s(k))^D && (4.29)\\
&\approx \frac{1}{2} S^2 A \int_1^{\infty} k^{-(\alpha+1)} \left(1 - Ak^{-(\alpha+1)}\right)^D dk, && (4.30)
\end{aligned}$$

where the expectation $\mathbf{E}_D$ is over all possible training sets of size D, and A is the normalization constant such that $\mathcal{P}(k) = Ak^{-(\alpha+1)}$. The difference in the loss $\Delta\mathcal{L} = \mathbf{E}_{D+1}[\mathcal{L}] - \mathbf{E}_D[\mathcal{L}]$ is

$$\begin{aligned}
\Delta\mathcal{L} &= \frac{1}{2} S^2 A \int_1^{\infty} k^{-(\alpha+1)} \left(1 - Ak^{-(\alpha+1)}\right)^D \left(\left(1 - Ak^{-(\alpha+1)}\right) - 1\right) dk && (4.31)\\
&= -\frac{1}{2} S^2 A^2 \int_1^{\infty} k^{-2(\alpha+1)} \left(1 - Ak^{-(\alpha+1)}\right)^D dk. && (4.32)
\end{aligned}$$

We can integrate $\Delta\mathcal{L}$ by parts.

$$\begin{aligned}
\Delta\mathcal{L} = \frac{1}{2} \Bigg[& -\frac{S^2 A k^{-\alpha}}{(\alpha+1)(D+1)} \left(1 - Ak^{-(\alpha+1)}\right)^{D+1} \Bigg]_1^{\infty} \\
& - \frac{S^2 A \alpha}{2(\alpha+1)(D+1)} \int_1^{\infty} k^{-(\alpha+1)} \left(1 - Ak^{-(\alpha+1)}\right)^{D+1} dk \\
\approx \mathcal{O}\left((1 - \mathcal{P}_s(1))^{D+1}\right) & - \frac{S^2 A \alpha}{2(\alpha+1)(D+1)} \int_1^{\infty} k^{-(\alpha+1)} \left(1 - Ak^{-(\alpha+1)}\right)^D \left(1 - Ak^{-(\alpha+1)}\right) dk \\
\approx & -\frac{\alpha}{(\alpha+1)(D+1)} \mathbf{E}_D[\mathcal{L}] + \frac{\alpha}{(\alpha+1)(D+1)} \Delta\mathcal{L}.
\end{aligned}$$

In the second line, the first term goes to 0 for $D \gg 1$. In the last line, we used the expression for $\Delta\mathcal{L}$ (Eq. (4.31)) and $\mathbf{E}_D[\mathcal{L}]$ (Eq. (4.28)). Rearranging the equation above and using that $D \gg 1$, we obtain the scaling law with exponent $-\alpha/(\alpha+1)$:

$$\frac{\Delta\mathcal{L}}{\mathbf{E}_D[\mathcal{L}]} = -\frac{\alpha}{1+(\alpha+1)D} \approx -\frac{\alpha}{(\alpha+1)}\frac{1}{D} \tag{4.33}$$

$$= -\frac{\alpha}{(\alpha+1)}\frac{\Delta D}{D}. \tag{4.34}$$

where in the last line, $\Delta D/D = 1/D$ as the change in the number of data points relative to D is one.

4.3.3 Parameter scaling law

The parameter scaling law assumes $T \to \infty$ and $D \to \infty$, with the parameters $N < n_s$ as the bottleneck. Because our model is a one-shot learner (Eq. (4.26)), learning of the k^{th} skill **only depends** on the existence of g_k in the model; the model with $[g_1, \cdots, g_N]$ will learn all $k \leq N$ skills with $\mathcal{L}_k = 0$. The $\mathcal{L}_k$ dependence on g_k is formalized as the following.

Equivalence between a basis function and a skill. *Given $T, D \to \infty$ and if the multilinear model has the N most frequent skill functions as a basis,*

$$\mathcal{L}_k(\infty) = \begin{cases} 0 & : k \leq N \\ S^2/2 & : k > N. \end{cases} \tag{4.35}$$

Proof. See Appendix B.1.3. □

Using Eq. (4.35) and Eq. (4.6), we can express the total loss as function of N:

$$\mathcal{L} \approx \frac{S^2}{2}\int_{N+1}^{\infty} Ak^{-(\alpha+1)}dk \propto (N+1)^{-\alpha}. \tag{4.36}$$

By approximating $N \approx N+1$ for $N \gg 1$, we obtain the power-law with exponent $-\alpha$.

4.3.4 Optimal compute scaling law

For analytical tractability, we define compute as $C := T \times N$. We start from Eq. (4.17) with $D \to \infty$

$$\mathcal{L} \approx \int_1^N Ak^{-(\alpha+1)}\mathcal{L}_k dk + \lim_{n_s\to\infty} \frac{S^2}{2}\int_N^{n_s} Ak^{-(\alpha+1)}dk \tag{4.37}$$

$$= \mathcal{L}_T + \frac{S^2 A}{2\alpha}N^{-\alpha}. \tag{4.38}$$

Using Eq. (4.23), the first term can be analyzed in two cases: $\mathcal{O}(T^{-\alpha/(\alpha+1)})$ if $\mathcal{L}_T \gg N^{-\alpha}$ or $\mathcal{O}(N^{-\alpha}T^{-2})$ if $\mathcal{L}_T = \Theta(N^{-\alpha})$. The second term can be integrated into $\mathcal{O}(N^{-\alpha})$. We can remove the irrelevant constant terms to express the loss

$$\mathcal{L} = T^{-\alpha/(\alpha+1)} + N^{-\alpha}, \tag{4.39}$$

and use the method of Lagrangian multiplier to obtain

$$-\frac{\alpha}{\alpha+1}T^{-\alpha/(\alpha+1)-1} + \lambda N = 0, \tag{4.40}$$

$$-\alpha N^{-(\alpha+1)} + \lambda T = 0, \tag{4.41}$$

$$NT - C = 0, \tag{4.42}$$

where λ is the Lagrange multiplier and C is compute. We can solve the above set of equations to obtain $T \propto N^{\alpha+1}$ or equivalently

$$T \propto C^{(\alpha+1)/(\alpha+2)}, \quad N \propto C^{1/(\alpha+2)}. \tag{4.43}$$

We can plug it in Eq. (4.39) to get

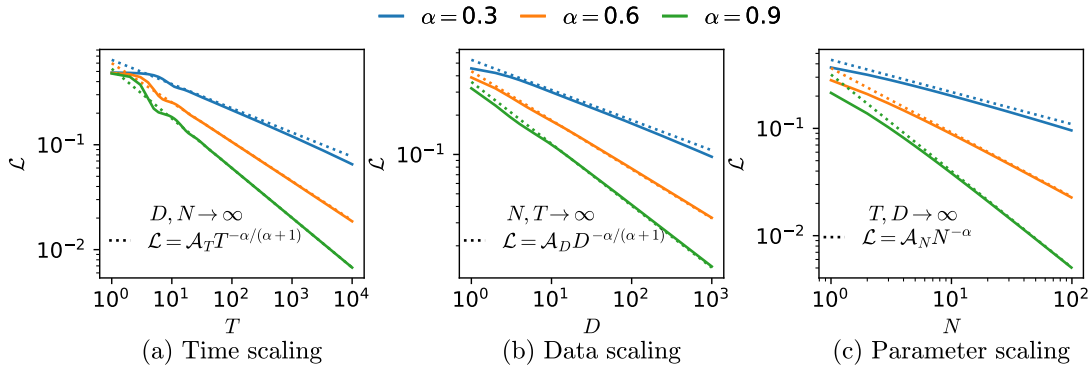
$$\mathcal{L} \propto C^{-\alpha/(\alpha+2)}. \tag{4.44}$$

The optimal compute, in Eq. (4.37), balances the contributions from the first term – loss from given basis functions – and second term – loss from withheld basis functions. Intuitively, the optimal compute means the model has sufficient time to learn most of the given basis functions (Fig. 4.3).

4.4 Experiments and results

To check the derivation of the scaling laws, we compare them to the simulation of our multilinear model. Fig. 4.2 compares the simulation of our model for T, D, N with the scaling laws in Table 4.1. In Fig. 4.3, we train our model for given C but with different tradeoffs between T and N and compare the loss with the theoretical optimal compute scaling law.

Results In Figs. 4.2 and 4.3, we confirm that our theoretical analysis of the scaling laws for the multilinear model holds in practice. The simulation decays faster than the scaling laws for the small α of 0.3, especially for Fig. 4.3 because smaller α amplifies the effect from the number of skills N being finite. See Appendix C.1 for the discussion. We can also calculate the constant prefactor constants of all scaling laws, but this is beyond the scope of the thesis. See [5].



(a) Time scaling (b) Data scaling (c) Parameter scaling

Figure 4.2: Scaling laws. The learning curve ($\mathcal{L}$ is the MSE loss) of the multilinear model (solid) and the theoretical power-law (dotted) for (a) time T, (b) data D, and (c) parameters N. Lower left legends show the condition (top) and the scaling law (bottom) where $\alpha + 1$ is the exponent of the power-law input data (Eq. (4.1)).

4.5 Summary

In this chapter, we introduced the multilinear model and derived the scaling laws observed in Michaud et al. [88] for the multitask sparse parity problem. The derivation of the scaling laws for time, data, and parameters relies on the decoupled dynamics of each skill and how each dynamics scale with the given resource: for

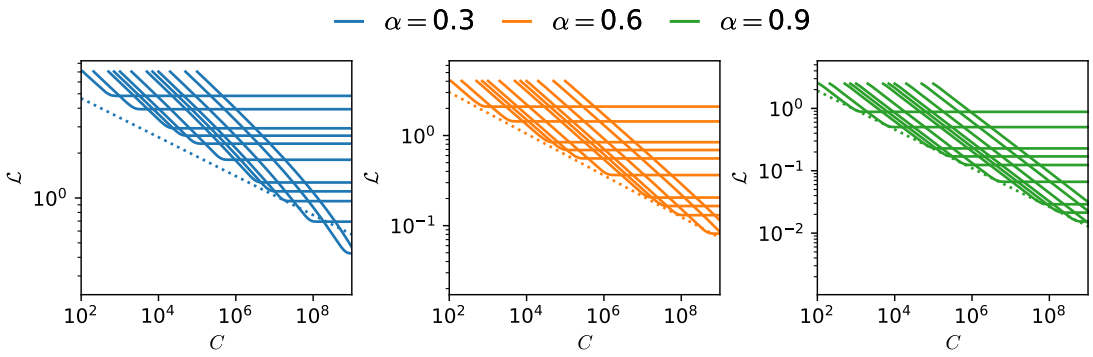


Figure 4.3: Scaling law for optimal compute. The solid lines are the learning curves of the multilinear model as a function of compute $C = T \times N$ with varying parameters N from 10^1 (top plateau) to 10^4 (bottom plateau). The dotted lines are optimal compute scaling laws with exponent $-\alpha/(\alpha + 2)$ (Section 4.3.4). For a given C, we achieve the optimal tradeoff when T is sufficient to fit all N skills (i.e. when the solid lines plateau).

the time scaling law, each $\mathcal{L}_k$ shares the same dynamics with T scaled by $\mathcal{P}_s(k)$ (Eq. (4.20)); for the data scaling law, each $\mathcal{L}_k$ depends only on the observation the k^{th} skill (Eq. (4.26)); for the parameter scaling law, each $\mathcal{L}_k$ depends on whether the model has g_k as a basis function (Eq. (4.35)). For the optimal compute scaling law, we showed the optimal tradeoff between T and N in which the model has sufficient time to learn all given skills (i.e. $T \propto N^{\alpha+1}$).

The decoupling of the skills originates from the mutually exclusive skill functions g_k's in our model (Eq. (4.3)), allowing the analytical tractability for deriving scaling laws. A natural question to ask is why such a strong assumption in the model agrees with the scaling laws observed in 2-layer MLP, as reported in Michaud et al. [88]. In the next chapter, we discuss this top in more detail.

5

Emergence in multitask sparse parity problem

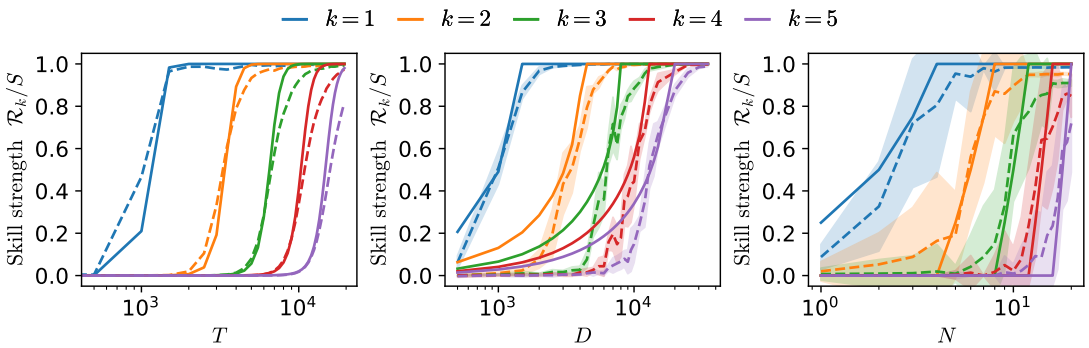
Contents

5.1 Introduction

Emergence in large language models (LLMs) has attracted a lot of recent attention [62, 93, 94, 95]. While the concept of emergence has been critiqued on the grounds that the sharpness of the transition to acquiring a new skill may be sensitive to the measure being used [96], the observation that important new skills are learned for larger models raises many challenging questions of when the skills emerge and what drives the emergence. These questions are complicated by difficulties in formally

defining skills or capabilities [97], and by our general limited understanding of the internal representations of deep neural networks [98].

In this chapter, we extend the multilinear model proposed in the last chapter to predict the emergence in a 2-layer MLP trained on the multitask sparse parity problem (Fig. 5.1). To justify how a multilinear model with fixed basis functions can approximate the dynamics of MLP with feature-learning and no skill basis functions, we explore the stage-like training [61] of the multilinear models. We then show that stage-like training – where the skills are learned in distinct stages – can formally recover the quanta model in Michaud et al. [88] and argue that MLP effectively decouples the skills by learning each skill in stages, justifying our model.



(a) Time emergence (b) Data emergence (c) Parameter emergence

Figure 5.1: Predicting emergence. The skill strength $\mathcal{R}_k$, defined as the k^{th} coefficient if a model is expanded in the basis of the skill functions (g_k), measures how well the k^{th} skill is learned, and is plotted against (a) time T, (b) data set size D, and (c) number of parameters N (width of the hidden layer). $\mathcal{R}_k$ is normalized by the target scale S such that $\mathcal{R}_k/S = 1$ means zero skill loss. The dashed lines show the abrupt growth – emergence – of 5 skills for a 2-layer MLP (Appendix C.3) trained on the multitask sparse parity problem with data power-law exponent $\alpha = 0.6$ (shaded area indicate 1-standard deviation over at least 10 runs). Solid lines are the predictions (Eqs. (5.2), (5.5) and (5.9), respectively) from our multilinear model calibrated on the first skill (blue) only.

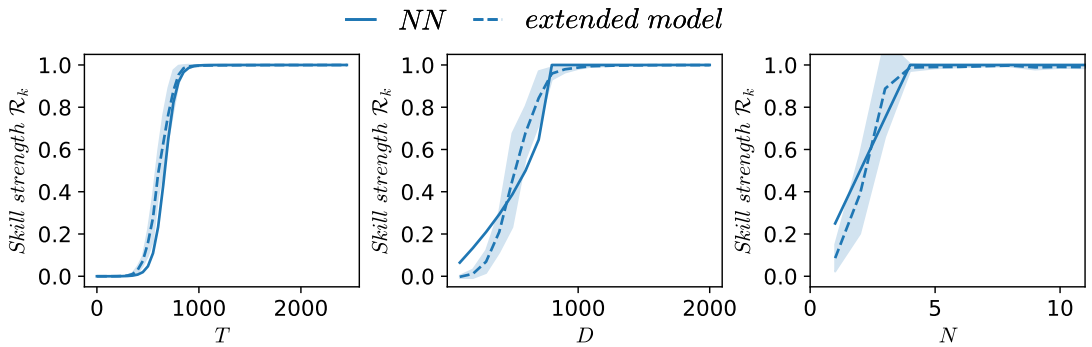
5.2 Predicting emergence

While the multilinear model (Eq. (4.9)) successfully predicts the scaling laws of 2-layer MLP (Chapter 4), it requires extension and calibration to predict the emergence observed in the NNs. Here, we present how the emergent dynamics in a 2-layer MLP can be approximated by extending the multilinear models.

Unlike the multilinear model with predefined skills, NNs **lack** data knowledge and must discover each g_k through feature-learning, requiring additional time, data, or parameters. For each emergence, we propose an extended model with a tunable hyperparameter to capture this inefficiency while preserving skill decoupling.

Methods We calibrate the additional hyperparameter of each extended model on a 2-layer MLP trained on one skill ($n_s = 1$) system (Fig. 5.2). We then use the calibrated models to predict the emergence of subsequent skills in the $n_s = 5$ setup (Fig. 5.1).

The 2-layer MLP in Figs. 5.1 and 5.2 takes the control and skill bits in Table 4.2 as inputs and outputs a scalar ($\{0,1\}^{n_s+n_b} \to \mathbb{R}$). We used 3 relevant bits out of 32 skill bits for the multitask sparse parity problem. The model has a width of 1000 and is trained with SGD unless otherwise stated. See Appendix C.3 for the details.



(a) Time calibration (b) Data calibration (c) Parameter calibration

Figure 5.2: Calibration of hyperparameter on the first skill. The calibration of the extended multilinear model (solid) on the 2-layer NN (dashed) for $n_s = 1$ system. For the calibrated hyperparameters, we have $\mathcal{B}^2 = 1/22$ for time (Eq. (5.2)), $D_c = 800$ for data (Eq. (5.5)), and $N_c = 4$ for hidden layer width (Eq. (5.9)).

5.2.1 Time emergence

In our multilinear model, the layerwise structure – the product of parameters $a_k b_k$ – leads to a sigmoidal saturation where an update of one layer hastens the update of the other layer. Feature learning dynamics in a 2-layer MLP shares the positive feedback between the layers but require a non-trivial update of parameters to express g_k.

Extended model. Given that feature learning, though nonlinear, involves parameter updates, we compensate for the additional delay in feature-learning by multiplying g_k by a calibration constant $0 < \mathcal{B} < 1$:

$$f_T(i, x; a, b) = \sum_{k=1}^{N} a_k(T) b_k(T) \mathcal{B} g_k(i, x), \qquad 0 < \mathcal{B} < 1. \tag{5.1}$$

The calibration constant $\mathcal{B}$ rescales the dynamics in T (Eq. (4.11)):

$$\frac{\mathcal{R}_k(T)}{S} = \frac{1}{1 + \left(\frac{S}{\mathcal{R}_k(0)} - 1\right) e^{-2\eta \mathcal{P}_s(k) \mathcal{B}^2 S T}}, \tag{5.2}$$

where $d_k/D \to \mathcal{P}_s(k)$ because we assume $D \to \infty$. We observe that $\mathcal{B}^2 = 1/22$ fits the NN trained on one skill (Fig. 5.2(a)), and the calibrated model predicts emergence in the $n_s = 5$ system (Fig. 5.1(a)), suggesting that the dynamics of feature-learning g_k in 2-layers NNs is similar to that of decoupled parameter learning $(a_k b_k)$ in a simple multilinear model. For further intuition of the extended model, see an example of time emergence in an NN in Appendix C.4.

5.2.2 Data point emergence

Our multilinear model (Eq. (4.9)) can learn the k^{th} skill with a single observation of the skill because of the prebuilt skill functions g_k (Eq. (4.26)). NNs, without the fixed basis functions, require multiple samples from the k^{th} skill before 'discovering' each g_k.

Extended model. To extend our model to a D_c-shot learner from a one-shot learner, we replace g_k with a sum of basis functions $e_{k,l}$:

$$f_T(i, x; a, B) = \sum_{k=1}^{N} a_k(T) \sum_{l=1}^{D_c} B_{k,l}(T) e_{k,l}(i, x). \tag{5.3}$$

The matrix $B \in \mathbb{R}^{N \times D_c}$ is an extension of $b \in \mathbb{R}^N$ in Eq. (4.9), D_c is a fixed scalar, and $e_{k,l}(i, x) : \{0, 1\}^{n_s + n_b} \to \mathbb{R}$ are functions with the following properties:

$$\mathbf{E}_{X|I=k}\left[e_{k,l} e_{k,l'}\right] = \delta_{ll'}, \qquad e_{k,l}(I \neq k, x) = 0, \qquad \sum_{l=1}^{D_c} \frac{1}{\sqrt{D_c}} e_{k,l} = g_k. \tag{5.4}$$

The first property states that e_k's, when $I = k$, are orthonormal in X. The second property asserts that, similar to g_k (Eq. (4.2)), $e_{k,l}$ is non-zero only when $I = k$, and fitting of the k^{th} skill only occurs among $e_{k,l}$'s, keeping the skills *decoupled*. The third property states that g_k can be expressed using $e_{k,l}$.

For the k^{th} skill, the extended model overfits g_k when there are fewer observations (d_k) than the dimension of the $e_{k,l}$ basis (D_c), and fits g_k when $d_k \geq D_c$: making the model a D_c shot learner.

D_c shot learner. *If we initialize the extended model in Eq. (5.3) with sufficiently small initialization and if the conditions in Eq. (5.4) are satisfied, then the skill strength after training ($T \to \infty$) on D datapoints is*

$$\mathcal{R}_k(\infty) = \begin{cases} S\left(1 - \sqrt{1 - d_k/D_c}\right) & : d_k < D_c \\ S & : d_k \geq D_c. \end{cases} \tag{5.5}$$

The number d_k is the number of samples in the training set for the k^{th} skill (i.e., datapoints with $g_k(i, x) \neq 0$).

Proof. See Appendix B.1.6. □

Using Eq. (5.5), we can calculate the emergence of $\mathcal{R}_k/S$ as a function of D. Note that Eq. (5.5) is similar to the model in Michaud et al. [88] in that, to learn a skill, the model requires a certain number of samples from the skill.

The derivation of Eq. (5.5) follows trivially from the dynamics of the extended model (Eq. (5.3)) and well-known results in linear/kernel regression [16, 20, 22, 99, 24]. To be more specific, the model finds the minimum norm solution as if we performed ridgeless regression on g_k with basis functions $[e_{k,1}, \cdots e_{k,D_c}]$. See Appendix B.1.6 for details.

We observe that $D_c = 800$ approximates the data emergence for the $n_s = 1$ system (Fig. 5.2(b)) and also the emergence for $n_s = 5$ system (Fig. 5.1(b)), suggesting that the NN discovers g_k when it observes D_c samples from the k^{th} skill.

5.2.3 Parameter emergence

Since our multilinear model has g_k's as the basis functions, it requires only one basis function (2 parameters) to express a skill (Eq. (4.35)). A 2-layer NN cannot express a skill (parity function) with a single hidden node (i.e., hidden layer of width 1), but requires multiple hidden nodes.

Extended model. To compensate for the need for multiple hidden nodes in expressing one skill, we extend our model similarly to Eq. (5.3). Because the number of parameters is now a bottleneck, we ensure the model has N basis functions ($e_{k,l}$'s):

$$f_T(i,x;a,B) = \sum_{k=1}^{q-1}\sum_{l=1}^{N_c} a_k(T)B_{k,l}(T)e_{k,l}(i,x) + \sum_{l'=1}^{r} a_q(T)B_{q,l'}(T)e_{q,l'}(i,x), \tag{5.6}$$

where N_c is the minimum number of basis functions needed to express a skill, quotient q is $\lfloor (N-1)/N_c \rfloor + 1$ and remainder r is such that $(q-1)N_c + r = N$. In short, the N basis functions are

$$[e_{1,1}, \cdots, e_{1,N_c}, \quad e_{2,1}, \cdots, e_{2,N_c} \quad \cdots \quad e_{q,1}, \cdots, e_{q,r}]. \tag{5.7}$$

Similar to Eq. (5.4), the basis functions satisfy the following properties

$$\mathbf{E}_{X|I=k}\left[e_{k,l}e_{k,l'}\right] = \delta_{ll'}, \qquad e_{k,l}(I \neq k, x) = 0, \quad \sum_{l=1}^{N_c} \frac{1}{\sqrt{N_c}} e_{k,l} = g_k. \tag{5.8}$$

N_c basis functions for a skill. *For the extended model in Eq. (5.6), the skill strength at $T, D \to \infty$ for a given N becomes*

$$\mathcal{R}_k(\infty) = \begin{cases} 0 & : k > q \\ S\frac{r}{N_c} & : k = q \\ S & : k < q\,. \end{cases} \tag{5.9}$$

Proof. See Appendix B.1.7. □

The equation Eq. (5.9) states that the model can or cannot express the k^{th} skill depending on the number of basis functions for the given skill. For $k < q$, it has all N_c basis functions $[e_{k,1}, \cdots, e_{k,N_c}]$ to express the k^{th} skill (Eq. (5.8)). The skills

with $k > q$ lack the basis functions and are inexpressible. The q^{th} skill are partially expressible depending on the number of basis functions for g_q: $[e_{q,1}, \cdots, e_{q,r}]$.

We observe that $N_c = 4$ fits the parameter emergence for the $n_s = 1$ system (Fig. 5.2(c)) and also the emergence for the $n_s = 5$ system (Fig. 5.1(c)), suggesting that the NN requires 4 nodes in expressing g_k. The results also suggest that an NN, while lacking the ordering of basis functions (Eq. (5.7)), prefers to use the hidden neuron in fitting more frequent skills. The 'preference' toward frequent skills agrees with Fig. 5.1(a) where the NN learns more frequent skills first. Note that for the parameter emergence experiment, Adam [100] was used, instead of SGD, to increase the chance of escaping the near-flat saddle points induced by an insufficient number of parameters [101].

5.3 Discussion

We return to the discussion of Fig. 5.1 or why NNs, despite their **lack** of the decoupling among the skills, behave similarly to the decoupled model with fixed basis functions g_k. We also address the limitations of our model.

5.3.1 Effective decoupling among the skills

Although NN feature-learn g_k's by non-tractable dynamics, we speculate that similar **stage-like** dynamics – induced by the model's layerwise structure and power-law frequencies of the skills – also hold: parameters 'useful' for expressing more frequent skills will be updated significantly faster than parameters useful for expressing less frequent skills. If skill discovery and saturation dynamics operate at different time scales (stages), with negligible interaction among the skills, the skill dynamics become effectively **decoupled**.

When the skill dynamics are decoupled, each skill's feature learning process scales with resource availability for that skill only (e.g., training time, data size, hidden neurons), leading to a similar emergence as in our multilinear model.

5.3.2 Limitations of the multilinear model

The strength of our extended multilinear model comes from the decoupled dynamics for each skill: leading to the prediction of the time, data, and parameter emergence with a single calibration. Here, we discuss the limitations of our model from simplifying the more complex dynamics of NNs.

Time emergence. We note that the NN and the multilinear model emerge at similar instances, but the NN takes longer to saturate fully. This is because, for a given skill, the dynamics of the NN is not one sigmoidal saturation but a sum of **multiple** sigmoidal dynamics with different saturation times. To express the parity function, the NN must use multiple hidden neurons, and the skill strength can be divided into the skill strength from each neuron whose dynamics follow a sigmoidal saturation. Because of the non-linearity and the function it expresses, each neuron is updated at different rates, and the slowly saturating neurons result in a longer tail compared to our multilinear model. For an example, see Fig. C.2 in Appendix C.4.

Data point emergence. Our extended model (Eq. (5.5)) deviates from NNs when $d_k \ll D_c$: NNs show a more abrupt change in $\mathcal{R}_k$ as a function of D. This is because our model asserts strict decoupling among the skills: even a few d_k will contribute to learning g_k from $e_{k,l}$. This differs from the NN, which lacks strict decoupling among the samples from different skills. We speculate that because NNs can perform benign [25] or tempered [102] overfitting, they treat a few data points from less frequent skills as 'noise' from more frequent skills: requiring more samples to learn the infrequent skills.

Parameter emergence. Note that Fig. 5.1(c) has high variance compared to other emergence plots in Fig. 5.1; this is because the NN sparsely, over many repeated trials, uses the hidden neurons to learn less frequent skills over more frequent ones. Table 5.1 is an example of such outliers. Because the 'preference' of NNs toward more frequent skills is not as strict as in our model, we speculate that initial conditions (ones that ease the learning of less frequent skills) play a role in creating outliers.

$k=1$	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98	0.98
$k=2$	4.5	0.95	0.95	0.95	0.96	0.96	0.04	0.96	0.96	0.95
$k=3$	0.6	0.0	0.72	0.90	0.92	0.64	0.88	0.8	0.58	0.52
$k=4$	0.0	0.78	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
$k=5$	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

Table 5.1: Samples of skill strength $\mathcal{R}_k/S$. The table shows the skill strength at $N = 10$ for 10 different runs of the parameter emergence experiment (Fig. 5.1(c)). Note that the variance of $\mathcal{R}_k/S$ is amplified by the outliers – shaded columns – that learn a less frequent skill at the cost of a more frequent skill (second column) or fail to learn a skill (seventh column).

5.4 Summary

We explored how the extended multilinear model predicts emergence in 2-layer MLPs for the multitask sparse parity problem. We showed that the layerwise structure and differences in skill frequencies lead to stage-like training, effectively decoupling skills. The success of our feature-learning-free model in describing this suggests that feature learning dynamics closely mirror parameter learning in layerwise models.

6

Lazy and Rich Regimes to Grokking

Contents

6.1 Introduction

In the last few chapters, we have looked at cases when the feedback principle leads to a greedy dynamics that learns only the minimal number of features (Chapter 2) or learns more relevant features significantly faster (Chapter 5). However, the dynamics of layerwise structure need not always need to be greedy or sigmoidal. For example, target downscaling [64, 21] or NTK initialization [15] can result in

linear dynamics where a linear model can approximate neural network's learning dynamics. Such regimes are called *lazy* regime [64, 103, 104], while the *rich* regime refers to a sufficient deviation from it, including the non-linear dynamics we observed in previous chapters.

Grokking [66] refers to a delayed generalization: the training accuracy reaches 100% quickly but the test accuracy requires a long over-training before achieving a generalizing value. Grokking, when first introduced, was considered as an artifact of algorithmic dataset, but recent studies [105, 106, 65] showed that grokking is a transition from a lazy overfitting regime to a rich generalizing regime.

In this chapter, we argue that the extreme rich regime correlates with the MF regime in Chapter 2, characterized by greedy dynamics and the feedback principle, while the lazy regime aligns with the extreme EF regime, lacking the feedback principle (Fig. 6.1). Using an exactly solvable linear neural network [6], we demonstrate that the known methods for entering the lazy regime also eliminate feedback among layers and linearize the model's dynamics. Leveraging this solved model, we then empirically show that multiple approaches for transitioning from linear to greedy (non-linear) dynamics in the toy model also remove the generalization delay in grokking models.

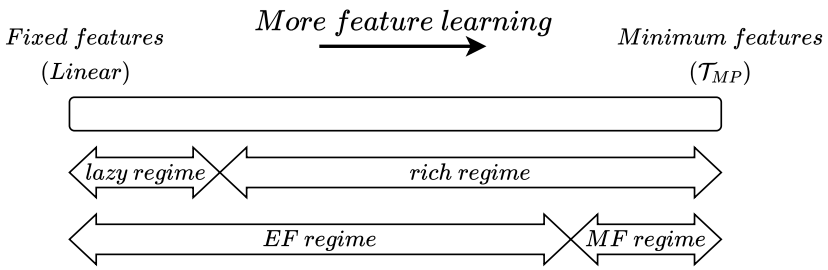


Figure 6.1: Intuitive relationship between MF, EF, rich, and lazy regimes The MF/EF and lazy/rich classifications both assess the amount of feature learning. The lazy/rich regime focuses on deviation from linear dynamics, while the MF/EF regime focuses on deviation from ideal, extreme feature learning dynamics.

6.2 Toy model: linear neural network

One of the major consequences of the feedback principle was its stronger inductive bias, compared to linear models, toward more 'useful' or aligned features. The stronger inductive bias led to the use of a minimum number of features (Chapters 2 and 3) and stage-like training of more frequent skills (Chapters 4 and 5).

To analyze how the dynamics transition from rich (greedy) to lazy (linear), we study a 2-layer p-wide linear neural network with scalar input x and scalar output ($d = c = 1$):

$$f(x) = \frac{x}{Z}\sum_{i=1}^{p} a_i b_i, \quad f^*(x) = xS, \tag{6.1}$$

where $x \in \mathbb{R}$, $S > 0$, and $Z \in \mathbb{R}$ is a normalization or output rescaling constant. In Nam et al. [6], we solve the exact dynamics of Eq. (6.1) for any initialization (e.g., Gaussian distribution). Before we look at the general solution, we study a special case where $a_i(0)$ and $b_i(0)$ are initialized from $\{-1, 1\}$ with equal probabilities.

The dynamics can be decomposed into two modes: the **aligned** mode $\mathcal{I}_+ = \{i : a_i(0)b_i(0)S > 0\}$ and the **anti-aligned** mode $\mathcal{I}_- = \{i : a_i(0)b_i(0)S < 0\}$, where $i \in \{1, \cdots, p\}$. We focus on the distinction between aligned and anti-aligned modes – how greedy the dynamics is – while assessing the linearity of the dynamics. The dynamic equations become

$$\sum_{i\in\mathcal{I}_\pm} \frac{da_i^2}{dt} = \mp\frac{2}{Z}(\theta - S)\sum_{i\in\mathcal{I}_\pm} a_i^2, \tag{6.2}$$

where we used the conserved quantity $a_i^2 - b_i^2 = 0$ and expressed $\theta := Z^{-1}\sum_{i=1}^{p} a_i b_i$ so that $f(x) = x\theta$. We define $\Delta\theta := Z^{-1}\left(\sum_{i\in\mathcal{I}_+} a_i b_i - \sum_{j\in\mathcal{I}_-} a_j b_j\right)$ as the difference between modes and the above equations rearranges into

$$\frac{d}{dt}\theta = -2\frac{\Delta\theta}{Z}(\theta - S), \tag{6.3}$$

$$\frac{d}{dt}(\Delta\theta) = -2\frac{\theta}{Z}(\theta - S). \tag{6.4}$$

Using Eq. (6.2), we find that $a_j/\sqrt{\sum_{i\in\mathcal{I}_+} a_i^2}$ is conserved for a_j in aligned mode ($j \in I_+$):

$$\frac{d}{dt}\left(\frac{a_j}{\sqrt{\sum_{i\in\mathcal{I}_+} a_i^2}}\right) = a_j \frac{d}{dt}\left(\frac{1}{\sqrt{\sum_{i\in\mathcal{I}_+} a_i^2}}\right) + \left(\frac{d}{dt}a_j\right)\left(\frac{1}{\sqrt{\sum_{i\in\mathcal{I}_+} a_i^2}}\right) \tag{6.5}$$

$$= \frac{a_j}{\sqrt{\sum_{i\in\mathcal{I}_+} a_i^2}} - \frac{a_j}{\sqrt{\sum_{i\in\mathcal{I}_+} a_i^2}} \tag{6.6}$$

$$= 0, \tag{6.7}$$

where in the second line, we used Eq. (6.2) for the first term and the gradient flow equation with $a_j = b_j$ for the second term. We obtain

$$a_j^2(t) = \left(\frac{\theta(t) + \Delta\theta(t)}{\theta(0) + \Delta\theta(0)}\right) a_j^2(0), \qquad j \in I_+. \tag{6.8}$$

Likewise, for anti-aligned mode,

$$a_k^2(t) = \left(\frac{\theta(0) + \Delta\theta(0)}{\theta(t) + \Delta\theta(t)}\right) a_k^2(0), \qquad k \in I_-. \tag{6.9}$$

6.2.1 Greedy case

Because $\theta(0) = 0$ and $\Delta\theta(0) > 0$ at initialization, θ and $\Delta\theta$ monotonically increase over time. Using the small initialization assumption $|a_i(0)b_i(0)| \ll S$ and $Z = 1$ as used in previous chapters, the dynamics of aligned mode dominates the dynamics.

Because both θ and $\Delta\theta$ are monotonically increasing and $S = \theta(\infty)$, the small initialization assumption ($\theta(0), \Delta\theta \ll S$) dictates that the aligned mode dominates (larger $a_j(\infty)/a_j(0)$ in Eq. (6.8)) while the anti-aligned mode decays (smaller $a_k(\infty)/a_k(0)$ in Eq. (6.9)).

Intuition As discussed in previous chapters, the feedback principle of layerwise models show amplifying dynamics under sufficiently small initialization. The aligned mode will receive positive feedback between the layers as a_j, b_j grow in magnitude, amplifying the dynamics. The anti-aligned mode will receive negative feedback as the magnitudes of a_k, b_k decay over time, slowing down the dynamics. This is illustrated in Fig. 6.2(a).

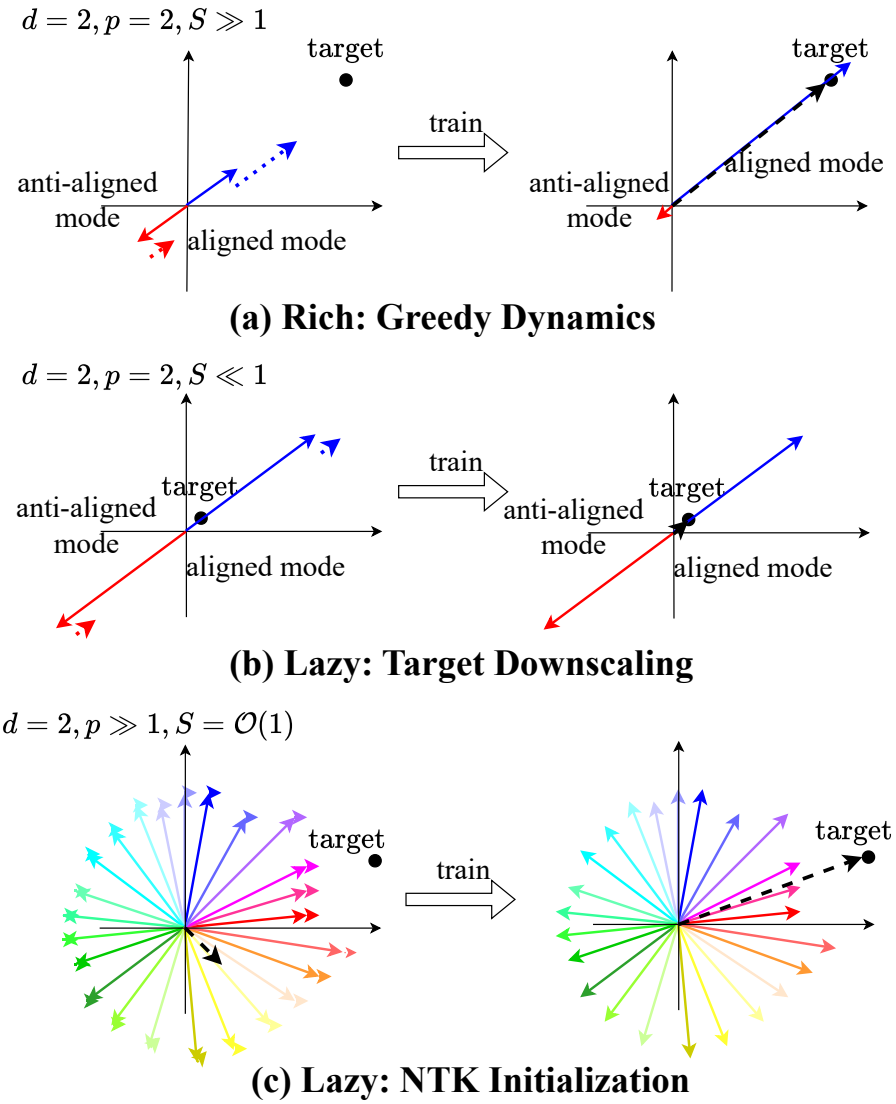


Figure 6.2: Illustration of rich vs. lazy dynamics. The colored arrows represent modes (e.g., $a_i b_i x_i$ in Eq. (6.1)), the dotted arrows represent their gradients, and the black dashed arrow represents the learned function in function space. **(a):** With smaller initialization and larger S, amplifying dynamics reinforce the growth of the aligned mode (larger parameters lead to faster growth) while mitigating the reduction of the anti-aligned mode (smaller parameters lead to slower decay), resulting in the aligned mode to dominate. **(b):** With a smaller target scale (or larger initialization), minimal adjustments per modes are sufficient to fit the target (lazy).**(c):** Under the NTK initialization with numerous modes, gradients toward the target are distributed across the modes, resulting in negligible movement for each mode – thus feature – to fit the target (lazy).

6.2.2 Target downscaling

Lazy regime by target downscaling was first proposed by Chizat et al., [64], which requires two assumptions: the initial function must be a zero function and the target scale must be sufficiently small. We can satisfy the conditions in Eq. (6.1) with $Z = 1$, $S \to 0$, and $\theta(0) = 0$ (or p even). Then $\frac{d\Delta\theta}{dt} \approx 0$ in Eq. (6.4) as both θ and S are close to 0 throughout the dynamics. As $\Delta\theta$ is constant, Eq. (6.3) reduces to linear dynamics (Eq. (3.7)). We can also check in Eqs. (6.8) and (6.9) that both aligned and anti-aligned modes do not evolve over time ($a_k(\infty) \approx a_k(0)$).

Intuition Because the target scale is near the origin, the parameter updates remain infinitesimal. As a result, the dynamics conclude before the feedback (between the layers) takes effect. Without feedback dynamics to distinguish aligned from anti-aligned modes, the dynamics closely resemble those of linear models (see Fig. 6.2(b)).

6.2.3 NTK initialization

NTK initialization, proposed by Jacot et al. [15], requires an infinite width, square root normalization, and $\mathcal{O}(1)$ target. We can achieve the conditions in Eq. (6.1) with $p \to \infty$ with $Z = \sqrt{p}$, and $S = \mathcal{O}(1)$, respectively. Similar to the target downscaling, Eq. (6.3) reduces to linear dynamics because $\Delta\theta$ is fixed in Eq. (6.4). At initializtion, the right hand side of Eq. (6.4) is 0, as $\theta(0) \to 0$ and $Z \to \infty$ for $p \to \infty$. Throughout the dynamics, $\theta \leq S = \mathcal{O}(1)$ ensures fixed $\Delta\theta$, linearizing the dynamics of Eq. (6.3). In Eqs. (6.8) and (6.9), $\Delta\theta = \mathcal{O}(\sqrt{p})$ and $S = \mathcal{O}(1)$ leaves the parameters unchanged for $p \to \infty$.

Intuition The difference between modes, $\Delta\theta$, remains fixed because the dynamics distribute the required movement $\theta - S = \mathcal{O}(1)$ among p features, resulting in minimal change for individual features. To be specific, each feature moves $\mathcal{O}(p^{-1/2})$, and the collective change of p features sums to $\mathcal{O}(1)$ change in θ. Again, such thinly spread dynamics conclude before the feedback takes effect, linearizing the dynamics (Fig. 6.2(c)). See Luo et al., [107] for more rigorous arguments.

6.3 General solution of the scalar-in scalar-out linear neural network

We demonstrated how the lazy regime completes the dynamics before the amplifying dynamics distinguishes modes with different alignments. Next, we present a general solution to Eq. (6.1) from Nam et al. [6] and explore the factors governing the transition between linear and greedy dynamics.

Theorem 1. *The solution for the gradient descent dynamics for MSE loss is given as*

$$a_i(t) = \frac{a_i(0) + b_i(0)}{2}\gamma(t)^{1/2} + \frac{a_i(0) - b_i(0)}{2}\gamma(t)^{-1/2}, \tag{6.10}$$

$$b_i(t) = \frac{a_i(0) + b_i(0)}{2}\gamma(t)^{1/2} - \frac{a_i(0) - b_i(0)}{2}\gamma(t)^{-1/2}, \tag{6.11}$$

where

$$\gamma(t) = \frac{\Sigma_0 - \mathcal{S}_0 + S + \sqrt{\Sigma_0^2 - \mathcal{S}_0^2 + S^2} + \left(-\Sigma_0 + \mathcal{S}_0 - S + \sqrt{\Sigma_0^2 - \mathcal{S}_0^2 + S^2}\right)\exp\left(-\frac{4\sqrt{\Sigma_0^2 - \mathcal{S}_0^2 + S^2}}{Z}t\right)}{\Sigma_0 + \mathcal{S}_0 - S + \sqrt{\Sigma_0^2 - \mathcal{S}_0^2 + S^2} + \left(-\Sigma_0 - \mathcal{S}_0 + S + \sqrt{\Sigma_0^2 - \mathcal{S}_0^2 + S^2}\right)\exp\left(-\frac{4\sqrt{\Sigma_0^2 - \mathcal{S}_0^2 + S^2}}{Z}t\right)} \tag{6.12}$$

and

$$\mathcal{S}_0 = \frac{1}{Z}\sum_{j=1}^{p} a_j(0)b_j(0), \quad \Sigma_0 = \frac{1}{Z}\sum_{j=1}^{p}\frac{a_j(0)^2 + b_j(0)^2}{2}. \tag{6.13}$$

In particular, we have

$$\gamma_+ := \lim_{t\to\infty}\gamma(t) = \gamma(\infty) = \frac{S + \sqrt{\Sigma_0^2 - \mathcal{S}_0^2 + S^2}}{\Sigma_0 + \mathcal{S}_0}. \tag{6.14}$$

The function γ is always monotone from $\gamma(0) = 1$ to $\gamma(\infty)$: if $S > \mathcal{S}_0$ then γ is increasing, and if $S < \mathcal{S}_0$ then γ is decreasing.

See Nam et al. [6] for a proof. For $S > S_0$, the constant γ_+, loosely speaking, determines how a_i and b_i scale with their initial values. A large γ_+ indicates more significant difference in contributions between more and less aligned features, while $\gamma_+ \approx 1$ implies negligible changes in a_i and b_i, resulting in lazy dynamics.

6.3.1 Reinterpretation from weight-to-target ratio

Note that Eq. (6.14) is bounded from below by the ratio between the target scale and weight norm:

$$\gamma_+ = \frac{\mathcal{S} + \sqrt{\Sigma_0^2 - \mathcal{S}_0^2 + S^2}}{\Sigma_0 + \mathcal{S}_0} \geq \frac{2\mathcal{S}}{2\Sigma_0} = \frac{\mathcal{S}}{\Sigma_0},$$

where we used $a_i(0)^2 + b_i(0)^2 \geq 2a_i(0)b_i(0)$, $\Sigma_0 \geq 0$ and $\Sigma_0 \geq \mathcal{S}_0$. Thus, for the amplifying dynamics to distinguish the more aligned from less aligned modes, we need the target to be sufficiently far away compared to the initial function determined by the initial weights.

The small initialization leads to larger γ_+, resulting in greedy dynamics. The target downscaling with $S_0 = 0$ and $S \to 0$ leads to $\gamma_+ \approx 1$. The NTK initialization with $p \to \infty$, $Z = \sqrt{p}$, and $a, b \sim N(0, 1)$ leads to $S_0 \to 0$ with the variance of $Var[S_0] = 1$ and $\Sigma_0 \to \infty$, resulting $\gamma_+ \approx 1$.

Several methods increase γ_+, which coincides with the transition out of the lazy regime. We can increase S by upscaling the target or analogously downscaling the input features (as S is the ratio between y and x). Alternatively, we can decrease Σ_0 by downscaling initialization [108] or by downscaling the output of the function by increasing Z. See also Kumar et al. [105].

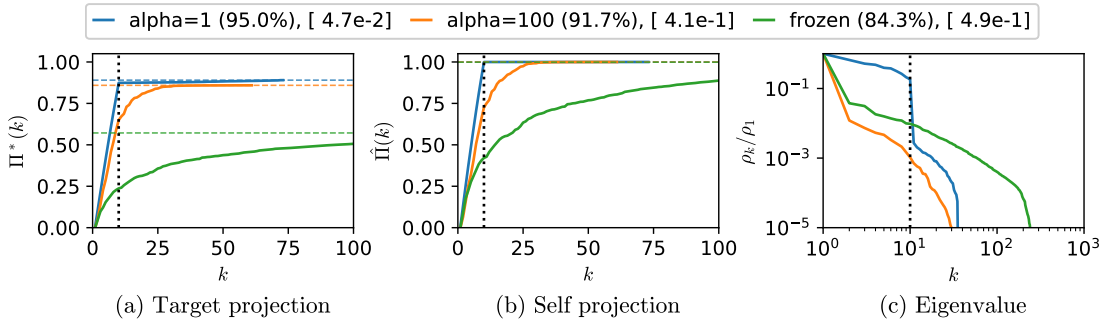
6.4 Empirical validation in Practical DNNs

Here, we empirically verify if our claims on the relationship between EF/MF to lazy/rich regime (Fig. 6.1) and findings on how to control the regimes (Theorem 1) extend to practical NNs with non-linear activations. We will first verify the lazy/rich to EF/MF connection in Fig. 6.1 by observing how target downscaling pushes the model further into the EF regime (Fig. 6.3). We then confirm that grokking is a transition from EF (lazy) to MF (rich) regime (Fig. 6.4). Finally, we verify that methods to increase γ_+ in Theorem 1 also remove the generalization delay in practical DNNs (Fig. 6.5).

6.4.1 Lazy/rich and EF/MF

As observed in Chapter 2, most DNNs in practice are in the rich regime. However, previous studies [44, 109, 110] observed that sufficient target downscaling with $y \to y/\alpha$ with $\alpha \gg \sqrt{p}$ – where p is the width of the last layer – leads to the lazy regime for DNNs, even at finite width.

In Fig. 6.3, we control the dynamics using the target downscaling constant α and compare it to a linear model (last-layer-only training) for a width $p = 256$ FCN. The richest model ($\alpha = 1$) is in the MF regime, the lazier model ($\alpha < 1$) in the EF regime, and the linear model further in the EF regime, as indicated by κ_{CKA}. This aligns with Fig. 6.1, where richer regimes are deeper in the MF regimes, while lazier regimes are in the EF regimes.



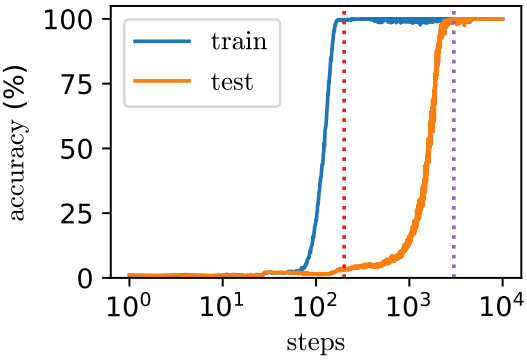
(a) Target projection (b) Self projection (c) Eigenvalue

Figure 6.3: Features with rescaled target compared to fixed features A 4-layer FCN with a width $p = 256$ is trained on 10,000 images of MNIST, with target downscaling (Section 6.2.2). We compare three cases: 1) The standard rich case ($\alpha = 1$), 2) lazier training with $\alpha = 100$, well above the threshold $\alpha/\sqrt{p} > 1$, and 3) last-layer-only (lazy) training with $\alpha = 1$. The standard (rich) scenario is in the MF regime while the lazy and linear dynamics scenarios are in the EF regime, suggesting a connection between MF/EF regimes and rich/lazy regimes.

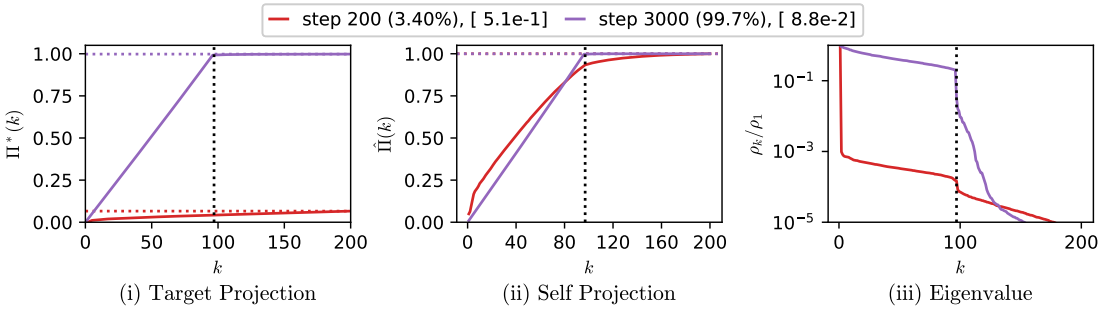
6.4.2 Grokking: transition from EF regime to MF regime

Grokking [66] refers to delayed generalization (Fig. 6.4(a)), where test accuracy improves significantly later than training accuracy. Recently, grokking has been studied as the transition from a lazy regime (overfitting phase) to a rich regime (generalizing phase) [111, 112].

Fig. 6.4 shows a 2-layer transformer trained on a modular p division problem, the first reported grokking example [66]. The metrics show that the model is in the EF regime in the overfitting phase, but transitions into an MF regime after grokking into a generalizing phase (Fig. 6.4(b)).



(a) Grokking



(b) Metrics before and after grokking

Figure 6.4: Grokking is a transition from EF to MF regime. A 2-layer transformer is trained on the modular p task. **(a):** The training and test accuracy as a function of steps, where the test accuracy reaches 100% significantly later than the training accuracy. The vertical dotted lines indicate when the metrics are measured. **(b):** The metrics of the model before grokking (step 200, red) and after grokking (step 3000, purple). The model shows a transition from the EF to the MF regime.

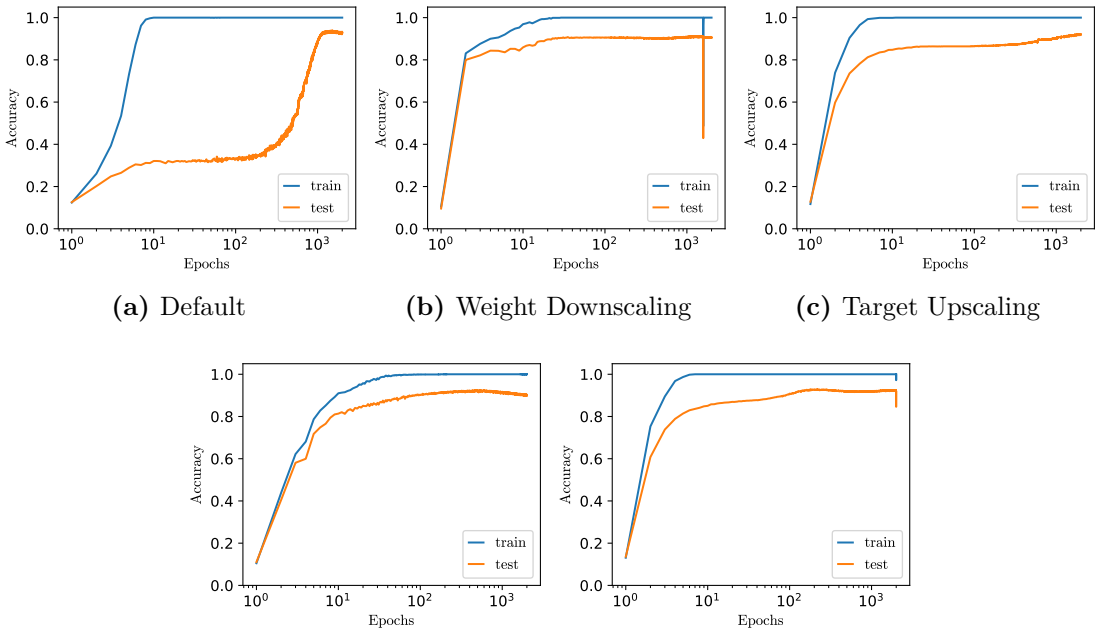
6.4.3 Empirical confirmation on MLP

Fig. 6.4 confirms the finding that grokking is a transition from lazy to rich regimes. If the grokking models were initialized in the rich regime, then models will not need to wait for the transition into the rich regime for generalization.

Fig. 6.5(a) shows a grokking depth 4, width 512 MLPs with tanh activation trained on 1000 datapoints of flattened (vectorized) MNIST dataset. We used

MSE loss with the class label as one-hot vector multiplied by the target scale. By default, the weights of the layers were multiplied by factor of 5 compared to LeCun initialization. For training, Adam with learning ratio of 10^{-3} and weight decay of 10^{-4} was used. Batch size was 128 and the model was trained for 2000 epochs. For input downscaling, the MNIST data vectors were multiplied by the input scaling factor. The output scale was multiplied to the output of the function (MLP).

To verify that our analysis of scalar input neural networks extends to DNNs, we apply techniques to increase γ_+ and escape the lazy regime, as summarized in Table 6.1. Models in Fig. 6.5(b-e) are initialized in richer regimes using these techniques, skipping the prolonged overfitting phase and showing no delay in generalization. This confirms that our analysis of linear neural networks extends to DNNs.



(a) Default **(b)** Weight Downscaling **(c)** Target Upscaling

(d) Input Downscaling **(e)** Output Downscaling

Figure 6.5: Grokking in MLP. We apply four techniques to remove (accelerate) grokking for a 4-layer MLP trained on MNIST.**(a):** Basic grokking scenario. **(b-e):** The methods that increase γ_+: discussed in Section 6.3.1 and summarized in Table 6.1. All methods remove the delay in generalization.

Name	Sub figure	Weight scale	Target scale	Input scale	Output scale
Default	(a)	5	3	1	1
Weight downscaling	(b)	**1**	3	1	1
Target upscaling	(c)	5	**30**	1	1
Input downscaling	(d)	5	3	**0.01**	1
Output downscaling	(e)	5	3	1	**0.1**

Table 6.1: Hyperparameters for Fig. 6.5 (a): The initial parameters for grokking. **(b-e):** The methods to increase γ_+.

6.5 Summary

In this chapter, we connected lazy and rich regimes to EF and MF regimes through their transition from linear to greedy dynamics. By solving a scalar-in, scalar-out linear neural network, we examined how initialization, input scale, output scale, and target scale influence the prevalence of amplifying dynamics and, consequently, the respective regimes.

Through out visualization techniques, we empirically verified our claim on the connection between lazy/rich and EF/MF regimes (Figs. 6.3 and 6.4). In Fig. 6.5, the application of techniques to increase γ_+ in a linear neural network successfully reduced the generalization delay in practical NNs, validating our analysis using linear neural networks.

7

Observations from the Framework

Contents

7.1 Introduction

So far, we have explored scenarios where simpler, solvable models provide good approximations or qualitative insights into practical systems. However, real-world systems are often more complex and various factors govern the level of amplifying dynamics (thus MF and EF regimes). Such factors are challenging to formalize in a solvable model.

Another important question unanswered is whether greedier dynamics lead to better features and performance. No definitive theory exists as the layerwise linear models – without non-linearities – do not gain expressivity boost from greedy dynamics. However, we empirically observe that tighter MF regimes often correlate with improved features and performances.

In this chapter, we present empirical results on how practical factors like training set size, learning rate, batch normalization, and architecture influence both dynamics, level of feature learning, and performance.

7.2 Effect of training set size

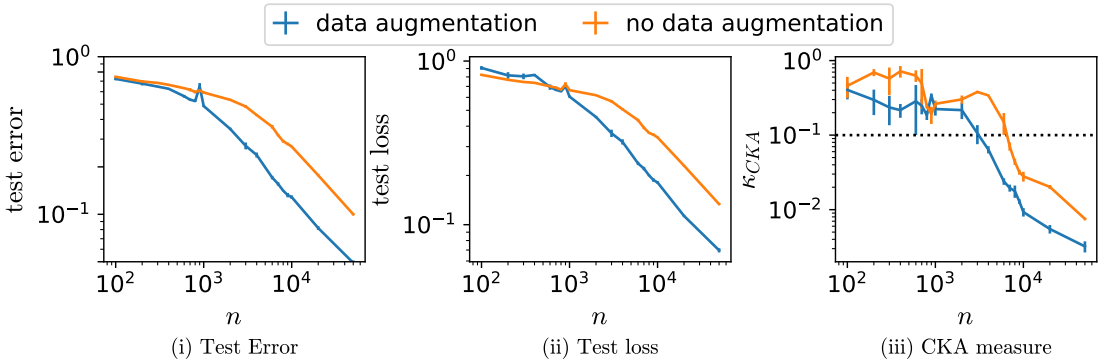
The learning curve, or how fast the loss scales with the training set size, measures the 'goodness' of inductive bias toward the target function. As discussed in Chapter 4, the learning curves often follow a power-law for a sufficient number of datapoints. Here, we explore how the inductive bias for the MF regime changes with additional datapoints.

Fig. 7.1(a) shows the learning curve of ResNet18 on CIFAR10 and the model's CKA measures after training (Fig. 7.1(a, iii)). The learning curve follows two phases: a slower power-law in fewer training samples and a faster power-law in larger numbers. The transition into faster scaling coincides with the transition from the EF to the MF regime, empirically supporting that greedy dynamics correlates with feature learning and thus better performance.

The transition suggests that feature learning (transition into the rich regime) requires a critical number of data points, though the theory behind this remains unclear. We speculate that with insufficient data, many features appear 'good', allowing greedy dynamics to promote those that would otherwise lag, mitigating its selective nature.

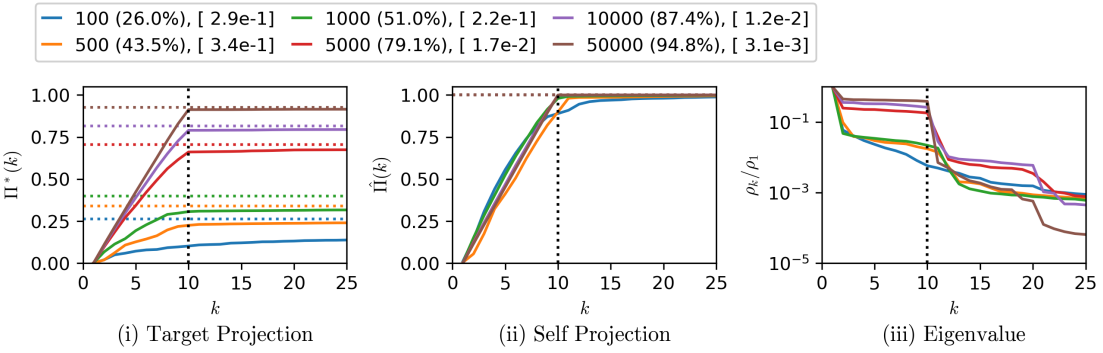
7.3 Effect of hyperparameters

In practice, various hyperparameters can effect the performance. Two most popular hyperameters are the learning rates and the weight decay. In Fig. 7.2, we explore their effects on performance and the greedy dynamics. Fig. 7.2 (a) shows that the two better-performing learning rates are in the MF regime while the underperforming model with the lowest learning rates is in the EF regime. Fig. 7.2(b) shows that



(i) Test Error (ii) Test loss (iii) CKA measure

(a) Learning curves of ResNet18 on CIFAR10



(i) Target Projection (ii) Self Projection (iii) Eigenvalue

(b) Feature learning as a function of training set size

Figure 7.1: Learning curve and feature learning metric. **(a):** Learning curves of ResNet18 on CIFAR10. Both error (i) and loss (ii) learning curves show a transition to a faster-decaying power law with additional data. This correlates with the decay of the MF tightness measure κ_{CKA} in (iii), with the shift occurring near the MF regime boundary (horizontal dotted line). **(b):** Metrics for data-augmented ResNet18 on CIFAR10 with varying data sizes. Adding more data leads to greater concentration of quality (i), utilization (ii), and intensity (iii) in the first C features, reducing κ_{CKA}.

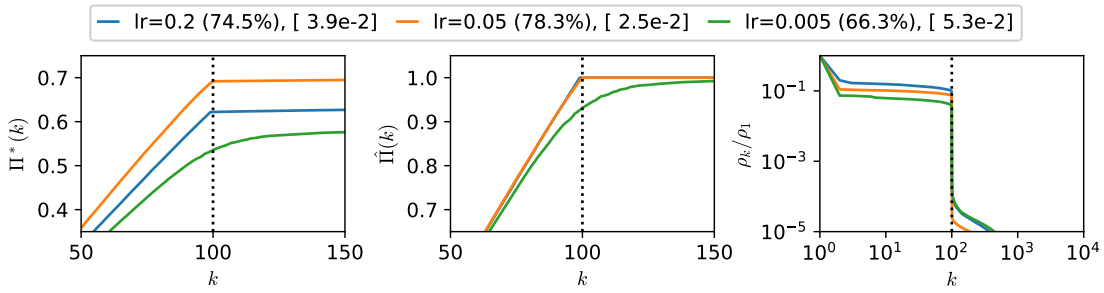
weight decay also flattens the distribution of first C eigenvalues, leading to a tighter MF regime and better performance.

Additionally, note that the model without weight decay also lies in the MF regime, suggesting that weight decay aids but is not solely responsible for the MF regime [113].

Again, we do not have a theory, but we speculate that finite effects of large learning rate enhances the selective nature of the greedy dynamics, selecting only the most relevant features. Likewise, we speculate that the weight decay prevents the less relevant features from gaining meaningful gradients.

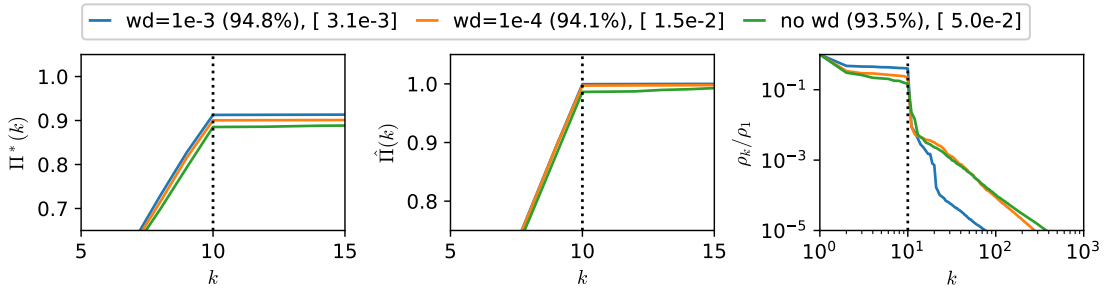
Fig. 7.3 shows that using batch normalization on VGG16 for CIFAR10 and

CIFAR100 not only leads to better performance [114] but also leads to a tighter MF regime. Notably, in Fig. 7.3(b) for CIFAR100, batch normalization shifts the model from EF to MF regime, with approximately 50% boost in test accuracy. We speculate that batch normalization lifts the lazy regime arising from the layer imbalance [63, 65, 115, 116].



(i) Target projection (ii) Self projection (iii) Eigenvalue

(a) ResNet18 on CIFAR100 with different learning rates



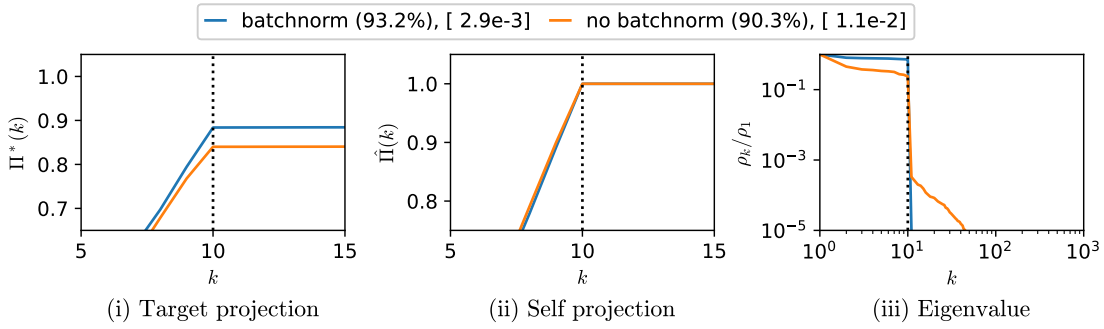
(i) Target projection (ii) Self projection (iii) Eigenvalue

(b) ResNet18 on CIFAR10 with different weight decay rates

Figure 7.2: Effects of learning rates weight decay on feature learning and generalization (a): We explore the effect of changing the learning rate for a ResNet18 on CIFAR100. Note that a lower learning rate leads to the EF regime and poorer performance. The best performance is for the intermediate learning rate in the MF regime with the tightest CKA measure. **(b):** we study the effect of weight-decay for a ResNet18 on CIFAR10 where a tighter MF regime correlates with a larger weight-decay and better performance.

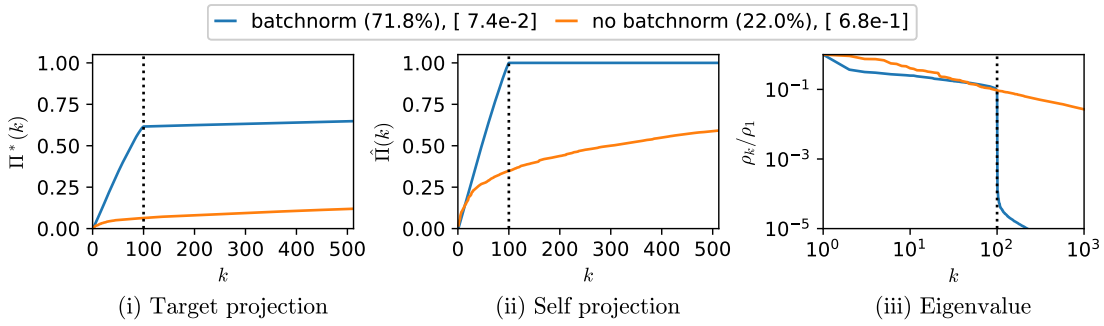
7.4 Effect of architecture

Architecture and dataset pair can influence the performance even when dynamics are greedy through the race toward more frequently used paths [117]. To be specific, different architectures show inductive bias on various functions even when they share



(i) Target projection (ii) Self projection (iii) Eigenvalue

(a) VGG16 on CIFAR10 with and without batch normalization



(i) Target projection (ii) Self projection (iii) Eigenvalue

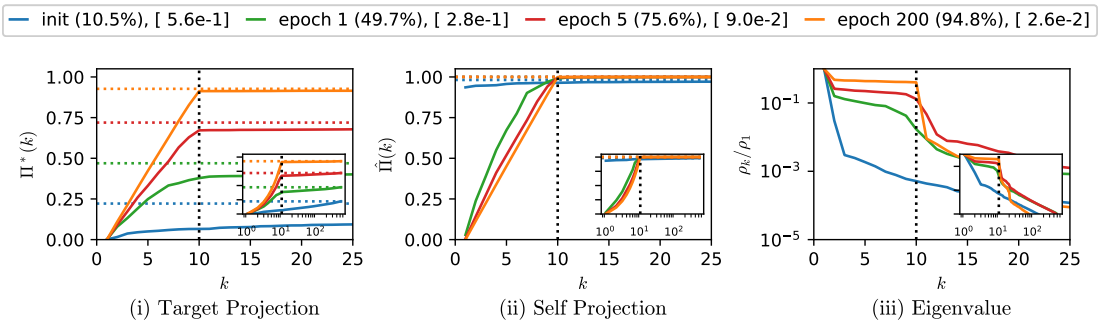
(b) VGG16 on CIFAR100 with and without batch normalization

Figure 7.3: Effect of batch normalization on feature learning and generalization (a): VGG16 on CIFAR10 with/without batch normalization. While both cases are in the MF regime, we observe that including batch normalization results in a tighter MF regime and better performance, including a flatter first C eigenvalues and a wider gap between ρ_C and ρ_{C+1}. **(b):** VGG16 on CIFAR100 with/without batch normalization. Adding batch normalization results in a significant performance boost and a transition into the MF regime. Despite both models achieving training accuracy of over 94%, the underlying dynamics and performance differ significantly.

the stage-like training [118, 119, 120, 121]. In Fig. 7.4, we examine the dynamics of MF and EF regimes by observing the features of ResNet18 (MF regime) and 4-layer 512-wide FCN (EF regime) on CIFAR10 at different epochs. For the MF regime in Fig. 7.4(a), the training dynamics decomposes into two phases: 1) the early collapse into the MF regime with a significant increase in the quality (up to epoch 5) and 2) a gradual increase of the quality and intensity of the first C features without using additional features. The first phase with significant feature learning is consistent with previous research that feature learning occurs at an early stage of training [46, 122, 65].

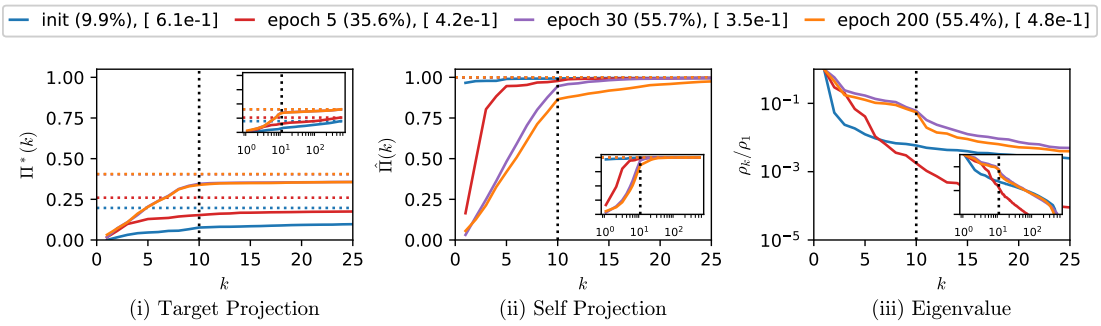
In Fig. 7.4(b), FCN shows less dramatic changes in quality, utilization, and

relative intensity. While it initially trends toward the MF regime, it deviates after epoch 30, using more features later in training. By epoch 200 in Fig. 7.4(b, ii), FCN uses more features to express the learned function than at epoch 30, with smaller $\hat{\Pi}(k)$ across all k. The additional use of features mirrors the behavior of linear models, where weaker intensity features are used in later training stages.



(i) Target Projection (ii) Self Projection (iii) Eigenvalue

(a) ResNet18 on CIFAR10 (MF regime)

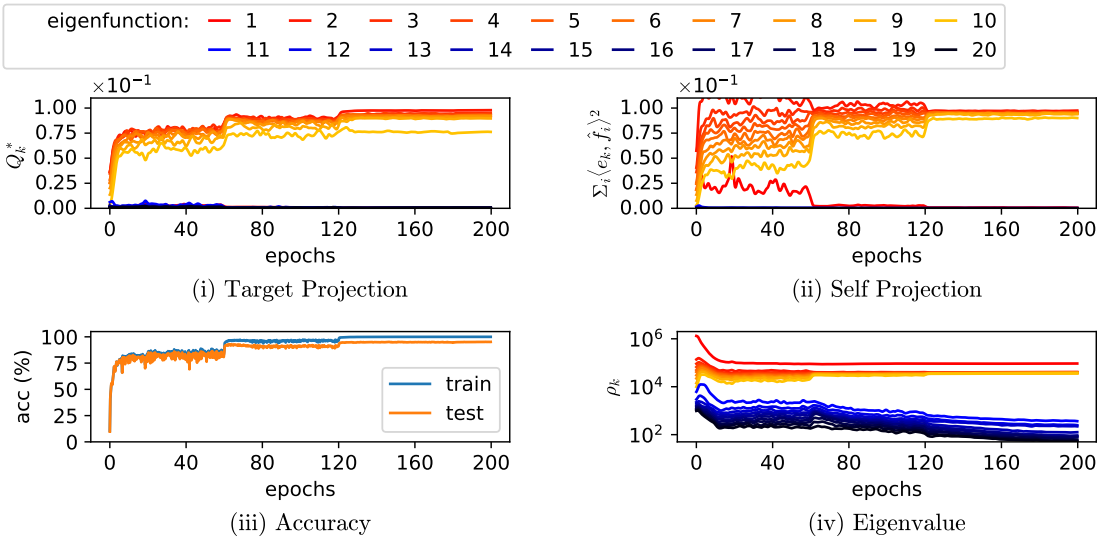


(i) Target Projection (ii) Self Projection (iii) Eigenvalue

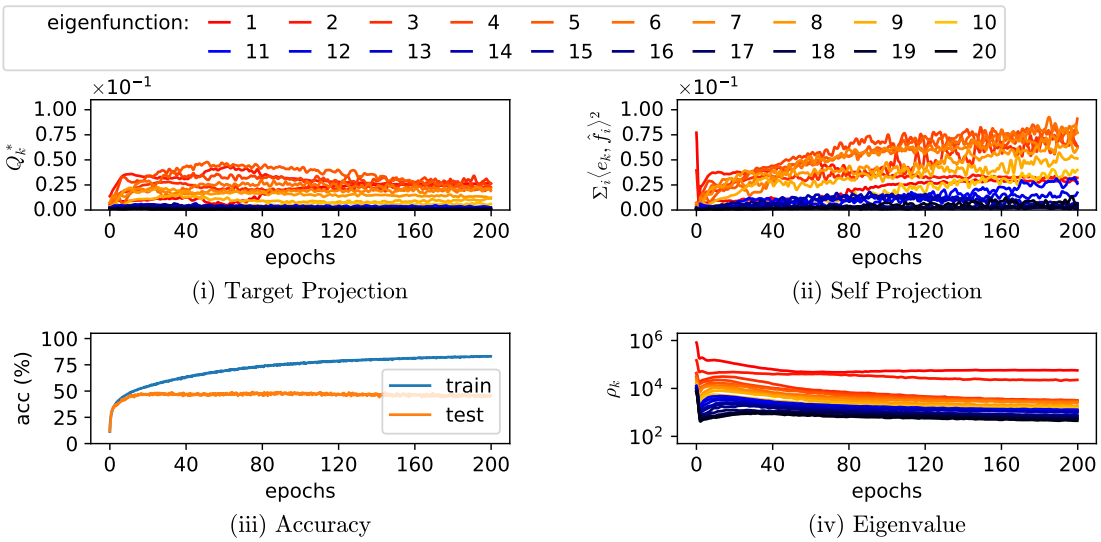
(b) FCN on CIFAR10 (EF regime)

Figure 7.4: Feature-learing as a function of epochs for the MF and EF regimes ResNet18 (a) and 4-layer FCN of width 512 (b) are trained on CIFAR10, and their metrics are shown at different epochs. Parentheses show test accuracy and square brackets indicate the CKA measure at given epochs. ResNet18 concentrates the metric on the first 10 features after just one epoch and reaches the MF regime from epoch 5. In contrast, FCN shows a less dramatic concentration on the first 10 features and deviates from the MF regime after epoch 5 by using more features.

Even when both models in Fig. 7.4 show greedy dynamics towards MF regime in the beginning, FCN eventually deviate from it by using more features. We speculate that the lack or difficulty in expressing the features cause such deviation. In Fig. 7.7, we observe the evolution of the first 20 features. Resnet18 on CIFAR10 behaves as a typical greedy dynamics with the first 10 features showing distinguished dynamics from others. For FCN, the distinction decays after epoch 5 and other



(a) ResNet18, CIFAR10

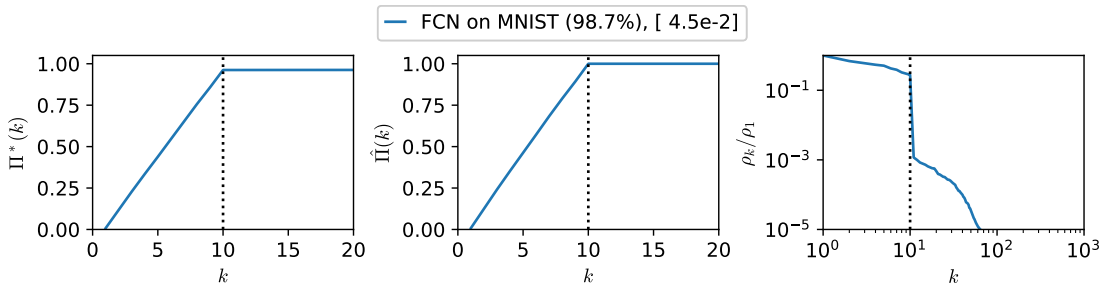


(b) FCN, CIFAR10

Figure 7.5: Temporal evolution of the first 20 features In (a) for ResNet18 on CIFAR10 and (b) for an FCN on CIFAR10, we plot, as a function of epoch and for the first 20 features (i) the quality of the features, (ii) a related projection of each feature onto the learned function, (iii) the test and training accuracies for the full DNNs, and (iv) the eigenvalue spectra. A Gaussian filter has been used for (i,ii,iv) to smooth the curve. We observe that the dynamics for (a) are in the MF regime and are restricted to the first 10 features. By contrast, the dynamics for (b) are in the EF regime. There is an initial regime of feature learning, but this slows down and further lowering of the loss is achieved primarily by using lower-intensity features, as evidenced in (ii).

features also gain utilization, more similar to linear dynamics. This is also when the test and training accuracies begin to diverge.

Although we lack a theory to explain such dynamics, we speculate that the difficulty in obtaining better features in the earlier layers lead to the transition into more lazy regime where lower quality features must be used to express the training set. The results in Fig. 7.6 supports this speculation as similar FCN can achieve MF regime on simpler MNIST dataset.



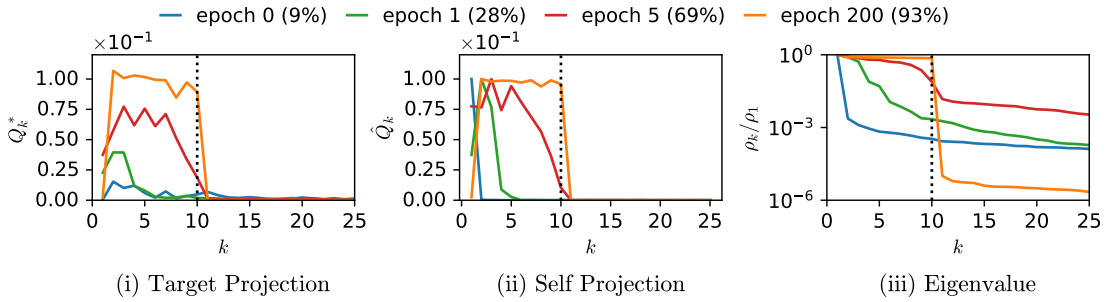
(i) Target projection (ii) Self projection (iii) Eigenvalue

Figure 7.6: FCN on MNIST FCN of width 512 and depth 4, similar to the FCN in Fig. 7.4, reaches the MF regime on MNIST dataset, suggesting the MF regime correlates with the difficulty of feature learning.

7.4.1 Amplifying dynamics with feature learning

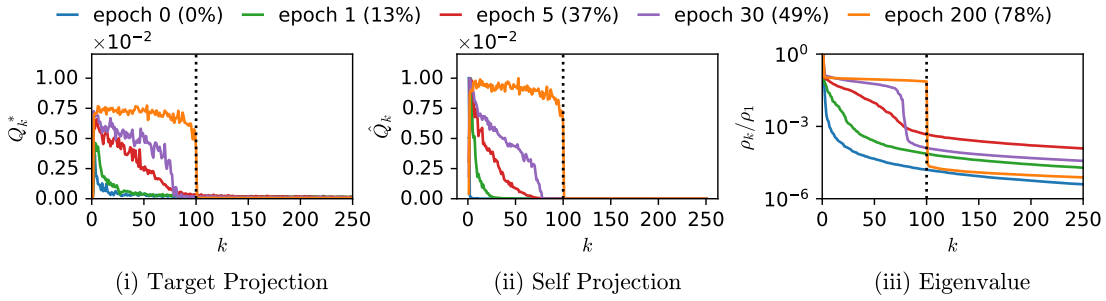
The amplifying dynamics state that the magnitude of one layer governs the change in the other layers. While we have only studied that change in the intensity of a feature in out theoretical search, we can extend the argument to say that sufficient update on the feature also increases the quality of the features.

The intuition is that more gradient means faster selection of more relevant features and thus better quality. In Fig. 7.7, we have supporting evidence for this claim because the quality of the feature correlates with its utilization and intensity ((i), (ii), and (iii) correlate in the figures). Indeed, larger features gain better quality faster. The extension of the feedback principle is left for future work.



(i) Target Projection (ii) Self Projection (iii) Eigenvalue

(a) VGG16, CIFAR10



(i) Target Projection (ii) Self Projection (iii) Eigenvalue

(b) ResNet18, CIFAR100

Figure 7.7: The quality/utilization of individual features at different epochs. The metrics at different epochs for VGG16 on CIFAR10 (a) and ResNet18 on CIFAR100 (b), using individual metrics (Q_k^* and $\hat{Q}_k$) instead of cumulative ones ($\Pi^*(k)$ and $\hat{\Pi}(k)$). A clear correlation is observed for models in the MF regime: features with higher intensity also exhibit higher quality and utilization.

7.5 Summary

In this chapter, we shared our empirical observations that may lead to further understanding of how various factors affect the dynamics and better performance. Widely used practices such as learning rate tuning, weight decay, batch normalization, and the use of complex architectures all direct toward both MF regime and better performance. While our observations link the MF regime to desirable inductive bias, the relationship is not exact—models can enter the MF regime even with randomly labeled data (Fig. 3.5), highlighting the need for deeper understanding.

8 Conclusion

In this thesis, we explored the dynamical inductive bias of deep neural networks (DNNs) through a guiding principle, solvable models, and a framework extended from well-understood linear models. We aimed for intuition by using clear examples with minimal setup, for rigor by analytically proving results in solvable models, and for empirical validation by verifying them in real neural networks.

The dynamical feedback principle (Chapter 3), formalized through solvable layerwise linear models, offers insights into key DNN phenomena. Amplifying dynamics of layerwise models explains neural collapse (Chapter 3), scaling laws (Chapter 4), and emergence (Chapter 5), while its absence accounts for lazy regimes and grokking (Chapter 6). With its simplicity, solvability, and empirical support, this principle makes abstracting DNNs through layerwise dynamics a compelling approach.

The visualization framework (Chapter 2) extends established measures from linear models (Chapter 1) to visualize the richness of network dynamics. Using prior knowledge from linear models, a structured analysis enables the quality, utilization, and intensity of the features to be analyzed. Our findings reveal intriguing, yet unresolved, correlations between architectural choices, hyperparameters, and dynamical richness (Chapter 7), paving the way for future investigations.

The journey toward a complete understanding of DNNs is far from over. In the worst scenario, DNNs may be too complicated for a complete elegant theory. Nevertheless, theoreticians must distill the core principles behind the rapid engineering developments, providing future researchers and practitioners with a clearer narrative to build upon. In this light, we hope this thesis serves as a step toward understanding one fundamental aspect of deep learning: the dynamics of layerwise structure.

Bibliography

[1] Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[2] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *CoRR*, abs/1708.07747, 2017.

[3] Alex Krizhevsky, Vinod Nair, and Geoffrey Hinton. Cifar-10 (canadian institute for advanced research).

[4] Yoonsoo Nam, Chris Mingard, Seok Hyeong Lee, Soufiane Hayou, and Ard Louis. Visualising feature learning in deep neural networks by diagonalizing the forward feature map, 2024.

[5] Yoonsoo Nam, Nayara Fonseca, Seok Hyeong Lee, Chris Mingard, and Ard A. Louis. An exactly solvable model for emergence and scaling laws in the multitask sparse parity problem. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[6] Yoonsoo Nam, Seok Hyeong Lee, Clementine Domine, Yea Chan Park, Charles London, Wonyl Choi, Niclas Goring, and Seungjai Lee. Position: Solve layerwise linear models first to understand neural dynamical phenomena (neural collapse, emergence, lazy/rich regime, and grokking), 2025.

[7] Ouns El Harzli, Yoonsoo Nam, Ilja Kuzborskij, Bernardo Cuenca Grau, and Ard A Louis. Algorithmic stability of minimum-norm interpolating deep neural networks. *NeurIPS 2023 Workshop on Mathematics of Modern Machine Learning*, 2024.

[8] Chris Mingard, Jessica Pointing, Charles London, Yoonsoo Nam, and Ard A. Louis. Exploiting the equivalence between quantum neural networks and perceptrons, 2024.

[9] George E. Andrews, Richard Askey, and Ranjan Roy. *Special Functions*. Encyclopedia of Mathematics and its Applications. Cambridge University Press, 1999.

[10] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*, volume 2. MIT press Cambridge, MA, 2006.

[11] James Mercer. Xvi. functions of positive and negative type, and their connection the theory of integral equations. *Philosophical transactions of the royal society of London. Series A, containing papers of a mathematical or physical character*, 209(441-458):415–446, 1909.

[12] Mark O Hill. Diversity and evenness: a unifying notation and its consequences. *Ecology*, 54(2):427–432, 1973.

[13] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[14] Radford M Neal. Priors for infinite networks (tech. rep. no. crg-tr-94-1). *University of Toronto*, 1994.

[15] Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pages 8571–8580, 2018.

[16] Abdulkadir Canatar, Blake Bordelon, and Cengiz Pehlevan. Spectral bias and task-model alignment explain generalization in kernel regression and infinitely wide neural networks. *Nature communications*, 12(1):2914, 2021.

[17] Abdulkadir Canatar and Cengiz Pehlevan. A kernel analysis of feature learning in deep neural networks. In *2022 58th Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pages 1–8. IEEE, 2022.

[18] Christopher TH Baker and RL Taylor. The numerical treatment of integral equations. *Journal of Applied Mechanics*, 46(4):969, 1979.

[19] Blake Bordelon, Abdulkadir Canatar, and Cengiz Pehlevan. Spectrum dependent learning curves in kernel regression and wide neural networks. In *International Conference on Machine Learning*, pages 1024–1034. PMLR, 2020.

[20] Arthur Jacot, Berfin Simsek, Francesco Spadaro, Clément Hongler, and Franck Gabriel. Kernel alignment risk estimator: Risk prediction from training data. *Advances in Neural Information Processing Systems*, 33:15568–15578, 2020.

[21] Stefano Spigler, Mario Geiger, and Matthieu Wyart. Asymptotic learning curves of kernel methods: empirical data versus teacher–student paradigm. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(12):124001, 2020.

[22] Hugo Cui, Bruno Loureiro, Florent Krzakala, and Lenka Zdeborová. Generalization error rates in kernel regression: The crossover from the noiseless to noisy regime. *Advances in Neural Information Processing Systems*, 34:10131–10143, 2021.

[23] Ouns El Harzli, Guillermo Valle-Pérez, and Ard A Louis. Double-descent curves in neural networks: a new perspective using gaussian processes. *arXiv preprint arXiv:2102.07238*, 2021.

[24] James B Simon, Madeline Dickens, Dhruva Karkada, and Michael R DeWeese. The eigenlearning framework: A conservation law perspective on kernel regression and wide neural networks. *Transactions on Machine Learning Research*, 2023. arXiv:2110.03922.

[25] Peter L Bartlett, Philip M Long, Gábor Lugosi, and Alexander Tsigler. Benign overfitting in linear regression. *Proceedings of the National Academy of Sciences*, 117(48):30063–30070, 2020.

[26] Niladri S Chatterji and Philip M Long. Foolish crowds support benign overfitting. *Journal of Machine Learning Research*, 23(125):1–12, 2022.

[27] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

[28] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[29] Zeyuan Allen-Zhu and Yuanzhi Li. What can resnet learn efficiently, going beyond kernels? *Advances in Neural Information Processing Systems*, 32, 2019.

[30] Amit Daniely and Eran Malach. Learning parities with neural networks. *Advances in Neural Information Processing Systems*, 33:20356–20365, 2020.

[31] Greg Yang and J. Edward Hu. Feature learning in infinite-width neural networks. *CoRR*, abs/2011.14522, 2020.

[32] Maria Refinetti, Sebastian Goldt, Florent Krzakala, and Lenka Zdeborová. Classifying high-dimensional gaussian mixtures: Where kernel methods fail and neural networks succeed. In *International Conference on Machine Learning*, pages 8936–8947. PMLR, 2021.

[33] Eran Malach, Pritish Kamath, Emmanuel Abbe, and Nathan Srebro. Quantifying the benefit of using differentiable learning over tangent kernels. In *International Conference on Machine Learning*, pages 7379–7389. PMLR, 2021.

[34] Gadi Naveh and Zohar Ringel. A self consistent theory of gaussian processes captures feature learning effects in finite cnns. *Advances in Neural Information Processing Systems*, 34:21352–21364, 2021.

[35] Nasim Rahaman, Aristide Baratin, Devansh Arpit, Felix Draxler, Min Lin, Fred A Hamprecht, Yoshua Bengio, and Aaron Courville. On the spectral bias of neural networks. *arXiv preprint arXiv:1806.08734*, 2018.

[36] Suriya Gunasekar, Jason Lee, Daniel Soudry, and Nathan Srebro. Characterizing implicit bias in terms of optimization geometry. In *International Conference on Machine Learning*, pages 1832–1841. PMLR, 2018.

[37] Kaifeng Lyu and Jian Li. Gradient descent maximizes the margin of homogeneous neural networks. *arXiv preprint arXiv:1906.05890*, 2019.

[38] Ziwei Ji and Matus Telgarsky. Directional convergence and alignment in deep learning. *Advances in Neural Information Processing Systems*, 33:17176–17186, 2020.

[39] Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*, pages 818–833. Springer, 2014.

[40] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[41] Christoph Molnar. *Interpretable machine learning*. Lulu. com, 2020.

[42] Daniel Beaglehole, Adityanarayanan Radhakrishnan, Parthe Pandit, and Mikhail Belkin. Mechanism of feature learning in convolutional neural networks. *arXiv preprint arXiv:2309.00570*, 2023.

[43] Lenaic Chizat and Francis Bach. On the global convergence of gradient descent for over-parameterized models using optimal transport. *Advances in neural information processing systems*, 31, 2018.

[44] Mario Geiger, Stefano Spigler, Arthur Jacot, and Matthieu Wyart. Disentangling feature and lazy training in deep neural networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2020(11):113301, 2020.

[45] Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M Roy, and Surya Ganguli. Deep learning versus kernel learning: an empirical study of loss landscape geometry and the time evolution of the neural tangent kernel. *Advances in Neural Information Processing Systems*, 33:5850–5861, 2020.

[46] Alexander Atanasov, Blake Bordelon, and Cengiz Pehlevan. Neural networks as kernel learners: The silent alignment effect. *arXiv preprint arXiv:2111.00034*, 2021.

[47] Blake Bordelon and Cengiz Pehlevan. Self-consistent dynamical field theory of kernel evolution in wide neural networks. *Advances in Neural Information Processing Systems*, 35:32240–32256, 2022.

[48] Ethan Dyer and Guy Gur-Ari. Asymptotics of wide networks from feynman diagrams. *arXiv preprint arXiv:1909.11304*, 2019.

[49] Sho Yaida. Non-gaussian processes and neural networks at finite widths. In *Mathematical and Scientific Machine Learning*, pages 165–192. PMLR, 2020.

[50] Greg Yang, Dingli Yu, Chen Zhu, and Soufiane Hayou. Tensor programs vi: Feature learning in infinite-depth neural networks, 2023.

[51] Nello Cristianini, John Shawe-Taylor, Andre Elisseeff, and Jaz Kandola. On kernel-target alignment. *Advances in neural information processing systems*, 14, 2001.

[52] Maithra Raghu, Justin Gilmer, Jason Yosinski, and Jascha Sohl-Dickstein. Svcca: Singular vector canonical correlation analysis for deep learning dynamics and interpretability. *Advances in neural information processing systems*, 30, 2017.

[53] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International conference on machine learning*, pages 3519–3529. PMLR, 2019.

[54] Aristide Baratin, Thomas George, César Laurent, R Devon Hjelm, Guillaume Lajoie, Pascal Vincent, and Simon Lacoste-Julien. Implicit regularization via neural feature alignment. In *International Conference on Artificial Intelligence and Statistics*, pages 2269–2277. PMLR, 2021.

[55] Yizhang Lou, Chris E Mingard, and Soufiane Hayou. Feature learning and signal propagation in deep neural networks. In *International Conference on Machine Learning*, pages 14248–14282. PMLR, 2022.

[56] Vardan Papyan, XY Han, and David L Donoho. Prevalence of neural collapse during the terminal phase of deep learning training. *Proceedings of the National Academy of Sciences*, 117(40):24652–24663, 2020.

[57] Dustin G Mixon, Hans Parshall, and Jianzong Pi. Neural collapse with unconstrained features. *arXiv preprint arXiv:2011.11619*, 2020.

[58] XY Han, Vardan Papyan, and David L Donoho. Neural collapse under mse loss: Proximity to and dynamics on the central path. *arXiv preprint arXiv:2106.02073*, 2021.

[59] Vignesh Kothapalli, Ebrahim Rasromani, and Vasudev Awatramani. Neural collapse: A review on modelling principles and generalization. *arXiv preprint arXiv:2206.04041*, 2022.

[60] Mariia Seleznova, Dana Weitzner, Raja Giryes, Gitta Kutyniok, and Hung-Hsu Chou. Neural (tangent kernel) collapse. *arXiv preprint arXiv:2305.16427*, 2023.

[61] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *Proceedings of the International Conference on Learning Representations 2014*, 2014. arXiv:1312.6120.

[62] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[63] Clémentine CJ Dominé, Nicolas Anguita, Alexandra M Proca, Lukas Braun, Daniel Kunin, Pedro AM Mediano, and Andrew M Saxe. From lazy to rich: Exact learning dynamics in deep linear networks. *arXiv preprint arXiv:2409.14623*, 2024.

[64] Lenaic Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in neural information processing systems*, 32, 2019.

[65] Daniel Kunin, Allan Raventos, Clémentine Carla Juliette Dominé, Feng Chen, David Klindt, Andrew M Saxe, and Surya Ganguli. Get rich quick: exact solutions reveal how unbalanced initializations promote rapid feature learning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[66] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv:2201.02177*, 2022.

[67] Arthur Jacot, François Ged, Berfin Şimşek, Clément Hongler, and Franck Gabriel. Saddle-to-saddle dynamics in deep linear networks: Small initialization training, symmetry, and sparsity, 2022.

[68] Scott Pesme and Nicolas Flammarion. Saddle-to-saddle dynamics in diagonal linear networks. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[69] Alexander Atanasov, Alexandru Meterez, James B Simon, and Cengiz Pehlevan. The optimization landscape of sgd across the feature learning strength. *arXiv preprint arXiv:2410.04642*, 2024.

[70] Ziwei Ji and Matus Telgarsky. Gradient descent aligns the layers of deep linear networks. *arXiv preprint arXiv:1810.02032*, 2018.

[71] Sanjeev Arora, Nadav Cohen, Noah Golowich, and Wei Hu. A convergence analysis of gradient descent for deep linear neural networks. *arXiv preprint arXiv:1810.02281*, 2018.

[72] Andrew K Lampinen and Surya Ganguli. An analytic theory of generalization dynamics and transfer learning in deep linear networks. *arXiv preprint arXiv:1809.10374*, 2018.

[73] Gauthier Gidel, Francis Bach, and Simon Lacoste-Julien. Implicit regularization of discrete gradient dynamics in linear neural networks. *Advances in Neural Information Processing Systems*, 32, 2019.

[74] Salma Tarmoun, Guilherme Franca, Benjamin D Haeffele, and Rene Vidal. Understanding the dynamics of gradient flow in overparameterized linear models. In *International Conference on Machine Learning*, pages 10153–10161. PMLR, 2021.

[75] Cong Fang, Hangfeng He, Qi Long, and Weijie J Su. Layer-peeled model: Toward understanding well-trained deep neural networks. *arXiv preprint arXiv:2101.12699*, 4, 2021.

[76] Suriya Gunasekar, Blake Woodworth, Srinadh Bhojanapalli, Behnam Neyshabur, and Nathan Srebro. Implicit regularization in matrix factorization, 2017.

[77] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. *Advances in Neural Information Processing Systems*, 32, 2019.

[78] Zhiyuan Li, Yuping Luo, and Kaifeng Lyu. Towards resolving the implicit bias of gradient descent for matrix factorization: Greedy low-rank learning. *arXiv preprint arXiv:2012.09839*, 2020.

[79] Lukas Braun, Clémentine Dominé, James Fitzgerald, and Andrew Saxe. Exact learning dynamics of deep linear networks with prior knowledge. *Advances in Neural Information Processing Systems*, 35:6615–6629, 2022.

[80] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint:1712.00409*, 2017.

[81] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint:2001.08361*, 2020.

[82] Jonathan S Rosenfeld, Amir Rosenfeld, Yonatan Belinkov, and Nir Shavit. A constructive prediction of the generalization error across scales. *arXiv preprint: 1909.12673*, 2019.

[83] Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint:2010.14701*, 2020.

[84] Mitchell A Gordon, Kevin Duh, and Jared Kaplan. Data and parameter scaling laws for neural machine translation. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5915–5922, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[85] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113, 2022.

[86] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint:2203.15556*, 2022.

[87] Gregor Bachmann, Sotiris Anagnostidis, and Thomas Hofmann. Scaling mlps: A tale of inductive bias. *Advances in Neural Information Processing Systems*, 36, 2024.

[88] Eric Michaud, Ziming Liu, Uzay Girit, and Max Tegmark. The quantization model of neural scaling. *Advances in Neural Information Processing Systems*, 36, 2023.

[89] Boaz Barak, Benjamin Edelman, Surbhi Goel, Sham Kakade, Eran Malach, and Cyril Zhang. Hidden progress in deep learning: Sgd learns parities near the computational limit. *Advances in Neural Information Processing Systems*, 35:21750–21764, 2022.

[90] Marcus Hutter. Learning curve theory. *arXiv preprint:2102.04074*, 2021.

[91] Blake Bordelon, Alexander Atanasov, and Cengiz Pehlevan. A dynamical model of neural scaling laws. *arXiv preprint:2402.01092*, 2024.

[92] Tamay Besiroglu, Ege Erdil, Matthew Barnett, and Josh You. Chinchilla scaling: A replication attempt. *arXiv preprint:2404.10102*, 2024.

[93] Deep Ganguli, Danny Hernandez, Liane Lovitt, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova Dassarma, Dawn Drain, Nelson Elhage, et al. Predictability and surprise in large generative models. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*, pages 1747–1764, 2022.

[94] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint:2206.04615*, 2022.

[95] Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. Emergent abilities of large language models. *arXiv preprint: 2206.07682*, 2022.

[96] Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of large language models a mirage? *Advances in Neural Information Processing Systems*, 36, 2023.

[97] Usman Anwar, Abulhair Saparov, Javier Rando, Daniel Paleka, Miles Turpin, Peter Hase, Ekdeep Singh Lubana, Erik Jenner, Stephen Casper, Oliver Sourbut, Benjamin L. Edelman, Zhaowei Zhang, Mario Günther, Anton Korinek, Jose Hernandez-Orallo, Lewis Hammond, Eric Bigelow, Alexander Pan, Lauro Langosco, Tomasz Korbak, Heidi Zhang, Ruiqi Zhong, Seán Ó hÉigeartaigh, Gabriel Recchia, Giulio Corsi, Alan Chan, Markus Anderljung, Lilian Edwards, Yoshua Bengio, Danqi Chen, Samuel Albanie, Tegan Maharaj, Jakob Foerster, Florian Tramer, He He, Atoosa Kasirzadeh, Yejin Choi, and David Krueger. Foundational challenges in assuring alignment and safety of large language models. *arXiv preprint: 2404.09932*, 2024.

[98] Trenton Bricken, Adly Templeton, Joshua Batson, Brian Chen, Adam Jermyn, Tom Conerly, Nick Turner, Cem Anil, Carson Denison, Amanda Askell, Robert Lasenby, Yifan Wu, Shauna Kravec, Nicholas Schiefer, Tim Maxwell, Nicholas Joseph, Zac Hatfield-Dodds, Alex Tamkin, Karina Nguyen, Brayden McLean, Josiah E Burke, Tristan Hume, Shan Carter, Tom Henighan, and Christopher Olah. Towards monosemanticity: Decomposing language models with dictionary learning. *Transformer Circuits Thread*, 2023. https://transformer-circuits.pub/2023/monosemantic-features/index.html.

[99] Ouns El Harzli, Bernardo Cuenca Grau, Guillermo Valle-Pérez, and Ard A Louis. Double-descent curves in neural networks: a new perspective using gaussian processes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 11856–11864, 2024.

[100] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint:1412.6980*, 2014.

[101] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

[102] Neil Mallinar, James Simon, Amirhesam Abedsoltan, Parthe Pandit, Misha Belkin, and Preetum Nakkiran. Benign, tempered, or catastrophic: Toward a refined taxonomy of overfitting. *Advances in Neural Information Processing Systems*, 35:1182–1195, 2022.

[103] Blake Woodworth, Suriya Gunasekar, Jason D Lee, Edward Moroshko, Pedro Savarese, Itay Golan, Daniel Soudry, and Nathan Srebro. Kernel and rich regimes in overparametrized models. In *Conference on Learning Theory*, pages 3635–3673. PMLR, 2020.

[104] Shahar Azulay, Edward Moroshko, Mor Shpigel Nacson, Blake E Woodworth, Nathan Srebro, Amir Globerson, and Daniel Soudry. On the implicit bias of initialization shape: Beyond infinitesimal mirror descent. In *International Conference on Machine Learning*, pages 468–477. PMLR, 2021.

[105] Tanishq Kumar, Blake Bordelon, Samuel J. Gershman, and Cengiz Pehlevan. Grokking as the transition from lazy to rich training dynamics. In *The Twelfth International Conference on Learning Representations*, 2024.

[106] Kaifeng Lyu, Jikai Jin, Zhiyuan Li, Simon Shaolei Du, Jason D. Lee, and Wei Hu. Dichotomy of early and late phase implicit biases can provably induce grokking. In *The Twelfth International Conference on Learning Representations*, 2024.

[107] Tao Luo, Zhi-Qin John Xu, Zheng Ma, and Yaoyu Zhang. Phase diagram for two-layer relu neural networks at infinite-width limit. *Journal of Machine Learning Research*, 22(71):1–47, 2021.

[108] Alan Jeffares, Alicia Curth, and Mihaela van der Schaar. Deep learning through a telescoping lens: A simple model provides empirical insights on grokking, gradient boosting & beyond. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

[109] Mario Geiger, Leonardo Petrini, and Matthieu Wyart. Landscape and training regimes in deep learning. *Physics Reports*, 924:1–18, 2021.

[110] Nikhil Vyas, Alexander Atanasov, Blake Bordelon, Depen Morwani, Sabarish Sainathan, and Cengiz Pehlevan. Feature-learning networks are consistent across widths at realistic scales. *arXiv preprint arXiv:2305.18411*, 2023.

[111] Tanishq Kumar, Blake Bordelon, Samuel J Gershman, and Cengiz Pehlevan. Grokking as the transition from lazy to rich training dynamics. *arXiv preprint:2310.06110*, 2023.

[112] Kaifeng Lyu, Jikai Jin, Zhiyuan Li, Simon S. Du, Jason D. Lee, and Wei Hu. Dichotomy of early and late phase implicit biases can provably induce grokking, 2024.

[113] Tomer Galanti, Zachary S Siegel, Aparna Gupte, and Tomaso Poggio. Sgd and weight decay provably induce a low-rank bias in neural networks. *arXiv preprint arXiv:2206.05794*, 2022.

[114] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? *Advances in neural information processing systems*, 31, 2018.

[115] Zhenfeng Tu, Santiago Aranguri, and Arthur Jacot. Mixed dynamics in linear networks: Unifying the lazy and active regimes. *arXiv preprint arXiv:2405.17580*, 2024.

[116] Yizhou Xu and Liu Ziyin. When does feature learning happen? perspective from an analytically solvable model. *arXiv preprint arXiv:2401.07085*, 2024.

[117] Andrew Saxe, Shagun Sodhani, and Sam Jay Lewallen. The neural race reduction: Dynamics of abstraction in gated networks. In *International Conference on Machine Learning*, pages 19287–19309. PMLR, 2022.

[118] Hannah Pinson, Joeri Lenaerts, and Vincent Ginis. Linear cnns discover the statistical structure of the dataset using only the most dominant frequencies. In *International Conference on Machine Learning*, pages 27876–27906. PMLR, 2023.

[119] Christiaan Heij, André CM Ran, and Freek Van Schagen. *Introduction to mathematical systems theory*. Springer, 2007.

[120] Kwangjun Ahn, Xiang Cheng, Minhak Song, Chulhee Yun, Ali Jadbabaie, and Suvrit Sra. Linear attention is (maybe) all you need (to understand transformer optimization). *arXiv preprint arXiv:2310.01082*, 2023.

[121] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR, 2020.

[122] Jimmy Ba, Murat A Erdogdu, Taiji Suzuki, Zhichao Wang, Denny Wu, and Greg Yang. High-dimensional asymptotics of feature learning: How one gradient step improves the representation. *Advances in Neural Information Processing Systems*, 35:37932–37946, 2022.

Appendices

A

Derivation for linear models

A.1 Derivation of Eq. (1.27)

We show that matrix O in Eq. (1.27) is an orthogonal matrix. The column vectors of matrix O,

$$O_{\cdot i} = \frac{1}{\sqrt{\rho_i}}[\langle e_i|\Phi_1\rangle, \cdots, \langle e_i|\Phi_p\rangle] \tag{A.1}$$

are orthogonal to each other:

$$\begin{aligned} O_{\cdot i}^T O_{\cdot j} &= \frac{1}{\sqrt{\rho_i \rho_j}} \sum_{k=1}^{p} \langle e_i|\Phi_k\rangle \langle \Phi_k|e_j\rangle & \text{(A.2)} \\ &= \frac{1}{\sqrt{\rho_i \rho_j}} \langle e_i| \left(\sum_{k=1}^{p} |\Phi_k\rangle \langle \Phi_k| \right) |e_j\rangle & \text{(A.3)} \\ &= \frac{1}{\sqrt{\rho_i \rho_j}} \langle e_i|\mathcal{T}|e_j\rangle & \text{(A.4)} \\ &= \sqrt{\frac{\rho_j}{\rho_i}} \langle e_i|e_j\rangle & \text{(A.5)} \\ &= \delta_{ij}. & \text{(A.6)} \end{aligned}$$

Because $OO^T = I$ and O is a finite p-dimensional square matrix, associativity of matrix multiplication shows that $O^T O = OO^T = I$ or O is an orthogonal matrix.

A.2 Gradient flow finds the minimum norm/rank solution

We show that a zero-initialized linear model trained with GF finds the minimum rank solution, which is also the minimum norm solution obtained from the ridgeless kernel regression.

Setup Given n samples of the training set, let $\Phi(X) \in \mathbb{R}^{p\times n}$ be the feature matrix and $Y \in \mathbb{R}^{1\times n}$ be the labels. The learned parameters $\hat{w}$ must interpolate the data or

$$\hat{w}^T\Phi(X) = Y. \tag{A.7}$$

Because $\Phi(X)$ is a rank n matrix in an overparameterized setup, performing SVD leads to $\Phi(X) = USV$ where $U \in \mathbb{R}^{p\times n}$, $V \in \mathbb{R}^{n\times n}$, and a diagonal matrix $S \in \mathbb{R}^{n\times n}$.

Minimum rank/norm solution The projection of $\hat{w}$ onto U must satisfy Eq. (A.7) to interpolate the data while any compliment of U (i.e. $\hat{w}(I_p - UU^T)$) does not affect the training set labels. Thus $\hat{w}$ becomes

$$\hat{w} = YV^TS^{-1}U^T + \hat{w}(I_p - UU^T), \tag{A.8}$$

where I_p is a p dimensional identity matrix. It is easy to check that the minimum **norm** and the minimum **rank** solution are when $\hat{w}(I_p - UU^T) = 0$.

GF solution The dynamic equation of the linear model trained with GF is

$$\frac{dw}{dt} = -\eta(w^T\Phi(X) - Y)\Phi(X)^T. \tag{A.9}$$

By replacing $\Phi(X)$ with U, S, and V,

$$\frac{dw}{dt} = -\eta(w^TUSV - Y)V^TSU^T. \tag{A.10}$$

If we multiply $I - UU^T$ on both sides,

$$\frac{dw}{dt}(I_p - UU^T) = -\eta(w^TUSV - Y)V^TSU^T(I_p - UU^T) \tag{A.11}$$

$$= 0, \tag{A.12}$$

where the last line follows because $U^T U = I_n$. Assuming $w = 0$ at initialization, Eq. (A.10) shows that the gradient is constrained on the image of U and GF finds the minimum rank/norm solution of $\hat{w} = YV^T S^{-1} U^T$.

A.2.1 Dynamics of diagonal linear neural network

Starting with the gradient flow equation in diagonal linear neural networks,

$$\frac{da_i}{dt} = -\frac{\partial \mathcal{L}}{\partial a_i}, \quad \frac{db_i}{dt} = -\frac{\partial \mathcal{L}}{\partial b_i}. \tag{A.13}$$

Analogous to Eq. (1.22), we can use the orthogonality condition ($\mathbf{E}[x_i x_j] = 0$ if $i \neq j$) to get

$$\begin{aligned} \frac{da_i}{dt} &= -\mathbf{E}\left[\frac{\partial}{\partial a_i}\frac{1}{2}(f(x) - f^*(x))^2\right] && \text{(A.14)} \\ &= -\mathbf{E}\left[b_i x_i \left(\sum_{j=1}^{p}(a_j b_j - S_j)x_j\right)\right] && \text{(A.15)} \\ &= -b_i \mathbf{E}\left[x_i^2\right](a_i b_i - S_i). && \text{(A.16)} \end{aligned}$$

We can analogously obtain $\frac{db_i}{dt}$, and the evolution of $a_i b_i$ is

$$\begin{aligned} \frac{d(a_i b_i)}{dt} &= \frac{da_i}{dt} b_i + a_i \frac{db_i}{dt} && \text{(A.17)} \\ &= -\mathbf{E}\left[x_i^2\right](b_i^2 + a_i^2)(a_i b_i - S) && \text{(A.18)} \\ &= -2\mathbf{E}\left[x_i^2\right] a_i b_i (a_i b_i - S), && \text{(A.19)} \end{aligned}$$

where we used Eq. (A.16) (and its equivalent for b_i) in the second line and the condition $a_i = b_i$ in the last line. Assuming $a_i(0)b_i(0) < S$, we can solve the differential equation to obtain

$$a_i(t)b_i(t) = \frac{S_i}{1 + \left(\frac{S_i}{a_i(0)b_i(0)} - 1\right) e^{-2S_i \mathbf{E}[x_i^2]t}}. \tag{A.20}$$

For a general derivation when $a_i \neq b_i$, see Appendix A of Saxe et al. (**(year?)**).

A.2.2 Derivation of the magnitude difference conservation in linear neural network

For the sake of readability, we restate the dimensions of inputs, weights, and outputs (labels) for linear neural networks:

$$x \in \mathbb{R}^{d\times 1}, \quad y \in \mathbb{R}^{c\times 1}, \quad W_1 \in \mathbb{R}^{d\times p}, \quad W_2 \in \mathbb{R}^{p\times c}. \tag{A.21}$$

For a linear neural network, the gradient flow equation is

$$\frac{dW_1}{dt} = -\frac{\partial \mathcal{L}}{\partial W_1}, \quad \frac{dW_2}{dt} = -\frac{\partial \mathcal{L}}{\partial W_2}. \tag{A.22}$$

The loss in the summand notation is

$$\mathcal{L} = \frac{1}{2}\mathbf{E}\left[\sum_k \left(\sum_{i,j} x_i w_{ij}^{(1)} w_{jk}^{(2)} - y_k\right)\left(\sum_{i',j'} x_{i'} w_{i'j'}^{(1)} w_{j'k}^{(2)} - y_k\right)\right], \tag{A.23}$$

where $w_{ij}^{(1)}$ and $w_{jk}^{(2)}$ are elements of matrix W_1 and W_2, respectively. The derivative for the first matrix W_1 is

$$\frac{\partial \mathcal{L}}{\partial w_{ij}^{(1)}} = \mathbf{E}\left[\sum_k \left(x_i w_{jk}^{(2)}\right)\left(\sum_{i',j'} x_{i'} w_{i'j'}^{(1)} w_{j'k}^{(2)} - y_k\right)\right] \tag{A.24}$$

$$= \mathbf{E}\left[x_i\left(\sum_{i',j',k} x_{i'} w_{i'j'}^{(1)} w_{j'k}^{(2)} w_{jk}^{(2)} - y_k w_{jk}^{(2)}\right)\right] \tag{A.25}$$

and this can be expressed in the matrix form as

$$\frac{\partial \mathcal{L}}{\partial W_1} = \mathbf{E}\left[x(x^T W_1 W_2 - y^T) W_2^T\right]. \tag{A.26}$$

Likewise for W_2,

$$\frac{\partial \mathcal{L}}{\partial w_{jk}^{(2)}} = \mathbf{E}\left[\left(\sum_i x_i w_{ij}^{(1)}\right)\left(\sum_{i',j'} x_{i'} w_{i'j'}^{(1)} w_{j'k}^{(2)} - y_k\right)\right], \tag{A.27}$$

which in the matrix form is

$$\frac{\partial \mathcal{L}}{\partial W_2} = \mathbf{E}\left[W_1^T x(x^T W_1 W_2 - y^T)\right]. \tag{A.28}$$

Using Eqs. (A.22), (A.26) and (A.28), we obtain

$$W_1^T \frac{dW_1}{dt} = \frac{dW_2}{dt} W_2^T \quad \Rightarrow \quad \frac{d}{dt}(W_1^T W_1 - W_2 W_2^T) = 0. \tag{A.29}$$

A.2.3 Dynamics of linear neural network with small initialization

Here, we show that a special initialization of W_1 and W_2 leads to Eq. (3.15) and argue that a sufficiently small initialization makes the assumption plausible. For a more rigorous arguments, see [61, 57, 70, 71, 72, 73, 74, 79, 63], especially more recent works for formal handling of initialization.

Let the singular value decomposition of the correlation matrix be

$$U^T P V = \mathbf{E}\left[xy^T\right]. \tag{A.30}$$

We assume whitened input $\mathbf{E}[xx^T] = I$, $W_1(0) = U^T A R$, and $W_2(0) = R^T A V$. The diagonal matrix A is of rank c and comprises sufficiently small singular values, while $R \in \mathbb{R}^{c\times c}$ is an orthogonal matrix. These assumptions on W_1 and W_2 immediately shows that layers are balanced:

$$W_1^T W_1 - W_2 W_2^T = 0. \tag{A.31}$$

Denote

$$\widetilde{W}_1 = U W_1 R^T, \quad \widetilde{W}_2 = R W_2 V^T. \tag{A.32}$$

Then from the gradient dynamics

$$\frac{dW_1}{dt} = -\mathbf{E}[xx^T] W_1 W_2 W_2^T + \mathbf{E}[xy^T] W_2^T = -W_1 W_2 W_2^T + U^T P V W_2^T, \tag{A.33}$$

we have

$$\frac{d\widetilde{W}_1}{dt} = \frac{d(UW_1R^T)}{dt} = -(UW_1R^T)(RW_2V^T)(VW_2^TR^T) + (UU^T)P(VW_2^TR^T) \tag{A.34}$$

$$= -\widetilde{W}_1\widetilde{W}_2\widetilde{W}_2^T + P\widetilde{W}_2^T. \tag{A.35}$$

Similarly, for $\widetilde{W}_2$,

$$\frac{d\widetilde{W}_2}{dt} = -\widetilde{W}_1^T\widetilde{W}_1\widetilde{W}_2 + \widetilde{W}_1^T P. \tag{A.36}$$

As both $\widetilde{W}_1$ and $\widetilde{W}_2$ starts with diagonal matrix $\widetilde{W}_1(0) = \widetilde{W}_2(0) = A$, and all matrices in Eq. (A.35) and Eq. (A.36) are diagonal, $\widetilde{W}_1$ and $\widetilde{W}_2$ are always diagonal. Also from

$$\widetilde{W}_1^T\widetilde{W}_1 - \widetilde{W}_2\widetilde{W}_2^T = R(W_1^TW_1 - W_2W_2^T)R^T = 0, \tag{A.37}$$

we have $\widetilde{W}_1(t) = \widetilde{W}_2(t)$ for any t. Denote by α_i the i^{th} diagonal entry of $\widetilde{W}_1$. We have either from Eq. (A.35) or Eq. (A.36),

$$\frac{d\alpha_i}{dt} = -\alpha_i^3 + \rho_i\alpha_i, \tag{A.38}$$

where ρ_i is the i^{th} diagonal entry of P or the i^{th} singular value of the correlation matrix. Solving this equation gives

$$\frac{d\alpha_i^2}{dt} = -2\alpha_i^2(\alpha_i^2 - \rho_i) \quad \Rightarrow \quad \frac{\alpha_i^2(t)}{\rho_i} = \frac{1}{1 + \left(\frac{\rho_i}{\alpha_i^2(0)} - 1\right)e^{-2\rho_i t}}, \tag{A.39}$$

and the dynamics of W_1W_2 is given as

$$W_1W_2 = U^TA(t)^2V, \quad A(t)^2 = \begin{pmatrix} \alpha_1(t)^2 & & \\ & \alpha_2(t)^2 & \\ & & \ddots \end{pmatrix}. \tag{A.40}$$

Discussion on the assumption The assumption specifies a particular form for the matrix, but with sufficiently small initialization, all matrices approximately meet the conditions.

B

Derivation for chapters 2 and 3

In this section, we show the derivation for the extended multilinear model.

B.1 Derivation of the multilinear model

In this section, we provide derivations of how the skill loss of our multilinear model evolves with a given resource: time (Lemma 1), data (Corollary B.1.1), and parameters (Corollary B.1.2). Note that two corollaries for data and parameters (Corollaries B.1.1 and B.1.2) follow from the decoupled dynamics (Lemma 1).

B.1.1 Decoupled dynamics of the multilinear model

Lemma 1. *Let the multilinear model Eq. (4.9) be trained with gradient flow on D i.i.d samples for the setup in Section 4.2.2 (input distribution: Eq. (4.1), target function: Eq. (4.4), and MSE loss: Eq. (4.5)). Let $k \leq N$ be a skill index in the multilinear model and the input distribution ($k \leq n_s$). Then assuming the following initialization $a_k(0) = b_k(0)$ and $0 < a_k(0)b_k(0) < S$, the dynamics of the k^{th} skill strength ($\mathcal{R}_k$) is*

$$\mathcal{R}_k(T) = \frac{S}{1 + \left(\frac{S}{\mathcal{R}_k(0)} - 1\right) e^{-2\eta S \frac{d_k}{D} T}} \tag{B.1}$$

and the skill loss is

$$\mathcal{L}_k(T) = \frac{S^2}{2\left(1 + \left(\frac{S}{\mathcal{R}_k(0)} - 1\right)^{-1} e^{2\eta S \frac{d_k}{D} T}\right)^2}, \tag{B.2}$$

where η is the learning rate and d_k is the number of observations with $g_k(I = k, x^{(j_k)}) \neq 0$.

Proof. For $j = 1, \cdots, D$, denote $(i^{(j)}, x^{(j)})$ be the j^{th} data point in the training set. Then the empirical loss for D datapoints is given as

$$\mathcal{L}^{(D)} = \frac{1}{2D} \sum_{j=1}^{D} \left(f^*(i^{(j)}, x^{(j)}) - f(i^{(j)}, x^{(j)})\right)^2. \tag{B.3}$$

We note that

$$\begin{aligned}
\left(f^*(i^{(j)}, x^{(j)}) - f(i^{(j)}, x^{(j)})\right)^2 &= \left(\sum_{k=1}^{n_s} (S - a_k b_k) g_k(i^{(j)}, x^{(j)})\right)^2 \\
&= (S - a_{i^{(j)}} b_{i^{(j)}})^2 g_{i^{(j)}}(i^{(j)}, x^{(j)})^2 \\
&= (S - a_{i^{(j)}} b_{i^{(j)}})^2,
\end{aligned}$$

as $g_i(i, j) \in \{1, -1\}$ and $g_k(i, j) = 0$ for $i \neq k$. So if we denote d_k the number of data points with $i^{(j)} = k$, then we can conclude

$$\mathcal{L}^{(D)} = \frac{1}{2D} \sum_{j=1}^{D} (S - a_{i^{(j)}} b_{i^{(j)}})^2 = \frac{1}{2D} \sum_{k=1}^{n_s} d_k (S - a_k b_k)^2, \tag{B.4}$$

which is the decoupled loss in the main text (Eq. (4.11)). Using the gradient descent equation and Eq. (B.4), we obtain

$$\frac{da_k}{dt} = -\eta \frac{d\mathcal{L}_D}{da_k} \tag{B.5}$$

$$= -\eta \frac{d_k}{D} b_k (a_k b_k - S). \tag{B.6}$$

Likewise, we can obtain the equation for b_k as

$$\frac{db_k}{dt} = -\eta \frac{d_k}{D} a_k (a_k b_k - S). \tag{B.7}$$

Because of symmetry between a and b (See [61]), assuming $a_k(0) = b_k(0)$, and $a_k(0)b_k(0) > 0$ results in $a_k(T) = b_k(T)$ for all T. The equation for $\mathcal{R}_k = a_k b_k$ is

$$\frac{d\mathcal{R}_k}{dt} = -\eta \frac{da_k}{dt} b_k + a_k \frac{db_k}{dt} = -\eta \frac{d_k}{D}(b_k^2 + a_k^2)(a_k b_k - S) \tag{B.8}$$

$$= -2\eta \frac{d_k}{D} \mathcal{R}_k(\mathcal{R}_k - S). \tag{B.9}$$

Assuming $a_k(0)b_k(0) < S$, we can solve the differential equation to obtain

$$\mathcal{R}_k(T) = \frac{S}{1 + \left(\frac{S}{\mathcal{R}_k(0)} - 1\right) e^{-2\eta S \frac{d_k}{D} T}}. \tag{B.10}$$

The equation for $\mathcal{L}_k$ follows from Eq. (4.10). □

B.1.2 One-shot learner

Corollary B.1.1. *For the setup in Lemma 1, the k^{th} skill loss ($\mathcal{L}_k$) at $T, N \to \infty$ is*

$$\mathcal{L}_k(\infty) = \begin{cases} 0 & : d_k > 0 \\ (S - \mathcal{R}_k(0))^2/2 \approx S^2/2 & : d_k = 0, \end{cases} \tag{B.11}$$

where d_k is the number of k^{th} skill's observations.

Proof. The corollary follows directly from Lemma 1. By taking $T, N \to \infty$,

$$\mathcal{R}_k(\infty) = \begin{cases} S & : d_k > 0 \\ \mathcal{R}_k(0) & : d_k = 0 \end{cases} \tag{B.12}$$

We obtain the result by using the relationship between $\mathcal{R}_k$ and $\mathcal{L}_k$ in Eq. (4.10). □

B.1.3 Equivalence between a basis function and a skill

Corollary B.1.2. *Let the multilinear model Eq. (4.9) be trained with gradient flow on D i.i.d samples for the setup in Section 4.2.3 (input distribution: Eq. (4.1), target function: Eq. (4.4), and MSE loss: Eq. (4.5)). Assume $a_k(0) = b_k(0)$, $0 < a_k(0)b_k(0) < S$, and that the model has the N most frequent skills as basis functions. Then $\mathcal{R}_k$ for the $k^{th} \leq n_s$ skill at $T, D \to \infty$ is*

$$\mathcal{L}_k(\infty) = \begin{cases} 0 & : k \leq N \\ S^2/2 & : k > N \end{cases} \tag{B.13}$$

Proof. The corollary follows directly from Lemma 1. By taking $T, D \to \infty$,

$$\mathcal{R}_k(\infty) = \begin{cases} S & : k \leq N \\ \mathcal{R}_k(0) & : k > N \end{cases} \tag{B.14}$$

We obtain the result by using the relationship between $\mathcal{R}_k$ and $\mathcal{L}_k$ in Eq. (4.10) and $\mathcal{R}_k(0) \ll S$. □

B.1.4 Gradient flow in the extended multilinear model

Lemma 2. *Let the extended multilinear model Eq. (5.3) be trained with gradient flow on D i.i.d samples for the setup in Section 4.2.2 (input distribution: Eq. (4.1), target function: Eq. (4.4), and MSE loss: Eq. (4.5)). For the skill index $k \leq N$ be a skill index in the multilinear model, let the feature matrix $\Phi \in \mathbb{R}^{D_c \times d_k}$ for the k^{th} skill be*

$$\Phi_{lj} = e_{k,l}(i^{(j)} = k, x^{(j)}), \tag{B.15}$$

and SVD on $\Phi = USV$. Assuming that the system is overparametrized ($d_k < D_c$), the gradient on $\vec{B}_k \in \mathbb{R}^{D_c}$ ($[B_{k,1}, \cdots, B_{k,D_c}]$) is contained in the column space of semi-orthogonal matrix $U \in \mathbb{R}^{D_c \times d_k}$:

$$UU^T \frac{d\vec{B}_k}{dt} = \frac{d\vec{B}_k}{dt}. \tag{B.16}$$

Proof. Similar to Lemma 1, the total loss can be decomposed into each skill such that the dynamics of $B_{k,l}$ relies only on d_k observations of the k^{th} skill:

$$\mathcal{L}_D = \frac{1}{2D} \sum_{k=1}^{n_s} \sum_{j=1}^{D} \left(f^*(i^{(j)}, x^{(j)}) - f(i^{(j)}, x^{(j)}) \right)^2 \tag{B.17}$$

$$= \frac{1}{2D} \sum_{k=1}^{n_s} \sum_{j_k=1}^{d_k} \left(S g_k(k, x^{(j_k)}) - \sum_{l=1}^{D_c} a_k B_{k,l} e_{k,l}(k, x^{(j_k)}) \right)^2 \tag{B.18}$$

$$= \frac{1}{2D} \sum_{k=1}^{n_s} \sum_{j_k=1}^{d_k} \left(\sum_{l=1}^{D_c} (\frac{S}{\sqrt{D_c}} - a_k B_{k,l}) e_{k,l}(k, x^{(j_k)}) \right)^2. \tag{B.19}$$

In the second line, we used Eq. (5.4) that $e_{k,l}(I \neq k, x) = 0$ and the orthogonality of g_k (Eq. (4.3)). In the last line, we used Eq. (5.4) that $g_k = {D_c}^{-1/2} \sum_l e_{k,l}$. We can find the gradient descent equation of $B_{k,l}$ from Eq. (B.19):

$$\frac{dB_{k,l}}{dt} = -\eta \sum_{j=1}^{d_k} \frac{1}{D} \left[a_k e_{k,l}(k, x^{(j)}) \sum_{l'=1}^{D_c} (a_k B_{k,l'} - \frac{S}{\sqrt{D_c}}) e_{k,l'}(k, x^{(j)}) \right], \tag{B.20}$$

which in the matrix form is

$$\frac{d\vec{B}_k}{dt} = -\frac{\eta a_k}{D}\Phi\Phi^T\left(B_k a_k - \frac{\vec{S}}{\sqrt{D_c}}\right), \tag{B.21}$$

where D_c dimensional vectors $\vec{B}_k$ and $\vec{S}$ are $[B_{k,1}, \cdots, B_{k,D_c}]$ and $[S, \cdots, S]$ respectively. It illustrates that $\frac{dB_k}{dt}$ is contained in $\mathrm{im}(\Phi)$, which is contained in $\mathrm{im}(U)$ (immediate from $\Phi = USV$). As $UU^T(Uz) = U(U^TU)z = Uz$, UU^T acts as identity on image of U, showing that $UU^T\frac{d\vec{B}_k}{dt} = \frac{d\vec{B}_k}{dt}$.

□

B.1.5 Conserved quantity of extended multilinear model

Lemma 3. *In the setup of Lemma 2, $a_k^2 - |\vec{B}_k|^2$ is conserved over time.*

Proof. We can use Eq. (B.19) to find the equation for a_k:

$$\frac{da_k}{dt} = -\eta\sum_{j=1}^{d_k}\frac{1}{D}\left[\sum_{l=1}B_{k,l}e_{k,l}(k, x^{(j)})\sum_{l'=1}^{D_c}(a_k B_{k,l'} - \frac{S}{\sqrt{D_c}})e_{k,l'}(k, x^{(j)})\right], \tag{B.22}$$

which in the matrix form is

$$\frac{da_k}{dt} = -\frac{\eta}{D}\vec{B}_k^T\Phi\Phi^T\left(\vec{B}_k a_k - \frac{\vec{S}}{\sqrt{D_c}}\right). \tag{B.23}$$

Then

$$a_k\frac{da_k}{dt} = -\frac{\eta a_k}{D}\vec{B}_k^T\Phi\Phi^T\left(\vec{B}_k a_k - \frac{\vec{S}}{\sqrt{D_c}}\right) \tag{B.24}$$

$$= \vec{B}_k^T\frac{d\vec{B}_k}{dt}, \tag{B.25}$$

where we used Eq. (B.21) in the last line. Thus, $a_k^2 - |\vec{B}_k|^2$ is conserved during the dynamics. □

B.1.6 Multi shot learner

Proposition B.1.1. *Let the setup be as that in Lemma 2. Suppose that $a_k(T)$ is eventually bounded away from zero, i.e. there exists $\delta > 0$ and $M > 0$ such that $T > M \Rightarrow |a_k(T)| \geq \delta$. Also assume that $U^\perp$-component of $\vec{B}_k(0)a_k(0)$ and $\vec{B}_k(0)S$ is negligible. Then the skill strength $\mathcal{R}_k$ is*

$$\mathcal{R}_k(\infty) = \begin{cases} d_k < D_c: & S\left(1 - \sqrt{1 - d_k/D_c}\right) \\ d_k \geq D_c: & S \end{cases} \tag{B.26}$$

Proof. First, we show that $\frac{d\mathcal{L}_k}{dt} \leq 0$ with equality only holding when the gradient is 0.

$$\frac{d\mathcal{L}_k}{dt} = \frac{d\mathcal{L}_k}{da_k}\frac{da_k}{dt} + \sum_i^{D_c} \frac{d\mathcal{L}_k}{dB_{k,i}}\frac{dB_{k,i}}{dt} \tag{B.27}$$

$$= -\eta \frac{d_k}{D}\left(\frac{d\mathcal{L}_k}{da_k}\frac{d\mathcal{L}_k}{da_k} + \sum_i^{D_c} \frac{d\mathcal{L}_k}{dB_{k,i}}\frac{d\mathcal{L}_k}{dB_{k,i}}\right) \leq 0. \tag{B.28}$$

The equality holds only when

$$\frac{d\mathcal{L}_k}{da_k} = \frac{da_k}{dt} = 0 \quad \text{and} \quad \frac{d\mathcal{L}_k}{dB_{k,i}} = \frac{dB_{k,i}}{dt} = 0\,. \tag{B.29}$$

We show that both a_k and $\vec{B}_k$ are bounded throughout whole dynamics. As

$$\mathcal{L}_k = \left|\Phi\left(\vec{B}_k a_k - \frac{\vec{S}}{\sqrt{D_c}}\right)\right|^2 \geq \sigma^2 \left|UU^T\left(\vec{B}_k a_k - \frac{\vec{S}}{\sqrt{D_c}}\right)\right|^2 \tag{B.30}$$

for σ^2 the smallest nonzero eigenvalue of $\Phi\Phi^T$, where $\Phi = USV$. This shows that

$$UU^T\left(\vec{B}_k a_k - \frac{\vec{S}}{\sqrt{D_c}}\right) \tag{B.31}$$

is bounded, so $UU^T\vec{B}_k a_k$ is bounded. Meanwhile, in Lemma 2, we showed that $(1-UU^T)\frac{d\vec{B}_k}{dt} = 0$, so $(1-UU^T)\vec{B}_k a_k$ is bounded. This shows that $\vec{B}_k a_k$ is bounded. As $a_k^2 - |\vec{B}_k|^2$ is constant (Lemma 3) and $|\vec{B}_k a_k| = |a_k||\vec{B}_k|$ is bounded, this shows that both a_k and $|\vec{B}_k|$ are bounded.

The dynamics moving in some bounded region always has at least one accumulation point, which we denote as p. We will show that $\frac{d\mathcal{L}_k}{dt} = 0$ at p. The function $\mathcal{L}_k(t)$ in t is a decreasing differential function which is positive. We also note that $\frac{d^2\mathcal{L}_k(t)}{dt^2}$ is globally bounded, as it can be expressed in polynomial expression in $(a_k, \vec{B}_k)$ and we showed that $(a_k(t), \vec{B}_k(t))$ is bounded. From Taylor's theorem, one can obtain

$$\inf \mathcal{L}_k(t) \leq \mathcal{L}_k(t_1 + t_2) \leq \mathcal{L}_k(t_1) + t_2\frac{d\mathcal{L}_k}{dt}(t_1) + \frac{t_2^2}{2}M \tag{B.32}$$

for $M = \sup|\frac{d^2\mathcal{L}_k(t)}{dt^2}|$. Choosing $t_2 = -\frac{d\mathcal{L}_k}{dt}(t_1)M^{-1}$ shows that

$$\mathcal{L}_k(t_1) - \frac{1}{2M}\left(\frac{d\mathcal{L}_k}{dt}(t_1)\right)^2 \geq \inf \mathcal{L}_k(t) \tag{B.33}$$

and letting $t_1 \to \infty$ here gives

$$\lim_{t_1\to\infty} \frac{1}{2M}\left(\frac{d\mathcal{L}_k}{dt}(t_1)\right)^2 \leq \lim_{t_1\to\infty}(\mathcal{L}_k(t_1) - \inf \mathcal{L}_k(t)) = 0 \tag{B.34}$$

so $\frac{d\mathcal{L}_k}{dt} \to 0$ as $t \to \infty$. Meanwhile, as p is accumulation point of (a_k, B_k), $\frac{d\mathcal{L}_k}{dt}(p)$ is accumulation point of $\frac{d\mathcal{L}_k}{dt}(a_k(t), \vec{B}_k(t))$. As $\lim_{t\to\infty} \frac{d\mathcal{L}_k}{dt}(t) = 0$, the only accumulation point of $\frac{d\mathcal{L}_k}{dt}(t)$ is zero, which shows that $\frac{d\mathcal{L}_k}{dt}(p) = 0$.

We have seen that $a_k^2 - |\vec{B}_k|^2$ and $(I - UU^T)\vec{B}_k$ are conserved in our dynamics. A quantity conserved in dynamics should also be conserved at p, so $p = (a, \vec{B})$ should satisfy the following conditions:

- $a^2 - |\vec{B}|^2 = a_k(0)^2 - |\vec{B}_k(0)|^2$ (Lemma 3);
- $(I - UU^T)\vec{B} = (I - UU^T)\vec{B}_k(0)$ (Lemma 2);
- $\frac{d\mathcal{L}_k}{dt}(a, \vec{B}) = 0$, or equivalently the gradient is 0 at p.

We will solve for p satisfying those three conditions. The third condition is equivalent to that

$$aUU^T\left(\vec{B}a - \frac{\vec{S}}{\sqrt{D_c}}\right) = 0. \tag{B.35}$$

As $a_k(T)$ is eventually bounded away from zero, we have $a \neq 0$, so

$$UU^T\left(\vec{B}a - \frac{\vec{S}}{\sqrt{D_c}}\right) = 0. \tag{B.36}$$

It follows that

$$\vec{B} = UU^T\vec{B} + (I - UU^T)\vec{B} = UU^T\frac{\vec{S}}{\sqrt{D_c}}a^{-1} + (I - UU^T)\vec{B}_k(0) \tag{B.37}$$

and substituting to first condition gives

$$a^2 - \frac{1}{a^2}\left|UU^T\frac{\vec{S}}{\sqrt{D_c}}\right|^2 - \left|(I - UU^T)\vec{B}_k(0)\right|^2 = a_k(0)^2 - |\vec{B}_k(0)|^2. \tag{B.38}$$

This is equivalent to a quadratic equation in a^2, and has a following solution of

$$a^2 = \sqrt{\left|UU^T\frac{\vec{S}}{\sqrt{D_c}}\right|^2 + \frac{(a_k(0)^2 - |UU^T\vec{B}_k(0)|^2)^2}{4}} + \frac{a_k(0)^2 - |UU^T\vec{B}_k(0)|^2}{2}. \tag{B.39}$$

This shows that there are two candidates for p, with a given as two square roots of Eq. (B.39) and B determined from a by Eq. (B.37). It is impossible for $\mathcal{L}_k(t)$ to have accumulation points both in regions $a > 0$ and $a < 0$, as it would imply $a_k(t) = 0$ happens infinitely many often, contradicting that a_k is eventually bounded away from zero. Thus it follows that $\mathcal{L}_k(t)$ can only have one accumulation point. As dynamics having unique accumulation point should converge, it follows that

$$(a, \vec{B}) = (a_k(\infty), \vec{B}_k(\infty)). \tag{B.40}$$

One can check that the $U^{\perp}$-component of $\vec{B}_k(\infty)a_k(\infty)$ is given as

$$(I - UU^T)\vec{B}_k(\infty)a_k(\infty) = (I - UU^T)\vec{B}_k(0)a_k(0) \tag{B.41}$$

and this is bounded by $|(1 - UU^T)B_k(0)|(S + a_k(0))$, so by our assumption this is negligible. Thus, we find that $\vec{B}_k(\infty)a_k(\infty)$ is the pseudo-inverse solution, which is also found by the linear model with $e_{k,l}$ as basis functions. We can calculate $\mathcal{L}_k(\infty)$ using the result from kernel (linear) regression [16, 20, 22, 99, 24] (for a summary, see tables 1 and 2 in appendix A of [24]). Using the terminology in table 1 of [24], the sample size is d_k; the number of parameters is D_c; ridge and noise are absent; the eigenfunctions are $[e_{k,1}, \cdots, e_{k,D_c}]$; the eigen coefficients are $\mathbf{E}_X[e_{k,i}(x)Sg_k(x)] = SD_c^{-1/2}$ (Eq. (5.4)); eigenvalues are uniform; the learnability is d_k/D_c for all i; and the overfitting coefficient is $(1 - d_k/D_c)^{-1}$. Taking into account that we have halved the MSE loss (Eq. (4.5)), the test loss is

$$\mathcal{L}_k(\infty) = \frac{S^2}{2}\left(1 - \frac{d_k}{D_c}\right). \tag{B.42}$$

We obtain the result by using Eq. (4.10). □

B.1.7 Multiple basis functions for a skill

Proposition B.1.2. *Let the extended multilinear model Eq. (5.6) be trained with gradient flow on $D \to \infty$ i.i.d samples for the setup in Section 4.2.3 with $n_s \to \infty$ (input distribution: Eq. (4.1), target function: Eq. (4.4), and MSE loss: Eq. (4.5),*

initialization: that of Proposition B.1.1). For a model with the following finite N basis functions

$$[e_{1,1},\ \cdots,\ e_{1,N_c},\ e_{2,1},\ \cdots,\ e_{q,r}], \tag{B.43}$$

where quotient $q = \lfloor (N-1)/N_c \rfloor + 1$ *and remainder* r *is such that* $(q-1)N_c + r = N$. *The skill strength at* $T \to \infty$ *becomes*

$$\mathcal{R}_k(\infty) = \begin{cases} k > q: & 0 \\ k = q: & S\frac{r}{N_c} \\ k < q: & S. \end{cases} \tag{B.44}$$

Proof. Because we have $D \to \infty$ and because $[e_{k,1}, \cdots e_{k,N_c}]$ can express g_k (Eq. (5.8)), it is trivial to show that $\mathcal{R}_k(\infty) = S$ for $k < q$. For $k = q$, the gradient descent dynamics (Eq. (B.21)) leads to

$$\frac{d\vec{B}_k}{dt} = -\frac{\eta a_k}{D}\Phi\Phi^T\left(\vec{B}_k a_k - \frac{\vec{S}}{\sqrt{N_c}}\right) \tag{B.45}$$

where the matrix $\Phi \in \mathbb{R}^{r \times d_k}$ and vector $\vec{B}_k \in \mathbb{R}^r$ are the feature matrix(Eq. (B.15)) and parameters for the k^{th} skill respectively. As $D \to \infty$, the matrix $\Phi\Phi^T$ becomes a rank r identity matrix scaled by the frequency of the skill:

$$\lim_{D\to\infty} \frac{1}{D}(\Phi\Phi^T)_{ll'} = \mathbf{E}_{I,X}\left[e_{k,l}(k,X)e_{k,l'}(k,X)\right] = \mathcal{P}(k)\delta_{l,l'}. \tag{B.46}$$

Plugging in $\Phi\Phi^T$,

$$\frac{dB_{k,l}}{dt} = -\eta\mathcal{P}(k)a_k\left(B_{k,l}a_k - \frac{S}{\sqrt{N_c}}\right). \tag{B.47}$$

Assuming the initialization in Proposition B.1.1, we can show that $a_k(\infty)B_{k,l}(\infty) = S/\sqrt{N_c}$ for $l \le r$. From Eq. (4.7), the skill strength $\mathcal{R}_k(\infty)$ is

$$\mathcal{R}_k(\infty) = \sum_{l=1}^{r} \frac{S}{\sqrt{N_c}}\mathbf{E}_X\left[e_{k,l}(k,X)g_k(k,X)\right] \tag{B.48}$$

$$= S\frac{r}{N_c}, \tag{B.49}$$

where we used Eq. (5.8) for the linear correlation between $e_{k,l}$ and g_k. □

C

Additional discussion for chapters 2 and 3

C.1 Finite data correction for slow decay of skill frequency.

In Fig. C.1, we observe that our model with $\alpha = 0.1$ deviates from the expected power-law with exponent $-\alpha/(\alpha+1)$. The deviation can be explained by the antiderivative term in Eq. (4.23):

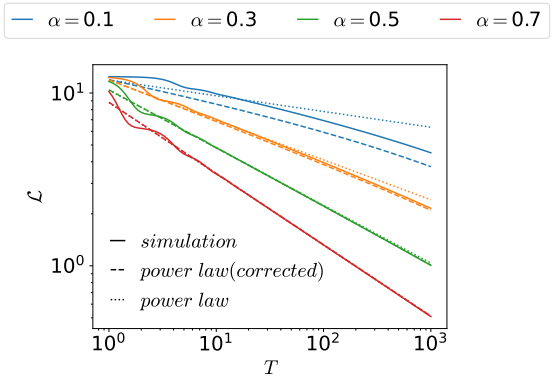


Figure C.1: Scaling law and corrected predictions. A simulation of our multilinear model with $N = 50,000$ (solid), a scaling law with exponent $-\alpha/(\alpha+1)$ (dotted), and a corrected scaling law considering finite N (dashed, Eq. (C.2)). The finite N corrected scaling law better predicts the dynamics, especially for smaller α.

$$\lim_{N\to\infty}\left[\frac{1}{2(\alpha+1)}\frac{S^2A}{\left(1+\frac{1}{S/\mathcal{R}_k(0)-1}e^{2\eta SAk^{-(\alpha+1)}T}\right)^2}\frac{k^{-\alpha}}{T}\right]_1^N = \lim_{N\to\infty}\left(\mathcal{O}\left(N^{-\alpha}\frac{1}{T}\right)-\mathcal{O}\left(\frac{1}{Te^T}\right)\right). \tag{C.1}$$

The second term ($k = 1$) goes to 0 faster than $\mathcal{O}(T^{-1})$ for sufficiently larger T but the first term ($k = N$) may not decay fast enough for finite N and sufficiently small α. For example, $N = 50,000$ and $\alpha = 0.1$ leads to $N^{-\alpha} \approx 0.3$, which is not negligibly small.

Assuming finite N and small α such that the first term in Eq. (C.1) is non-negligible, we can rewrite Eq. (4.23) as

$$\frac{d\mathcal{L}}{dT} \approx -\frac{\alpha}{(\alpha+1)}\frac{\mathcal{L}+\mathcal{L}_C}{T}, \qquad \mathcal{L}_C \approx S^2AN^{-\alpha}/2\alpha, \tag{C.2}$$

where we assumed a small initialization $S/\mathcal{R}_k(0) \gg 1$ and sufficiently large number of parameters $N^{\alpha+1} \gg T$ to approximate $\mathcal{L}_C$. Because the total loss at initialization is $\mathcal{L}(0) = S^2/2$, $\mathcal{L}_C$ is non-negligible compared to the loss for sufficiently small α. Thus considering $\mathcal{L}_C$, we obtain the corrected power-law which better approximates the time scaling law (dashed lines in Fig. C.1).

C.2 Connection to linear models.

In Section 4.3, we have shown how the scaling laws follow from the basis functions g_k that decouple the loss. To analyze the role of g_k, we can ask whether a simpler linear model with g_k as basis functions (Eq. (C.3)) also recovers the scaling laws. The answer is yes and we outline how a linear model can recover all scaling laws. In addition, we also outline how extended linear models – extended similar to Section 5.2 such that skills are decoupled – can recover all emergence behaviors shown in Chapter 5 except the time emergence.

By replacing a_kb_k with w_k, we obtain the linear model with skill basis functions:

$$f_T(i,x;w) = \sum_{k=1}^{N} w_k(T)g_k(i,x). \tag{C.3}$$

The dynamics of the linear model under gradient flow is

$$\mathcal{R}_k(T) = w_k(T) = S(1 - e^{-\eta \frac{d_k}{D} T}), \tag{C.4}$$

where we assumed $w_k(0) = 0$. The linear model follows an exponential saturation of the skill strength in contrast to the sigmoidal saturation of the multilinear model (Fig. 3.3).

Nevertheless, the linear model Eq. (C.4) results in the same scaling laws in Section 4.3. For the time scaling law, we recover the relationship between $d\mathcal{L}_k/dT$ and $d\mathcal{L}_k/dk$ in Section 4.3.1 because $\mathcal{R}_k(T)$ is a function of $\frac{d_k}{D}T$ only (where $d_k/D = \mathcal{P}_s(k)$ for $D \to \infty$). For the data scaling law, we recover Corollary B.1.1 because each w_k (i.e. $\mathcal{R}_k$) is decoupled. For the parameter scaling law, we recover Corollary B.1.2 trivially as the linear model shares the same basis functions.

The data and parameter emergence in Section 5.2 can be obtained from the linear model in Eq. (C.3) if we extend the model analogous to Eqs. (5.3) and (5.6). For example, we can extend the model for data emergence as

$$f_T(i, x; W) = \sum_{k=1}^{N} \sum_{l=1}^{D_c} W_{k,l}(T) e_{k,l}(i, x), \tag{C.5}$$

where the matrix $W \in \mathbb{R}^{N \times D_c}$ is an extension of $w \in \mathbb{R}^N$ in Eq. (C.3), D_c is a fixed scalar, and $e_{k,l}(i, x) : \{0, 1\}^{n_s + n_b} \to \mathbb{R}$ are functions with the following properties:

$$\mathbf{E}_{X|I=k} \left[e_{k,l} e_{k,l'} \right] = \delta_{ll'}, \quad e_{k,l}(I \neq k, x) = 0, \quad \sum_{l=1}^{D_c} \frac{1}{\sqrt{D_c}} e_{k,l} = g_k. \tag{C.6}$$

The equivalence can be shown by Lemma 2 which states that the multilinear model finds the minimum norm solution: the solution that the linear model finds in a ridgeless regression setup.

Thus, for our setup, the basis functions play a critical role in the scaling laws and data/parameter emergences. The choice of basis functions, also known as the task-model alignment (see [16, 24]), determines the linear model's scaling laws and emergence behaviors. See Bordelon et al. [91] for a study of the scaling laws in linear models.

C.3 Details of the 2-layer MLP

In this section, we detail the methods used to train a 2-layer fully connected neural network (MLP) with ReLU activation. All parameters of the MLP were initialized with a Gaussian distribution with a standard deviation of 0.001. The input dimension of the model was $n_s + n_b = 5 + 32$ where n_s is the length of control bits (number of skills) and n_b is the length of the skill bits. Each skill has $m = 3$ mutually exclusive sparse bits that are used to express the skill function. The target scale was $S = 5$. The model was trained with SGD without momentum and no weight decay (the exception is the parameter emergence experiment where Adam with learning rate 0.001 and weight decay of 5×10^{-5} was used to escape the local minima).[1] For the data emergence experiment, the learning rate was halved every $50,000$ step.

The skill strength $\mathcal{R}_k(T)$ (Eq. (4.7)) was measured using $20,000$ i.i.d samples from the k^{th} skill.[2] For the time emergence, the skill strengths were measured every 50 steps, while for other experiments, they were measured after training. To mimic the infinite parameter $N \to \infty$, we used the model of width 1000 (for the hidden layer). To mimic the infinite time $T \to \infty$, we trained for 5×10^5 steps (3×10^4 steps for time emergence) where each step had the batch size of 4000 (2000 for the data emergence experiment). To mimic $D \to \infty$, we sampled new data points for every batch. The details are given in the following table.

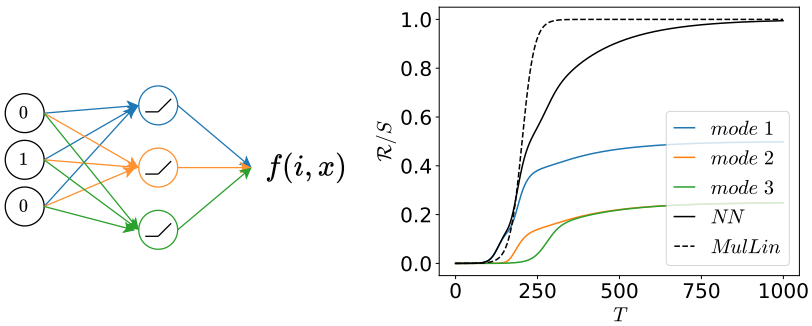
Name	Values
width	1000
learning rate	0.05
initialization standard deviation	0.01
activation	ReLU
batch size	4000
steps	500,000
target scale	5
number of skill bits	32
number of skills	5

[1] We are free to choose any optimizer as long as it preserves the order in which the skills are learned. Additionally, the parameter emergence experiment uses infinite data; we expect the same solution for Adam and SGD.

[2] Note that except the data scaling law experiment, the training set size is infinite.

C.4 Time emergence example in NN

In this section, we discuss an example for the time emergence case (Fig. 5.1(a)) in which the saturation of skill in an NN consists of multiple saturating 'modes' as in Fig. C.2.



(a) neuron modes for a parity function

(b) mode/skill strength

Figure C.2: Modes in NN. A 2-layer MLP with ReLU activations with a width of 3 and weight sharing (Eq. (C.9)) is trained to fit the parity function. **(a):** The skill strength $\mathcal{R}$, because of the last layer's linearity, can be decomposed into skill strength from each hidden neuron or each 'mode' (shown in different colors, Eq. (C.14)). **(b):** The skill strength for each mode follows a near-sigmoidal curve with different emergent/saturation times (colors) whose sum results in the total skill strength (solid black). Note that different saturation times of each mode result in a deviation from the prediction of the multilinear model with $\mathcal{B}^2 = 1/3$ (dashed black).

Task. We assume an input $X \in \mathbb{R}^{3\times 8}$ (note that we are not using X as a random variable) that is all 8 possible inputs for bits with dimension 3. The target Y is the parity function scaled by S.

$$X = \begin{pmatrix} 0&0&0&0&1&1&1&1 \\ 0&0&1&1&0&0&1&1 \\ 0&1&0&1&0&1&0&1 \end{pmatrix}, \qquad Y = \begin{pmatrix} S & -S & -S & S & -S & S & S & -S \end{pmatrix} \tag{C.7}$$

NN. We assume a 2-layer width 3 NN with ReLU activation with the input dimension 3 (Fig. C.2(a)). The NN has 16 parameters, but to simplify the argument, we use weight sharing so NN has only 4 parameters:

$$f(x;\alpha,\beta,\gamma,c) = w^T\sigma(Wx+b)+c \tag{C.8}$$

where σ is the ReLU activation and W, b, w are

$$W = \begin{pmatrix} -\alpha & \alpha & -\alpha \\ -\beta & \beta & -\beta \\ \gamma & -\gamma & \gamma \end{pmatrix}, \qquad b = \begin{pmatrix} 0 \\ \beta \\ -\gamma \end{pmatrix}, \qquad w = \begin{pmatrix} -2\alpha \\ \beta \\ \gamma \end{pmatrix}. \tag{C.9}$$

Modes. It is easy to see that $\alpha = \beta = \gamma = \sqrt{2S}$ and $c = -S$ leads to the target parity function. We note that one parameter except c (i.e. α, β, γ) maps to one neuron or a mode (colors in Fig. C.2(a)). We define the first mode $f^{(1)}$ as

$$f^{(1)}(x) = w_1 \sigma(W_1^T x + b_1) = -2\alpha^2 \sigma(x_2 - x_1 - x_3) \tag{C.10}$$

$$= -2\alpha^2 h_1(x), \qquad h_1(x) := \sigma(x_2 - x_1 - x_3), \tag{C.11}$$

where w_1, b_1 are the first entry of w, b respectively and W_1 is the first row of W. Note that $f^{(1)}(x)$ takes a form similar to the multilinear model (Eq. (4.9)) but with h_1 as the respective basis. We define $f^{(2)}, f^{(3)}$ similarly, and the sum of modes becomes the NN:

$$f(x) = \sum_{q=1}^{3} f^{(i)}(x) + c, \tag{C.12}$$

which resembles the multilinear model with different skills.

Mode strength. Analogous to the skill strength in Eq. (4.7), we define mode q's strength $\mathcal{R}^{(q)}$ as

$$\mathcal{R}^{(q)} = \frac{1}{8S^2} Y^T f^{(q)}(X), \tag{C.13}$$

where $f^{(q)}(X) = [f^{(q)}(X_1), \cdots, f^{(q)}(X_8)]$ and X_j are the j^{th} column of X. By the linearity of the expectation,

$$\mathcal{R} = \sum_{q=1}^{3} \mathcal{R}^{(q)}. \tag{C.14}$$

Note that constant c always has zero correlation (inner product) to the target (Y).

Analysis. The dynamics of each mode $\mathcal{R}^{(q)}(x)$ differs from that of the multilinear model (Eq. (4.11)) because $h_q(x)$ often depends on the parameter, and the dynamics are no longer decoupled among each mode. Nevertheless, each mode follows a sigmoid-like growth (Fig. C.2(b)). We note that each mode has a different saturation time scale or is updated at different frequencies. A mode with a longer time scale leads to a longer 'tail' of saturation as discussed in the main text.

Update frequency. Because of the non-linearity, each mode differs in the gradients it receives. We can explicitly calculate the gradient for each parameter as:

$$\frac{d\alpha^2}{dt} = 2\eta\alpha^2(-S - (-2\alpha^2 + 2\beta^2 + c)) \tag{C.15}$$

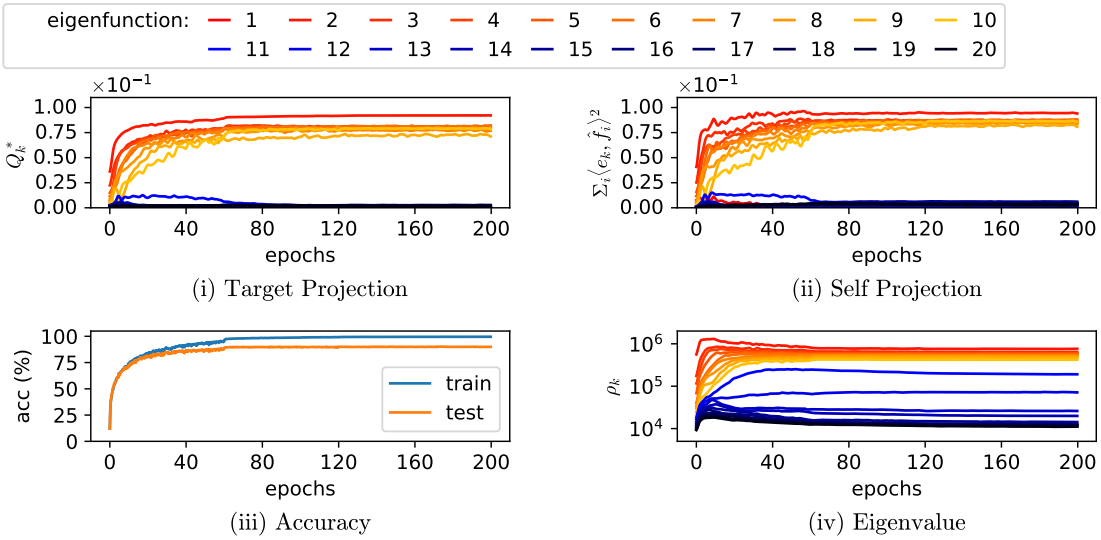
$$\frac{d\beta^2}{dt} = -\eta\beta^2(S - (-2\alpha^2 + 5\beta^2 + 5c)) \tag{C.16}$$

$$\frac{d\gamma^2}{dt} = -\eta\gamma^2(S - (\gamma^2 + c)) \tag{C.17}$$

$$\frac{dc}{dt} = -\eta(2\alpha^2 - 5\beta^2 - \gamma^2 - 8c). \tag{C.18}$$

We immediately notice that c will grow the fastest for small initialization ($\alpha, \beta, \gamma, c \ll 1$) because it saturates exponentially while other parameters saturate sigmoidally. Considering that S is always the largest term and c saturate to S quickly, we notice that the saturation is in the order of α^2 ($\approx 2S + 2c \approx 4S$), β^2 ($\approx -S + 5c \approx 4S$), and $\gamma^2 (\approx 2S)$. We observe that our crude approximation holds in Fig. C.2(b): the first (α) and the second (β) modes saturate at similar timescale, while the third mode (γ) requires approximately twice the time for saturation.

C.5 Additional plots



(i) Target Projection (ii) Self Projection

(iii) Accuracy (iv) Eigenvalue

(a) ResNet18, CIFAR10

Figure C.3: Temporal evolution of the ResNet18 features for slower learning rate In (a) for ResNet18 on CIFAR10 and (b) for an FCN on CIFAR10, we plot, as a function of epoch and for the first 20 features (i) the quality of the features, (ii) a related projection of each feature onto the learned function, (iii) the test and training accuracies for the full DNNs, and (iv) the eigenvalue spectra. A Gaussian filter has been used for (i,ii,iv) to smooth the curve. We observe that the dynamics for (a) are in the MFR and are primarily driven by feature learning. By contrast, the dynamics for (b) are in the EFR. There is an initial regime of feature learning, but this slows down and further lowering of the loss is achieved primarily by coefficient learning, as evidenced in (ii).

D

Proofs related to Neural Collapse

In this section, we state the relevant terms and the four NC conditions given in [58]. Please note that the current study assumes column-vectors for all vectors (e.g. $\mu_i \in \mathbb{R}^{p\times 1}$). The last layer can be defined in the following manner

$$\hat{f}_i(x) = w_i^T \Phi(x) + b_i \tag{D.1}$$

where $w_i \in \mathbb{R}^{p\times 1}$ is the parameter of the last that projects the feature onto the i^{th} entry of the output function. $b_i \in \mathbb{R}$ is the corresponding bias term. The feature class mean or the mean of features with label i is

$$\mu_i = \frac{C}{n} \sum_{x^{(j)}\in A_i} \Phi(x^{(j)}). \tag{D.2}$$

The global mean is

$$\mu_g = \frac{1}{C} \sum_{i}^{C} \mu_i. \tag{D.3}$$

The feature within-class covariance $\Sigma_W \in \mathbb{R}^{p\times p}$ is

$$\Sigma_W := \frac{1}{n} \sum_{i}^{C} \sum_{x^{(j)}\in A_i} (h_i(x^{(j)}) - \mu_i)(h_i(x^{(j)}) - \mu_i)^T. \tag{D.4}$$

The feature between-class covariance $\Sigma_b \in \mathbb{R}^{p\times p}$ is

$$\Sigma_b := \frac{1}{C} \sum_{i}^{C} (\mu_i - \mu_g)(\mu_i - \mu_g)^T. \tag{D.5}$$

The four NC conditions are the following:

1. **Within class variance tends to 0**

$$\Sigma_W \Sigma_b^T \to 0 \tag{D.6}$$

2. **Convergence to simplex ETF**

$$\frac{(\mu_i - \mu_g)^T(\mu_j - \mu_g)}{\|(\mu_i - \mu_g)\|_2 \|(\mu_j - \mu_g)\|_2} \to \frac{C\delta_{ij} - 1}{C - 1} \tag{D.7}$$

3. **Convergence to self duality**

$$\frac{w_i}{\|w_i\|_2} - \frac{\mu_i - \mu_g}{\|(\mu_i - \mu_g)\|_2} \to 0 \tag{D.8}$$

4. **Simplification to nearest class center**

$$\arg\max_i w_i \Phi(x) + b_i \to \arg\min_i \|\Phi(x) - \mu_i\|_2 \tag{D.9}$$

Proposition 2.5.3. *If $\mathcal{T}_{MP}$ is MP-operator and $\hat{f}_i(x) = \mathbf{1}_{A_i}(x)$ for a balanced partition $\{A_1, \cdots, A_c\}$ of $\mathcal{X}$, then*

$$\langle C\hat{f}_i | \mathcal{T}_{MP}[C\hat{f}_j - \mathbf{1}] \rangle = a_2(\delta_{ij} - C^{-1}) \qquad \langle \mathbf{1} | \mathcal{T}_{MP}[C\hat{f}_j - \mathbf{1}] \rangle = 0, \tag{2.19}$$

where C is the number of classes.

Proof. By the definition of indicator functions of equal partitions,

$$\langle \hat{f}_i | \hat{f}_j \rangle = \frac{1}{C}\delta_{ij}, \qquad \sum_i^C \hat{f}_i = \mathbf{1}, \qquad \langle \mathbf{1} | \hat{f}_j \rangle = \frac{1}{C} \tag{D.10}$$

Since $C\hat{f}_j - \mathbf{1}$ is orthogonal to $\mathbf{1}$,

$$T_{MP}[C\hat{f}_j - \mathbf{1}] = a_2(C\hat{f}_j - \mathbf{1}) \tag{D.11}$$

where we used the definition of MP operators Eq. (2.8). Then from Eq. (D.10) and Eq. (D.11),

$$\langle C\mathbf{f}_i | T_{MP}[C\hat{f}_j - \mathbf{1}] \rangle = a_2 \langle C\mathbf{f}_i | C\hat{f}_j - \mathbf{1} \rangle = a_2(C\delta_{ij} - \mathbf{1}) \tag{D.12}$$

Likewise,

$$\langle \mathbf{1} | T_{MP}[C\hat{f}_j - \mathbf{1}] \rangle = a_2 \langle \mathbf{1} | C\hat{f}_j - \mathbf{1} \rangle = 0. \tag{D.13}$$

□

Proposition 2.5.2. *Let $\mathcal{T}_{MP}$ be the MP-operator on data taken from the empirical distribution of the training set, and $\hat{f}$ the empirical output function of balanced C-way classification task; The following NC2 condition holds:*

$$(\mu_i - \mu_g)^T(\mu_j - \mu_g) \propto \delta_{ij} - \frac{1}{C} \tag{2.11}$$

Proof. From the translation of class mean function Eq. (2.15),

$$\begin{aligned}
\mu_i^T \mu_j &= C^2 \left(\int_\chi \hat{f}_i(x)\Phi(x)q(x)dx\right)^T \int_\chi \hat{f}_j(x')\Phi(x')q(x')dx' && \text{(D.14)}\\
&= C^2 \int_\chi \hat{f}_i(x)q(x) \int_\chi \Phi(x)^T\Phi(x')\hat{f}_j(x')q(x')dx'dx && \text{(D.15)}\\
&= C^2 \int_\chi \hat{f}_i(x)q(x)T[\hat{f}_j](x)dx && \text{(D.16)}\\
&= C^2 \langle \hat{f}_i, T[\hat{f}_j]\rangle . && \text{(D.17)}
\end{aligned}$$

Likewise, using Eq. (2.14), we get

$$\begin{aligned}
\mu_i^T \mu_g &= C \left(\int_\chi \hat{f}_i(x)\Phi(x)q(x)dx\right)^T \int_\chi \Phi(x')q(x')dx' && \text{(D.18)}\\
&= C \int_\chi \hat{f}_i(x)q(x) \int_\chi \Phi(x)^T\Phi(x')q(x')dx'dx && \text{(D.19)}\\
&= C \int_\chi \hat{f}_i(x)q(x)T[\mathbf{1}](x)dx && \text{(D.20)}\\
&= C \langle \hat{f}_i, T[\mathbf{1}]\rangle . && \text{(D.21)}
\end{aligned}$$

Then

$$\begin{aligned}
(\mu_i - \mu_g)^T(\mu_j - \mu_g) &= \langle C\hat{f}_i - \mathbf{1}, T[C\hat{f}_j - \mathbf{1}]\rangle && \text{(D.22)}\\
&= Ca_2(\delta_{ij} - \frac{1}{C}) && \text{(D.23)}
\end{aligned}$$

which is the NC2 condition, where we used Eq. (D.14) and Eq. (D.18) in the first line and Proposition 2.5.3 in the second line. □

Proposition 2.5.1. *Let $\mathcal{T}_{MP}$ be MP-operator, the inputs x are taken from q' the empirical distribution of the training set, and $\hat{f}$ the empirical output function of balanced C-way classification task. The following NC1 condition holds:*

$$Tr\left(\left(\sum_i \sum_{x\in A_i} (h_i(x) - \mu_i)(h_i(x) - \mu_i)^T\right)\left(\sum_j (\mu_j - \mu_g)(\mu_j - \mu_g)^T\right)^T\right) = 0 \tag{2.10}$$

Proof. Using the condition that χ is the training set, and q is its empirical distribution, and $\hat{f}$ a class indicator function, we obtain

$$NC1 = Tr\left(\sum_i \sum_{x\in A_i} (h_i(x)-\mu_i)(h_i(x)-\mu_i)^T \sum_j (\mu_j-\mu_g)(\mu_j-\mu_g)^T\right) \tag{D.24}$$

$$= \sum_{i,j}\sum_{x\in A_i} Tr\left(\left((\Phi(x)-\mu_i)^T(\mu_j-\mu_g)\right)^2\right) \tag{D.25}$$

$$= \sum_{i,j}\int_\chi \hat{f}_i(x)\left((\hat{f}_i(x)\Phi(x)-\mu_i)^T(\mu_i-\mu_g)\right)^2 q(x)dx \tag{D.26}$$

$$= \sum_{i,j}\int_\chi \hat{f}_i(x)\left((\Phi(x)-\mu_i)^T(\mu_j-\mu_g)\right)^2 q(x)dx \tag{D.27}$$

where we have used the fact that $\hat{f}_i$ are indicator functions for class i in line 2, 3, and 4. From Eq. (D.14) and Eq. (D.18),

$$\mu_i^T(\mu_j-\mu_g) = \langle C\hat{f}_i | T[C\hat{f}_j - \mathbf{1}]\rangle \tag{D.28}$$

and from Eq. (2.15) and Eq. (2.14)

$$\Phi(x)^T(\mu_j-\mu_g) = T[C\hat{f}_j - \mathbf{1}](x). \tag{D.29}$$

Applying Proposition 2.5.3 into the two equations above, we obtain

$$(\Phi(x)-\mu_i)^T(\mu_j-\mu_g) = a_2 C(\hat{f}_j(x)-\delta_{ij}). \tag{D.30}$$

Plugging Eq. (D.30) into Eq. (D.27), we obtain

$$NC1 = a_2^2C^2\sum_{i,j}\int_\chi \hat{f}_i(x)\left(\hat{f}_j(x)-\delta_{ij}\right)^2 q(x)dx \tag{D.31}$$

$$= a_2^2C^2\sum_{i,j}\int_\chi \hat{f}_i(x)(\hat{f}_j(x)(1-2\delta_{ij})+\delta_{ij})q(x)dx \tag{D.32}$$

$$= a_2^2C\sum_{i,j}(\delta_{ij}(2-2\delta_{ij})) \tag{D.33}$$

$$= 0 \tag{D.34}$$

where in the third line, we used $\langle \hat{f}_i|\hat{f}_j\rangle = \delta_{ij}C^{-1}$ and $\langle \hat{f}_i|1\rangle = C^{-1}$. □