THE IMPACT OF DATA ON GENERALIZATION IN DEEP LEARNING

Rubing Yang

A DISSERTATION

in

Applied Mathematics and Computational Science

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2026

Supervisor of Dissertation

Pratik Chaudhari, Associate Professor of Electrical and Systems Engineering

Graduate Group Chairperson

Yoichiro Mori, Calabi-Simons Chair in Mathematics and Biology

Dissertation Committee

Surbhi Goel, Magerman Term Assistant Professor of Computer and Information Science
Edgar Dobriban, Associate Professor of Statistics and Data Science
Joshua Vogelstein, Associate Professor of Biomedical Engineering, Johns Hopkins University

THE IMPACT OF DATA ON GENERALIZATION IN DEEP LEARNING

ACKNOWLEDGEMENT

First and foremost, I would like to express my deepest gratitude to my advisor, Professor Pratik Chaudhari, for his guidance and support throughout my Ph.D. journey. There were many challenges along the way, but he was always generous with his help. He always encourages me to view research from a broader perspective. His commitment to pursuing meaningful and original questions, rather than merely extending existing work, has profoundly shaped how I think and work as a researcher. I thank my committee members for their valuable feedback, especially Professor Joshua Vogelstein for his mentorship on a project in this thesis and Professor Surbhi Goel for inspiring me to expand the related-work sections.

Throughout my Ph.D., I am grateful to my friends for stimulating academic discussions, generous support, and companionship: Yajing An, Ashwin De Silva, Zirui Fan, Yansong Gao, Chris Hsu, Rohit Jena, Jiaxin Liu, Jialin Mao, Shreyam Mishra, Amrut Nadgir, Rahul Ramesh, Yifei Shao, Yan Sun, Yuhao Tan, Hanwen Wang, Rongguang Wang, Zixuan Wen, Jin Wu, Ke Xu, Shiyun Xu, Yinshuang Xu, Fanyang Yu, and many others. You guys made this journey so meaningful and memorable. Thank you for the inspiring conversations, the laughter, the encouragement during difficult moments, and all the wonderful memories we created along the way.

I would also like to thank my internship mentor, Claudia Iriondo at Genentech, for introducing me to the field of AI for biology. I am grateful for her generous guidance on seeking jobs in biotech industry, and her warm hospitality, including organizing events that helped me get to know others. I also thank the MLTD group for the stimulating discussions, insightful talks, and welcoming research environment.

Lastly, I would like to thank my parents, Xiuguang Zu and Runsheng Yang. You have always given me unconditional love and support, respected my ideas and decisions, and comforted and encouraged me whenever I felt discouraged or lost. Everything I have achieved, academically and beyond, rests on the foundation you built for me through your sacrifices and unwavering belief in me. I could not have come this far without you.

ABSTRACT

THE IMPACT OF DATA ON GENERALIZATION IN DEEP LEARNING

Rubing Yang

Pratik Chaudhari

Understanding why deep neural networks generalize remains a central problem in modern machine learning. Classical Probably Approximately Correct (PAC) framework attributes generalization to controlling hypothesis-class capacity, but this does not fully explain the strong performance of highly overparameterized networks. Architectural inductive biases provide only a partial explanation, since weakly biased models such as transformers also generalize across diverse tasks. Moreover, this behavior often persists without explicit regularization or constraints that enforce compact representations.

In this thesis, we explore the hypothesis that neural networks tend to explore only a small subset of its functional space when trained on typical data. The implicit regularization from dataset and training together guarantees the good generalization performance of deep neural networks. We also extend the learning framework to the setting in which the data distribution and optimal hypothesis changes over time.

Chapter 1 formulates the central perspective of this thesis and reviews existing approaches to understanding generalization in deep learning. In Chapter 2, we show that the spectral structure of typical datasets is inherited by trained networks, leading to a small effective dimensionality and data-dependent capacity control. In Chapter 3, we characterize the evolution of the generalization gap through contraction and perturbation dynamics, where perturbation drives the gap while contraction suppresses its accumulation, and derive an effective Gram matrix whose alignment with the initial residual predicts generalization. In Chapter 4, we introduce "prospective learning" for settings in which data distributions and optimal predictors evolve over time, and develop "prospective ERM" with theoretical guarantees and empirical validation. In Chapter 5, we outline compositional generalization as a future direction that inspires new approaches to distribution-free generalization.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF ILLUSTRATIONS

CHAPTER 1
INTRODUCTION

The generalization of deep neural networks remains a central puzzle in modern machine learning. Classical PAC theory suggests that generalization should arise from controlling the capacity of the hypothesis class, but this viewpoint alone does not explain why highly overparameterized neural networks generalize so well. Architectural inductive biases, such as those in convolutional or equivariant networks, account for part of this success, but they cannot be the whole story: transformers, which impose much weaker built-in biases, achieve state-of-the-art performance across a wide range of tasks. Moreover, deep networks often generalize well even without explicit regularization mechanisms such as sparse coding, norm penalties, or other constraints designed to enforce compact representations.

Existing approaches do not yet provide a complete explanation for this phenomenon. As discussed in Sec. 1.1, simplified models, such as linear regression, deep linear networks, and neural tangent kernel analyses, provide useful theoretical insight but rely on assumptions that do not fully capture feature learning in finite neural networks. Worst-case bounds, including VC-dimension, Rademacher-complexity, and classical PAC and PAC-Bayes analyses, are often too loose for highly overparameterized models. Non-worst-case approaches, such as norm-based bounds, stability analysis, and information-theoretic analyses, incorporate properties of the data, learned solution, or training trajectory, but are often post-hoc or depend on restrictive assumptions. These limitations motivate a framework that more directly connects data geometry and training dynamics to generalization.

In this thesis, we investigate the hypothesis that, although deep neural networks are highly expressive, training on typical data restricts them to a much smaller subset of their functional space. We argue that this restriction arises jointly from the structure of the data and the dynamics of optimization, rather than from explicit architectural or regularization constraints alone. From this perspective, generalization is governed by an implicit form of capacity control induced by the interaction between the dataset and the training process.

To analyze this viewpoint, we introduce several quantities that characterize the effective function space explored during training. In Chapter 2, we define the effective dimensionality as the number of directions in which parameter changes significantly affect the loss, providing a measure of the model capacity that is effectively accessible on a given dataset. We show that the "sloppy" structure of Hessian, as induced by the data, ensures small effective dimensionality, and hence good generalization, and this geometry can be used to further optimize generalization bound. In Chapter 3, we introduce the perturbation and contraction factors to describe the evolution of the generalization gap along training. The perturbation factor quantifies how replacing training samples changes the gradient field and drives nearby learning trajectories apart, while the contraction factor measures how the geometry of the loss landscape causes these trajectories to move closer together under training. Their simoutaneous decay guarantees controlled accumulation of generalization gap, and we constructed theory for understanding the mechanism of this. We further integrate these dynamics into an effective Gram matrix, characterizing the direction that generalization gap accumulates while training. The alignment of the non-stiff eigen space of effective Gram matrix with the initial residual aids to explaining the good generalization of deep neural networks.

Chapter 4 extends this perspective beyond the independently and identically distributed (IID) data

distribution setting by introducing "prospective learning" framework, where the data distribution and optimal predictor may change over time. We define "prospective risk", and provide distribution free conditions for "prospective ERM" being optimal. This shows that incorporating time into the hypothesis can enable learning in dynamic environments wherefails. Chapter 5 develops compositional generalization as a future direction, inspiring approches to explore the intrinsic invariances between data from different domain in a distribution free way, that ensures generalization.

1.1. Related Work

We start by introducing previous works on the topic of generalization.

1.1.1. Geometry of loss landscape and model manifold

Hessian and the FIM of deep networks have been studied to understand the local geometry of the energy landscape and the behavior of SGD, see Hochreiter and Schmidhuber (1997); Chaudhari et al. (2017); Fort and Ganguli (2019b); Sagun et al. (2016); Dinh et al. (2017), among others. Similarly, FIM has been used to study optimization (Amari, 1998; Martens and Grosse, 2016; Karakida et al., 2019), gradient diversity (Yin et al., 2018; Chaudhari and Soatto, 2018), and also generalization (Amari et al., 2002; Sun and Nielsen, 2020). A number of these works have pointed out that the Hessian and the FIM have spiky/large eigenvalues (Papyan, 2019) along with a bulk of near-zero eigenvalues (Papyan, 2018; Pennington and Bahri), and that this indicates that the energy landscape, or the prediction space, is locally flat. In Chapter 2 We focus on the decay pattern of the eigenspectra of these matrices and discover that it mirrors the decay pattern of the inputs for typical datasets. We observed a strong overlap of the subspace spanned by the stiff eigenvectors of the Hessian/FIM at initialization with the corresponding subspaces at the end of training; this is consistent with the analysis in Gur-Ari et al. (2018) and the literature on the lazy training regime (Chizat et al., 2019).

Brown et al. (2004) and Gutenkunst et al. (2007) noticed that regression models fitted to systems biology data have few stiff parameters that determine the outcome and a large number of sloppy parameters which only weakly determine the outcome. These authors have developed an elaborate geometric understanding of this phenomenon, see Transtrum et al. (2011) and references therein. While sloppiness is thought to be a universal property of parametric models (Waterfall et al., 2006), the mechanism that causes models to be sloppy has not been studied yet. In Chapter 2 we explain this by showing that the "effective dimensionality" of a model can usually be much smaller than its number of parameters.

1.1.2. Simplified models of deep networks

A direct examination of the exact solutions of deep neural networks is an appealing approach. However, due to the inherent complexity of these networks, such analyses are often intractable. As a first step, various solvable models from statistics and physics have been employed to partially characterize deep neural network behavior, offering valuable theoretical insights. Linear regression, for instance, has been widely used to explore phenomena like benign over-fitting (Bartlett et al., 2020) and double descent (Hastie et al., 2022; Belkin et al., 2020). To investigate the effects of depth, deep linear networks have served as a useful abstraction for studying multi-layer dynamics (Laurent and von Brecht, 2018). One of the most prominent frameworks in this area is the Neural Tangent Kernel (NTK) approach (Jacot et al., 2018), which models the training dynamics of deep neural networks in the infinite-width regime under a kernel-based approximation. Mallinar et al. (2022) and Belkin et al. (2020) characterized different regimes of kernel regression and analyzed its resemblance

with deep learning. The NTK method has enabled significant results on convergence (Du et al., 2019; Li and Liang, 2018) and generalization (Arora et al., 2019; Jacot et al., 2020). Bowman and Montúfar (2022) analyzed the divergence of finite-width neural networks with NTK regime in different eigenspaces. Similarly, mean-field analysis (Chizat and Bach, 2018; Mei et al., 2019) has been used to study the evolution of neuron distribution in the infinite-width limit. These models often rely on assumptions such as convexity in the loss landscape and constraining the dynamics to a region around initialization. This precludes feature learning—a critical aspect of modern deep neural networks.

1.1.3. PAC framework and weight norm-based bounds

The Probably-approximately correct (PAC) frameworks (Valiant, 1984) provides generalization bounds for models trained on independently and identically distributed (i.i.d.) data, using measures such as Vapnik-Chervonenkis (VC) dimension or Rademacher complexity to characterize the hypothesis space. The PAC-Bayes framework (McAllester, 1999a; Langford and Caruana, 2002) extends these ideas by deriving generalization error bounds for randomized estimators. However, both frameworks are limited in their ability to explain the generalization behavior of modern deep neural networks. Despite their rich hypothesis class and extremely large VC dimensions, deep neural networks consistently achieve remarkable generalization, a phenomenon that defies the worst-case assumptions inherent in these classical frameworks.

A more refined line of work replaces the size of the entire hypothesis class with complexity measures evaluated at the learned solution. Norm-based bounds characterize complexity using quantities such as products of layer-wise norms, spectral norms, and classification margins (Neyshabur et al., 2015; Bartlett et al., 2017; Neyshabur et al., 2018). Because these quantities depend on the learned weights, such bounds can be sharper than parameter-count- or VC-dimension-based bounds. Nevertheless, they remain uniform over all networks satisfying the norm constraints. This set can still be much larger than the subset of solutions that is accessible to a training algorithm on typical data, and consequently the resulting numerical bounds can remain loose.

Another direction studies neural networks in the lazy-training or neural tangent kernel regime. For a network that is sufficiently wide, the gradient kernel remains approximately constant during training, reducing the analysis to that of a kernel method. The displacement from initialization has a closed form and provide a generalization bound (Arora et al., 2019; Cao and Gu, 2019). The analysis of these methods require sufficiently wide networks, and often in the lazy training regime, where the end of training point is not too far away from initialization.

1.1.4. Mutual information-based generalization bounds

Another important line of work studies generalization from an information-theoretic perspective. The central idea is to quantify how much information the output of a learning algorithm retains about the particular training sample. Let S denote the training set and W the learned parameters. Under suitable assumptions on the loss, the expected generalization gap can be bounded in terms of the mutual information $I(S;W)$ (Russo and Zou, 2015; Xu and Raginsky, 2017). Informally, if the learned parameters reveal little information about the identities or values of the training examples, then the learning algorithm is less likely to overfit.

For stochastic iterative algorithms, this endpoint information can be further decomposed along the training trajectory. At each iteration, the data-dependent gradient may transmit information

about the selected samples to the parameters, whereas the injected noise limits how accurately this information can be retained. For algorithms such as stochastic gradient Langevin dynamics, this leads to bounds of the representative form $I(S;W_T) \lesssim \sum_{t=0}^{T-1} \frac{\eta_t^2}{\sigma_t^2}\mathbb{E}\left[\operatorname{tr}\widehat{\Sigma}_g(t)\right]$, where σ_t measures the Langevin noise, and $\widehat{\Sigma}_g(t)$ measures the covariance of the sample gradients, both at iteration t (Mou et al., 2018; Negrea et al., 2019; Neu et al., 2021). Thus, information about the training sample accumulates over time according to the variability of the sample gradients, relative to the amount of noise injected by the algorithm.

Conditional mutual-information bounds further refines this bound. One constructs pairs of independent candidate samples and randomly selects one sample from each pair to form the training set. The conditional mutual information measures how much the learned predictor reveals about which candidate in each pair was selected for training (Hafez-Kolahi et al., 2020; Steinke and Zakynthinou, 2020), conditioning on all samples. By focusing on the sample-selection information rather than all information contained in the dataset, this approach can produce sharper and more data-dependent generalization guarantees.

These information-theoretic methods are particularly natural for randomized or noisy learning algorithms. Although the general framework is not formally restricted to stochastic algorithms, standard mutual information can become infinite or uninformative for deterministic algorithms. Moreover, trajectory-based bounds for noisy optimization can be highly sensitive to the noise scale: as σ_t becomes small, terms proportional to $1/\sigma_t^2$ may become large and the resulting bound can become vacuous.

1.1.5. Stability and robustness-based bounds

Algorithmic stability studies how sensitive a learning algorithm is to a small perturbation of its training data. An algorithm is stable implies that no individual training example has a large influence on the learned predictor, allowing the empirical loss to remain representative of the population loss (Bousquet and Elisseeff, 2002). This framework has been used to derive algorithm-dependent generalization guarantees for gradient-based optimization. For example, Hardt et al. (2016) analyzed the stability of stochastic gradient descent under Lipschitzness and smoothness conditions and related its generalization behavior to the learning rate, the number of iterations, and properties of the loss.

Training trajectory-related quantities are used to analyze generalization. Kozachkov et al. (2023b) use Riemannian contraction theory to derive dynamical robustness and generalization guarantees for supervised learning. This relies on uniform assumptions of the loss landscape. Data-dependent stability analyses are used to replaced global worst-case constants with quantities determined by the data and local curvature, showing that the stability of SGD can depend on the geometry encountered near its initialization and along optimization (Kuzborskij and Lampert, 2018). From a complementary convex-analytic perspective, Lugosi and Neu (2022) derive generalization bounds by tracking the growth of a potential function associated with the dependence between the training data and the learned output. Another closely related direction represents the geometry explored during training through a data-dependent kernel. Chen et al. (2023b) introduced the loss path kernel, which aggregates inner products between per-example loss gradients along optimization. More recent work used this kernel to derive gradient-flow generalization bounds involving its RKHS norm and trace (Chen et al., 2025). Unlike the static neural tangent kernel, the loss path kernel changes with the parameters and can therefore capture feature learning. Training-trajectory-dependent

quantities can be computationally expensive and difficult to estimate accurately, which may limit the practical applicability of the resulting generalization bounds.

A complementary perspective is provided by algorithmic robustness. Rather than retraining the model after replacing a training sample, robustness-based analyses fix the learned predictor and compare its losses on nearby training and test examples. If the sample space has a small over number, then the gap between empirical loss and population loss can be small (Xu and Mannor, 2012). Kawaguchi et al. (2022) developed data-dependent bounds in which these regions and the associated loss consistency are induced by the observed sample. This can yield more localized guarantees than bounds based on a fixed, worst-case partition of the entire sample space. Similarly, Chen et al. (2022b) exploit the low-dimensional manifold structure of the data to derive statistical recovery guarantees that depend on the intrinsic rather than the ambient dimension.

1.1.6. Learning Frameworks for Non-IID Data

A changing data distribution could be modeled as a sequence of tasks. Depending upon the stochastic process, different concepts are relevant, e.g., multi-task learning (Baxter, 2000a) is useful for scenarios when there are a finite number of tasks. Much of continual or lifelong learning (Ramesh and Chaudhari, 2022; Vogelstein et al., 2020) focuses on "task-incremental" and "class-incremental" settings (Van de Ven and Tolias, 2019), in which the learner knows when the task switches. The "Prospective Learing (PL)" framework does not make this assumption, and therefore, the problem is substantially more difficult. "Data-incremental" (or task-agnostic) setting (De Lange and Tuytelaars, 2021) is similar to PL. But the main difference is the goal: continual or lifelong learning seeks to minimize past error. As a consequence, continual learning methods are poor prospective learners; see Sec. 4.5. Online meta-learning (Thrun and Pratt, 1998; Dhillon et al., 2020; Finn et al., 2017a; Fakoor et al., 2020) is close to task-agnostic continual learning, except that the former models tasks as being sampled IID from some distribution of tasks. Due to this, one cannot predict which task is next, and therefore cannot prospect.

1.2. Contributions

In Chapter 2, we study how the dataset may be the cause of the anomalous generalization performance of deep networks. We show that the data correlation matrix of typical classification datasets has an eigenspectrum where, after a sharp initial drop, a large number of small eigenvalues are distributed uniformly over an exponentially large range. This structure is mirrored in a network trained on this data: we show that the Hessian and the Fisher Information Matrix (FIM) have eigenvalues that are spread uniformly over exponentially large ranges. We call such eigenspectra "sloppy" because sets of weights corresponding to small eigenvalues can be changed by large magnitudes without affecting the loss. Networks trained on atypical, non-sloppy synthetic data do not share these traits. We show how this structure in the data can give to non-vacuous PAC-Bayes generalization bounds analytically; we also construct data-distribution dependent priors that lead to accurate bounds using numerical optimization.

In Chapter 3, we derive a differential equation that governs the evolution of the generalization gap when a model is trained by gradient descent-based methods. This differential equation is driven by two key quantities, a contraction factor that brings together trajectories corresponding to slightly different datasets, and a perturbation factor that accounts for them training on different datasets. The coupled decay of contraction and perturbation guarantees a controlled accumulation of generalization gap during training. We analyze this differential equation to show that the generalization

gap is given by a quadratic form that consists of an "effective Gram matrix" that depends upon the training trajectory and a certain residual of the predictor at initialization. Our framework is applicable to general deep networks and smooth loss functions. In numerical experiments on different neural network architectures, datasets and sample sizes, we show that this quadratic form accurately captures the actual generalization gap. We also show how to instantiate our framework in a number of examples via analytical calculations. For example, for high-dimensional linear regression, our framework matches existing calculations of generalization gap in the literature exactly in under-parameterized, over-parameterized and critical regimes.

In Chapter 4, we develop a theoretical framework called "Prospective Learning" that is tailored for situations when the optimal hypothesis changes over time. In PAC learning, empirical risk minimization (ERM) is known to be consistent. We develop a learner called Prospective ERM, which returns a sequence of predictors that make predictions on future data. We prove that the risk of prospective ERM converges to the Bayes risk under certain assumptions on the stochastic process generating the data. Prospective ERM, roughly speaking, incorporates time as an input in addition to the data. We show that standard ERM as done in PAC learning, without incorporating time, can result in failure to learn when distributions are dynamic. Numerical experiments illustrate that prospective ERM can learn synthetic and visual recognition problems constructed from MNIST and CIFAR-10.

In Chapter 5, we develop compositional generalization as a future direction. We consider settings in which the training and test data contain the same underlying operations and instance families but differ in how these components are combined, so that test tasks require new pairings of previously learned skills and contexts. We propose that generalization to these unseen compositions depends both on the residual error remaining after training and on whether the functional directions required by the new combinations are sufficiently covered by those activated by the observed training data.

CHAPTER 2

DOES THE DATA INDUCE CAPACITY CONTROL IN DEEP LEARNING?

As reviewed in Sec. 1.1, deep neural networks often generalize far better than predicted by worst-case complexity measures. This observation has motivated generalization bounds based on properties of the learned solution and the training process. For example, norm-based bounds characterize complexity through quantities such as the learned weight norms and margins, as discussed in Sec. 1.1.3, while stability- and trajectory-based analyses use quantities accumulated along optimization, as reviewed in Sec. 1.1.5. Although these quantities are data-dependent in the sense that they are evaluated on a model trained on the observed dataset, their connection to the structure of the data is often indirect. In this chapter, we seek a more intuitive connection between data and generalization. Specifically, we ask whether the geometry of the input data is inherited by training-related quantities, thereby restricting the effective capacity of the trained network and leading to improved generalization guarantees.

This motivates the central question of this chapter: what structure effectively restricts the behavior of deep networks in practice? Prior work on the geometry of neural loss landscapes has shown that the Hessian and the Fisher Information Matrix often contain only a few stiff directions and many nearly flat directions (Papyan, 2018, 2019; Pennington and Bahri), echoing the notion of sloppy models studied in physics and systems biology (Brown et al., 2004; Gutenkunst et al., 2007). We argue that an important part of the answer lies in the geometry induced by the data. Empirically, typical datasets exhibit highly anisotropic, or "sloppy," eigenspectra: only a small number of directions carry large variance, while many directions have very small eigenvalues. In this chapter, we show that this structure is inherited by trained neural networks, appearing in the spectra of the Hessian and the Fisher Information Matrix. Thus, although the parameter space is extremely high-dimensional, the directions that significantly affect the loss can be much fewer. These quantities can be well related to the data and provide a data-dependent mechanism of effective capacity control, helping explain why deep networks can generalize even in the absence of explicit regularization.

2.1. Introduction

Consider the experiment in Fig. 2.1. For a convolutional network trained on CIFAR-10, we calculated the eigenspectrum of the data correlation matrix ($n^{-1}XX^\top$ where each column of X is one input sample) and compared it to the eigenspectra of the Fisher Information Matrix (FIM) and the Hessian of a deep network trained on this dataset. Eigenspectrum of the FIM and the Hessian have a similar decay pattern as that of the data matrix. There are very few (less than 5% of the input dimensionality) large—let us call them stiff—eigenvalues after which there is a sharp drop in magnitude and a long tail of small—let us call them "sloppy"—eigenvalues follows. Some other correlation matrices that one can obtain from the network also show the same decay pattern, e.g., that of activations of different layers, Jacobians of any of the logits with respect to the weights, and gradients of the loss with respect to activations of different layers. All these matrices have eigenvalues that span exponentially large ranges, e.g., the top 1000 eigenvalues drop by about 5 orders of magnitude. Sloppy eigenvalues are distributed uniformly across such exponentially large ranges. Eigenspectra of many other typical datasets and neural networks share these traits. On the other hand, eigenspectra corresponding to synthetic, atypical datasets, which we create by sampling inputs randomly and labeling them either randomly, or using a random teacher network, do not share these traits (Sec. 2.4).

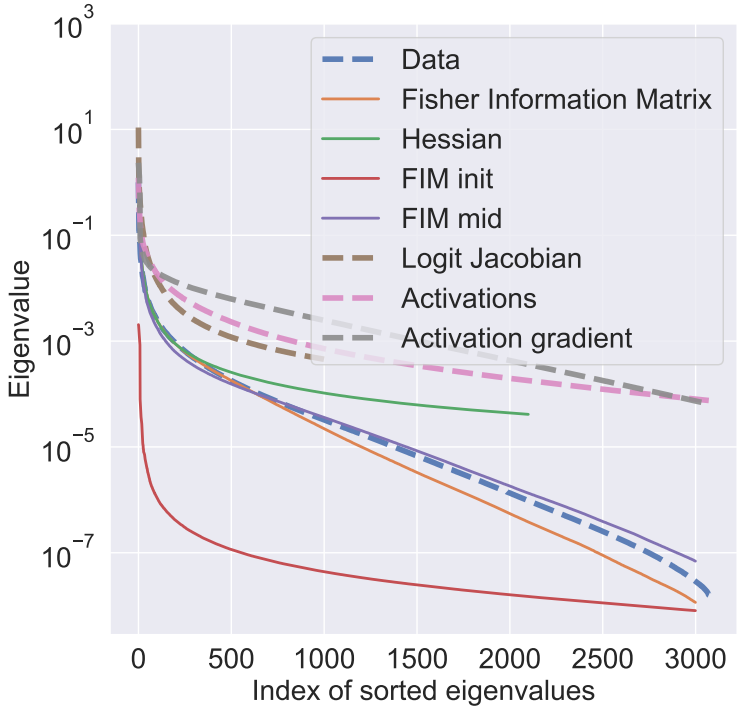

Figure 2.1: Eigenspectra of the correlation matrices of the data, the activations and gradients with respect to the activations of the second layer, Jacobian of one of the logits with respect to the weights, the FIM, empirical FIM, Hessian at the end of training and FIM at initialization and in the middle of training. Activation gradients are scaled up by 10^{12} to bring them to this scale. All eigenspecta are scaled by the largest eigenvalue of the data correlation matrix. All eigenspectra are qualitatively similar with eigenvalues that span exponentially large ranges. After an initial regime where the top few eigenvalues drop sharply (which we call stiff), the slope of the tail (which we call sloppy) of these eigenspectra on a logarithmic scale is almost the same as that of the data matrix. We call this phenomenon "sloppiness". Eigenspectra corresponding to activations/activation gradients of other layers and logit Jacobians of other logits are very similar (see Figs. 2.5 to 2.7). The eigenspectra at initialization or in the middle of training are also qualitatively the same (see Fig. 2.10). This plot shows the eigenspectra for a wide residual network on CIFAR-10; corresponding eigenspectra of other networks on MNIST are qualitatively the same (see Figs. 2.3 and 2.9).

The Hessian determines the local geometry in the weight space of the loss function of a deep network, small eigenvalues correspond to directions in the weight space where the loss is insensitive to changes in the weights. Similarly, the FIM governs the local geometry in the prediction space, i.e., if we think of a deep network as a parameterized probability distribution $p_w(y \mid x)$, the FIM is the correlation matrix of the gradient of log-likelihood $\log p_w(y \mid x)$. Eigenvectors of small eigenvalues of the FIM correspond to sets of weights which can be modified significantly without affecting the probability distribution $p_w(y \mid x)$ much.

This leads us to the hypothesis that the training data effectively controls the complexity of the trained model. This chapter investigates this hypothesis and makes the following contributions.

(1) We show that **for typical datasets and deep networks, eigenspectra of correlation matrices of the data, activations of different layers, Jacobian of the logits with respect to**

the weights, gradients of the loss with respect to the activations, as also the Hessian and the FIM, are all qualitatively the same. These eigenspectra consists of a few large eigenvalues and a large number of small eigenvalues that are distributed uniformly across an exponentially large range. We call such eigenspectra and the corresponding quantities "sloppy" and define this notion in Theorem 2.17. Atypical datasets, e.g., those with random inputs or labels, can be constructed for whom these quantities do not have such eigenspecta. This suggests that typical inputs are "simple" (in the precise sense of having a sloppy data correlation matrix) and this results in a "simple" trained model (in the sense of a sloppy Hessian or FIM). We show using theory that (a) the trace of the correlation of the activations, logit Jacobians, Hessian and the FIM can be upper bounded by the trace of the data correlation matrix, (b) if we assume that the activations are sloppy then we can show that the eigenspectrum of the FIM is sloppy, (c) under the assumption of bounded norm of weights, we can show that the eigenvalues of activations decays faster than $\mathcal{O}(1/i)$.

(2) As a demonstration of how sloppiness of data controls the capacity of the trained model, we calculate **non-vacuous PAC-Bayes bounds analytically.** For a Gaussian isotropic prior $N(w_0, \epsilon^{-1}I)$ centered at the initial weights w_0, we calculate the optimal covariance of a Gaussian posterior $N(w, \Sigma_q)$ (where w are weights of the trained network) that minimizes a loose version of PAC-Bayes generalization bound. We show that if the Hessian at w is sloppy, we can obtain a non-vacuous generalization bound—without any numerical optimization. Specifically, we show that the optimal PAC-Bayes posterior inverse covariance has the same eigenvectors as Hessian, and the corresponding eigenvalues are

$$\bar{\lambda}_i = 2(n-1)\lambda_i + \epsilon \quad \forall i \leq p,$$

where λ_i are eigenvalues of the Hessian, n is the number of samples and p is the number of weights. A posterior that leads to a small bound is less spread out along stiff directions but more spread out along sloppy directions of the Hessian.

(3) We can think of the prior inverse variance scaled by the number of data $\epsilon/(2(n-1))$ as a threshold beyond which the eigenvalues of the Hessian λ_i are small enough and the loss changes so little that the optimal PAC-Bayes posterior focuses on accurately capturing the prior covariance to obtain a small KL-term. Motivated by this, we define an **effective dimensionality of a deep network** as

$$p(n, \epsilon) = \sum_{i=1}^{p} \mathbf{1}_{\left\{|\lambda_i| > \frac{\epsilon}{2(n-1)}\right\}},$$

i.e., the number of eigenvalues of the Hessian that are greater than a threshold $\epsilon/(2(n-1))$ for a user-chosen PAC-Bayes prior $N(0, \epsilon^{-1}I)$. For sloppy eigenspectra, this dimensionality is typically a tiny fraction of the number of weights (0.29% for a fully-connected network on MNIST).

(4) We find that the **stiff sub-space of the FIM at initialization has a strong overlap with its counterpart at the end of training, and that updates to the weights take place in the subspace spanned by the stiff eigenvectors of the FIM (as also the Hessian)**. We exploit this observation to compute tight data-distribution/data-dependent PAC-Bayes bounds for fully-connected and convolutional networks on MNIST using a Gaussian prior whose covariance is proportional to the FIM and a Gaussian posterior whose eigenvectors are the same as those of the FIM at initialization.

2.2. Background

This section develops notation and PAC-Bayesian generalization bounds.

2.2.1. Problem Setup

Consider a dataset $D_n = \{(x_i, y_i)\}_{i=1}^n$ with n samples, $x_i \in X \subset \mathbb{R}^d$ and $y_i \in Y = \{1, \ldots, m\}$. We assume that this dataset is drawn from a joint distribution D on $X \times Y$. A classifier $h_w : X \mapsto [0,1]^m$ parameterized by weights $w \in \mathbb{R}^p$ belongs to a hypothesis space $\{h_w : w \in \mathbb{R}^p\}$; this classifier maps inputs $x \in X$ to m-dimensional categorical distributions $p_w(y \mid x) \in [0,1]^m$. Let Q be a distribution on hypotheses which is implicitly a measure on $\mathbb{R}^p$. We define

(a) training error of a hypothesis $\hat{e}(h_w, D_n) = \frac{1}{n} \sum_{i=1}^n \mathbf{1}_{\left\{y_i \neq \arg\max_y (p_w(y \mid x_i))\right\}}$;
(b) cross-entropy objective $\check{e}(h_w, D_n) = -\frac{1}{n \log(2)} \sum_{i=1}^n \log p_w(y_i \mid x_i)$ which is used as a differentiable surrogate and convex upper-bound of the error for purposes of training;
(c) population error of a hypothesis $e(h_w) = \mathbb{E}_{D_n \sim D^n} [\hat{e}(h_w, D_n)]$;
(d) empirical error and loss of the distribution of hypotheses $\hat{e}(Q, D_n) = \mathbb{E}_{w \sim Q} [\hat{e}(h_w, D_n)]$ and $\check{e}(Q, D_n) = \mathbb{E}_{w \sim Q} [\check{e}(h_w, D_n)]$, respectively; and
(e) with some abuse of notation, population error of Q given by $e(Q) = \mathbb{E}_{D_n \sim D^n} [\hat{e}(Q, D_n)]$.
(f) we use $\check{e}(Q) = \mathbb{E}_{D_n \sim D^n} [\check{e}(Q, D_n)]$ to denote the population loss.

Hessian and Fisher Information Matrix (FIM) The Hessian $H \in \mathbb{R}^{p \times p}$ is defined to be the second derivative of the empirical loss with respect to the weights w, i.e., $H_{ij} = \partial_i \partial_j \check{e}(h_w, D_n)$. The entries F_{ij} of the Fisher Information Matrix (FIM) $F \in \mathbb{R}^{p \times p}$ are

$$\frac{1}{n} \sum_{k=1}^n \sum_{y=1}^m p_w(y \mid x_k) \partial_i \log p_w(y \mid x_k) \partial_j \log p_w(y \mid x_k);$$

it is important to note the expectation over the outputs y. A related quantity is the empirical FIM where one sets $y = y_k$ in the expression above. Although the FIM and the empirical FIM are different in general, especially when used in algorithms like natural gradient descent (Kunstner et al., 2019), their eigenspectra are qualitatively similar (Fig. 2.1). Both the Hessian and FIM are large matrices and it is prohibitive to compute them for modern deep networks. Some of our experiments use a Kronecker-factor approximation (Martens and Grosse, 2016) of block diagonal Hessian and FIM where cross-terms $\partial_i \partial_j$ across different layers of a deep network are set to zero.

2.2.2. PAC-Bayes generalization bounds

The PAC-Bayesian framework developed in Langford and Seeger (2001); McAllester (1999b) allows us to estimate the population error of a randomized hypothesis with distribution Q using its empirical error and its Kullback–Leibler (KL) divergence from a prior distribution P.

Theorem 2.1 (PAC-Bayes generalization bound; McAllester, 1999b; Langford and Seeger, 2001). Let P be fixed independently of the observed dataset D_n. For every $\delta > 0$, with probability at least $1 - \delta$ over $D_n \sim D^n$, the following inequality holds simultaneously for all distributions Q on the hypothesis space:

$$\mathrm{kl}(\hat{e}(Q, D_n), e(Q)) \leq \frac{\mathrm{KL}(Q, P) + \log(n/\delta)}{(n-1)},$$

where $\mathrm{KL}(Q, P) = \int \log\left(\frac{\mathrm{d}Q}{\mathrm{d}P}(w)\right) \mathrm{d}Q(w)$.

Here, $\mathrm{kl}(q, p) = q \log(q/p) + (1-q) \log((1-q)/(1-p))$ is the KL divergence between Bernoulli

distributions with means q and p. Pinsker's inequality gives

$$2(q-p)^2 \le \mathrm{kl}(q,p).$$

Applying it to Theorem 2.1 yields

$$e(Q) \le \hat{e}(Q, D_n) + \sqrt{\frac{\mathrm{KL}(Q,P) + \log(n/\delta)}{2(n-1)}}.$$

Moreover, for every classifier h_w and example (x_i, y_i),

$$\mathbf{1}_{\{y_i \neq \arg\max_y p_w(y \mid x_i)\}} \le -\frac{1}{\log 2} \log p_w(y_i \mid x_i).$$

Averaging over the dataset and over $w \sim Q$ therefore gives

$$\hat{e}(Q, D_n) \le \check{e}(Q, D_n).$$

We use this result with a Gaussian posterior $Q = N(w, \Sigma_q)$ and a family of Gaussian priors whose precision is selected from a countable grid. For fixed $b, c > 0$, define

$$\epsilon_j = c\exp(-j/b), \qquad P_j = N(w_0, \epsilon_j^{-1} I), \qquad j = 1, 2, \ldots,$$

where each P_j is chosen independently of D_n. Apply Theorem 2.1 to P_j with failure probability $\delta_j = 6\delta/(\pi^2 j^2)$. Since $\sum_{j=1}^{\infty} \delta_j = \delta$, a union bound shows that, with probability at least $1-\delta$, the resulting inequalities hold simultaneously for every $j \ge 1$ and every posterior Q. Using $j = b\log(c/\epsilon_j)$ and the cross-entropy upper bound above, we obtain

$$e(Q) \le \check{e}(Q, D_n) + \left(\frac{\mathrm{KL}(Q,P_j) + 2\log(b\log(c/\epsilon_j)) + \log\bigl(\pi^2 n/(6\delta)\bigr)}{2(n-1)} \right)^{1/2}, \tag{2.1}$$

simultaneously for all $j \ge 1$ and all Q. This differentiable upper bound can be optimized using SGD (Langford and Caruana, 2002; Dziugaite and Roy, 2017b).

2.2.3. Data-dependent priors

The posterior Q in Theorem 2.1 may depend upon the training samples D_n, e.g., it could be the distribution on the weight space induced by a randomized training algorithm like stochastic gradient descent (SGD). The prior P can depend upon the data distribution D, but not the samples D_n themselves. Although it is common to use priors that do not depend upon the data at all, it is has been increasingly noticed that data-distribution dependent priors may provide tighter bounds (Dziugaite and Roy, 2018). To gain intuition, notice that in the expression for the KL-divergence between two Gaussians, we have a term of the form $(w-w_0)^\top \Sigma_p^{-1}(w-w_0)$ depends upon the distance between trained weights w and initialization w_0 (weighted by the inverse prior covariance Σ_p^{-1}). It is difficult to pick a prior P such that this term is small.

Priors that depend upon the FIM and the Hessian In order to compute a tight PAC-Bayes bound, one may choose a prior using a subset of the training samples (Ambroladze et al., 2007). For instance, we can center the Gaussian prior on weights pre-trained on this subset to obtain a better

PAC-Bayes bound—the theory allows this. Doing so leads to a worse denominator in Theorem 2.1, although this may be mitigated by a smaller numerator. Parrado-Hernández et al. (2012) also define expectation-priors, i.e., where we choose a prior that depends on the data *distribution* but in practice, we evaluate this prior using the dataset. For example, choosing the prior covariance to be proportional to the FIM, or the Gauss-Newton approximation of the Hessian,

$$\Sigma_p \propto F_{w_0}, \text{or, } \Sigma_p \propto \tilde{H}_{w_0}, \tag{2.2}$$

are both valid choices in the PAC-Bayes framework. The crucial distinction between these two choices is that while we may use all the training samples to compute the FIM in the former case, we should compute the Hessian on a separate subset of the data in the latter case.

2.3. Methods

In Sec. 2.3.1, we prove how such a sloppiness in the Hessian and the FIM arises if it is present in the input correlation matrix. We then discuss different methods to compute PAC-Bayes bounds on the generalization error using sloppiness (Sec. 2.3.2) and develop an expression for the effective dimensionality of a deep network (Sec. 2.3.3).

2.3.1. A sloppy data correlation matrix leads to a sloppy FIM and Hessian

Consider a deep network with L layers with the input of the network $x \in \mathbb{R}^d$ and m outputs. The activations of the k^{th} layer are given by $h^k = \sigma(w^{k-1}h^{k-1})$,for $k = 1, \cdots, L$. Preactivations will be denoted by $u^k = w^{k-1}h^{k-1}$ for $k = 1, \cdots, L+1$, and for clarity, we use a special notation $z \equiv u^{L+1}$ to denote the logits of the network. The weights of the network are $w = (w^0, w^1, \ldots, w^L)$. We have $h^k \in \mathbb{R}^{d_k}$, $w^k \in \mathbb{R}^{d_{k+1} \times d_k}$ and $w^L \in \mathbb{R}^{m \times d_L}$. In this notation, we have $h_0 = x \in \mathbb{R}^d$. The linear map represented by w^k can model both fully-connected layers and convolutional layers. For the sake of exposition, we set all the bias terms to zero. The non-linearity σ acts element-wise upon its argument and we assume that it has a bounded derivative $|\sigma'(x)| \leq a$ with $\sigma(x) = 0$, in which case, $|\sigma(x)| \leq a\,|x|$. Popular nonlinearities like rectified linear units (ReLU), leaky ReLUs and tanh satisfy this assumption. In this section, we use $\mathbb{E}$ to denote the expectation over x. The following lemmas holds for all distribution of x. In particular, we can choose the distribution of x to be the point mass distribution on the dataset D_n, i.e. $x \sim \frac{1}{n}\sum_{i=1}^{n} \delta_{x_i}$, in this case, $\mathbb{E}\left[xx^\top\right] = \frac{1}{n}XX^T \in \mathbb{R}^{d \times d}$ is the input corelation matrix.

The following lemma bounds the trace of the activation correlations and the norm of the gradient of each logit with respect to the activations.

Lemma 2.2 (Bounding the trace of the correlations of activations and norm of activation gradients). We have

$$\text{tr}\left(\mathbb{E}\left[h^k h^{k\top}\right]\right) \leq a^2 \|w^{k-1}\|_2^2 \text{tr}\left(\mathbb{E}\left[h^{k-1}h^{k-1\top}\right]\right), \tag{2.3}$$

and

$$\left\|\frac{\mathrm{d}z_i}{\mathrm{d}h^k}\right\|_2 \leq a \left\|\frac{\mathrm{d}z_i}{\mathrm{d}h^{k+1}}\right\|_2 \left\|w^k\right\|_2. \tag{2.4}$$

Proof. For the first inequality in Eq. (2.3), observe that

$$\begin{aligned}
\mathrm{tr}\left(\mathbb{E}\left[h^k {h^k}^\top\right]\right) &\leq \sum_{j=1}^{d_k} \mathbb{E}\left[\sigma(u_j^k)^2\right] \\
&\leq a^2 \sum_{j=1}^{d_k} \mathbb{E}\left[(u_j^k)^2\right] \\
&= a^2 \mathrm{tr}\left(\mathbb{E}\left[u^k {u^k}^\top\right]\right) \\
&= a^2 \mathrm{tr}\left(\mathbb{E}\left[\left(w^{k-1} h^{k-1}\right)\left(w^{k-1} {h^{k-1}}^\top\right)\right]\right) \\
&= a^2 \mathrm{tr}\left(w^{k-1}\, \mathbb{E}\left[h^{k-1} {h^{k-1}}^\top\right] {w^{k-1}}^\top\right) \\
&\leq a^2 \left\|w^{k-1}\right\|_2^2 \mathrm{tr}\left(\mathbb{E}\left[h^{k-1} {h^{k-1}}^\top\right]\right).
\end{aligned}$$

The final inequality uses $\mathrm{tr}(WAW^\top) \leq \|W\|_2^2 \mathrm{tr}(A)$ for every positive semi-definite matrix A.

For the second inequality in Eq. (2.4), observe that

$$\begin{aligned}
\frac{\mathrm{d}z_i}{\mathrm{d}h^k} &= \frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}} w^k \\
&= a\left(\frac{\mathrm{d}z_i}{\mathrm{d}h^{k+1}} \mathbb{1}_{u^{k+1} \geq 0}\right) w^k \\
\Rightarrow \left\|\frac{\mathrm{d}z_i}{\mathrm{d}h^k}\right\|_2 &\leq a \left\|\frac{\mathrm{d}z_i}{\mathrm{d}h^{k+1}}\right\|_2 \left\|w^k\right\|_2.
\end{aligned}$$

where $\mathbb{1}_{\text{cond}}$ is a vector of 1s at elements where the condition is true. □

The above inequalities can be used in Theorem 2.3 to bound the trace of the gradient correlation of any logit z_i with respect to weights of a layer w^k.

Lemma 2.3 (Bounding the trace of logit-Jacobian correlations). For logit z_i, $i = 1, ..., m$

$$\mathrm{tr}\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w^k} \frac{\mathrm{d}z_i}{\mathrm{d}w^k}^\top\right]\right) \leq a^{2L} \mathrm{tr}\left(\mathbb{E}\left[xx^\top\right]\right) \prod_{j=0, j\neq k}^{L} \left\|w^j\right\|_2^2. \tag{2.5}$$

for $k = 0, ..., L$. As a result,

$$\mathrm{tr}\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w} \frac{\mathrm{d}z_i}{\mathrm{d}w}^\top\right]\right) \leq a^{2L} \mathrm{tr}\left(\mathbb{E}\left[xx^\top\right]\right) \prod_{j=0}^{L} \left\|w^j\right\|_2^2 \left(\sum_{j=0}^{L} \frac{1}{\left\|w^j\right\|_2^2}\right).$$

Proof. The proof follows via an application of Theorem 2.2. For $k = 0, 1, ..., L-1$,

$$\begin{aligned}
\operatorname{tr}\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w^k}\frac{\mathrm{d}z_i}{\mathrm{d}w^k}^\top\right]\right) &= \operatorname{tr}\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}^\top \otimes h^k h^{k\top}\right]\right) \\
&= \mathbb{E}\left[\operatorname{tr}\left(\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}^\top\right)\operatorname{tr}\left(h^k h^{k\top}\right)\right] \\
&\leq a^2\left\|\frac{\mathrm{d}z_i}{\mathrm{d}h^{k+1}}\right\|_2^2 \operatorname{tr}\left(\mathbb{E}\left[h^k h^{k\top}\right]\right) \\
&\leq a^2\left\|\frac{\mathrm{d}z_i}{\mathrm{d}h^{L}}\right\|_2^2 \left(\prod_{j=k+1}^{L-1}\left\|w^j\right\|_2^2\right) a^{2(L-k-1)} \\
&\quad a^{2k}\prod_{j=0}^{k-1}\left\|w^j\right\|_2^2 \operatorname{tr}\left(\mathbb{E}\left[xx^\top\right]\right) \\
&\leq a^{2L}\operatorname{tr}\left(\mathbb{E}\left[xx^\top\right]\right)\prod_{j=0,j\neq k}^{L}\left\|w^j\right\|_2^2.
\end{aligned}$$

The third line comes from the fact that the matrix $\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}^\top$ is rank one and its trace is the same as 2-norm. The last inequality comes from the fact that $\left\|w_i^L\right\|_2 \leq \left\|w^L\right\|_2$. For $k = L$,

$$\begin{aligned}
\operatorname{tr}\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w^L}\frac{\mathrm{d}z_i}{\mathrm{d}w^L}^\top\right]\right) &= \operatorname{tr}\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w_i^L}\frac{\mathrm{d}z_i}{\mathrm{d}w_i^L}^\top\right]\right) \\
&= \operatorname{tr}\left(\mathbb{E}\left[h^L h^{L\top}\right]\right) \\
&\leq a^{2L}\operatorname{tr}\left(\mathbb{E}\left[xx^\top\right]\right)\prod_{j=0}^{L-1}\left\|w^j\right\|_2^2.
\end{aligned}$$

□

We first bound the trace of the FIM and the Hessian in terms of the trace of the input correlation matrix.

Theorem 2.4 (Trace of the FIM and Hessian are bounded by that of the data correlation matrix). For the FIM F and the Hessian H, we have

$$\operatorname{tr}(F), (\log 2)\operatorname{tr}(H) \leq m a^{2L}\operatorname{tr}\left(\mathbb{E}[xx^T]\right)\prod_{j=0}^{L}\left\|w^j\right\|_2^2\left(\sum_{j=0}^{L}\frac{1}{\left\|w^j\right\|_2^2}\right). \tag{2.6}$$

Notice that the $\log 2$ factor in front of $\operatorname{tr}(H)$ comes from the rescaling factor in the definition of $\check{e}(h_w, D_n)$.

Proof. We first calculate an inequality for the Fisher Information Matrix (FIM)

$$
\begin{aligned}
F &= \mathbb{E}\left[\sum_{y=1}^{m} p_w(y\,|\,x)(\partial_w \log p_w(y\,|\,x))(\partial_w \log p_w(y\,|\,x))^\top\right] \\
&= \mathbb{E}\left[\partial_w z\left[\sum_{y=1}^{m} p_w(y\,|\,x)\frac{\mathrm{d}\log p_w(y\,|\,x)}{\mathrm{d}z}\frac{\mathrm{d}\log p_w(y\,|\,x)}{\mathrm{d}z}^\top\right]\partial_w z^\top\right]
\end{aligned}
$$

For an output distribution $p_w(y\,|\,x)$ obtained using the softmax operator on the logits z_y

$$
p_y \equiv p_w(y\,|\,x) = \frac{e^{z_y}}{\sum_{y'} e^{z_{y'}}}
$$

we have

$$
\frac{\mathrm{d}}{\mathrm{d}z}\log p_w(y\,|\,x) = e_y - p
$$

where e_y is the one-hot vector of the class y and $p = [p_1, ..., p_m]$.

$$
\begin{aligned}
\sum_{y=1}^{m} p_w(y\,|\,x)\frac{\mathrm{d}\log p_w(y\,|\,x)}{\mathrm{d}z}\frac{\mathrm{d}\log p_w(y\,|\,x)}{\mathrm{d}z}^\top &\preceq \sum_{y=1}^{m} p_w(y\,|\,x)\left\|\frac{\mathrm{d}\log p_w(y\,|\,x)}{\mathrm{d}z}\right\|_2^2 I \\
&= (1 - \|p\|_2^2)I \\
&\preceq I
\end{aligned}
$$

Hence we have

$$
F \preceq \mathbb{E}\left[(\partial_w z)\,(\partial_w z)^\top\right].
$$

In the case of the Hessian for the cross-entropy loss we make a similar calculation following the calculation of Fort and Ganguli (2019b). For the calculation of Hessian, the expectation $\mathbb{E}$ denotes the expectation with respect to inputs and labels in the training set. We write

$$
\begin{aligned}
(\log 2)H &\approx \mathbb{E}\left[(\partial_w z)\,\nabla_z^2(-\log p_w(y\,|\,x))\,(\partial_w z)^\top\right] \\
&= \mathbb{E}\left[(\partial_w z)\left(\mathrm{diag}(p) - pp^\top\right)(\partial_w z)^\top\right] \\
&\preceq \mathbb{E}\left[(\partial_w z)\,(\mathrm{diag}(p))\,(\partial_w z)^\top\right] \\
&\preceq \mathbb{E}\left[(\partial_w z)\,(\partial_w z)^\top\right].
\end{aligned}
\tag{2.7}
$$

In the above calculation, we have kept only the so-called G-term of the Hessian and neglected an additional H-term.

$$
\mathbb{E}\left[\sum_{i=1}^{m}(y_k - p_k)\,\frac{\partial^2 z_i}{\partial w_\alpha \partial w_\beta}\right]
$$

which is typically small in practice for a well-trained network because the terms $1 - p_i$ are close to zero for all logits (Papyan, 2019; Sagun et al., 2016). Hence, both $\mathrm{tr}\,(F)$ and $(\log 2)\mathrm{tr}\,(H)$ can be

bounded by

$$\operatorname{tr}(F), (\log 2)\operatorname{tr}(H) \leq \sum_{i=1}^{m} \operatorname{tr}\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w}\frac{\mathrm{d}z_i}{\mathrm{d}w}^{\top}\right]\right) \leq ma^{2L}\operatorname{tr}\left(\mathbb{E}\left[xx^{\top}\right]\right)\prod_{j=0}^{L}\|w^j\|_2^2\left(\sum_{j=0}^{L}\frac{1}{\|w^j\|_2^2}\right). \tag{2.8}$$

Notice that the $\log 2$ factor in front of $\operatorname{tr}(H)$ comes from the rescaling factor in the definition of $\check{e}(h_w, D_n)$. □

Remark 2.5. The G-term is always positive semi-definite since the output distribution $p \in \mathbb{R}^C$ is always convex on the logits $z \in \mathbb{R}^C$, i.e., $\left(-\log\left(\frac{e^{z_y}}{\sum_{y'} e^{z_{y'}}}\right)\right)_{y=1}^{C}$ is convex in z.

Remark 2.6. The expression in Eq. (2.8) gives the form of the bound used in Theorem 2.10; in particular, the product of layer-wise weight norms appears explicitly in the term that controls the trace.

Remark 2.7. Empirically, the trace of FIM and Hessian at the end of training (Fig. 2.4) is usually much smaller than the trace of correlation matrix of logit Jacobians (Fig. 2.5). In this case, the prediction of the bound in Eq. (2.8) seems very loose. However from the above calculation, we also know that

$$\operatorname{tr}(F) \leq \mathbb{E}\left[(1-\|p\|_2^2)\operatorname{tr}\left((\partial_w z)(\partial_w z)^{\top}\right)\right],$$
$$\operatorname{tr}(H) \leq \operatorname{tr}\left(\mathbb{E}\left[(\partial_w z)\left(\operatorname{diag}(p) - pp^{\top}\right)(\partial_w z)^{\top}\right]\right).$$

For trained network that predicts accurately, we usually get the probabilities p that are very close to one-hot vectors of the correct classes. In this case, both $1-\|p\|_2^2$ and $\operatorname{diag}(p) - pp^{\top}$ are close to zero. This explains why in our experiments the trace of F and H at the end of training are much smaller than that of logit Jacobians.

We would next like to show that if the eigenvalues of the input correlation matrix $\frac{1}{n}XX^{\top} \in \mathbb{R}^{d\times d}$ are sloppy, then the FIM and the Hessian are also sloppy. Note that for a regression problem with a deep linear network, the Hessian and the FIM are actually the input matrix itself and therefore have the same eigenspectrum. It is however quite difficult to control the eigenspectrum of these matrices because they are a result of multiple nonlinear operations on the inputs. In general, the eigenspectrum of even the correlation of the activations can evolve arbitrarily. We therefore first bound the eigenvalues of a block-diagonal approximation of the FIM in terms of the eigenvalues of the activations in the following lemma.

Remark 2.8. For FIM F and G-term of Hessian H, Theorem 2.4 holds for all states during training. In fact, all of the bounds proved in this section about logit gradients depend only on the 2-norm of the weights, instead of the evolution of the weights or the training or test accuracy, which shows that the sloppiness can be preserved in places where 2-norm of the weights are not too large, which is prevalent throughout training. Note that for full Hessian H, the bound only holds at the end of training where the H-term of H is negligible, in other states of training, H is not guaranteed to be positive semi-definite.

The above lemma depends upon the following result that controls the eigenvalues of the sum of Hermitian matrices.

Lemma 2.9 (Weyl's inequality). For Hermitian matrices $A, B, C \in \mathbb{R}^{p\times p}$, if $C = A + B$, then

$$\lambda_{i+j-1}(C) \leq \lambda_i(A) + \lambda_j(B), \quad \lambda_{p-i-j}(C) \geq \lambda_{p-i}(A) + \lambda_{p-j}(B) \tag{2.9}$$

for all $1 \leq i, j \leq p$. In particular if $B \succeq 0$, then $\lambda_i(C) \geq \lambda_i(A)$ for all $i \leq p$.

Lemma 2.10 (Block-diagonal approximation of the FIM is sloppy if the activations are sloppy). Let $\text{spec}(A)$ denote the eigenvalues $(\lambda_1(A), \ldots, \lambda_p(A))$ of a matrix A in descending order. For a constant c, we define the notation $\text{spec}(A) \preceq c\,\text{spec}(B)$ to denote that $\lambda_i(A) \leq c\lambda_i(B)$ for all $i \leq p$. For any logit z_i, for all layers $k \leq L$, we have

$$\text{spec}\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w^k}\frac{\mathrm{d}z_i}{\mathrm{d}w^k}^\top\right]\right) \preceq\ a^{2(L-k)}\prod_{j=k+1}^{L}\left\|w^j\right\|_2^2 \text{spec}(I_{d_{k+1}}) \otimes \text{spec}\left(\mathbb{E}\left[h^k h^{k\top}\right]\right), \tag{2.10}$$

where we use the notation $\prod_{j=L+1}^{L}\|w_j\|_2^2 = 1$.

Proof.

$$\begin{aligned}
\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w^k}\frac{\mathrm{d}z_i}{\mathrm{d}w^k}^\top\right] &= \mathbb{E}\left[\left(\frac{\mathrm{d}z_i}{\mathrm{d}h^{k+1}} \odot \frac{\mathrm{d}h^{k+1}}{\mathrm{d}u^{k+1}}\right)\left(\frac{\mathrm{d}z_i}{\mathrm{d}h^{k+1}} \odot \frac{\mathrm{d}h^{k+1}}{\mathrm{d}u^{k+1}}\right)^\top \otimes h^k h^{k\top}\right] \\
&\preceq \mathbb{E}\left[a^2\left\|\frac{\mathrm{d}z_i}{\mathrm{d}h^{k+1}}\right\|^2 I_{d_{k+1}} \otimes h^k h^{k\top}\right] \\
&= a^2\left\|\frac{\mathrm{d}z_i}{\mathrm{d}h^{k+1}}\right\|^2 I_{d_{k+1}} \otimes \mathbb{E}\left[h^k h^{k\top}\right] \\
&= a^{2(L-k)}\left(\prod_{j=k+1}^{L}||w_j||^2\right) I_{d_{k+1}} \otimes \mathbb{E}\left[h^k h^{k\top}\right]
\end{aligned}$$

Hence, by Eq. (2.9)

$$\text{spec}\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w_k}\frac{\mathrm{d}z_i}{\mathrm{d}w_k}^\top\right]\right) \preceq \text{spec}\left(a^{2(L-k)}\prod_{j=k+1}^{L}||w_j||^2 I_{d_{k+1}} \otimes \mathbb{E}\left[h^k h^{k\top}\right]\right)$$

so we have

$$\text{spec}\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w_k}\frac{\mathrm{d}z_i}{\mathrm{d}w_k}^\top\right]\right) \preceq a^{2(L-k)}\prod_{j=k+1}^{L}||w_j||^2 \text{spec}\left(I_{d_{k+1}}\right) \otimes \text{spec}(\mathbb{E}\left[h^k h^{k\top}\right])$$

□

Corollary 2.11. Denote the FIM and Hessian w.r.t the kth layer $F(w_k), H(w_k)$ respectively, then we have,

$$\text{spec}\left(F(w_k)\right), \text{spec}\left((\log 2)H(w_k)\right) \preceq m a^{2(L-k)}\prod_{j=k+1}^{L}\left\|w^j\right\|_2^2 \text{spec}(I_{d_{k+1}}) \otimes \text{spec}\left(\mathbb{E}\left[h^k h^{k\top}\right]\right).$$

As in Theorem 2.10, an empty product is defined to be one.

Proof. From Theorem 2.3 we know that

$$F(w_k), (\log 2)H(w_k) \preceq \mathbb{E}\left[(\partial_{w_k} z)(\partial_{w_k} z)^\top\right]$$

Let $s = \sum_{i=1}^m z_i$ be the sum of logits, then we have

$$\begin{aligned} F(w_k), (\log 2)H(w_k) &\preceq \mathbb{E}\left[\left(\frac{\mathrm{d}s}{\mathrm{d}w_k}\right)\left(\frac{\mathrm{d}s}{\mathrm{d}w_k}\right)^\top\right] \\ &\preceq m a^{2(L-k)} \prod_{j=k+1}^{L} \left\|w^j\right\|_2^2 \mathrm{spec}(I_{d_{k+1}}) \otimes \mathrm{spec}\left(\mathbb{E}\left[h^k {h^k}^\top\right]\right) \end{aligned}$$

The second inequality comes from a similar calculation as in Theorem 2.10 for network with one added layer where $h_{L+1} = u_{L+1} = z$, $u_{L+2} = w_{L+1} h_{L+1}$, and $w_{L+1} = [1, ..., 1]$, $\|w_{L+1}\|_2^2 = m$. □

By developing similar result for the sum of logits $s = \sum_{i=1}^m z_i$ as in Theorem 2.10 (See Theorem 2.11 for details), we can see that eigenspectra of the block-diagonal approximation of the FIM (which are simply a concatenation of the eigenspectra of different blocks) are controlled by the eigenspectra of the corresponding layer's activation correlations. Our experiments in Fig. 2.1 and Sec. 2.4 show that activations of all layers (except the logits) of a trained deep network are sloppy. As we discuss in the proof, we can repeat this analysis for the Gauss-Newton matrix which is the dominant G-term in the Hessian to also show that its block-diagonal approximation is sloppy if the activations are sloppy.

Remark 2.12 (Modification using sloppiness of activation gradients). Fig. 2.1 shows that the slope of decay of FIM and the activations are essentially the same. However, in Eq. (2.10) if $\mathrm{spec}\left(\mathbb{E}\left[h^k {h^k}^\top\right]\right)$ decays as $\mathcal{O}(\exp(-ci))$, the decay of $\mathrm{spec}\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w^k}\frac{\mathrm{d}z_i}{\mathrm{d}w^k}^\top\right]\right)$ is $\mathcal{O}(\exp(-ci/d_{k+1}))$. This is a loose bound, especially when d_{k+1} is large, e.g., the spectrum could decay much more faster. But note that if we can write a KFAC-approximation

$$\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w^k}\frac{\mathrm{d}z_i}{\mathrm{d}w^k}^\top\right] \approx \mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}^\top\right] \otimes \mathbb{E}\left[h^k {h^k}^\top\right].$$

then we obtain a stronger decay for the logit gradient when d_{k+1} is large, if we assume that the activation *gradients* are sloppy (see Fig. 2.1).

Calculation. As we discussed in Theorem 2.12, the expression in Eq. (2.10) gives a very loose bound when d_k is large. Write $\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w_k}\frac{\mathrm{d}z_i}{\mathrm{d}w_k}^\top\right]$ using KFAC approximation

$$\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w^k}\frac{\mathrm{d}z_i}{\mathrm{d}w^k}^\top\right] \approx \mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}^\top\right] \otimes \mathbb{E}\left[h^k {h^k}^\top\right].$$

If $\mathrm{spec}\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}\frac{\mathrm{d}z_i}{\mathrm{d}u^{k+1}}^\top\right]\right)$ decays as $\exp\{-c_1 i\}$ and $\mathrm{spec}\left(\mathbb{E}\left[h^k {h^k}^\top\right]\right)$ decays as $\exp\{-c_2 j\}$, then

the $(i+j)^2$th largest eigenvalue of $\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w^k}\frac{\mathrm{d}z_i}{\mathrm{d}w^k}^\top\right]$ is smaller than $\exp(-\min\{c_1,c_2\}(i+j))$, hence the kth largest eigenvalue of $\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w^k}\frac{\mathrm{d}z_i}{\mathrm{d}w^k}^\top\right]$ is smaller than $\exp\left(-\min\{c_1,c_2\}\sqrt{k}\right)$. Hence, the decay rate of spec $\left(\mathbb{E}\left[\frac{\mathrm{d}z_i}{\mathrm{d}w^k}\frac{\mathrm{d}z_i}{\mathrm{d}w^k}^\top\right]\right)$ is $\mathcal{O}\left(\exp\left(-\min\{c_1,c_2\}\sqrt{k}\right)\right)$. □

Remark 2.13 (Eigenspectra of inner product kernel is controlled by data). By Theorem 2.1 of Karoui (2010), for n i.i.d. random vectors $x_i \in \mathbb{R}^{d_0}$, we denote the data matrix as $X = [x_1^\top, .., x_n^\top]^\top$. The kernel matrix with entries $M_{i,j} = f\left(\frac{x_i^\top x_j}{d_0}\right)$ can be approximated by

$$K = \left(f(0) + f''(0)\frac{\mathrm{tr}(\Sigma_{d_0}^2)}{2d_0^2}\right)11^\top + f(0)\frac{XX^\top}{d_0} + v_{d_0}I_n$$

where

$$v_{d_0} = f\left(\frac{\mathrm{tr}(\Sigma_{d_0})}{d_0}\right) - f(0) - f'(0)\frac{\mathrm{tr}(\Sigma_{d_0})}{d_0}$$

In other words, $\|M-K\|_2 \to 0$ in probability when $d_0, n \to \infty$ for fixed d_0/n. v_{d_0} is small when $\frac{\mathrm{tr}(\Sigma_{d_0})}{d_0}$ is small. Hence, the eigenepectrum of M is preserved from $XX^\top$.

spec $\left(\mathbb{E}\left[h^1 h^{1\top}\right]\right)$ = spec (M'/n) except for the zero eigenvalues, where $M'_{i,j} = \sigma(w^0 x_i)\sigma(w^0 x_j)$, which means that the activation spectrum can also be written in the form of the spectrum of a kernel matrix (although it is not a inner product kernel as M). This strongly indicates that the decay pattern of activations can be inherited from that of the data correlation matrix $\frac{1}{n}XX^\top$.

Remark 2.14 (Infinitely wide network with bounded weight norm). Although our experiments show that activations are sloppy if the data are, it seems rather difficult to prove in general. If we assume that the ℓ_2 norm of the weights is bounded, we can iterate over Eq. (2.3) for layers $k, k-1, \ldots, 0$ down to the input layer to see that if eigenvalues of $\mathbb{E}\left[h^k h^{k\top}\right]$ are to be summable, then the i^{th}th eigenvalue of $\mathbb{E}\left[h^k h^{k\top}\right]$ is smaller than c/i for some constant c. Under the setting that the width of the k^{th} layer does to infinity, we have that the eigenvalues of $\mathbb{E}\left[h^k h^{k\top}\right]$ decays faster than $\mathcal{O}(1/i)$.

2.3.2. PAC-Bayes generalization bounds that exploit sloppiness

We next discuss four different ways that exploit sloppiness of a deep network to pick the prior and posterior in PAC-Bayes theory.

Method 1: Posterior covariance has the same eigenvectors as that of the Hessian at end (Analytical) Consider a deep network trained to minimize the loss $\check{e}(h_{w'}, D_n)$, and assume that w is a local minimum. The Hessian H_w is then positive semi-definite, with eigendecomposition $H_w = U_w \Lambda_w U_w^\top$, where $\Lambda_w = \mathrm{diag}(\lambda_1, \ldots, \lambda_p)$ and $\lambda_1 \geq \cdots \geq \lambda_p \geq 0$. We consider a Gaussian posterior $Q = N(w, \Sigma_q)$ and an isotropic Gaussian prior $P = N(w_0, \epsilon^{-1}I)$. For two multivariate Gaussians $Q = N(\mu_q, \Sigma_q)$ and $P = N(\mu_p, \Sigma_p)$, their KL divergence is

$$\mathrm{KL}(Q,P) = \frac{1}{2}\left(\mathrm{tr}(\Sigma_p^{-1}\Sigma_q) - p + (\mu_p - \mu_q)^\top \Sigma_p^{-1}(\mu_p - \mu_q) + \log\frac{\det\Sigma_p}{\det\Sigma_q}\right). \tag{2.11}$$

To obtain an analytical posterior covariance, we use $\sqrt{x} \leq x + 1/4$ in Eq. (2.1) and omit terms that do not depend on Σ_q. This reduces the calculation to minimizing the surrogate objective

$$L(\Sigma_q) := \breve{e}(Q, D_n) + \frac{\mathrm{KL}(Q, P)}{2(n-1)} \tag{2.12}$$

over $\Sigma_q \succeq 0$. Let $P_w = N(w, \epsilon^{-1} I)$. Because P and P_w have the same covariance and Q is centered at w,

$$\mathrm{KL}(Q, P) = \mathrm{KL}(Q, P_w) + \frac{\epsilon}{2} \|w - w_0\|_2^2.$$

It follows that

$$\begin{aligned} L(\Sigma_q) &= \int \breve{e}(h_{w'}, D_n) \, \mathrm{d}Q(w') + \frac{1}{2(n-1)} \int \log \frac{\mathrm{d}Q}{\mathrm{d}P_w}(w') \, \mathrm{d}Q(w') + \frac{\epsilon}{4(n-1)} \|w - w_0\|_2^2 \\ &= \frac{1}{2(n-1)} \left(\mathrm{KL}(Q, B) - \log Z \right) + \frac{\epsilon}{4(n-1)} \|w - w_0\|_2^2, \end{aligned}$$

where

$$\begin{aligned} B(w') &= \frac{\exp(-2(n-1)\breve{e}(h_{w'}, D_n)) \, P_w(w')}{Z}, \\ Z &= \int \exp(-2(n-1)\breve{e}(h_{w'}, D_n)) \, \mathrm{d}P_w(w'). \end{aligned}$$

Therefore, the unconstrained minimizer of the surrogate objective is

$$Q = B \propto \exp(-2(n-1)\breve{e}(h_{w'}, D_n)) \, P_w, \tag{2.13}$$

which need not be Gaussian for a general loss. To obtain a closed-form Gaussian posterior, we use the local quadratic approximation

$$\breve{e}(h_{w'}, D_n) = \breve{e}(h_w, D_n) + \frac{1}{2} \left\langle w' - w, H_w (w' - w) \right\rangle.$$

Under this approximation, B is Gaussian and its covariance satisfies

$$\Sigma_q^{-1} = 2(n-1) H_w + \epsilon I.$$

Equivalently,

$$\Sigma_q = U_w (\bar{\Lambda}_w)^{-1} U_w^\top, \tag{2.14}$$

where

$$\bar{\lambda}_i = 2(n-1)\lambda_i + \epsilon \quad \text{for all } i \leq p. \tag{2.15}$$

Thus, the analytically optimal posterior covariance has the same eigenvectors as the Hessian and assigns larger variance to sloppier directions.

Using this posterior, we obtain a non-vacuous bound on the generalization error (see Sec. 2.4). For example, the bound for a fully-connected network on MNIST with one hidden layer of 600 neurons is 0.32, $e(Q) \approx 0.089$ while an upper bound computed by Dziugaite and Roy (2017b) by numerically optimizing the right hand-side of Theorem 2.1 is 0.161.

Note that even if we computed the posterior covariance analytically using the surrogate objective in Eq. (2.12), we could substitute this posterior into Eq. (2.1). Given eigenvalues of the Hessian Λ_w, if we fix the eigenvectors of Σ_q to be that of H_w, we can also optimize the eigenvalues of the inverse posterior covariance $\bar{\Lambda}_w$ directly using nonlinear optimization.

Remark 2.15. Hessian and Gaussian posterior covariance matrix for Bayesian inference, which is mentioned in Maddox et al. (2019), shares similar relation as Eq. (2.14). This is because the targets for optimizing PAC-Bayes bound and ELBO used to find Gaussian approximation of Bayesian inference are similar – the right side of pac-bayes is a balance of training loss and KL term, and while training Bayesian network, ELBO creates a balance balance between loss and KL, which results in similar form of posterior.

Remark 2.16 (PAC-Bayes posterior is more spread out along sloppy eigenvectors). By Eq. (2.15), we can think of the scaled prior inverse variance $\epsilon/(2(n-1))$ as a threshold beyond which the sloppy eigenvalues of the Hessian λ_i are small enough and the loss changes so little that the optimal PAC-Bayes posterior focuses on accurately capturing the prior covariance to obtain a small KL-term. For eigenvalues above this threshold, e.g., the stiff eigenvalues, the optimal posterior has to strike a balance between the empirical loss term and the KL term. We will see in Sec. 2.4 that this observation also holds for cases when posteriors are optimized.

2.3.3. Effective dimensionality of a deep network

Motivated by Theorem 2.16, we define the **effective dimensionality for a deep network at local minimum** w as

$$p(n,\epsilon) = \sum_{i=1}^{p} \mathbf{1}_{\left\{|\lambda_i| \geq \frac{\epsilon}{2(n-1)}\right\}}, \tag{2.16}$$

i.e., the number of eigenvalues of the Hessian H_w with magnitude at least $\frac{\epsilon}{2(n-1)}$. Define the **strength of the model at** w as

$$s(n,\epsilon) = \sum_{i=1}^{p(n,\epsilon)} \log\left(\frac{2(n-1)\lambda_i}{\epsilon} + 1\right). \tag{2.17}$$

We characterize the decay pattern of sorted eigenvalues of H_w at the tail by the **sloppy factor** as defined below.

Definition 2.17 (Sloppy factor). If $\lambda_i(A)$ denote eigenvalues of a positive semi-definite matrix $A \in \mathbb{R}^{p\times p}$ in descending order $\lambda_1 \geq \cdots \geq \lambda_p$, then the sloppy factor of A at index r is

$$c(A,r) = \sup\{c' \geq 0 : \lambda_i(A) \leq \lambda_r(A) e^{-c'(i-r)} \text{ for } i \geq r \geq 1\} \tag{2.18}$$

This definition implicitly means that the small eigenvalues beyond λ_r are uniformly distributed across an exponentially large range (λ_r, λ_p) if $c(A,r) > 0$. Let us emphasize that sloppiness is a phenomenon pertaining to the non-zero eigenvalues of a matrix and is relevant even if the matrix is singular, e.g., the FIM loses rank for non-identifiable models like deep networks (Amari et al., 2002).

We call a positive semidefinite matrix **sloppy** if the eigenspectra of the matrix have a sharp drop at initial then decays linearly afterwards in the log scale. In our experiments in Fig. 2.1 and Sec. 2.4,

the Hessian, FIM and the correlation matrix of the input data are all sloppy.

Let $r = p(n, \epsilon)$ and abbreviate $c(H_w, r)$ as $c(n, \epsilon)$. To control the tail of the Hessian eigenspectrum, we assume the stronger exponential-decay condition

$$\lambda_i \leq \frac{\epsilon}{2(n-1)} \exp(-c(n,\epsilon)(i-r)) \quad \text{for all } i > r, \tag{2.19}$$

where $c(n, \epsilon) > 0$. It follows that

$$\sum_{i=r+1}^{p} \lambda_i \leq \frac{\epsilon}{2(n-1)} \sum_{k=1}^{\infty} e^{-c(n,\epsilon)k} = \frac{\epsilon}{2(n-1)(e^{c(n,\epsilon)}-1)} \leq \frac{\epsilon}{2(n-1)c(n,\epsilon)}.$$

Recall from Method 1 that, under the local quadratic approximation, the posterior $Q = N(w, \Sigma_q)$ minimizing the surrogate objective in Eq. (2.12) satisfies

$$\Sigma_q = U_w \bar{\Lambda}_w^{-1} U_w^\top, \qquad \bar{\lambda}_i = 2(n-1)\lambda_i + \epsilon.$$

The increase in the expected empirical loss is therefore

$$\begin{aligned} \breve{e}(Q, D_n) - \breve{e}(h_w, D_n) &= \frac{1}{2} \sum_{i=1}^{p} \frac{\lambda_i}{\bar{\lambda}_i} \\ &\leq \frac{r + 1/c(n,\epsilon)}{4(n-1)}. \end{aligned}$$

For the KL term, Eq. (2.11) gives

$$\begin{aligned} \frac{\mathrm{KL}(Q,P)}{2(n-1)} &= \frac{1}{4(n-1)} \left(\epsilon \|w - w_0\|_2^2 - p + \sum_{i=1}^{p} \left(\log \frac{\bar{\lambda}_i}{\epsilon} + \frac{\epsilon}{\bar{\lambda}_i} \right) \right) \\ &\leq \frac{1}{4(n-1)} \left(\epsilon \|w - w_0\|_2^2 + \sum_{i=1}^{r} \log \left(\frac{2(n-1)\lambda_i}{\epsilon} + 1 \right) + \sum_{i=r+1}^{p} \frac{2(n-1)\lambda_i}{\epsilon} \right) \\ &\leq \frac{1}{4(n-1)} \left(\epsilon \|w - w_0\|_2^2 + s(n,\epsilon) + \frac{1}{c(n,\epsilon)} \right). \end{aligned}$$

The first inequality uses $-1 + 1/(1+x) \leq 0$ for the stiff directions and $\log(1+x) \leq x$ for the sloppy directions; the second uses Eq. (2.19). Combining the loss and KL terms yields

$$L(\Sigma_q) = \breve{e}(Q, D_n) + \frac{\mathrm{KL}(Q,P)}{2(n-1)} \leq \breve{e}(h_w, D_n) + \frac{p(n,\epsilon) + s(n,\epsilon) + 2/c(n,\epsilon) + \epsilon \|w - w_0\|_2^2}{4(n-1)}. \tag{2.20}$$

Under the additional assumption that the empirical cross-entropy vanishes, $\breve{e}(h_w, D_n) = 0$, only the effective-dimensionality, stiff-subspace-strength, tail-decay, and distance-from-initialization terms remain on the right-hand side.

Lemma 2.18 (Generalization bound in terms of effective dimensionality). Fix $\epsilon > 0$, let $P = N(w_0, \epsilon^{-1} I)$. Suppose that w is a local minimum with positive-semidefinite Hessian $H_w =$

$U_w \Lambda_w U_w^\top$. Suppose further that the expected error of the local quadratic approximation under $Q \sim N(w, \Sigma_q)$ is bounded from above by $\rho \geq 0$, in the sense that

$$\breve{e}(Q, D_n) \leq \breve{e}(h_w, D_n) + \frac{1}{2}\mathrm{tr}(H_w \Sigma_q) + \rho.$$

Then, for every $\delta > 0$, with probability at least $1 - \delta$ over $D_n \sim D^n$,

$$\begin{aligned} e(Q) \leq \breve{e}(h_w, D_n) + \rho + \frac{p(n,\epsilon) + 1/c(n,\epsilon)}{4(n-1)} \\ + \sqrt{\frac{\frac{1}{2}\left[\epsilon \|w - w_0\|_2^2 + s(n,\epsilon) + \frac{1}{c(n,\epsilon)}\right] + \log(n/\delta)}{2(n-1)}}. \end{aligned} \tag{2.21}$$

Proof. On the event in Theorem 2.1, the cross-entropy upper bound on the empirical classification error gives

$$e(Q) \leq \breve{e}(Q, D_n) + \sqrt{\frac{\mathrm{KL}(Q, P) + \log(n/\delta)}{2(n-1)}}.$$

By the assumed control of the quadratic-approximation error and the expected empirical-loss calculation above,

$$\breve{e}(Q, D_n) \leq \breve{e}(h_w, D_n) + \rho + \frac{p(n,\epsilon) + 1/c(n,\epsilon)}{4(n-1)}.$$

The Gaussian KL calculation above and the definition of $s(n,\epsilon)$ also give

$$\frac{\mathrm{KL}(Q, P)}{2(n-1)} \leq \frac{\epsilon \|w - w_0\|_2^2 + s(n,\epsilon) + 1/c(n,\epsilon)}{4(n-1)}.$$

Substituting these two estimates into the PAC-Bayes inequality yields Eq. (2.21). □

From the above calculations we know that the generalization of deep network at a point w depends on the strength of top few eigenvectors and the decay of the tail. The sloppiness of H_w guarantees that for the optimal ϵ, the effective dimensionality $p(n,\epsilon)$ is small, which ensures that both $s(n,\epsilon)$ and $1/c(n,\epsilon)$ are small compare to n while $\epsilon\|w - w_0\|^2$ is not too large (for FC-600-2 on MINST, $p(n,\epsilon) = 2429$, $s(n,\epsilon) = 6810$, $1/c(n,\epsilon) = 2545$, $\epsilon\|w - w_0\|^2 = 8526$, with $n = 55000, \epsilon = 101.3$). Such ϵ can be found when $\epsilon/2(n-1)$ is around the elbow of the eigenspectra of H_w. This ensures a good generalization bound. For comparison, if we repeat the calculation for an isotropic Hessian $\lambda_i \equiv \lambda$, either $s(n,\epsilon)$ or $1/c(n,\epsilon)$ will be very large ($\mathcal{O}(p)$). Tight upper and lower bounds for the VC-dimension of deep networks are of the form $\mathcal{O}(L\, p \log p)$ which is likely much larger than the complexity term $\mathcal{O}(s(n,\epsilon) + 2/c(n,\epsilon) + \epsilon\|w - w_0\|^2)$. While the VC-dimension (up to constant) of an MNIST network (FC-600-2) with one hidden layer is about 66 million, the value of $s(n,\epsilon) + 2/c(n,\epsilon) + \epsilon\|w - w_0\|^2$ is about 20K for $\epsilon = 101.3$. This suggests that **even if the hypothesis class of deep networks is very large, sloppiness of H_w, which is inherited from sloppiness of the input data, ensures a small effective dimensionalty and severely restricts the set of hypotheses that are accessible during training**. The corresponding quantities for the other models are reported in Table 2.3.

Remark 2.19 (Why does the effective dimensionality depend upon ϵ?). Our definition of the effective dimensionality may seem unusual because it depends upon ϵ, which is essentially a user-chosen parameter. This is an artifact of the PAC-Bayes analysis. As $\epsilon \to 0$, the effective dimensionality converges to the number of weights p, but for non-zero values of ϵ, where the prior covariance restricts the set of hypotheses that the PAC-Bayes learner searchers over, this expression coupled with the analytical calculation in Eq. (2.15) may provide a useful way to perform model selection. By optimizing PAC-Bayes bound (Sec. 2.3.4 Method 3), we usually selects ϵ s.t. $\frac{\epsilon}{2(n-1)}$ is close to the elbow of the eigenspectra H_w (Fig. 2.2). For practical purposes, we can also replace eigenvalues of the Hessian in Eq. (2.15) by those of the FIM (see Fig. 2.1), which is easier to compute.

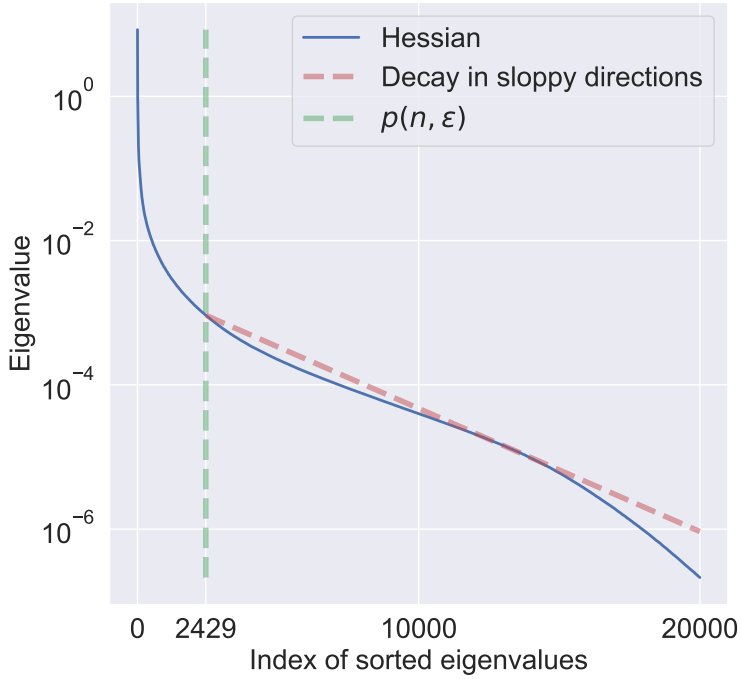

Figure 2.2: The blue line is the eigenspectra of Hessian (KFAC approximation) at the mean of posterior at the end of optimization of PAC-Bayes bound using Sec. 2.3.4 Method 3. We calculate $p(n, \epsilon)$ using the ϵ at the end of optimization, and the red line shows the linear decay of sloppy eigenvalues in the log scale with sloppy factor $c(n, \epsilon) = 0.0004$. The green line, which is close to the elbow of the eigenspectra, splits the stiff eigenvalues and sloppy ones.

Remark 2.20 (Sloppiness is prevalent for deep neural nets). As we have seen in Fig. 2.1, for deep neural nets such as wide residual networks, the sloppiness of data eigenspectrum is preserved in FIM, Hessian, and other related quantities. The prevalence of sloppiness for deep networks is also reflected in theory: In Theorem 2.4, if we set Lipchitz factor $a = 1$ (which is the case for ReLU activation), the trace bound depends on the product of the weight norm instead of the number of layers. Theorem 2.2 and Theorem 2.10 shows that the sloppiness of layer-wise eigenspectrum can be preserved through layers, which is a theoretical evidence that deep networks are also sloppy. In addition, the effective dimensionality of LENET-5 is small (0.184 %) and the PAC-Bayes bound is non-vacuous, which means that for deep networks (see Table 2.3 for details), sloppiness also leads to good generalization.

2.3.4. Numerical methods to compute PAC-Bayes bounds

Method 2: Eigenvectors of the posterior covariance are fixed to those of FIM at initialization ($\mathrm{E}(\Sigma_q) = \mathrm{E}(F_{w_0})$) Our experiments show that there is a large overlap between the subspace spanned by the stiff eigenvectors of the FIM at the end of training with the corresponding subspace at the beginning of training (Fig. 2.12). Similarly, there is a large overlap between the subspace spanned by the stiff eigenvectors of the Hessian with that of the FIM (Fig. 2.4). Therefore, as the simplest extension of the analytical computation of the posterior in Method 1, we pick the posterior covariance matrix to have the same eigenvectors as that of FIM at initialization and optimize only its eigenvalues, i.e.,

$$P = N(w_0, \epsilon^{-1} I),\ Q = N(w, \Sigma_q = U_{w_0} \bar{\Lambda}_w U_{w_0}^\top), \tag{2.22}$$

where $F_{w_0} = U_{w_0} \Lambda U_{w_0}^\top$ is the orthonormal decomposition of the FIM at initialization w_0. We use the notation $\mathrm{E}(A)$ to denote the set of eigenvectors of the matrix A, arranged in decreasing order of eigenvalues. We can optimize the PAC-bound in Eq. (2.1) numerically. The variables of optimization are the mean of the posterior w, eigenvalues of the covariance $\bar{\Lambda}_w$ and the scale of the prior ϵ. Note that in this method, we do not modify the eigenvectors of the posterior covariance even if the mean changes during optimization.

Method 3: Eigenvectors of the posterior covariance are the same as those of the Hessian, but the Hessian is revalauted at different instants while optimizing the bound ($\mathrm{E}(\Sigma_q) = \mathrm{E}(H_w)$) This method is similar to Method 2 except that we update the eigenvectors of the posterior covariance matrix while optimizing the bound

$$P = N(w_0, \epsilon^{-1} I),\ Q = N(w, \Sigma_q = U_w \bar{\Lambda}_w U_w^\top), \tag{2.23}$$

where $H_w = U_w \Lambda U_w^\top$ is the orthonormal decomposition of the Hessian at weights w. The variables of optimization here the same, i.e., mean of the posterior w, eigenvalues of the covariance $\bar{\Lambda}_w$, and the scale of the prior ϵ.

Method 4: Prior covariance is proportional to FIM at initialization, eigenvectors of posterior covariance the same as those of FIM at initialization ($\Sigma_p = aF_{w_0} + \epsilon^{-1}$; $\mathrm{E}(\Sigma_q) = \mathrm{E}(F_{w_0})$) This is a data-distribution dependent prior. We exploit the large projection of the change of weights during training onto the stiff eigenvectors of FIM at initialization to construct the prior (prior has more variance in stiff directions where the training process mostly takes place, Fig. 2.12), and use posterior align to F_{w_0}:

$$P = N(w_0, aF_{w_0} + \epsilon^{-1} I),\ Q = N(w, \Sigma_q = U_{w_0} \bar{\Lambda}_w U_{w_0}^\top), \tag{2.24}$$

where $F_{w_0} = U_{w_0} \Lambda U_{w_0}^\top$ is the orthonormal decomposition of the FIM at initialization w_0. The variables that are modified during optimization of the bound are the posterior mean w, the eigenvalues $\bar{\Lambda}_w$, and scalar constants a, ϵ^{-1}. In this case, we incur a larger penalty from the union bound (see Sec. 2.4.3) because there are two constants being optimized in the prior.

2.4. Empirical Study

2.4.1. Setup

We use fully-connected (of varying widths and up to two hidden layers) and convolutional (LeNet, ALL-CNN of Springenberg et al. (2015) and wide residual network of Zagoruyko and Komodakis (2016)) deep networks of varying sizes on MNIST (LeCun et al., 1990) and CIFAR-10 (Krizhevsky, 2009a) datasets to study sloppy eigenspectra. PAC-Bayes bounds are computed on MNIST for binary classification problems (digits 0–4 and 5–9, respectively) [1].

Implementation details We make approximations while working with large matrices like Hessian and FIM. In some cases we compute the full Hessian/FIM, for larger networks we compute only the top 1000–3000 eigenvalues of the Hessian. We also use Kronecker-factor (KFAC) approximation of the Gauss-Newton matrix in Backpack (Dangel et al., 2020) as a replacement for FIM/Hessian; this is the so-called G-term Eq. (2.7). See the technical details at the end of Sec. 2.4.3. We also exploit a PyTorch wrapper described there that allows us to draw a large number of samples to estimate the gradient of $\check{e}(Q, D_n)$ more accurately (we use 150, whereas Dziugaite and Roy (2017b) use 1) and thereby optimize the PAC-Bayes bound for fewer iterations.

Data We use the MNIST dataset for experiments on fully-connected networks and LeNet. We setup a binary classification problem (we map {0,1,2,3,4} to label 0 and {5,6,7,8,9} to label 1). We use 55000 samples from the training set to train the model and to optimize the PAC-Bayes bound. We set aside 5000 samples for calculating the FIM, which is used in Method 4 of PAC-Bayes bound optimization. Strictly speaking, it is not required to do so because a prior that depends upon the FIM is an expectation-prior (as discussed in Parrado-Hernández et al. (2012)) but we set aside these samples to compare in a systematic manner to existing methods in the literature that use 55,000 samples. Test error of all models is estimated using the validation set of MNIST. We use the CIFAR-10 dataset for experiments using two architectures, an All-CNN network and a wide residual network. For CIFAR-10, we use 50, 000 samples for training and 10, 000 samples for estimating the test error. No data augmentation is performed for MNIST, for CIFAR-10 we randomly flip images (left to right) with probability 0.5 and select random crops of size 32×32 after adding a padding of 4 pixels on the width and height.

Architectures For experiments on MNIST, we use LeNet-5 (this is a network with two convolutional layers of 20 and 50 channels respectively, both of 5×5 kernel size, and a fully-connected layer with 500 hidden neurons) and fully-connected net with one or two layers and 600 or 1200 neurons on each layer. The latter are denoted as FC-600-1, or FC-1200-2 in our experimental section. For CIFAR-10, we use ALL-CNN (in order to reduce the number of weights, we reduced the number of channels in the first set of blocks to 64, and in the second set of blocks to 128; this is down from 96 and 192 respectively in the original network) and wide residual net with depth 10 and a widening factor of 8. In the latter case, in order to reduce the number of weights which makes computing Hessian amenable, we reduce the number of channels in each block of the WRN to [4, 32, 64, 128], down from [16, 128, 256, 512] for a widen factor of 8.

Training procedure We train for 30 epochs on MNIST and for 100 epochs on CIFAR-10. The batch-size is fixed to 500 for both datasets. For all experiments with train with Adam and reduce the learning rate using a cosine annealing schedule starting from an initial learning rate of 10^{-3} and ending at a learning rate of 10^{-5}.

[1] The code for all experiments in this chapter can be found at https://github.com/rubingy/sloppy.git

2.4.2. Hessian and FIM are sloppy if the data correlation matrix is sloppy

Sloppiness across architectures and datasets Figure 2.3 complements Fig. 2.1 by showing the eigenspectra for a two-layer fully-connected network with 600 hidden neurons on MNIST. The two figures are qualitatively similar: despite its lower dimensionality, MNIST has roughly the same range of eigenvalues as CIFAR-10, but it has a much smaller threshold r in Theorem 2.17, indicating that the data have fewer effective dimensions. The FIM (the empirical FIM is essentially the same curve) decays particularly strongly for MNIST. Since the trace of the FIM has been used as an indicator of the information stored in the weights (Achille et al., 2018), this suggests that relatively little information must be stored in the weights to predict MNIST well. Although the Hessian and FIM have very different eigenvalues on MNIST, Fig. 2.4 shows that their leading eigenspaces have substantial overlap.

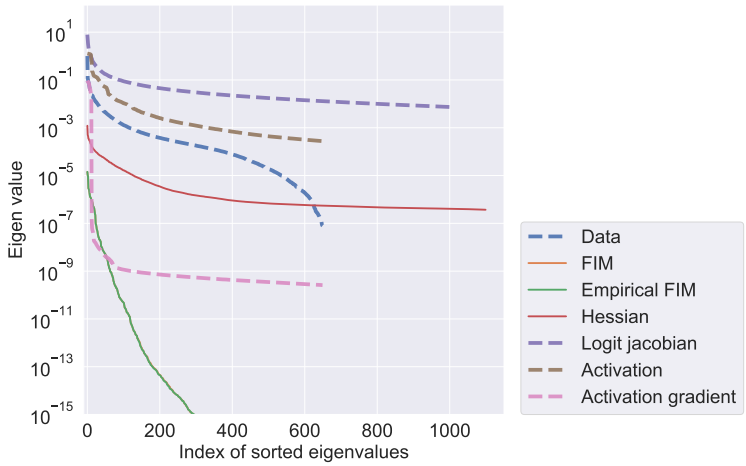

Figure 2.3: Eigenspectra for a two-layer fully-connected network on MNIST. The eigenspectra are qualitatively the same as those of Fig. 2.1, e.g., there is a sharp drop at the beginning and a long, linear tail of small eigenvalues follows. Slopes of the eigenspectra of activations, activation gradients, Jacobians and Hessian mirror those of the data. In contrast to Fig. 2.1, the slope of the FIM is quite different here. The Empirical FIM and FIM overlaps with each other since the model is trained to nearly perfect train and validation error.

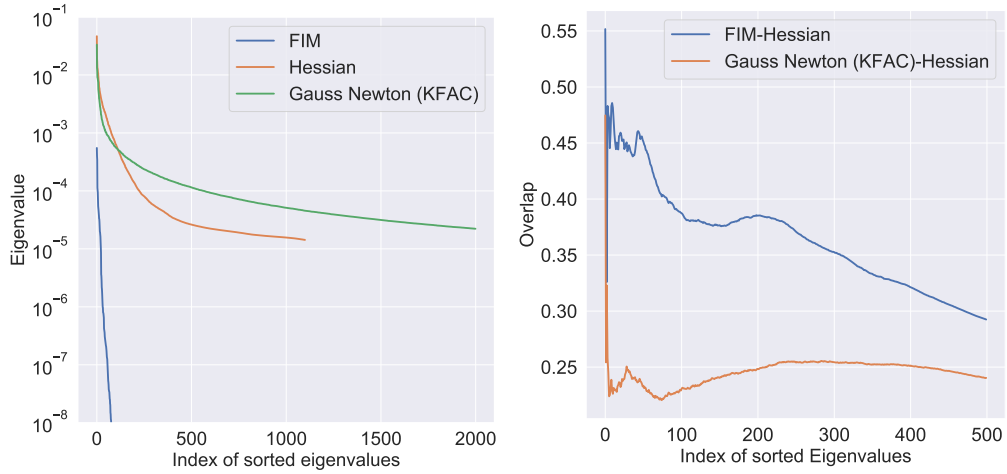

Figure 2.4: (Left) Eigenspectra of a two-layer fully-connected network on MNIST (same network as Fig. 2.3) for the FIM, Hessian and a KFAC approximation of the Gauss-Newton matrix. Even if FIM's eigenvalues are quite different, its eigenvectors have a large inner product with those of the Hessian (right), much larger than a random vector. Even if KFAC is a good approximation for the eigenvalues of the Hessian, the eigenvectors computed from KFAC are quite different from those of the Hessian.

Sloppiness is not limited to the Hessian and FIM. The correlation matrices of the logit Jacobians also have rapidly decaying eigenspectra, and this behavior is consistent across different logits (Fig. 2.5). Likewise, the correlation matrices of activations and activation gradients are sloppy across different layers of the network (Figs. 2.6 and 2.7). These results show that the spectral structure of the data persists in both the forward representations and the derivatives propagated through the network. We then examine whether this phenomenon depends on the architecture. ALL-CNN and the wide residual network exhibit qualitatively similar spectra on CIFAR-10 (Figs. 2.1 and 2.8), while FC-1200-1 and FC-600-2 do so on MNIST (Figs. 2.3 and 2.9). The agreement between different architectures trained on the same dataset supports the conclusion that the observed sloppiness is inherited primarily from the dataset rather than from a particular architecture. Figure 2.10 further shows that the FIM and Hessian eigenspectra remain qualitatively similar throughout training.

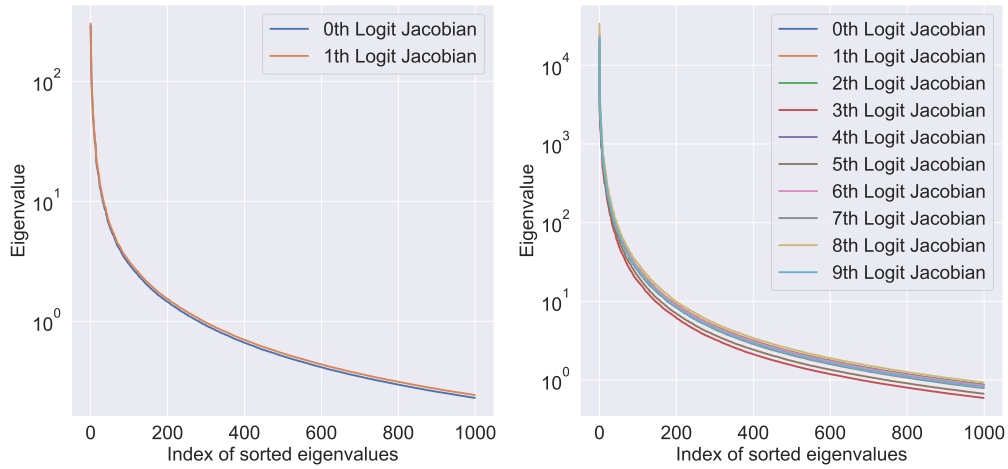

Figure 2.5: Eigenspectra of the correlation matrices of logit Jacobians for FC-600-2 on MNIST (left) and a wide residual network on CIFAR-10 (right). The eigenspectra are similar across logits.

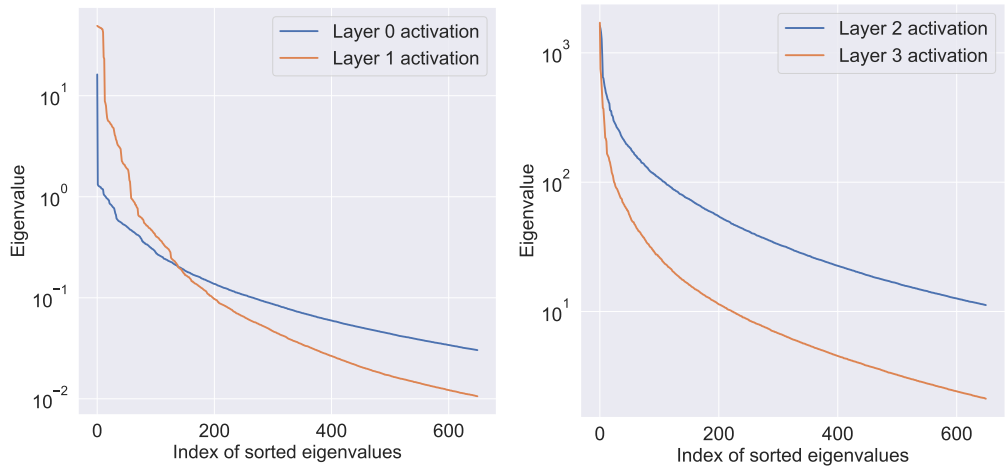

Figure 2.6: Eigenspectra of the correlation matrices of activations in different layers for FC-600-2 on MNIST (left) and a wide residual network on CIFAR-10 (right). The eigenspectra are similar across layers.

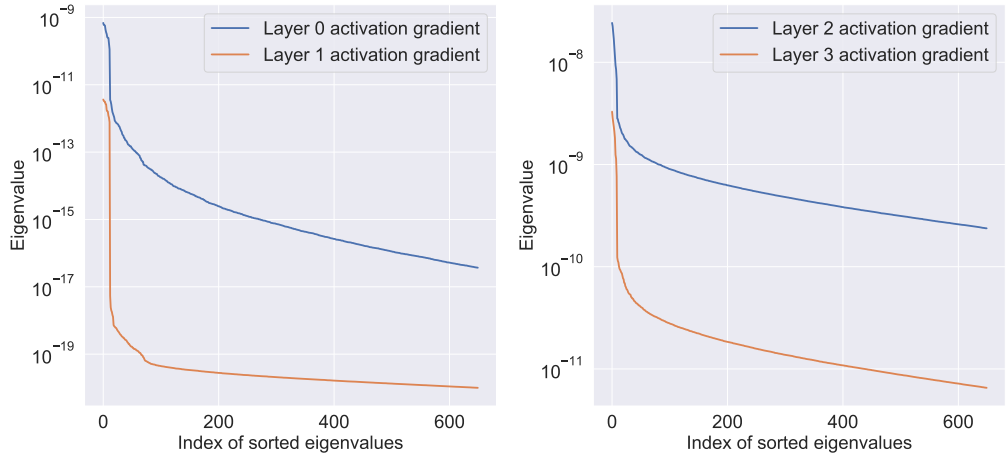

Figure 2.7: Eigenspectra of the correlation matrices of gradients with respect to activations in different layers for FC-600-2 on MNIST (left) and a wide residual network on CIFAR-10 (right). The eigenspectra are qualitatively similar across layers, while their eigenvalues become smaller in higher layers.

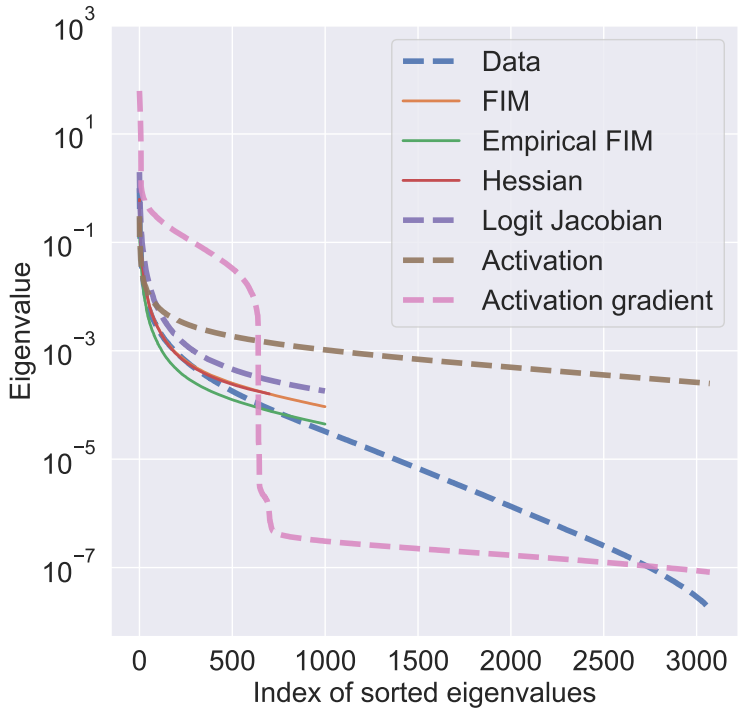

Figure 2.8: Eigenspectra for ALL-CNN on CIFAR-10. They are qualitatively similar to those of the wide residual network on CIFAR-10 in Fig. 2.1.

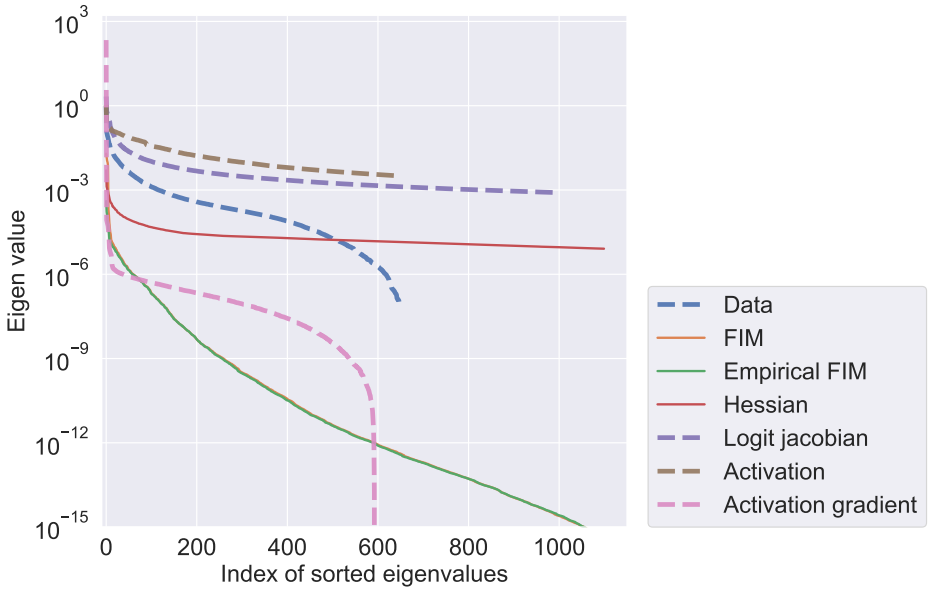

Figure 2.9: Eigenspectra for FC-1200-1 on MNIST. They are qualitatively similar to those of FC-600-2 on MNIST in Fig. 2.3.

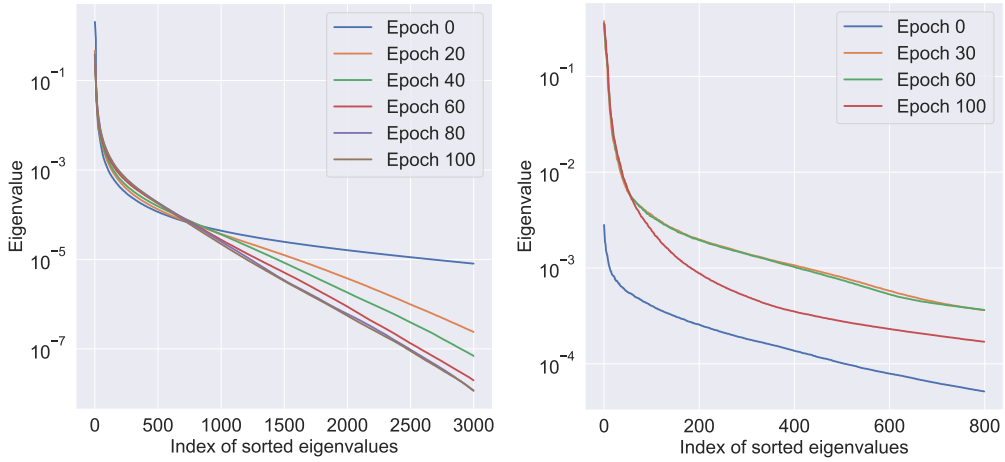

Figure 2.10: (Left) Eigenspectra of the FIM for a wide residual network on CIFAR-10 throughout training. The eigenspectrum at epoch 0 is scaled by 10^3 to bring it to the displayed scale. (Right) Eigenspectra of the Hessian throughout training, using the absolute values of its eigenvalues. The eigenspectra remain qualitatively similar throughout training.

Figure 2.11 shows the eigenspectra of empirical FIM at the end of training, in which the eigenspectra is less sloppy for data sets with random labels, and the top eigenvalues increases as we increase the fraction of random labels. This shows that even if we have the same input data set, the sloppiness and the top few eigenvectors can still be affected by the task. The eigenspectra becomes less sloppy and has larger head when the the model is used to learn more difficult tasks (more random labels).

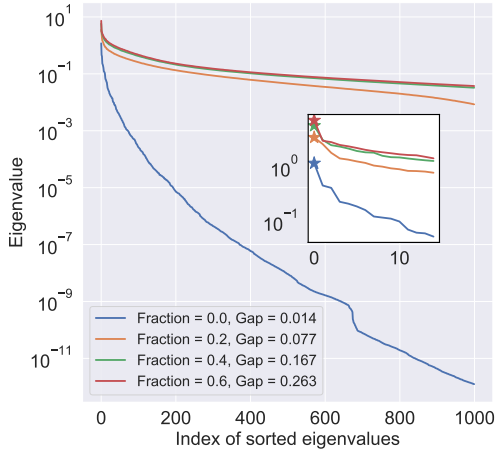

Figure 2.11: Eigenspectra at the end of training for data sets with random labels. The plots shows the eigenspectra of empirical FIM at the end of training. The experiment is done on MNIST using fully connected net FC-600-2. The label "Fraction=a" indicates the data set with random label of fraction a. The inset plot shows the top 15 eigenvalues. The line for Fraction=0.0 are scaled up by 10^7. The plots shows that the FIM at the end of training is less sloppy for data sets with random labels, and the top eigenvalues increases as we increase the fraction of random labels.

Overlap of the stiff subspace of the FIM/Hessian at the end of training with that at initialization is large Fig. 2.12 shows the results of this experiment. We also constructed a third network, denoted -v2, to test whether weights can move back toward the initialization in directions that are unimportant for the loss. The v2 model is trained in two phases. In the first phase, we initialize the model at w_0 and train it as usual to obtain the trained weights w^1. In the second phase, starting from w^1, we train for 20 additional epochs with an objective equal to the original training objective plus a spring-force-like penalty:

$$\hat{e}(h_w, D_n) + \alpha \|w - w_0\|_2^2.$$

The second term pulls the weights toward the initialization w_0 while maintaining nearly the same error; we set α to be twice the learning rate. We find in Fig. 2.12 (right) that this variant has a much larger projection into the stiff eigenspace, and thereby a smaller overlap with the sloppy eigenvectors. Figure 2.13 further decomposes this effect: the cumulative projection of $\Delta w = w - w_0$ for the v2 model onto the stiff directions of F_{w_0} is larger than that of the original FC model, and the projection onto individual stiff eigenvectors is also more concentrated. Since the projections onto the stiff and sloppy orthogonal subspaces sum to one, this means that the v2 model has moved back toward initialization primarily in the sloppy subspace. Thus weights can effectively "come back" toward the initialization in directions that have little effect on the loss, even though training predominantly evolves in the stiff subspace.

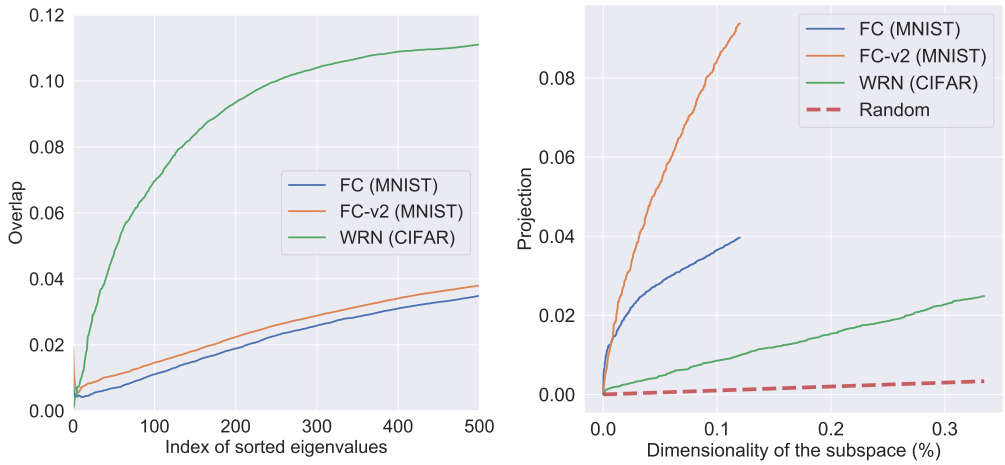

Figure 2.12: (Left) Overlap between the subspace of the top k eigenvectors (X-axis) of the FIM at the end of training with that at initialization ($\left\|\mathrm{E}_k(F_w)^\top \mathrm{E}_k(F_{w_0})\right\|_F^2/k$) for fully-connected (FC) and convolutional networks (WRN). (Right) Projection $\|\mathrm{E}_k(F_{w_0})\Delta w\|_2^2/\|\Delta w\|_2^2$ of the change in weights during training (where $\Delta w = w - w_0$) into the sub-space of the top k eigenvectors (shown as percentage because different networks have different size) of the FIM. They are much larger than the projection onto a random vector. Thus weights change predominantly in the stiff eigenspace of the FIM at initialization; also see Fig. 2.4 which shows that the FIM eigenvectors have a strong overlap with those of the Hessian.

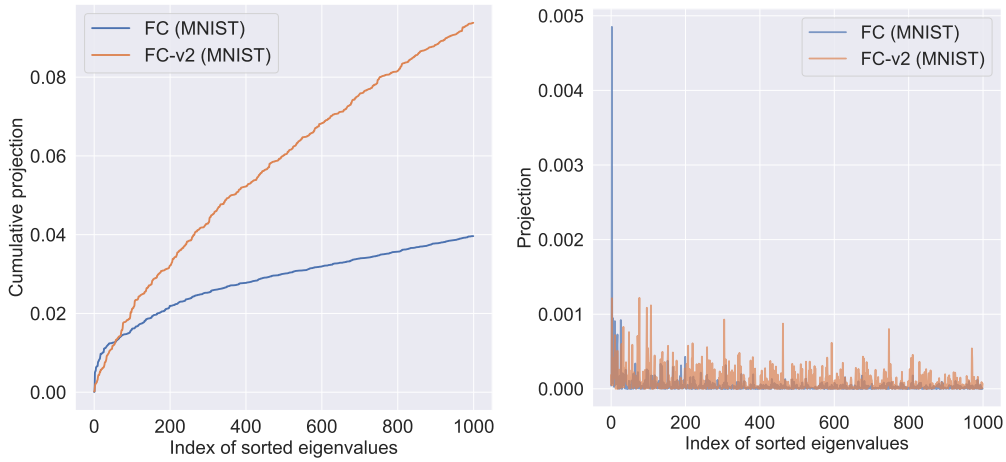

Figure 2.13: (Left) Cumulative projection $\|E_k(F_{w_0})\Delta w\|_2^2/\|\Delta w\|_2^2$ of the change in weights during training (where $\Delta w = w - w_0$) onto the eigenspace of the top kth eigenvalues of F_{w_0}. (Right) Projection $\|\nu_k(F_{w_0})\Delta w\|_2^2/\|\Delta w\|_2^2$ of the change in weights during training onto the eigenvector $\nu_k(F_{w_0})$ of kth largest eigenvalue of F_{w_0}, which is the derivative of the curve in the left plot. We use FC-600-2 on MNIST for this experiment.

A study of the data correlation matrix, FIM and Hessian for atypical problems We construct synthetic training and validation sets containing 50,000 and 10,000 samples, respectively. Each input $x_i \in \mathbb{R}^{200}$ is sampled from $N(0, \Lambda)$, where $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_{200})$ and $\lambda_i = b\exp(-ci)$. We fix $b/c = 50$ so that the trace of the data correlation matrix remains approximately constant

across datasets, while varying the "sloppy factor" c controls the decay of its eigenspectrum. Larger values of c correspond to sharper spectral decay. We randomly initialize a two-layer fully-connected teacher network with one hidden layer and ten output classes, and generate labels according to $y_i = \arg\max_{y \in [m]} p_w^t(y \mid x_i)$. Although the input covariance is diagonal, the weights in the teacher's first layer mix the inputs, so the correlation matrix of the first-layer activations is non-diagonal. We train two-layer fully-connected student networks for 50 epochs using Adam with a batch size of 500 and a cosine learning-rate schedule decreasing from 10^{-3} to 10^{-5}, until they interpolate the training data.

Our goals are to study (i) how the quantities discussed in this chapter, including the Hessian, FIM, activations, activation gradients, and logit Jacobians, depend on the sloppiness of the data matrix; and (ii) whether the student can interpolate sloppy datasets without overfitting. Figure 2.14 shows the results. Using the ϵ derived from PAC-Bayes bound optimization, we also calculate that the effective dimensionalities $p(n, \epsilon)$ are 542 for $c = 0.1$ and 1730 for $c = 0.001$, the strengths $s(n, \epsilon)$ are 2325 for $c = 0.1$ and 4962 for $c = 0.001$, and the inverse sloppy factors of the Hessian at the end of training $1/c(n, \epsilon)$ are 209 for $c = 0.1$ and 452 for $c = 0.001$. These results show that the spectra have heavier tails and more stiff directions as the datasets become less sloppy; the complete results are reported in Table 2.1.

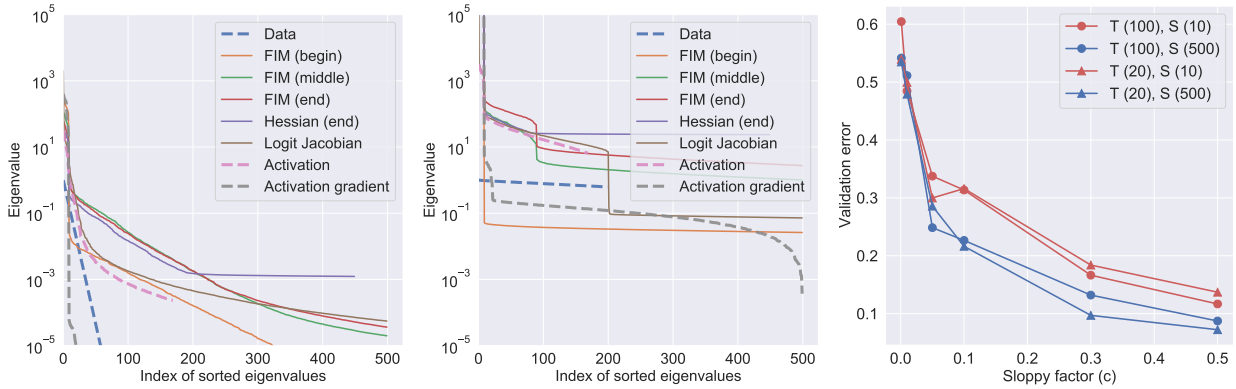

Figure 2.14: (Left, Middle) Eigenspectra of the data correlation matix, FIM at begining, middle and end of training and the Hessian, correlation matrix of logit Jacobian, activation, activation gradient at the end of training, for sloppy factor $c = 0.1$ (left) and $c = 10^{-3}$ (right). We see that if the data is sloppy, both FIM/Hessian are sloppy but if data is not sloppy then even if there is a sharp drop after the top few eigenvalues (around 100 for middle panel), the eigenspectrum is flat. In comparison, the FIM/Hessian decay by about 3 orders of magnitude on the left.

(Right panel) Validation error on synthetic datasets of different sloppiness (X-axis) for different number of hidden neurons (numbers in brackets in the legend) for the two-layer teacher (T) and student networks (S). All students in this plot interpolate the training data perfectly. For isotropic data correlation matrices, i.e., small sloppiness factor, interpolation results in poor generalization. For sloppy data, interpolation is not detrimental to generalization. Also note that as the number of student neurons increases, fixed the teacher's size and the sloppiness factor, the validation error is better. Fixed teacher size, say 20, if inputs are sloppier (sloppy factor of 0.5 vs. 0.3) then we can generalize—roughly equally well in this experiment—even if the student is smaller (10 vs. 500).

Quantity/Model	Random-0.1	Random-0.001
Training and validation error of the trained model		
$\hat{e}(h_w, D_n)$	0.0000	0.0749
$\log 2 * \check{e}(h_w, D_n)$	0.0869	0.5745
$e(h_w)$	0.1150	0.5035
$\log 2 * \check{e}(h_w)$	0.2983	1.3797
$\mathbf{E}(\Sigma_q) = \mathbf{E}(F_{w_0})$ **(Method 2)**		
$\hat{e}(Q, D_n)$	0.0155	0.0117
$\log 2 * \check{e}(Q, D_n)$	0.0509	0.2368
$e(Q)$	0.1191	0.4596
$\log 2 * \check{e}(Q)$	0.3965	1.2574
PAC-Bayes bound	0.3560	0.6781
$\mathrm{KL}(Q, P)$	18468.6914	53052.6094
$\mathbf{E}(\Sigma_q) = \mathbf{E}(H_w)$ **(Method 3, our implementation)**		
$\hat{e}(Q, D_n)$	0.0102	0.0128
$\log 2 * \check{e}(Q, D_n)$	0.0420	0.2508
$e(Q)$	0.1133	0.4614
$\log 2 * \check{e}(Q)$	0.3793	1.2556
PAC-Bayes bound	0.2986	0.6769
$\mathrm{KL}(Q, P)$	15311.6416	52592.7852
ϵ	1.60	0.266
$p(n, \epsilon)$	542 (0.256%)	1730 (0.820 %)
$s(n, \epsilon)$	2325	4962
$1/c(n, \epsilon)$	209	452
$\Sigma_p = aF_{w_0} + \epsilon^{-1}, \mathbf{E}(\Sigma_q) = \mathbf{E}(F_{w_0})$ **(Method 4)**		
$\hat{e}(Q, D_n)$	0.0093	0.0083
$\log 2 * \check{e}(Q, D_n)$	0.0262	0.2217
$e(Q)$	0.1199	0.5035
$\log 2 * \check{e}(Q)$	0.6493	1.3797
PAC-Bayes bound	0.2514	0.6577
$\mathrm{KL}(Q, P)$	12346.2207	50933.4063
$\mathbf{diag}(\Sigma_q) = \Lambda$ **(our implementation)**		
$\hat{e}(Q, D_n)$	0.0094	0.0083
$\log 2 * \check{e}(Q, D_n)$	0.0275	0.0452
$e(Q)$	0.1325	0.4599
$\log 2 * \check{e}(Q)$	0.6041	2.1837
PAC-Bayes bound	0.5096	0.7512
$\mathrm{KL}(Q, P)$	32949.1836	66678.6016

Table 2.1: Comparison of PAC-Bayes bounds on synthetic data sets. The table shows the PAC-Bayes bound optimization results for the synthetic data sets introduced in Sec. 2.4.2. The first three method blocks correspond to Methods 2–4 described in Sec. 2.3.2, and the final block is our reproduction of Dziugaite and Roy (2017b) on the synthetic data sets. Using the ϵ derived by Method 3, we calculate the effective dimensionality, strength, and inverse sloppy factor of the Hessian at the end of training, as reported in the Method 3 block. The less-sloppy data set has heavier spectral tails and more stiff directions, resulting in worse generalization.

2.4.3. PAC-Bayes bounds

To assess whether sloppiness can yield informative generalization guarantees, we compute PAC-Bayes bounds for fully-connected networks and LeNet-5 trained on the MNIST binary classification task described in Sec. 2.4. We compare four constructions that exploit the eigenspaces of the Hessian or FIM through the posterior covariance and, in Method 4, a data-distribution-dependent prior. We first describe how each bound is optimized and then report the resulting bounds.

PAC-Bayes Optimization Methods

We optimize the problem,

$$\min \check{e}(Q, D_n) + \sqrt{\frac{\mathrm{KL}(Q, P) + \varphi}{2(n-1)}} \tag{2.25}$$

where Q, P are multivariate normal distribution, φ is the penalty we added for including a trainable parameter in prior (say its scale), and n is the number of samples. For Gaussian distributions on the weight space Q, P, as we saw in Eq. (2.11), the KL-divergence is

$$\frac{1}{2}\left(\mathrm{tr}(\Sigma_p^{-1}\Sigma_q) - p + (w - w_0)^\top \Sigma_p^{-1}(w - w_0) + \log(\det \Sigma_p / \det \Sigma_q)\right).$$

The penalty for the case when $P = N(w_0, \epsilon_j^{-1} I)$ comes from the union bound over the grid $\epsilon_j = c\exp(-j/b)$ for $j = 1, 2, \ldots$ and is given by

$$\varphi = 2\log\left(b\log(c/\epsilon_j)\right) + \log\left(\pi^2 n/(6\delta)\right).$$

Note that for Method 4, we need more than one trainable parameters for the prior, and the penalty φ should also be modified according to Sec. 2.4.3. We calculate $\check{e}(Q, D_n)$ using Monte Carlo samples from Q. After the optimization process, we calculate the PAC-Bayes bound on $e(Q)$ using

$$\mathrm{kl}(\hat{e}(Q, D_n), e(Q)) \leq \frac{\mathrm{KL}(Q, P) + \varphi}{n-1}, \tag{2.26}$$

which involves finding an approximation of $\mathrm{kl}^{-1}(b, a) := \sup\{a' \in [0,1] : \mathrm{kl}(b, a') \leq a\}$ (see Dziugaite and Roy (2017b) for details). We next discuss the various methods for calculating PAC-Bayes bounds developed in this chapter and provide their implementation details.

Method 1 The tightest bound in this case is obtained using the v2 model described in Fig. 2.12. To recall, this involves a second post-training phase where the trained model is updated to be closer to the initialization w_0. In the context of the PAC-Bayes upper bound, this reduces the distance between the means of the Gaussian prior and posterior. We choose Σ_q as in Eq. (2.14) and Eq. (2.15). For $\epsilon_j = c\exp(-j/b)$ and $j = 1, \ldots, 60$, we evaluate $\mathrm{KL}(Q, P_j)$ by using Eq. (2.11), and $\hat{e}(Q, D_n)$ is estimated by sampling. The covariance Σ_q is approximated by the top eigenvalues and eigenvectors of the Hessian as discussed in Sec. 2.4.3. The PAC-Bayes bound is calculated by Eq. (2.1) and we choose the smallest bound among all choices of ϵ_j.

We also set $\Sigma_q = U_w \bar{\Lambda}_w U_w^\top$ and calculate $\bar{\Lambda}$ by directly minimizing Eq. (2.1) where the variables of optimization are $\bar{\lambda}_i$ for $i \leq k$ using nonlinear optimization in scipy (using the BFGS algorithm), and the PAC-Bayes bound is calculated in the same way as above. This is denoted as Method 5 (Numerical) in Table 2.3.

For comparison, we also choose $\Sigma_q = \epsilon^{-1} I$ and calculate the PAC-Bayes bound. This is denoted as Method 6 (Isotropic) in Table 2.3.

Methods 2 and 3 We choose P and Q as described in Sec. 2.3.4. We set $\epsilon^{-1} = \exp(2\rho)$, $\bar{\Lambda}_w = \exp(2\xi)$. The parameters ρ, w, ξ are optimized while optimizing the PAC-Bayes upper bound. We initialize ϵ^{-1} at $\exp(-6)$ and $\bar{\Lambda}_w$ at $(\Lambda^F + \epsilon^{-1})/10$ where Λ^F are the eigenvalues of the FIM at initialization. For fully-connected networks and LeNet, we evaluate $\hat{e}(Q, D_n)$ using the methods described in Sec. 2.4.3 and Sec. 2.4.3 respectively.

We use the Gauss-Newton matrix as an approximation of the FIM for Method 2.

Method 4 We choose P and Q as described in Sec. 2.3.4. We set $a = \exp(2\rho_1)$, $\epsilon^{-1} = \exp(2\rho_2)$, $\sigma = \exp(2\xi)$ and train parameters ρ_1, ρ_2, w, ξ. In our experiments, ϵ^{-1} is initialized to $\exp(-6)$, a is initialized to $\exp(-1)$ and Σ_q is initialized to be $(aF_{w_0} + \epsilon)/10$. In this case,

$$\mathrm{KL}(Q, P) = \frac{1}{2}\left(\sum_i \frac{\sigma_i}{a\lambda_i^F + \epsilon^{-1}} - d + (w - w_0)^\top (aF_{w_0} + \epsilon^{-1})^{-1}(w - w_0) + \sum_i \log \frac{a\lambda_i^F + \epsilon^{-1}}{\sigma_i}\right)$$

where λ_i^F are eigenvalues of F_{w_0}. For fully-connected networks and LeNet, we approximate $(w - w_0)^\top (aF_{w_0} + \epsilon^{-1})^{-1}(w - w_0)$ using the methods described in Sec. 2.4.3 and Sec. 2.4.3 respectively.

We use the Gauss-Newton matrix as an approximation of the FIM for Method 4.

Results

Model	Method					
	1	**2**	**3**	**4**		
	Analytical	$\mathrm{E}(\Sigma_q) = \mathrm{E}(F_{w_0})$	$\mathrm{E}(\Sigma_q) = \mathrm{E}(H_w)$	$\mathrm{E}(\Sigma_q) = \mathrm{E}(F_{w_0})$	$\mathrm{diag}(\Sigma_q) = \Lambda$	$\mathrm{E}(\Sigma_q) = \mathrm{E}(\hat{H}_w)$
FC-600-1	0.3241	0.1590	0.1357	0.1323	0.161	0.1198
FC-600-2	0.3794	0.1767	0.1540	0.1397	0.186	0.1443
FC-1200-1	0.3509	0.1523	0.1515	0.1486	0.179	0.1413
FC-1200-2	0.3915	0.2017	0.1817	0.1702	0.223	-
LENET-5	0.0572	0.0099	0.0188	0.0092	-	-

Table 2.2: Comparison of PAC-Bayes bounds on MNIST for different methods. The first four methods correspond to our Methods 1–4 described in Sec. 2.3.2. The prior for Method 4 is $\Sigma_p = aF_{w_0} + \epsilon^{-1}$; all other methods use $P = N(w_0, \epsilon^{-1} I)$. The notation $\mathrm{E}(A)$ denotes eigenvectors of the matrix A. The penultimate column where the posterior covariance is diagonal corresponds to numbers from Dziugaite and Roy (2017b). The final column corresponds to the approach of Wu et al. (2021) who set the eigenvectors of the posterior covariance to be the same as those of the layer-wise Hessian. For fully-connected nets, the error $e(Q)$ ranges from 6–8 $\times 10^{-2}$ for Method 1 and 1–4 $\times 10^{-2}$ for all other methods. For LENET-5 the error $e(Q)$ ranges from 1–2 $\times 10^{-2}$ for all methods. All bounds hold with probability at least 0.975.

Tables 2.2 and 2.3 show the PAC-Bayes bounds obtained using different methods. For all networks, Method 1 gives a non-vacuous bound, and Methods 2–4 give bounds comparable to existing approaches. Method 4 gives the tightest bounds among our methods by using a data-dependent prior based on the FIM at initialization. These results show that the sloppiness of the Hessian and FIM can be used to obtain informative generalization bounds.

Quantity/Model	FC-600-1	FC-600-2	FC-1200-1	FC-1200-2	LeNet
Training and validation error of the trained model					
$\hat{e}(h_w, D_n)$	0.0000	0.0000	0.0000	0.0000	0.0000
$\log 2 * \check{e}(h_w, D_n)$	0.0008	0.0000	0.0010	0.0000	0.0000
$e(h_w)$	0.0150	0.0143	0.0146	0.0139	0.0111
$\log 2 * \check{e}(h_w)$	0.0641	0.0956	0.0584	0.0977	0.0669
Analytic (Method 1)					
$\hat{e}(Q, D_n)$	0.0901	0.0766	0.0534	0.0678	0.0074
$\log 2 * \check{e}(Q, D_n)$	0.2299	0.1997	0.1410	0.1776	0.0263
$e(Q)$	0.0897	0.0827	0.0553	0.0729	0.0167
$\log 2 * \check{e}(Q)$	0.2384	0.2314	0.1492	0.2015	0.0927
PAC-Bayes bound	0.3241	0.3794	0.3509	0.3915	0.0572
$\mathrm{KL}(Q, P)$	8512.5098	13417.4023	14088.1738	15308.4170	1965.8048
ϵ	199.4836	401.7107	328.8929	443.9590	36.4424
$\mathbf{E}(\Sigma_q) = \mathbf{E}(F_{w_0})$ **(Method 2)**					
$\hat{e}(Q, D_n)$	0.0309	0.0288	0.0267	0.0298	0.0053
$\log 2 * \check{e}(Q, D_n)$	0.0895	0.0798	0.0742	0.0829	0.0160
$e(Q)$	0.0346	0.0331	0.0327	0.0348	0.0147
$\log 2 * \check{e}(Q)$	0.0995	0.0959	0.0947	0.0995	0.0590
PAC-Bayes bound	0.1590	0.1767	0.1523	0.2017	0.0099
$\mathrm{KL}(Q, P)$	4772.4854	5953.1523	4841.5972	7268.2832	46.5822
$\mathbf{E}(\Sigma_q) = \mathbf{E}(H_w)$ **(Method 3, our implementation)**					
$\hat{e}(Q, D_n)$	0.0202	0.0165	0.0169	0.0178	0.0043
$\log 2 * \check{e}(Q, D_n)$	0.0556	0.0451	0.0466	0.0487	0.0133
$e(Q)$	0.0268	0.0253	0.0245	0.0249	0.0141
$\log 2 * \check{e}(Q)$	0.0781	0.0781	0.0761	0.0742	0.0564
PAC-Bayes bound	0.1357	0.1540	0.1515	0.1817	0.0188
$\mathrm{KL}(Q, P)$	4645.1128	6122.5703	5919.6455	7589.6387	430.4026
ϵ	46	101	53	172	1360
$p(n, \epsilon)$	2301 (0.487%)	2429 (0.292 %)	2315 (0.245 %)	2287 (0.095 %)	82 (0.184 %)
$s(n, \epsilon)$	6435	6810	6420	6280	231
$1/c(n, \epsilon)$	2236	2497	2604	2841	38
$\Sigma_p = aF_{w_0} + \epsilon^{-1}, \mathbf{E}(\Sigma_q) = \mathbf{E}(F_{w_0})$ **(Method 4)**					
$\hat{e}(Q, D_n)$	0.0237	0.0218	0.0226	0.0220	0.0048
$\log 2 * \check{e}(Q, D_n)$	0.0663	0.0611	0.0631	0.0614	0.0147
$e(Q)$	0.0270	0.0265	0.0266	0.0264	0.0145
$\log 2 * \check{e}(Q)$	0.0806	0.0956	0.0789	0.0801	0.0573
PAC-Bayes bound	0.1323	0.1397	0.1486	0.1702	0.0092
$\mathrm{KL}(Q, P)$	4090.7241	4679.0293	5074.4102	6369.7505	23.2886

Table 2.3: Full comparison of PAC-Bayes bounds and effective dimensionalities on MNIST. Complete results for Methods 1–6 and the comparison methods, including errors, KL divergences, and effective dimensionality statistics.

Quantity/Model	FC-600-1	FC-600-2	FC-1200-1	FC-1200-2	LeNet
			diag$(\Sigma_q) = \Lambda$ **(our implementation)**		
$\hat{e}(Q, D_n)$	0.0283	0.0249	0.0284	0.0285	0.0079
$\log 2 * \check{e}(Q, D_n)$	0.0795	0.0700	0.0795	0.0797	0.0236
$e(Q)$	0.0330	0.0311	0.0326	0.0331	0.0161
$\log 2 * \check{e}(Q)$	0.0942	0.0923	0.0940	0.0963	0.0637
PAC-Bayes bound	0.1707	0.1846	0.1886	0.2167	0.0131
KL(Q, P)	5674.5186	6854.7871	6668.9023	8332.5869	37.5598
			Numerical optimization of Method 1 calculations (Method 5)		
$\hat{e}(Q, D_n)$	0.0711	0.0630	0.0805	0.0580	0.0087
$\log 2 * \check{e}(Q, D_n)$	0.1805	0.1673	0.2072	0.1510	0.0331
$e(Q)$	0.0717	0.0683	0.0800	0.0644	0.0168
$\log 2 * \check{e}(Q)$	0.1902	0.1925	0.2092	0.1811	0.0955
PAC-Bayes bound	0.3182	0.3917	0.3539	0.4366	0.0792
KL(Q, P)	9920.2510	15908.7813	11271.7246	20162.2891	2941.7917
ϵ	243.6499	490.6506	269.2748	599.2820	54.3656
			Isotropic Posterior (Method 6)		
$\hat{e}(Q, D_n)$	0.0473	0.0879	0.0653	0.0638	0.0094
$\log 2 * \check{e}(Q, D_n)$	0.1266	0.2538	0.1661	0.1757	0.0385
$e(Q)$	0.0524	0.0937	0.0677	0.0697	0.0191
$\log 2 * \check{e}(Q)$	0.1409	0.2935	0.1759	0.2057	0.1076
PAC-Bayes bound	0.3694	0.5461	0.4288	0.5490	0.1146
KL(Q, P)	16261.0205	26160.2773	18533.2422	30034.0645	4807.3564
ϵ	401.7107	808.9461	443.9590	894.0237	89.6338
			diag$(\Sigma_q) = \Lambda$ **(from Dziugaite and Roy (2017b))**		
$e(h_w)$	0.018	0.016	0.018	0.015	-
$e(Q)$	0.034	0.033	0.035	0.035	-
PAC-Bayes bound	0.161	0.186	0.179	0.223	-
KL(Q, P)	5144	6534	5977	8558	-
			E$(\Sigma_q) =$ **E**(H_w) **(from Wu et al. (2021))**		
$e(h_w)$	0.0153	0.0148	0.0161	-	-
$e(Q)$	0.02347	0.02523	0.02316	-	-
PAC-Bayes bound	0.1198	0.1443	0.1413	-	-
KL(Q, P)	3766.1	4956.8	5021.1	-	-

Table 2.3: Full comparison of PAC-Bayes bounds and effective dimensionalities on MNIST (continued).

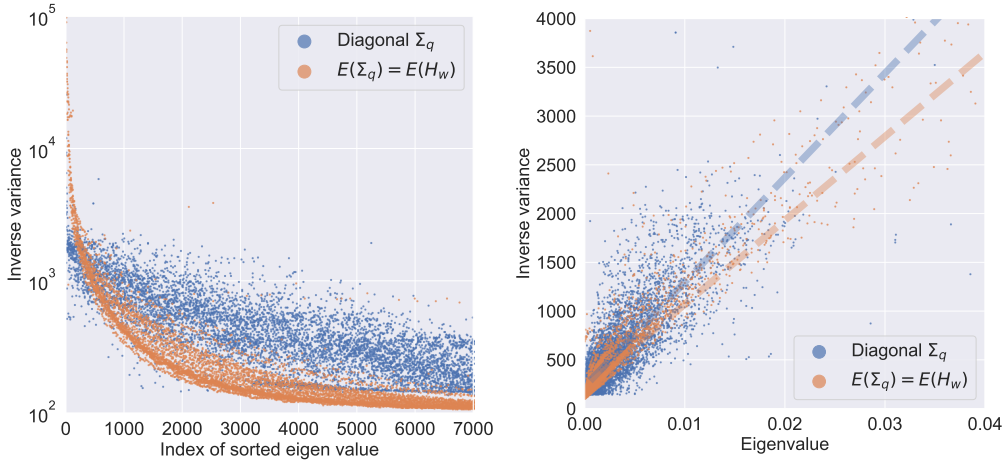

Figure 2.15: Posterior covariance computed by optimizing PAC-Bayes bound is aligned with sloppy directions. (Left) Inverse eigenvalues of the posterior covariance ($\bar{\lambda}_i^{-1}$) indexed by eigenvalues of the Hessian (X-axis) for $\mathrm{E}(\Sigma_q) = \mathrm{E}(H_w)$ (orange) and a diagonal-Σ_q (blue). For the latter (which is the method of Dziugaite and Roy (2017b)), the inverse eigenvalues are indexed by the sorted diagonal of the Hessian because their method is akin to assuming a diagonal Hessian. (Right) Inverse eigenvalues of the posterior covariance ($\bar{\lambda}_i^{-1}$) plotted against eigenvalues of the Hessian for $\mathrm{E}(\Sigma_q) = \mathrm{E}(H_w)$ (orange) and for diagonal-Σ_q (blue); for the latter the X-axis is the corresponding entry on the diagonal of the Hessian. For the orange pointcloud, we obtain a surprisingly accurate fit for our analytical expression ($\bar{\lambda}_i^{-1} = 2(n-1)\lambda_i + \epsilon$: we get $n \sim 43,000$ (true $n = 55,000$) and $\epsilon \sim 210.6$ (true $\epsilon = 101.3$).

Figures 2.15 and 2.16 study the posterior covariance computed while optimizing the bound. We observe that our Method 3, where the posterior covariance has the same eigenvectors as those of the Hessian (which we prove in Sec. 2.3.2 Method 1 to optimize pac-bayes bound), puts a large variance along sloppy directions of the Hessian. The method of Dziugaite and Roy (2017b) can be thought of as assuming a diagonal Hessian. As Fig. 2.15 (left) shows, the latter approach has a smaller variance in the sloppy subspace, and correspondingly their KL-term and the total bound is worse than ours, even if they have a sharper posterior. This coincides with our calculation in Method 1 that the eigenvectors of the optimal posterior is the same as that of the Hessian H_w. It is interesting to note that both methods clearly follow the trend that sloppier eigenvalues correspond to larger variance in the posterior. The inverse eigenvalues of covariance matrix has a strong linear relation with the eigenvalues of Hessian H_w, which also coincides with our analytical calculation Eq. (2.15) in Method 1.

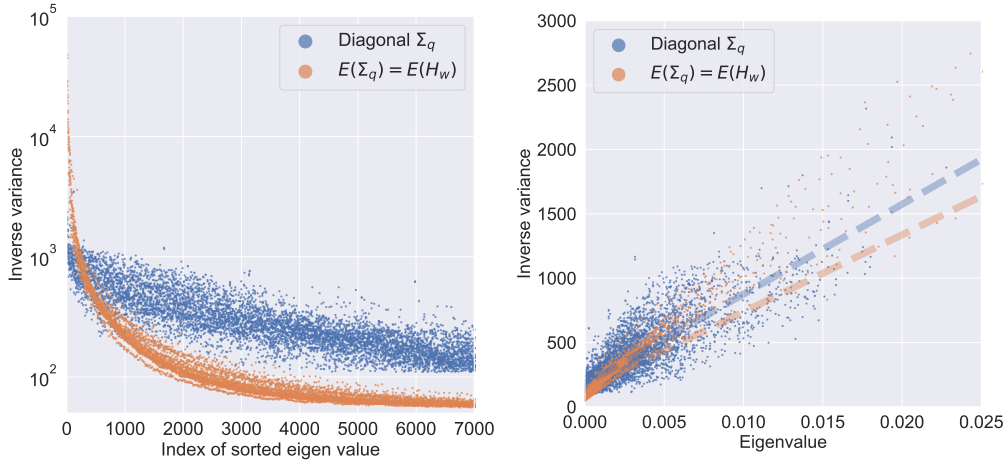

Figure 2.16: Alignment between the optimized posterior covariance and the sloppy directions for FC-1200-1. The inverse posterior variances exhibit an approximately linear relationship with the Hessian eigenvalues, reproducing the qualitative behavior observed for FC-600-2 in Fig. 2.15. The fitted values are $n \sim 30{,}000$ and $\epsilon \sim 138.2$, compared with the true values $n = 55{,}000$ and $\epsilon = 53.3$.

Technical Details for Computing PAC-Bayes Bounds

Optimization procedure We use a batch size of 1100 and draw 150 samples from the posterior Q to estimate $\hat{e}(Q, D_n)$ for each weight update. The implementation for efficiently computing these samples is described below. Adam is used to optimize the PAC-Bayes bound. For Methods 2, 3, and 4, we first train for 100 epochs with learning rate 10^{-3} and then train for another 150 epochs while decaying the learning rate by a multiplicative factor of 0.95 every 5 epochs. We found that, for this problem, maintaining a constant learning rate at the beginning is beneficial compared with decaying it immediately using, for example, a cosine schedule. For the reproduction of the approach of Dziugaite and Roy (2017b), which we denote by $\mathrm{diag}(\Sigma_q) = \Lambda$, we train for 300 epochs in the second phase with a decaying learning rate.

Efficient posterior sampling in Bayesian neural networks Bayesian neural networks are typically implemented using Bayesian variants of standard layers. For example, a Bayesian linear layer maintains a mean and standard deviation for each weight and samples a weight vector using the reparameterization trick during each forward pass. Although efficient, this approach requires complex architectures to be reimplemented using Bayesian layers. We instead use a PyTorch-specific wrapper around an existing network. The wrapper applies the reparameterization trick using the mean and log standard deviation of the weights and temporarily swaps the network parameters with the sampled weights during the forward pass. This makes the 150-sample estimates used above practical without rewriting the architecture and may also be useful for tasks such as estimating predictive uncertainty. The implementation in Fig. 2.17 is adapted from the PyTorch functional autograd benchmark utilities.

```python
def del_attr(obj, names):
    #names: one name in the list names_all, a.b.c, split by "."
    # list of format names = [a,b,c]
    # delete the attribute obj.a.b.c
    if len(names) == 1: delattr(obj, names[0])
    else: del_attr(getattr(obj, names[0]), names[1:])

def set_attr(obj, names, val):
    #names: one name in the list names_all, a.b.c, split by ".",
    # list of format names = [a,b,c],
    # set the attribute obj.a.b.c to val if obj.a.b.c is nn.Parameter
    if len(names) == 1: setattr(obj, names[0], val)
    else: set_attr(getattr(obj, names[0]), names[1:], val)

def get_names_params(mod):
    # names_all: a list of all names of mod.parameters of type
    # [a1.b1.c1, a2.b2.c2, ...]
    # orig_params: tuple of parameters of type nn.Parameter
    orig_params = tuple(mod.parameters())
    names_all = []
    for name, p in list(mod.named_parameters()):
        names_all.append(name)
    return orig_params, names_all

def load_weights(mod, names_all, params):
    for name, p in zip(names_all, params):
        set_attr(mod, name.split("."), p)

class bayesian_nn(nn.Module):
    def __init__(self, c, args, ns=1):
        # c: class of the model and args
        # ns: number of Monte Carlo samples
        super().__init__()
        self.w = c(*args)
        self.mu_std = nn.ModuleList([c(*args), c(*args)])
        self.ns, self.args = ns, args
        orig_params_w, self.names = get_names_params(self.w)

    def forward(self, x):
        ys = []
        for _ in range(self.ns):
            for name, m, v in zip(self.names,
                                  list(self.mu_std[0].parameters()),
                                  list(self.mu_std[1].parameters())):
                r = torch.randn_like(m).mul(torch.sqrt(torch.exp(2*v))).add(m)
                del_attr(self.w, name.split("."))
                set_attr(self.w, name.split("."), r)
            ys.append(self.w(x))
        return torch.stack(ys)
```

Figure 2.17: PyTorch wrapper for efficiently sampling weights from a Gaussian posterior.

Computing the PAC-Bayes term that corresponds to the distance from initialization In Method 4, we need to calculate

$$E = (w - w_0)^\top (aF_{w_0} + \epsilon^{-1})^{-1}(w - w_0).$$

In Methods 2 and 3, we need to sample from a posterior of the form $N(0, U\Lambda U^\top)$ for various different values of U and Λ. Doing either of these is not easy for high-dimensional weight spaces. We employ

two different methods to deal with this problem. For fully-connected networks we use a KFAC approximation of the Hessian/FIM while for LeNet which has much fewer weights, we approximate these matrices using their top few eigenvalues and eigenvectors.

KFAC approximation of the FIM and Hessian We approximate the Hessian/FIM by a variation of Kronecker decomposition of block-diagonal Hessian/FIM (KFRA; Botev et al., 2017). We use the BACKPACK library for implementing this (Dangel et al., 2020). For the weight of the k^{th} layer $w_k \in \mathbb{R}^{d_{k+1} \times d_k}$, the KFRA approximation of the corresponding block in the Hessian/FIM which is denoted by F^k or H^k can be written as $A_k \otimes B_k$. Denote by U_{A_k}, U_{B_k} the eigenspaces of A_k and B_k. To estimate E, we can first decompose E as the summation where each term is for a particular layer k

$$E = \sum_{k=0}^{L} E^k$$

where

$$\begin{aligned} E^k &= (w^k - w_0^k)^\top (a\,(F_{w_0})^k + \epsilon^{-1})^{-1} (w^k - w_0^k) \\ &= (w^k - w_0^k)^\top U^k (a\Lambda^k + \epsilon^{-1})^{-1} {U^k}^\top (w^k - w_0^k) \\ &= \left((w^k - w_0^k)^\top U^k \left(a\Lambda^k + \epsilon^{-1} \right)^{-1/2} \right) \left((w^k - w_0^k)^\top \; U^k \left(a\Lambda^k + \epsilon^{-1} \right)^{-1/2} \right)^\top \end{aligned}$$

$(E^k)^{1/2}$ can be calculated by

$$\begin{aligned} {E^k}^{1/2} &= (w^k - w_0^k)^\top U^k \left(a\Lambda^k + \epsilon^{-1} \right)^{-1/2} \\ &= (U_{A_k}^\top (w^k - w_0^k) U_{B_k}) \odot \left(a\Lambda^k + \epsilon^{-1} \right)^{-1/2} \end{aligned}$$

where in the last line, $(w^k - w_0^k) \in \mathbb{R}^{d_{k+1} \times d_k}$. We use $\odot$ to denote element wise multiplication. $U_{A_k}^T (w^k - w_0^k) U_{B_k}$ can now be easily calculated using the KFAC factors.

To sample from the posterior $N(w, U\Lambda U^\top)$, we can concatenate the samples of the weights of each layer. We first sample $r^k \sim N(0, I_{d_k})$, then calculate $\sqrt{\Lambda^k} \odot r_k$ and thereby

$$\nu_k := U_k \left(\sqrt{\Lambda^k} \odot r_k \right) = U_{A_k} \left(\sqrt{\Lambda^k} \odot r_k \right) U_{B_k}^\top.$$

The final sample is therefore $w + [\nu_1, ..., \nu_k]$ which is distributed as $N(w, U\Lambda U^\top)$.

Approximate FIM and Hessian using its top eigenvalues and eigenvectors For symmetric Σ with orthogonal decomposition $\Sigma = U\Lambda U^\top$, $U = [U_1, U_2]$, $\Lambda = \text{diag}(\Lambda_1, \Lambda_2)$, we have

$$\begin{aligned} \Sigma &= U_1 \Lambda_1 U_1^\top + U_2 \Lambda_2 U_2^\top \\ \text{where } I &= U_1 U_1^\top + U_2 U_2^\top. \end{aligned}$$

In this case, to calculate E, we approximate $aF_{w_0} + \epsilon^{-1}$ by

$$aF_{w_0} + \epsilon^{-1} = U_1 (a\Lambda_1 + \epsilon_1^{-1}) U_1^\top + \epsilon_2^{-1} U_2 U_2^\top$$

where Λ_1, U_1 are the stiff (largest k) eigenvalues and corresponding eigenvectors for F_{w_0} and U_2, Λ_2 are the sloppy ones. Notice that we use two scalar parameters ϵ_1 and ϵ_2 to set the additive constant in the prior covariance.

$$\begin{aligned} E &= (w-w_0)^\top U_1(a\Lambda_1 + \epsilon_1^{-1})^{-1}U_1^\top(w-w_0) + \epsilon_2(w-w_0)^\top U_2U_2^\top(w-w_0) \\ &= (w-w_0)^\top U_1(a\Lambda_1 + \epsilon_1^{-1})^{-1}U_1^\top(w-w_0) + \epsilon_2\left(\|w-w_0\|_2^2 - (w-w_0)^\top U_1U_1^\top(w-w_0)\right) \end{aligned}$$

Notice that the term $(w-w_0)^\top U_1$ is not hard to calculate because $U_1 \in \mathbb{R}^{p\times k}$ and since we are choosing the top few eigenvalues of the Hessian/FIM, the value of k is small (about 300).

To sample from the posterior $N(w, U\Lambda U^\top)$, we first set $\Lambda = \mathrm{diag}(\Lambda_1, \Lambda_2)$ where Λ_1 are the top k stiff eigenvectors and Λ_2 are the $p-k$ other eigenvectors. Correspondingly, we have $U = [U_1, U_2]$. We use an isotropic variance for the sloppy subspace and set $\Lambda_2 = \epsilon^{-1}I_{p-k}$. We first sample $r \sim N(0, I_k)$, then calculate

$$\begin{aligned} \nu_1 &= U_1\sqrt{\Lambda_1}U_1^\top r \\ \nu_2 &= \epsilon^{-1/2}U_2U_2^\top r = \epsilon^{-1/2}(r - U_1U_1^\top r) \end{aligned}$$

Notice that $U_1U_1^\top r$, and $U_1\sqrt{\Lambda_1}U_1^\top r$ are easy to calculate. The result $w + [\nu_1, \nu_2]$ is distributed as $N(w, U\Lambda U^\top)$.

For cases when we recompute the FIM/Hessian while optimizing the PAC-Bayes bound (Method 2 and 3 respectively), we recompute the eigenvalues Λ_1 and the corresponding eigenvectors U_1. Note that the parameter ϵ in the covariance of the posterior is also optimized when we optimize the PAC-Bayes bound.

Optimizing parameters of the prior in the PAC-Bayes bound The prior should be fixed before looking at the training set, but for all methods above, we optimize the scale of the prior. We do this by adding an additional penalty in the KL term. Assume that a^i for $i = 1, \ldots, m'$ are the number of parameters in the prior that we can select, we choose $a^i = (1/c^i)\exp\left(-j^i/b^i\right)$ for $j^i \in \mathbb{N}$. We reindex j^i as a single index $k = (\sum_i j^i)^{m'}$, then if the PAC-Bayes bound for each index k is designed to hold with probability at least $1 - \frac{6\delta}{\pi^2k^2}$, then by union bound, it will hold uniformly for all $k \in \mathbb{N}$ with probability at least $1-\delta$. For a bound that holds with probability $1-\delta'$, the penalty we should add is $\log\frac{n}{\delta'}$, hence, using the relation

$$a^i = (1/c^i)\exp\left(j^i/b^i\right), \quad \delta' = \frac{6\delta}{\pi^2k^2}, \quad k = (\sum_i j^i)^{m'}$$

we add the penalty

$$\varphi(a^1, ..., a^{m'}) = 2m'\log\left(\sum_i b_i \log\left(c^i a^i\right)\right) + \log\frac{\pi^2 n}{6\delta}$$

Similarly, for any positive or negative integer j^i, we can set $k = (\sum_i 2|j|^i)^{m'}$ to get the penalty

$$\varphi(a^1, ..., a^{m'}) = 2m'\log\left(2\sum_i \left|b_i \log\left(c^i a^i\right)\right|\right) + \log\frac{\pi^2 n}{6\delta}$$

In Methods 1, 2, 3 we choose $a^1 = \epsilon^{-1}$, in Method 4, we choose $a^1 = a$ and $a^2 = \epsilon^{-1}$.

Hyperparameter choices In Methods 2, 3, and 4, we choose $b = 0.01$ and $c = 0.1$ for the penalty on the scaling parameters in the prior. In Method 1, we choose $b = 0.1$ and $c = 0.05$. For all PAC-Bayes bound optimization experiments, we use the confidence parameter $\delta = 0.025$.

2.5. Discussion

We showed that for typical datasets, the sloppy decay pattern of eigenvalues of the input correlation matrix is mirrored in key quantities of the deep network, e.g., eigenspectra of the activation correlations, activation gradients, logit Jacobians, Hessian and the FIM. This suggests that the "simplicity", more precisely, low-dimensional nature, of inputs of high-dimensional datasets directly affects the representations learned by the network. We showcased the benefit of sloppiness by providing non-vacuous PAC-Bayes generalization bounds for deep networks, including analytical ones.

CHAPTER 3

THE DYNAMICS OF GENERALIZATION IN DEEP LEARNING

Chapter 2 provides a data-dependent explanation for the effective capacity control of deep networks: the sloppiness of the input geometry is inherited by quantities such as the Hessian and the Fisher Information Matrix, leading to a small effective dimensionality at the trained solution. However, this viewpoint is inherently post-hoc. It analyzes geometric quantities at, or near, the end of training, and it primarily relates generalization to the spectrum of the input data matrix. As a result, it does not fully capture how the labels, the loss, and the entire optimization trajectory interact to produce good generalization.

This motivates the study in this chapter. Existing non-worst-case approaches move beyond classical capacity control, but each captures only part of the phenomenon. Weight-dependent bounds often reason in parameter space and depend on properties of the final weights or reachable hypothesis class (Bartlett et al., 2017; Neyshabur et al., 2018; Arora et al., 2019); PAC-Bayes analyses can yield non-vacuous estimates, but are often post-hoc and tied to properties of the trained solution (Dziugaite and Roy, 2017a); stability and robustness analyses typically require uniform assumptions over the loss landscape or algorithmic behavior (Bousquet and Elisseeff, 2002; Hardt et al., 2016; Xu and Mannor, 2012); and information-theoretic bounds often apply most naturally to randomized algorithms or track accumulated trajectory-dependent quantities without explicitly accounting for contraction along training (Xu and Raginsky, 2017; Mou et al., 2018; Negrea et al., 2019; Neu et al., 2021). In this chapter, we instead analyze the evolution of the generalization gap throughout training: we approximate it by an averaged loss difference, study how this quantity is driven by perturbation and damped by contraction, and show that the eventual gap is governed by the alignment between the initial residual and an effective Gram matrix.

3.1. Introduction

Consider a training dataset that has n data points sampled independently from the uniform distribution on a two-point domain $\{(x_1, y_1), (x_2, y_2)\}$ where $x_1, x_2 \in \mathbb{R}^d$ are orthogonal vectors with unit norm. There are many ways to understand the generalization loss of a parameterized predictor fitted to such data. For example, standard weight-norm based techniques, such as those of Arora et al. (2019) give a bound of $\mathcal{O}(n^{-1/2})$ on the generalization loss. Mutual information-based techniques, such as those by Pensia et al. (2018) and Negrea et al. (2019), lead to a better bound, $\mathcal{O}(n^{-1})$. As we will show in Sec. 3.4.2, the true generalization gap of this problem is only $\Theta(2^{-n})$. Therefore, even in this simple toy setting, these techniques drastically over-estimate the generalization loss.

The central idea behind both approaches above is to characterize the "volume" of the hypothesis class explored during training, though they do so in distinct ways. Weight-norm based approaches compute the Rademacher complexity of weight configurations reachable from initialization, say, using gradient descent. These bounds can be quite loose for neural networks. Rather than directly dealing with the hypothesis space, mutual information-based approaches characterize how the generalization gap accumulates along the training trajectory in terms of gradient covariance. This leads to tighter bounds but these techniques apply only to randomized algorithms, such as stochastic gradient descent with additive isotropic noise. And the quality of the bounds is sensitive to the magnitude of this added noise.

3.1.1. Contributions of this chapter

Our work introduces two key technical advances to address the limitations of the above frameworks. First, we analyze the generalization gap directly in terms of the loss rather than the model parameterization. This effectively handles the non-uniformity of the Lipschitz constant of neural loss landscape and scenarios where multiple distinct regions of the weight space have the same loss. Second, we derive an estimate of the generalization gap for deterministic training algorithms that mirrors the structure of mutual information-based bounds. By tracking the training trajectory directly, our analysis models the learning dynamics more faithfully and yields significantly more accurate estimates of the generalization gap. In the simple example above, our technique accurately recovers the $\Theta(2^{-n})$ rate.

We investigate the dynamics of a quantity called the "averaged loss difference" $\bar{\Delta}_n(t)$ which is akin to a leave-one-out estimate of the generalization gap. We show that

$$\frac{\mathrm{d}\bar{\Delta}_n(t)}{\mathrm{d}t} = -\bar{c}_n(t)\bar{\Delta}_n(t) + \bar{\epsilon}_n(t), \quad t > 0, \quad \bar{\Delta}_n(0) = 0.$$

Here, $\bar{c}_n$ is a "contraction factor" that describes how quickly two training trajectories, one trained on all n samples in the dataset and another trained on $(n-1)$ samples, come together. The quantity $\bar{\epsilon}_n$ is a "perturbation factor" that keeps these two trajectories apart because of the one distinct sample. Both these quantities are positive and typically decrease in magnitude during training. This explains why generalization gap is controlled in neural networks. We develop detailed mathematical interpretations of the contraction and perturbation factors. The former can be approximated by a generalized Rayleigh quotient between the per-sample gradient and difference between weights along the two training trajectories weighted by the Hessian. The perturbation factor is directly related to the trace of the covariance of the gradients. Our technique for analyzing $\bar{\Delta}_n$ in terms of the contraction and perturbation factors is a generalization of contraction theory for time-varying dynamical systems that evolve on non-convex energy landscapes.

The key result of this chapter is to show that, for general deep networks and smooth loss functions, after gradient flow runs for time t, the averaged loss difference

$$\bar{\Delta}_n(t) = \vec{r}_n(0)^\top K_n(0,t)\ \vec{r}_n(0),$$

where K_n is a quantity that we call the "effective Gram matrix" and $\vec{r}_n(0)$ is the gradient of the loss function with respect to the predictor at initialization. The effective Gram matrix is positive semi-definite, it scales as $\mathcal{O}(n^{-1})$ and depends upon a certain covariance of the gradients integrated along the entire training trajectory. The generalization gap is small if the initial residual projects on the subspace of the effective Gram matrix with small eigenvalues. The quadratic form for $\bar{\Delta}_n(t)$ provides a new perspective into similarities between deep networks with kernel machines, not in terms of their training dynamics (where the equivalence holds only under certain modeling assumptions such as infinite width), but in terms of the generalization gap. Across numerous experiments on different neural network architectures, datasets and sample sizes, we show that the quadratic form accurately captures the actual generalization gap. For high-dimensional linear regression, we analytically calculate the contraction and perturbation factors to show that the asymptotic averaged loss difference $\lim_{t\to\infty} \bar{\Delta}_\infty(t)$ exactly matches existing calculations in the literature of the generalization gap in under-parameterized, over-parameterized and critical regimes. We provide

an example where these quantities can be calculated analytically for a simplified one-layer self-attention-based network. Finally, we also show how our framework can be adapted to stochastic gradient descent.

3.2. Preliminaries

Let $[n]$ denote the set of integers $\{1, ..., n\}$. The notation $a \cdot b$ denotes the inner product of vectors a and b. For a function h, we write $h(w)|_a^b \equiv h(b) - h(a)$ and $h(w)|_a \equiv h(a)$. We use $|\cdot|$ for the absolute value of a scalar, , $\|\cdot\|_2$ for the ℓ_2-norm of a vector or a matrix, and $\|\cdot\|_F$ for the Frobenius norm of a matrix.

Let $\mathcal{X}$ and $\mathcal{Y}$ be input and output spaces respectively, and let $\mathcal{Z} = \mathcal{X} \times \mathcal{Y}$ be the sample space. If $\mathcal{W}$ denotes the parameter (weight) space, consider the predictor $f : \mathcal{W} \times \mathcal{X} \to \mathcal{Y}$ and a loss function $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}^+$. As an example, for the cross-entropy loss on a C-class classification problem, we could set $\mathcal{Y} = \mathbb{R}^C$ and $\ell(f(w, x), y) = -\sum_{j=1}^{C} y^{(j)} \log p^{(j)}$, where $p^{(j)} \propto \exp(f^{(j)}(w, x))$ (normalized appropriately) is the softmax probability of class C. We use the shorthand $\ell(w, z) \equiv \ell(f(w, x), y)$.

Consider a dataset $S_n = \{z_i = (x_i, y_i)\}_{i\in[n]}$ of size n with each $z_i \in \mathcal{Z}$ drawn i.i.d. from a distribution D. Let D^n denote the distribution of such datasets, i.e., $S_n \sim D^n$. Let $S_n^{-i} = \{z_1, ..., z_{i-1}, z_{i+1}, ..., z_n\}$ denote a dataset obtained by removing the i-th datum. The quantity $\bar{\ell}(w, S_n) = n^{-1} \sum_{i=1}^{n} \ell(w, z_i)$ is the average loss over the dataset S_n. Let $w_n(t)$ and $w_n^{-i}(t)$ be the solutions of gradient flows

$$\frac{\mathrm{d}w}{\mathrm{d}t} = -\nabla \ell(w, S_n), \quad \frac{\mathrm{d}w}{\mathrm{d}t} = -\nabla \ell(w, S_n^{-i}), \tag{3.1}$$

respectively. Our calculations will often use quantities that depend upon $w_n(t)$ and $w_n^{-i}(t)$, i.e., two points along weight trajectories that are trained on slightly different datasets. The numerical experiments that validate our calculations will be conducted by omitting m samples. All expressions will therefore replace $z_i \equiv S_{(m)}$ and $S_n^{-i} \equiv S_n^{-(m)}$.

To reduce clutter, we denote $\delta w_i(t) = w_n^{-i}(t) - w_n(t)$. For the first-order derivatives, we denote the single-sample gradient by $g_i = \nabla \ell(w_n, z_i)$ and the average gradient over the leave-one-out dataset as $g^{-i} = \nabla \bar{\ell}(w_n, S_n^{-i})$. Similarly, for the second-order derivatives, we distinguish the single-sample Hessian $H_i = \nabla^2 \ell(w_n, z_i)$ from the leave-one-out dataset Hessian $H^{-i} = \nabla^2 \bar{\ell}(w_n, S_n^{-i})$. In such expressions, we will typically omit the subscript n indicating the size of dataset and t that indicates time; we will clarify in places where there could be ambiguity.

Given a predictor f, a loss function ℓ, and an initialization $w_n(0)$,

$$R(S_n, t) = \mathbb{E}_z[\ell(w_n(t), z)], \quad R_{\text{train}}(S_n, t) = \bar{\ell}(w_n(t), S_n),$$

are the **generalization loss** and the **train loss** of gradient flow trained on S_n at time t, respectively. Our main quantity of interest is their difference, **the generalization gap**

$$\delta R(S_n, t) = R(S_n, t) - R_{\text{train}}(S_n, t). \tag{3.2}$$

We will typically use the averaged versions of the quantities, e.g., $\mathbb{E}_{S_n}[R(S_n, t)]$ which is the average generalization loss over the training samples. We will often omit the subscript S_n in the expectation when there is no ambiguity.

3.2.1. Contraction Theory

This section introduces some preliminary material on contraction theory (Lohmiller and Slotine, 1998, 2000), which provides a way to analyze solutions of slightly different dynamical systems. Contraction theory reformulates Lyapunov stability (Isidori, 1995; Marino and Tomei, 1995) using a quadratic Lyapunov function corresponding to a Riemannian contraction metric and its uniform positive definite matrix to characterize the necessary and sufficient conditions for exponential convergence of trajectories to each other and the stability of these trajectories to perturbations of the dynamics. Consider a nonlinear dynamical system

$$\frac{\mathrm{d}\xi}{\mathrm{d}t} = h(\xi, t). \tag{3.3}$$

The following theorem gives guarantees of the exponential convergence of trajectories with different initializations.

Theorem 3.1 (Theorem 2.1 from Tsukamoto et al. (2021)). If there exists a uniformly positive definite matrix $M(\xi, t) \succ 0$ for all ξ, t, such that the following condition holds for some $\alpha > 0$,

$$\forall \xi, t: \quad \dot{M} + M\nabla_\xi h + \nabla_\xi h^\top M \preceq -2\alpha M, \tag{3.4}$$

then all trajectories of Eq. (3.3) converge to a single trajectory under the metric induced by M exponentially fast regardless of their initial conditions, i.e., for all trajectories ξ, ξ' of Eq. (3.3), $d(\xi(t), \xi'(t))_M \leq d(\xi(0), \xi'(0))_M e^{-\alpha t}$, where $d(\cdot,\cdot)_M$ denotes the distance under the metric induced by M. A dynamical system Eq. (3.3) satisfying Eq. (3.4) is said to be "contracting", under the "contraction metric" induced by M. The factor α is defined as the "contraction factor".

Using Theorem 3.1, we can also analyze trajectories of a perturbed dynamical system

$$\frac{\mathrm{d}\xi}{\mathrm{d}t} = h(\xi, t) + b(\xi, t). \tag{3.5}$$

Let $\xi_0(t)$, $\xi_1(t)$ be solutions of Eq. (3.3) and Eq. (3.5), respectively. The next theorem shows that for a contracting system, the solution of the perturbed system does not differ too much from that of the original system, under certain conditions.

Theorem 3.2 (Theorem 2.3 from Tsukamoto et al. (2021)). Assume that the dynamical system Eq. (3.3) is contracting under M with factor α. If $\overline{b} = \sup_{\xi,t} \|b(\xi, t)\|$ and there exist constants $\underline{m}, \overline{m} > 0$ such that $\underline{m} I \preceq M(\xi, t) \preceq \overline{m} I$ for all ξ, t, then we have

$$d(\xi_1(t), \xi_0(t)) \leq \frac{d(\xi_1(0), \xi_0(0))_M}{\sqrt{\underline{m}}} e^{-\alpha t} + \frac{\overline{b}}{\alpha}\sqrt{\frac{\overline{m}}{\underline{m}}}\left(1 - e^{-\alpha t}\right),$$

$$d(\xi_1(t), \xi_0(t))_M \leq d(\xi_1(0), \xi_0(0))_M e^{-\alpha t} + \frac{\overline{b}\sqrt{\overline{m}}}{\alpha}\left(1 - e^{-\alpha t}\right),$$

where $d(\cdot,\cdot)$, $d(\cdot,\cdot)_M$ denote the distance under Euclidean metric and metric induced by M, respectively.

In short, for contracting systems, for large times t, the bound on the distance between the solution of the original dynamics and the perturbed dynamic is determined by the perturbation of the system,

the contraction factor, and the eigenvalues of the metric. In this chapter, we will be interested in using these ideas to understand the difference between two trajectories evaluated on certain loss functions that are fitted using slightly different datasets.

3.3. Results

The **pointwise loss difference**

$$\Delta_n^{-i}(t) = \ell(w_n^{-i}(t), z_i) - \ell(w_n(t), z_i),$$

is the difference of the loss along two weight trajectories on slightly different datasets. The **averaged loss difference**

$$\bar{\Delta}_n(t) = \frac{1}{n}\sum_{i=1}^{n} \Delta_n^{-i}(t) \tag{3.6}$$

is similar to the Leave-One-Out-Cross-Validation (LOOCV) loss. The following lemma shows how the expected generalization gap can be approximated using the averaged loss difference.

Lemma 3.3. Assume that the expected generalization loss $\mathbb{E}\left[R(S_n, t)\right]$ is non-increasing in n, the expected training loss $\mathbb{E}\left[R_{\text{train}}(S_n, t)\right]$ is non-decreasing in n, and the expected generalization gap $\mathbb{E}\left[\delta R(S_n, t)\right]$ is non-negative for all n, t. Then

$$\mathbb{E}\left[\delta R(S_n, t)\right] \leq \mathbb{E}\left[\bar{\Delta}_n(t)\right] \leq \mathbb{E}\left[\delta R(S_{n-1}, t)\right].$$

If we also have $\mathbb{E}[\delta R(S_n, t)] / \mathbb{E}[\delta R(S_{n-1}, t)] \to 1$ as $n \to \infty$, then,

$$\mathbb{E}\left[\delta R(S_n, t)\right] = \mathbb{E}\left[\bar{\Delta}_n(t)\right] + o\left(\mathbb{E}\left[\delta R(S_n, t)\right]\right).$$

Proof. By the definition of the averaged loss difference $\bar{\Delta}_n(t)$,

$$\bar{\Delta}_n(t) = \frac{1}{n}\left(\sum_{i=1}^{n} \ell(w_n^{-i}(t), z_i)\right) - \bar{\ell}(w_n(t), S_n)$$

Taking the expectation on both sides, we have

$$\mathbb{E}\left[\bar{\Delta}_n(t)\right] = \mathbb{E}\left[R(S_{n-1}, t)\right] - \mathbb{E}\left[R_{\text{train}}(S_n, t)\right]$$

Using the definition of the generalization gap, this identity can also be written as

$$\mathbb{E}\left[\bar{\Delta}_n(t)\right] = \mathbb{E}\left[\delta R(S_n, t)\right] + \mathbb{E}\left[R(S_{n-1}, t)\right] - \mathbb{E}\left[R(S_n, t)\right].$$

Therefore, if

$$\mathbb{E}\left[R(S_{n-1}, t)\right] - \mathbb{E}\left[R(S_n, t)\right] = o\left(\mathbb{E}\left[\delta R(S_n, t)\right]\right),$$

then

$$\mathbb{E}\left[\delta R(S_n, t)\right] = \mathbb{E}\left[\bar{\Delta}_n(t)\right] + o\left(\mathbb{E}\left[\delta R(S_n, t)\right]\right).$$

Under the two monotonicity assumptions, we have

$$\mathbb{E}\left[R(S_n,t)\right] \leq \mathbb{E}\left[R(S_{n-1},t)\right], \quad \mathbb{E}\left[R_{\text{train}}(S_{n-1},t)\right] \leq \mathbb{E}\left[R_{\text{train}}(S_n,t)\right].$$

Therefore,

$$\mathbb{E}\left[\delta R(S_n,t)\right] \leq \mathbb{E}\left[\bar{\Delta}_n(t)\right] \leq \mathbb{E}\left[\delta R(S_{n-1},t)\right].$$

Finally, under these monotonicity assumptions, if $\mathbb{E}[\delta R(S_n,t)]/\,\mathbb{E}[\delta R(S_{n-1},t)] \to 1$ as $n \to \infty$, then

$$\frac{\mathbb{E}[\delta R(S_{n-1},t)] - \mathbb{E}[\delta R(S_n,t)]}{\mathbb{E}[\delta R(S_n,t)]} \to 0$$

as $n \to \infty$. The sandwich bound then implies

$$0 \leq \mathbb{E}\left[\bar{\Delta}_n(t)\right] - \mathbb{E}\left[\delta R(S_n,t)\right] \leq \mathbb{E}\left[\delta R(S_{n-1},t)\right] - \mathbb{E}\left[\delta R(S_n,t)\right] = o\left(\mathbb{E}\left[\delta R(S_n,t)\right]\right).$$

Hence,

$$\mathbb{E}\left[\delta R(S_n,t)\right] = \mathbb{E}\left[\bar{\Delta}_n(t)\right] + o\left(\mathbb{E}\left[\delta R(S_n,t)\right]\right).$$

□

This lemma shows that the expected generalization gap $\mathbb{E}_{S_n}\left[\delta R(S_n,t)\right]$ can be approximated by the expected averaged loss difference $\mathbb{E}_{S_n}\left[\bar{\Delta}_n(t)\right]$. This lemma motivates the theory developed in this chapter. Our strategy will be to characterize the generalization gap using quantities derived from $\Delta_n(t)$.

Remarks on the assumptions The expected generalization loss $\mathbb{E}[R(S_n,t)]$ is non-increasing in n: This holds for ridgeless linear regression without label noise. When label noise is non-zero, the generalization loss decays monotonically when the number of samples is greater than the number of features. This result also holds for ridge regression when the ridge coefficient λ decays with n, but not too fast, i.e., $\lambda > \frac{\sigma^2}{\sigma^2+\|\theta^*\|^2} \cdot \frac{1}{n}$ where, σ is the noise variance and θ^* is the true regressor. See Hastie et al. (2022) for reference. Similar results hold for kernel regression and random feature regression-based architectures (Mei and Montanari, 2022), which are both widely used models in the analysis of neural networks. For consistent estimators, the generalization loss converges to the Bayes risk asymptotically, although this decrease need not be strictly monotonic. Estimators like Empirical Risk Minimization (ERM), Structural Risk Minimization (SRM) are consistent under mild assumptions on the hypothesis class, e.g., having finite capacity.

The expected training loss $\mathbb{E}[R_{\text{train}}(S_n,t)]$ is non-decreasing: this holds for ridgeless linear regression in general, and therefore for kernel regression and random feature-based models of neural networks. Note that this assumption can be modified slightly to be $\mathbb{E}[R_{\text{train}}(S_{n-1},t)] \leq \mathbb{E}[R_{\text{train}}(S_n,t)] + B/n$. This new condition holds for empirical risk minimization (ERM) with bounded loss $|\ell(w,z)| \leq B$ in general. The resulting left-hand side of the inequality in Lemma 3.3 gets an additive term of B/n correspondingly. The rest of our calculations stay as they are. The expected generalization gap $\mathbb{E}[\delta R(S_n,t)]$ is non-negative: This holds for empirical risk minimization (ERM) in general.

As we show next, the concentration of $\bar{\Delta}_n(t)$ to $\mathbb{E}\left[\bar{\Delta}_n(t)\right]$ can also be guaranteed if algorithm stability is assumed.

Concentration of $\bar{\Delta}_n(t)$ to $\mathbb{E}[\bar{\Delta}_n(t)]$ We first define the notion of stability for a deterministic algorithm $\mathcal{A}$ that maps from space of datasets to weight space, i.e. $\mathcal{A} : \cup_{n=0}^{\infty} \mathcal{Z}^n \to \mathcal{W}$.

Definition 3.4. An algorithm $\mathcal{A}$ is uniformly ε-stable if for all datasets S, S' differing in at most one sample, we have

$$\sup_z \left|\ell(\mathcal{A}(S), z) - \ell(\mathcal{A}(S'), z)\right| \leq \varepsilon$$

Now we define the set of algorithms Γ that map the dataset S to points on the gradient flow trajectory trained on S at certain time points.

$$\Gamma = \left\{ \mathcal{A} : \mathcal{A}(S) = w(t), t \geq 0, \text{ where } w \text{ satisfies } \frac{\mathrm{d}w}{\mathrm{d}t} = -\nabla \bar{\ell}(w, S), w(0) \in \mathcal{W}, S \in \cup_{n \in \mathbb{N}} \mathcal{Z}^n \right\}$$

Lemma 3.5. Assume that (1) $|\ell(w, z)| \leq B$ for all $w \in \mathcal{W}, z \in \mathcal{Z}$, (2) $\forall \mathcal{A} \in \Gamma$, $\mathcal{A}$ is ε-stable, then for all $t > 0$, with probability $1 - \delta$,

$$\left|\bar{\Delta}_n(t) - \mathbb{E}[\bar{\Delta}_n(t)]\right| \leq (n\varepsilon + 2B)\sqrt{\frac{2\log(2/\delta)}{n}}$$

Proof. Let $\tilde{S}_n$ denote a modified dataset of S_n by replacing the sample z_j with a different sample $\tilde{z}_j$. Let $\tilde{w}_n(t)$, $\tilde{w}_n^{-i}(t)$ be the corresponding trajectories trained with $\tilde{S}_n$ and $\tilde{S}_n^{-i}$ (the removed-ith sample version of S_n). Note that $\tilde{w}_n^{-j}(t) = w_n^{-j}(t)$. Let $\bar{\Delta}(\tilde{S}_n, t)$ and $\bar{\Delta}(S_n, t)$ be the averaged loss difference calculated on $\tilde{S}_n$ and S_n respectively. By assumptions (1) and (2), we have

$$\begin{aligned}
|\bar{\ell}(\tilde{w}_n(t), \tilde{S}_n) - \bar{\ell}(w_n(t), S_n)| &\leq \frac{(n-1)\varepsilon}{n} + \frac{2B}{n} \leq \varepsilon + \frac{2B}{n} \\
|\ell(\tilde{w}_n^{-i}(t), z_i) - \ell(w_n^{-i}(t), z_i)| &\leq \varepsilon \quad \forall i \neq j \\
|\ell(\tilde{w}_n^{-j}(t), z_j) - \ell(w_n^{-j}(t), z_j)| &\leq \frac{2B}{n}
\end{aligned}$$

Hence we have

$$\left|\bar{\Delta}(\tilde{S}_n, t) - \bar{\Delta}(S_n, t)\right| \leq 2\varepsilon + \frac{4B}{n} \tag{3.7}$$

Inequality Eq. (3.7) gives the replace-one-sample difference of $\bar{\Delta}_n$, hence by McDiarmid's inequality (McDiarmid, 1989), we have the following concentration inequality,

$$\mathbb{P}_{S_n}\left[|\bar{\Delta}_n - \mathbb{E}[\bar{\Delta}_n]| \geq a\right] \leq 2\exp\left(-\frac{2a^2}{n(2\varepsilon + 4B/n)^2}\right)$$

Setting the right-hand side to δ, we have with probability at least $1 - \delta$,

$$\left|\bar{\Delta}_n - \mathbb{E}[\bar{\Delta}_n]\right| \leq (n\varepsilon + 2B) \cdot \sqrt{\frac{2\log(2/\delta)}{n}}.$$

□

Remark 3.6. In general, the convergence of $\bar{\Delta}_n$ to $\mathbb{E}\left[\bar{\Delta}_n\right]$ can be guaranteed by different versions of algorithm stability (e.g., hypothesis stability, pointwise hypothesis stability and uniform stability; Bousquet and Elisseeff, 2002). Charles and Papailiopoulos (2018) show that the algorithm $\mathcal{A}$ is $C(L,\mu)/(n-1)$-uniformly stable if $\ell(w,z)$ is L-Lipchitz in w and $\bar{\ell}(w,S)$ is μ-PL (Polyak Lojasiewicz) in w, where $C(L,\mu)$ is a constant depending on L and μ. Other versions of stability can also be guaranteed by PL and QG (quadratic growth) conditions as shown in Charles and Papailiopoulos (2018). The averaged loss difference $\bar{\Delta}_n$ is in nature similar to cross-validation loss estimator. Under certain stability conditions, Austern and Zhou (2025) prove central limit theorems and Berry-Esseen bounds for the cross validation loss estimator.

3.3.1. Evolution of the averaged loss difference (Δ_n^{-i} and $\bar{\Delta}_n$)

We begin with a lemma that describes the evolution of $\Delta_n^{-i}(t)$ in terms of two quantities, a contraction factor $c_n^{-i}(t)$ and a perturbation factor $\epsilon_n^{-i}(t)$.

Lemma 3.7. For a loss function $\ell(w,z)$ that is differentiable in w for all z,

$$\frac{\mathrm{d}\Delta_n^{-i}(t)}{\mathrm{d}t} = -c_n^{-i}(t)\Delta_n^{-i}(t) + \epsilon_n^{-i}(t), \quad t>0, \quad \Delta_n^{-i}(0)=0,$$

where

$$c_n^{-i}(t) = \frac{\nabla\ell(w,z_i)\cdot\nabla\bar{\ell}(w,S_n^{-i})\big|_{w_n(t)}^{w_n^{-i}(t)}}{\Delta_n^{-i}(t)} \text{ and}$$
$$\epsilon_n^{-i}(t) = \nabla\ell(w,z_i)\cdot\left(\nabla\bar{\ell}(w,S_n) - \nabla\bar{\ell}(w,S_n^{-i})\right)\Big|_{w_n(t)}$$

are the pointwise contraction and perturbation factors, respectively.

Proof. By taking the derivative of the pointwise loss difference $\Delta_n^{-i}(t)$, we have,

$$\begin{aligned}
\frac{\mathrm{d}\Delta_n^{-i}(t)}{\mathrm{d}t} &= \frac{\mathrm{d}\left(\ell(w_n^{-i}(t),z_i) - \ell(w_n(t),z_i)\right)}{\mathrm{d}t} \\
&= -\nabla\ell(w,z_i)\cdot\nabla\bar{\ell}(w,S_n^{-i})\big|_{w_n^{-i}(t)} - \left(-\nabla\ell(w,z_i)\cdot\nabla\bar{\ell}(w,S_n)\big|_{w_n(t)}\right) \\
&= -\left(\nabla\ell(w,z_i)\cdot\nabla\bar{\ell}(w,S_n^{-i})\big|_{w_n^{-i}(t)} - \nabla\ell(w,z_i)\cdot\nabla\bar{\ell}(w,S_n^{-i})\big|_{w_n(t)}\right) \\
&+ \left(-\nabla\ell(w,z_i)\cdot\nabla\bar{\ell}(w,S_n^{-i})\big|_{w_n(t)} + \nabla\ell(w,z_i)\cdot\nabla\bar{\ell}(w,S_n)\big|_{w_n(t)}\right) \\
&= -\left(\nabla\ell(w,z_i)\cdot\nabla\bar{\ell}(w,S_n^{-i})\big|_{w_n^{-i}(t)} - \nabla\ell(w,z_i)\cdot\nabla\bar{\ell}(w,S_n^{-i})\big|_{w_n(t)}\right) \\
&+ \nabla\ell(w,z_i)\left(\nabla\bar{\ell}(w,S_n) - \nabla\bar{\ell}(w,S_n^{-i})\right)\big|_{w_n(t)}
\end{aligned}$$

Hence,

$$\frac{\mathrm{d}\Delta_n^{-i}(t)}{\mathrm{d}t} = -c_n^{-i}(t)\Delta_n^{-i}(t) + \epsilon_n^{-i}(t),$$

where

$$c_n^{-i}(t) = \frac{\nabla \ell(w, z_i) \cdot \nabla \bar{\ell}(w, S_n^{-i})\big|_{w_n(t)}^{w_n^{-i}(t)}}{\Delta_n^{-i}(t)},$$

and

$$\epsilon_n^{-i}(t) = \nabla \ell(w, z_i) \cdot \left(\nabla \bar{\ell}(w, S_n) - \nabla \bar{\ell}(w, S_n^{-i})\right)\Big|_{w_n(t)}.$$

□

The numerator of the contraction factor $c_n^{-i}(t)$ is the discrepancy between weights $w_n(t)$ and $w_n^{-i}(t)$ along training trajectories of two different datasets (S_n and S_n^{-i} respectively). We can interpret the contractor factor as a "spring constant" that pulls these two weight trajectories closer due their shared $(n-1)$ samples. The perturbation factor $\epsilon_n^{-i}(t)$ is the force that keeps the two trajectories apart because of one distinct sample, it depends upon the term $\nabla \bar{\ell}(w, S_n) - \nabla \bar{\ell}(w, S_n^{-i})$. This lemma can be extended to any piecewise differentiable loss if we define the gradient $\nabla \ell(w, z)$ at the non-differentiable point to belong to the sub-differential and have a bounded norm. This covers all commonly used neural architectures and activation functions.

Averaging the pointwise evolution equation in Lemma 3.7 over the samples gives:

$$\frac{\mathrm{d}\bar{\Delta}_n(t)}{\mathrm{d}t} = -\bar{c}_n(t)\bar{\Delta}_n(t) + \bar{\epsilon}_n(t). \tag{3.8}$$

The **averaged contraction factor**

$$\bar{c}_n(t) = \frac{\frac{1}{n}\sum_{i=1}^{n} \nabla \ell(w, z_i) \cdot \nabla \bar{\ell}(w, S_n^{-i})\big|_{w_n(t)}^{w_n^{-i}(t)}}{\bar{\Delta}_n(t)}, \tag{3.9}$$

and the **averaged perturbation factor**

$$\bar{\epsilon}_n(t) = \frac{\mathrm{tr}\hat{\Sigma}_n(t)}{n-1}, \tag{3.10}$$

where $\hat{\Sigma}_n(t) = \mathrm{Cov}_{z \sim \mathrm{Unif}(S_n)}\left(\nabla \ell(w_n(t), z)\right)$ is the covariance matrix of $\nabla \ell(w_n(t), z)$ for z sampled uniformly from the dataset S_n.

Derivation of the averaged dynamics. The evolution of $\bar{\Delta}_n(t)$ can be derived through that of $\Delta_n^{-i}(t)$.

$$\begin{aligned}
\frac{\mathrm{d}\bar{\Delta}_n(t)}{\mathrm{d}t} &= \frac{1}{n}\sum_{i=1}^{n}\frac{\mathrm{d}\Delta_n^{-i}(t)}{\mathrm{d}t} \\
&= \frac{1}{n}\sum_{i=1}^{n}\left(-c_n^{-i}(t)\Delta_n^{-i}(t)+\epsilon_n^{-i}(t)\right) \\
&= -\frac{\frac{1}{n}\sum_{i=1}^{n}c_n^{-i}(t)\Delta_n^{-i}(t)}{\bar{\Delta}_n(t)}\bar{\Delta}_n(t)+\frac{1}{n}\sum_{i=1}^{n}\epsilon_n^{-i}(t) \\
&= -\bar{c}_n(t)\bar{\Delta}_n(t)+\bar{\epsilon}_n(t).
\end{aligned}$$

Here we have,

$$\begin{aligned}
\bar{c}_n(t) &= \frac{\frac{1}{n}\sum_{i=1}^{n}c_n^{-i}(t)\Delta_n^{-i}(t)}{\bar{\Delta}_n(t)} \\
&= \frac{\frac{1}{n}\sum_{i=1}^{n}\nabla\ell(w,z_i)\cdot\nabla\bar{\ell}(w,S_n^{-i})\big|_{w_n(t)}^{w_n^{-i}(t)}}{\bar{\Delta}_n(t)},
\end{aligned}$$

and

$$\begin{aligned}
\bar{\epsilon}_n(t) &= \frac{1}{n}\sum_{i=1}^{n}\epsilon_n^{-i}(t) \\
&= \frac{1}{n}\sum_{i=1}^{n}\nabla\ell_i^{\top}\left(\frac{1}{n}\sum_{j=1}^{n}\nabla\ell_j-\frac{1}{n-1}\sum_{j\neq i}\nabla\ell_j\right) \\
&= \frac{1}{n}\sum_{i=1}^{n}\nabla\ell_i^{\top}\left(\frac{1}{n}\nabla\ell_i-\frac{1}{n(n-1)}\sum_{j\neq i}\nabla\ell_j\right) \\
&= \frac{1}{n^2}\sum_{i=1}^{n}\nabla\ell_i^{\top}\nabla\ell_i-\frac{1}{n^2(n-1)}\sum_{i\neq j}\nabla\ell_i^{\top}\nabla\ell_j.
\end{aligned}$$

In the calculation above, we use $\nabla\ell_i$, $\nabla\bar{\ell}$ as an abbreviation for $\nabla\ell(w_n(t),z_i)$, $\nabla\bar{\ell}(w_n(t),S_n)$ respectively. Notice that we have the following decomposition of the gradient covariance matrix $\hat{\Sigma}(t)$:

$$\begin{aligned}
\hat{\Sigma}(t) &= \frac{1}{n}\sum_{i=1}^{n}\left(\nabla\ell_i-\nabla\bar{\ell}\right)\left(\nabla\ell_i-\nabla\bar{\ell}\right)^{\top} \\
&= \frac{1}{n}\sum_{i=1}^{n}\nabla\ell_i\nabla\ell_i^{\top}-\frac{1}{n^2}\left(\sum_{i=1}^{n}\nabla\ell_i\right)\left(\sum_{i=1}^{n}\nabla\ell_i\right)^{\top} \\
&= \frac{n-1}{n^2}\sum_{i=1}^{n}\nabla\ell_i\nabla\ell_i^{\top}-\frac{1}{n^2}\sum_{i\neq j}\nabla\ell_i\nabla\ell_j^{\top}
\end{aligned}$$

Hence, we have

$$\bar{\epsilon}_n(t) = \frac{\mathrm{tr}\hat{\Sigma}(t)}{n-1}, \quad \hat{\Sigma}_n(t) = \underset{z\sim\mathrm{Unif}(S_n)}{\mathrm{Cov}} \nabla\ell(w_n(t), z),$$

where $\hat{\Sigma}_n(t)$ represents the covariance matrix of $\nabla\ell(w_n(t), z)$ for z sampled uniformly from the dataset S_n.

The solution of this differential equation can be written in the integral form as

$$\bar{\Delta}_n(t) = \int_0^t \bar{\epsilon}_n(s) \exp\left(\int_s^t -\bar{c}_n(u)\,\mathrm{d}u\right) \mathrm{d}s \tag{3.11}$$

with the assumption that $w_n(0) = w_n^{-i}(0)$ for all i.

Remark 3.8. Contraction theory (Lohmiller and Slotine, 1998, 2000) studies dynamical systems of the form $\frac{\mathrm{d}\xi}{\mathrm{d}t} = h(\xi, t)$. Consider a uniformly positive definite matrix $M(\xi, t) \succ 0$. If $\|\xi(t) - \xi'(t)\|_M$ is the distance between points along two trajectories $\xi(\cdot)$ and $\xi'(\cdot)$ under the metric M, the two trajectories are said to contract with a contraction factor α if

$$\frac{\mathrm{d}}{\mathrm{d}t}\|\xi(t) - \xi'(t)\|_M \leq -\alpha\|\xi(t) - \xi'(t)\|_M.$$

The classical contraction results reviewed in Sec. 3.2.1 provide the background for this comparison. Kozachkov et al. (2023a) gave a bound on the generalization gap by analyzing gradient flow trained on datasets with one sample replaced under the assumption that the dynamics contracts uniformly on the weight space with a contraction factor α. The pointwise contraction c_n^{-i} in Lemma 3.7 satisfies a similar equation,

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\ell(w_n^{-i}, z_i) - \ell(w_n, z_i)\right) = -c_n^{-i}(t)\ \left(\ell(w_n^{-i}, z_i) - \ell(w_n, z_i)\right).$$

Our development thus exploits the key idea behind contraction theory but it is different in two important ways. First, we measure the discrepancy between the trajectories in terms of the loss instead of constructing a metric on the weight space. This is important because it allows us to address the fact that the Lipschitz coefficient of neural networks is non-uniform in the weight space, and in non-convex energy landscapes such as those in deep learning because points along trajectories that are far away in the weight space can have a similar loss. Second, our contraction factor $c_n^{-i}(t)$ is local and non-uniform across both time and weight space. In other words, this chapter develops a generalization of the ideas in contraction theory and allows for a more refined analysis of the generalization gap. When a dynamical system is subject to a perturbation, contraction pulls the perturbed trajectory towards the unperturbed one, whereas perturbation drives the two trajectories apart. The distance between the two trajectories is controlled by the balance between these two quantities and this is what keeps the generalization gap bounded. In fact, if we assume a uniform bound on perturbation and contraction factors: $\bar{\epsilon}_n(t) \leq \epsilon^*$ and $\bar{c}_n(t) \geq c^*$ for all t, Eq. (3.8) yields

$$\bar{\Delta}_n(t) \leq \frac{\epsilon^*}{c^*}(1 - e^{-c^*t}).$$

This recovers the bound of Kozachkov et al. (2023a).

Remark 3.9. The quantity $\mathbb{E}\left[\bar{\Delta}_n(t)\right]$ in Lemma 3.3 is the expected value over training datasets, of the pointwise loss difference averaged over the samples in the dataset, and it is closely related to the expected generalization gap $\mathbb{E}\left[\delta R(S_n, t)\right]$. Its evolution has the same form, with modified contraction and perturbation factors:

$$\begin{aligned} &\frac{\mathrm{d}}{\mathrm{d}t}\mathbb{E}\left[\bar{\Delta}_n(t)\right] = -\bar{c}_n(t)\,\mathbb{E}\left[\bar{\Delta}_n(t)\right] + \bar{\epsilon}_n(t), \text{ where} \\ &\bar{c}_n = \frac{\mathbb{E}\left[\frac{1}{n}\sum_{i=1}^n \nabla\ell(w, z_i)\cdot\nabla\bar{\ell}(w, S_n^{-i})\big|_{w_n(t)}^{w_n^{-i}(t)}\right]}{\mathbb{E}\left[\bar{\Delta}_n(t)\right]}, \quad \bar{\epsilon}_n = \frac{\mathbb{E}\left[\mathrm{tr}\hat{\Sigma}(t)\right]}{n-1}, \end{aligned} \tag{3.12}$$

Analyzing this equation is quite difficult because it involves an expectation over the sample set. This is why we focus our analytical development on $\bar{\Delta}_n(t)$ and instead provide an experimental characterization of the differences between $\bar{\Delta}_n(t)$ and the true generalization gap $\delta R(S_n, t)$ in the sequel.

3.3.2. Understanding the contraction factor $\bar{c}_n$

The following lemma and the adjoining experiments allow an interpretation of the contraction factor $c_n^{-i}(t)$ as a generalized Rayleigh quotient.

Lemma 3.10. Assume that the loss function $\ell(w, z)$ is twice continuously differentiable with respect to w for all $z \in \mathcal{Z}$ and there exist constants $G, L_1, L_2 > 0$ such that for all $w, w_1, w_2 \in \mathcal{W}$ and $z \in \mathcal{Z}$,

(i) $\|\nabla\ell(w, z)\|_2 \le G$;
(ii) $\|\nabla^2\ell(w, z)\|_2 \le L_1$;
(iii) $\|\nabla^2\ell(w_1, z) - \nabla^2\ell(w_2, z)\|_2 \le L_2\|w_1 - w_2\|_2$.

Suppose there exists a constant $\iota_{\mathrm{nd}} > 0$ such that $\delta w_i(t) \neq 0$ and $|g_i \cdot \delta w_i(t)| \ge \iota_{\mathrm{nd}}\|\delta w_i(t)\|_2$ for all indices and times under consideration. If $\Delta_n^{-i}(t) \neq 0$, then the pointwise contraction factor

$$c_n^{-i}(t) = \hat{c}_1^{-i}(t) + \hat{c}_2^{-i}(t) + O(\|\delta w_i(t)\|_2), \tag{3.13}$$

where

$$\hat{c}_1^{-i} \triangleq \frac{g^{-i}\cdot H_i\delta w_i}{g_i\cdot\delta w_i}, \quad \hat{c}_2^{-i} \triangleq \frac{g_i\cdot H^{-i}\delta w_i}{g_i\cdot\delta w_i};$$

all quantities being functions of time t. Similarly, assuming $\overline{\Delta}_n(t) \neq 0$ and $|\sum_{i=1}^n g_i\cdot\delta w_i(t)| \ge \iota_{\mathrm{nd}}\sum_{i=1}^n\|\delta w_i(t)\|_2$, the averaged contraction factor

$$\bar{c}_n(t) = \hat{c}_1(t) + \hat{c}_2(t) + O(\max_i\|\delta w_i(t)\|_2) \tag{3.14}$$

where

$$\hat{c}_1 = \sum_{i=1}^n \hat{c}_1^{-i}\underbrace{\left(\frac{g_i\cdot\delta w_i}{\sum_{j=1}^n g_j\cdot\delta w_j}\right)}_{\alpha_i} = \sum_{i=1}^n \hat{c}_1^{-i}\alpha_i.$$

The expression for $\hat{c}_2$ is analogous. Note that $\sum_i\alpha_i = 1$.

Proof. Let $h(u) = w_n + u\delta w_i$ for $u \in [0,1]$ parameterize the line segment between w_n and w_n^{-i}. Let $\phi(w) = \nabla\ell(w, z_i)^\top \nabla\bar{\ell}(w, S_n^{-i})$ be the inner product of the gradients. The numerator of c_n^{-i} is $\phi(w_n^{-i}) - \phi(w_n)$. We prove the lemma by Taylor expansion of the numerator and the denominator.

The assumptions imply that $\nabla\phi$ is Lipschitz with a constant depending only on G, L_1, L_2. By Taylor's theorem and the product rule,

$$\begin{aligned}\phi(w_n^{-i}) - \phi(w_n) &= A_i + O(d_i^2),\\ A_i &= g^{-i} \cdot H_i \delta w_i + g_i \cdot H^{-i} \delta w_i,\end{aligned}$$

where $d_i = \|\delta w_i\|_2$. Similarly,

$$\ell(w_n^{-i}, z_i) - \ell(w_n, z_i) = B_i + O(d_i^2), \qquad B_i = g_i \cdot \delta w_i.$$

Since $A_i = O(d_i)$ and $|B_i| \geq \iota_{\text{nd}} d_i$, division gives

$$c_n^{-i} = \frac{A_i + O(d_i^2)}{B_i + O(d_i^2)} = \frac{A_i}{B_i} + O(d_i) = \hat{c}_1^{-i} + \hat{c}_2^{-i} + O(d_i).$$

For the averaged contraction factor, let $d_{\max} = \max_i d_i$. Summing the expansions above gives

$$\bar{c}_n = \frac{\sum_i A_i + O(\sum_i d_i^2)}{\sum_i B_i + O(\sum_i d_i^2)}.$$

Using $\sum_i d_i^2 \leq d_{\max} \sum_i d_i$ and the averaged non-degeneracy condition, we obtain

$$\bar{c}_n = \frac{\sum_i A_i}{\sum_i B_i} + O(d_{\max}) = \hat{c}_1 + \hat{c}_2 + O(d_{\max}),$$

which proves the result. □

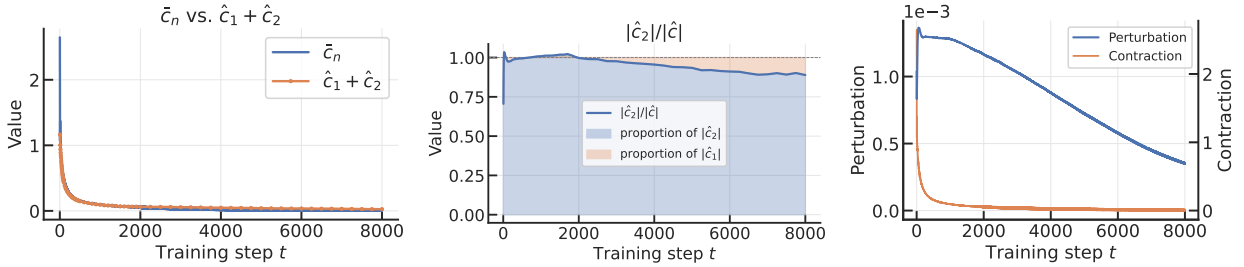

Figure 3.1: Left: True averaged contraction factor ($\bar{c}_n$) and its approximation by $\hat{c}_1 + \hat{c}_2$ on a full-connected network trained on MNIST for classifying two classes. Number of samples $n = 50,000$, and we conduct experiments by omitting $m = 100$ samples, averaged over 100 trajectories. The model is trained to the endpoint with train loss 0.007. **Middle:** The contraction factor is dominated by the second component $\hat{c}_2$. **Right:** The contraction and the perturbation factors decay as training progresses.

Fig. 3.1 shows that $\bar{c}_n$ is well approximated by $\hat{c}_1 + \hat{c}_2$ and all three quantities are mostly positive throughout the training process. Notice that $\hat{c}_2(t) = \sum_{i=1}^n \alpha_i \left(\frac{g_i \cdot H^{-i} \delta w_i}{g_i \cdot \delta w_i}\right)$ which is a weighted gen-

eralized Rayleigh quotient. Both contraction and perturbation factors decay as training progresses. This is useful because when the magnitude of the perturbation factor is large during early training, the magnitude of the contraction factor which shrinks the generalization gap is also large. We calculate the decay of the contraction and perturbation factors precisely in the high-dimensional regression setting in Lemma 3.16, reflecting the similar coupled decay trend.

Fig. 3.2 analyses the Rayleigh quotient further. It shows that even if the trace of the Hessian decreases during training, the gradient is strongly aligned with the large eigenvalues of the Hessian in the later stages of training. On the other hand, the parameter difference δw_i increasingly aligns with the small eigenvalues of the Hessian as training progresses. This behavior arises from the driven, Hessian-damped dynamics of δw_i,

$$\frac{\mathrm{d}\delta w_i}{\mathrm{d}t} \approx -H^{-i}(t)\delta w_i + \frac{1}{n}\left(\nabla \ell(w_n(t), z_i) - \nabla \bar{\ell}(w_n(t), S_n^{-i})\right).$$

its components along large-eigenvalue directions are rapidly suppressed, whereas those along small-eigenvalue directions persist and accumulate. The decreasing alignment of δw_i and H^{-i} is what causes the weighted generalized Rayleigh quotient $\hat{c}_2(t)$ to decrease during training. We can calculate this decay precisely in the high-dimensional regression setting in Lemma 3.18.

3.3.3. Understanding the perturbation factor $\bar{\epsilon}_n$

Let $r(w, z) = \frac{\mathrm{d}\ell(w,z)}{\mathrm{d}f(w,z)} \in \mathcal{Y}$ denote the gradient of the loss function with respect to the output of the predictor f. Let $r_i(t) = r(w_n(t), z_i)$ denote this gradient for sample z_i and weights $w_n(t)$. We can now define the **residual** on dataset S_n at time t to be

$$\vec{r}_n(t) = \frac{1}{\sqrt{n}}[r_1(t), \ldots, r_n(t)]^\top \in \mathcal{Y}^n. \tag{3.15}$$

The residual is the collection of loss-predictor gradients on the dataset S_n. It effectively describes how well the weights at time t have fitted to the data and indicates the direction in the predictor

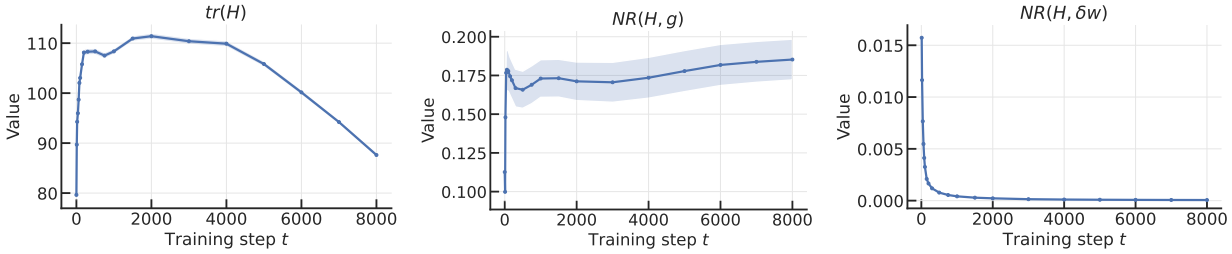

Figure 3.2: Statistics of the generalized Rayley quotient approximation of $\bar{c}_n$ for a full-connected network trained on a binary classification problem on MNIST for $n = 50,000$ samples by omitting $m = 100$ samples, averaged over 100 trajectories. **Left:** The averaged trace of the Hessian $\mathrm{tr}(H) = \mathbb{E}_{(m)}\left[H^{-(m)}\right]$ decreases during training. **Middle:** The average of the normalized Rayley quotient of Hessian and the gradient $\mathrm{NR}(H, g) = \mathbb{E}_{(m)}\left[\frac{g_{(m)}^\top H^{-(m)} g_{(m)}}{\lambda_{\max}(H^{-(m)})\|g_{(m)}\|^2}\right]$ remains large at the end of training. **Right:** The average of the normalized Rayley quotient of Hessian and parameter difference $\mathrm{NR}(H, \delta w) = \mathbb{E}_{(m)}\left[\frac{\delta w_{(m)}^\top H^{-(m)} \delta w_{(m)}}{\lambda_{\max}(H^{-(m)})\|\delta w_{(m)}\|^2}\right]$ decays to zero at the end of training.

space in which the training will proceed. Our goal in this section is to show that the averaged perturbation factor $\bar{\epsilon}_n$ is controlled by the residual.

As an example, if we consider the squared loss $\ell(y, y') = \frac{1}{2}(y - y')^2$ for $y, y' \in \mathbb{R}$, the residual is the normalized displacement vector from the predictor to the target, i.e., $\vec{r}_n(t) = \frac{1}{\sqrt{n}}(\vec{f} - \vec{y})$, where $\vec{y} = [y_1, \ldots, y_n]^\top$, and $\vec{f} = [f(w_n(t), x_1), \ldots, f(w_n(t), x_n)]^\top$. If we initialize the predictor such that $f(w_n(0), x_i) = 0$ for all $i \in [n]$, then the ℓ_2-norm of the residual $\|\vec{r}_n(0)\|_2$ is the largest at initialization and vanishes at interpolation following the gradient flow Eq. (3.1). Our definition of residual above generalizes the displacement vector in the squared loss case, and can be applied to any loss function with global minimum of zero. We make some remarks on the normalization by $\sqrt{n}$ in Sec. 3.3.6.3.

If weights evolve according to the gradient flow in Eq. (3.1), then

$$\begin{aligned} \frac{\mathrm{d}\vec{r}_n(t)}{\mathrm{d}t} &= -\frac{1}{n} P_n(t) \vec{r}_n(t), \\ P_n(t) &= \Big[\nabla r(w_n(t), z_i)^\top \nabla f(w_n(t), x_j)\Big]_{i,j \in [n]}. \end{aligned} \tag{3.16}$$

Derivation of the residual dynamics. We derive the equation governing the evolution of $\vec{r}_n(t)$ by calculating its time derivative.

$$\begin{aligned} \frac{\mathrm{d}\vec{r}_n(t)}{\mathrm{d}t} &= \frac{1}{\sqrt{n}} \begin{bmatrix} \frac{\mathrm{d}r(w_n(t), z_1)}{\mathrm{d}t} \\ \cdots \\ \frac{\mathrm{d}r(w_n(t), z_n)}{\mathrm{d}t} \end{bmatrix} = \frac{1}{\sqrt{n}} \begin{bmatrix} \nabla r(w_n(t), z_1)^\top \frac{\mathrm{d}w_n(t)}{\mathrm{d}t} \\ \cdots \\ \nabla r(w_n(t), z_n)^\top \frac{\mathrm{d}w_n(t)}{\mathrm{d}t} \end{bmatrix} \\ &= \frac{1}{\sqrt{n}} \begin{bmatrix} -\frac{1}{n} \sum_{j=1}^n \nabla r(w_n(t), z_1)^\top \nabla f(w_n(t), x_j) \cdot r(w_n(t), z_j) \\ \cdots \\ -\frac{1}{n} \sum_{j=1}^n \nabla r(w_n(t), z_n)^\top \nabla f(w_n(t), x_j) \cdot r(w_n(t), z_j) \end{bmatrix} = -\frac{1}{n} P_n(t) \vec{r}_n(t). \end{aligned}$$

Here the third equality follows from the evolution of $w_n(t)$:

$$\begin{aligned} \frac{\mathrm{d}w_n(t)}{\mathrm{d}t} &= -\nabla \ell(w_n(t), S_n) \\ &= -\frac{1}{n} \sum_{i=1}^n \nabla f(w_n(t), x_i) \frac{\mathrm{d}\ell(f(w_n(t), x_i), y_i)}{\mathrm{d}f(w_n(t), x_i)} \\ &= -\frac{1}{n} \sum_{i=1}^n \nabla f(w_n(t), x_i) r(w_n(t), x_i). \end{aligned}$$

Note that

$$\begin{aligned} \nabla r(w, z_i), \nabla f(w, z_i) &\in \mathcal{W} \times \mathcal{Y} \\ P_n(t) = \Big[\nabla r(w_n(t), z_i)^\top \nabla f(w_n(t), x_j)\Big]_{i,j \in [n]} &\in \mathcal{Y}^n \times \mathcal{Y}^n. \end{aligned}$$

The residual dynamics in Eq. (3.16) form a linear time-varying ordinary differential equation. In general, its solution can be written as

$$\vec{r}_n(t) = \Omega_n(t_0, t)\ \vec{r}_n(t_0), \tag{3.17}$$

where $\Omega_n(t_0, t)$ is called the propagator. In numerical experiments, we approximate $\Omega_n(t_0, t)$ by multiplying the discrete-time update matrices for the residual along the training trajectory, i.e., using Euler discretization.

Lemma 3.11. The trace of the gradient covariance[2]

$$\operatorname{tr}\left(\hat{\Sigma}_n(t)\right) = \vec{r}_n(t)^\top F_n(t) \vec{r}_n(t), \tag{3.18}$$

where $F_n \in \mathcal{Y}^n \times \mathcal{Y}^n$ has entries

$$F_n(t)_{ij} = \begin{cases} \left(1 - \frac{1}{n}\right) \nabla f(w_n(t), x_i)^\top \nabla f(w_n(t), x_i) & \text{if } i = j, \\ -\frac{1}{n} \nabla f(w_n(t), x_i)^\top \nabla f(w_n(t), x_j) & \text{if } i \neq j. \end{cases} \tag{3.19}$$

Proof. We have the following decomposition of the gradient covariance $\hat{\Sigma}_n(t)$:

$$\begin{aligned} \hat{\Sigma}_n(t) &= \frac{1}{n} \sum_{i=1}^{n} \left(\nabla \ell_i - \nabla \bar{\ell}\right) \left(\nabla \ell_i - \nabla \bar{\ell}\right)^\top \\ &= \frac{1}{n} \sum_{i=1}^{n} \nabla \ell_i \nabla \ell_i^\top - \nabla \bar{\ell} \nabla \bar{\ell}^\top \end{aligned}$$

where

$$\begin{aligned} \nabla \ell_i &= \nabla f(w_n(t), x_i) r_i(t) \\ \nabla \bar{\ell} &= \frac{1}{n} \sum_{i=1}^{n} \nabla f(w_n(t), x_i) r_i(t). \end{aligned}$$

Taking traces and using the normalized residual in Eq. (3.15), we obtain

$$\begin{aligned} \operatorname{tr} \hat{\Sigma}_n(t) &= \frac{1}{n} \sum_{i=1}^{n} r_i(t)^2 \nabla f(w_n(t), x_i)^\top \nabla f(w_n(t), x_i) \\ &\quad - \frac{1}{n^2} \sum_{i,j=1}^{n} r_i(t) r_j(t) \nabla f(w_n(t), x_i)^\top \nabla f(w_n(t), x_j) \\ &= \vec{r}_n(t)^\top F_n(t) \vec{r}_n(t), \end{aligned}$$

where the diagonal and off-diagonal blocks of $F_n(t)$ are given in Eq. (3.19). □

[2]Let us emphasize that $P_n(t)$ and $F_n(t)$ are elements of $\mathcal{Y}^n \times \mathcal{Y}^n$. For regression problems, we might have $\mathcal{Y} \subseteq \mathbb{R}$ in which case they are simply matrices in $\mathbb{R}^{n \times n}$. For classification problems with C categories, $\mathcal{Y} \subset \mathbb{R}^C$, and therefore both quantities are four-dimensional tensors. But we can interpret them as elements of $\mathbb{R}^{nC \times nC}$. This amounts to vectorizing the tensor as a matrix.

Eqs. (3.10) and (3.17) and the above lemma give

$$(n-1)\bar{\epsilon}_n = \operatorname{tr}\left(\hat{\Sigma}_n(t)\right) = \vec{r}_n(0)^\top \Omega_n(t)^\top F_n(t)\Omega_n(t)\vec{r}_n(0), \tag{3.20}$$

where we denote $\Omega(0,t)$ as $\Omega(t)$ for short. In Eq. (3.18), the term F_n pertains to the covariance of the predictor on the training dataset S_n. We can see that the residual $\vec{r}_n(t)$ controls the magnitude of gradients $\nabla\ell(w,z_i)$ for $i\in[n]$ and therefore also the covariance. For networks that train quickly, the residual norm $\|\vec{r}_n(t)\|_2$ vanishes quickly, leading to a smaller accumulation of the perturbation term $\bar{\epsilon}_n$ above, and hence a smaller generalization gap.

3.3.4. An effective Gram matrix governs the generalization gap

We next derive an expression of averaged loss difference $\bar{\Delta}_n(t)$ in terms of a certain quadratic form of the initial residual and an "effective Gram matrix", by analyzing the evolution of $\bar{\Delta}_n(t)$ and $\vec{r}_n(t)$ during training. The following theorem combines the solution of $\bar{\Delta}_n(t)$ in Eq. (3.11) and $\vec{r}_n(t)$ in Eq. (3.17), along with the decomposition of $\hat{\Sigma}_n(t)$ in Eq. (3.18).

Theorem 3.12. Assume that the evolution of $w_n(t)$ and $w_n^{-i}(t)$ follows Eq. (3.1) and the loss function $\ell(w,z)$ is smooth in w for every $z\in\mathcal{Z}$. We have

$$\bar{\Delta}_n(t) = \vec{r}_n(0)^\top K_n(0,t)\vec{r}_n(0), \tag{3.21}$$

where

$$K_n(0,t) = \frac{\int_0^t \Omega_n(s)^\top F_n(s)\Omega_n(s)\exp\left(-\int_s^t \bar{c}_n(u)\mathrm{d}u\right)\mathrm{d}s}{(n-1)} \tag{3.22}$$

is positive semi-definite. Let

$$K_n \triangleq \lim_{t\to\infty} K_n(0,t)$$

when the limit exists, then we have

$$\bar{\Delta}_n(\infty) \triangleq \lim_{t\to\infty}\bar{\Delta}_n(t) = \vec{r}_n(0)^\top K_n \vec{r}_n(0).$$

We call K_n the **effective Gram matrix** because it is a weighted average of Gram matrices of the form $V^\top V$, where $V = \sqrt{\frac{F_n(s)}{n-1}}\Omega_n(s)$.

Proof. By Eq. (3.17) and Lemma 3.11,

$$\operatorname{tr}\hat{\Sigma}_n(t) = \vec{r}_n(0)^\top \Omega_n(t)^\top F_n(t)\Omega_n(t)\vec{r}_n(0). \tag{3.23}$$

Combining with the solution of $\bar{\Delta}_n(t)$ Eq. (3.11), we have

$$\bar{\Delta}_n(t) = \vec{r}_n(0)^\top \left(\frac{\int_0^t \Omega_n(s)^\top F_n(s)\Omega_n(s)\exp\left(-\int_s^t \bar{c}_n(u)\mathrm{d}u\right)\mathrm{d}s}{n-1}\right)\vec{r}_n(0)$$

Hence, we have

$$\bar{\Delta}_n(t) = \vec{r}_n(0)^\top K_n(0,t)\vec{r}_n(0)$$

where

$$K_n(0,t) = \frac{\int_0^t \Omega_n(s)^\top F_n(s)\Omega_n(s) \exp\left(-\int_s^t \bar{c}_n(u)\mathrm{d}u\right)\mathrm{d}s}{n-1}.$$

For any $q = [q_1, \ldots, q_n] \in \mathcal{Y}^n$, let $a_i = \nabla f(w_n(s), x_i)q_i$. Then

$$q^\top F_n(s) q = \sum_{i=1}^n \|a_i\|_2^2 - \frac{1}{n}\left\|\sum_{i=1}^n a_i\right\|_2^2 \geq 0,$$

where the inequality follows from Cauchy–Schwarz. Thus $F_n(s)$ is PSD for every s, and $K_n(0,t)$ is PSD as an integral of PSD matrices with non-negative scalar weights. □

The quadratic form $\vec{r}_n(0)^\top K_n \vec{r}_n(0)$ in Theorem 3.12 gives a data and architecture dependent measure of complexity that characterizes the generalization gap of general deep neural networks. The key conclusion of this theorem is that if the initial residual $\vec{r}_n(0)$ projects on the subspace of K_n with small eigenvalues, then the eventual generalization gap $\bar{\Delta}_n(\infty)$ is small. This result is slightly stronger than "post-hoc" estimates of generalization. While the effective Gram matrix depends upon the entire training trajectory, the initial residual can be computed before training a model. The following lemma suggests conditions under which the limiting object, the effective Gram matrix, exists.

Remark 3.13 (The effective Gram matrix under a fixed training kernel). Consider the squared loss and suppose $P_n(t) = P$ is fixed for all $t \geq 0$ with $P \succ 0$ symmetric. Then $F_n(t) = F$ is also fixed and $\Omega_n(t) = \exp(-tP/n)$. If the contraction factor satisfies $\bar{c}_n(t) \geq 0$ and $0 \preceq F \preceq \beta I$, then the effective Gram matrix

$$K_n \preceq \frac{1}{n-1}\int_0^\infty \exp\Big(-\frac{s}{n}P\Big) F \exp\Big(-\frac{s}{n}P\Big)\mathrm{d}s \preceq \frac{\beta n}{2(n-1)}P^{-1}.$$

Consequently, since $\vec{r}(0) = -\vec{y}/\sqrt{n}$,

$$\bar{\Delta}_n(\infty) \leq \frac{\beta}{2(n-1)}\vec{y}^\top P^{-1}\vec{y}.$$

For the ReLU kernel, assuming unit-norm inputs, we have $0 \preceq F = \frac{1}{2}I - \frac{1}{n}P \preceq \frac{1}{2}I$, hence we can take $\beta = \frac{1}{2}$. This gives a bound on the generalization gap that has a similar form as that of Arora et al. (2019).

Lemma 3.14 (Existence of the effective Gram matrix). Let $\lambda_{\min}(t)$ be the smallest eigenvalue of $(P_n(t) + P_n(t)^\top)/2$. Let $m(t) = \frac{1}{n-1}\|F_n(t)\|_2$ and $\omega(t) = \exp\left(-\frac{2}{n}\int_0^t \lambda_{\min}(s)\mathrm{d}s\right)$. If

(i) $\int_0^\infty \omega(s)\ m(s)\mathrm{d}s < \infty$, and
(ii) the contraction factor $\bar{c}_n(t) \geq 0$ for all $t \geq 0$,

then $\lim_{t\to\infty} K_n(0,t)$ exists in the ℓ_2-norm.

Proof. For any $x \in \mathcal{Y}^n$, let $y(t) = \Omega_n(t)x$. Using $\mathrm{d}\Omega_n(t)/\mathrm{d}t = -P_n(t)\Omega_n(t)/n$, we have

$$\frac{\mathrm{d}\|y(t)\|_2^2}{\mathrm{d}t} = -\frac{1}{n}y(t)^\top \left(P_n(t) + P_n(t)^\top\right) y(t) \le -\frac{2\lambda_{\min}(t)}{n}\|y(t)\|_2^2.$$

Since $\Omega_n(0) = I$, we have $y(0) = x$. Grönwall's inequality gives

$$\|y(t)\|_2^2 \le \omega(t)\|x\|_2^2.$$

Since this holds for every x, taking the supremum over $\|x\|_2 = 1$ gives $\|\Omega_n(t)\|_2^2 \le \omega(t)$. Let

$$A(t) = \frac{\Omega_n(t)^\top F_n(t)\Omega_n(t)}{n-1}, \qquad \tilde{c}(s,t) = \exp\left(-\int_s^t \bar{c}_n(u)\mathrm{d}u\right).$$

Then $\|A(t)\|_2 \le \omega(t)m(t)$. Moreover, since $\bar{c}_n(t) \ge 0$, for every fixed s, $\tilde{c}(s,t)$ is non-increasing in t and converges to a limit $\tilde{c}_\infty(s) \in [0,1]$.

We can write

$$K_n(0,t) = \int_0^\infty 1_{[0,t]}(s)A(s)\tilde{c}(s,t)\mathrm{d}s.$$

The integrand converges pointwise to $A(s)\tilde{c}_\infty(s)$ and its matrix 2-norm is bounded by the integrable function $\omega(s)m(s)$. Therefore, by the dominated convergence theorem and condition (i),

$$\lim_{t\to\infty}\left\|K_n(0,t) - \int_0^\infty A(s)\tilde{c}_\infty(s)\mathrm{d}s\right\|_2 = 0,$$

which proves the result. □

Sometimes the effective Gram matrix calculated from the propagator derived from $P_n(t)$ in Eq. (3.16) may not converge. In such cases, we can create a perturbed version of $P_n(t)$ with a controlled $\lambda_{\min}(t)$ such that the conditions of Lemma 3.14 are satisfied. This guarantees the convergence of $\lim_{t\to\infty} K_n(0,t)$ while preserving the trajectory of $\vec{r}_n(t)$ given $\vec{r}_n(0)$. We use this perturbed version in Sec. 3.4.2 for calculations on linear regression problems.

3.3.5. An empirical characterization of the quantities in Theorem 3.12

We now show that the quantities in Theorem 3.12 capture the true generalization gap faithfully for different configurations of neural network training. Eigenvalues of K_n represent the relative contribution to the generalization gap accumulated in the different subspaces. We will also see that if the initial residual predominantly projects onto the subspace of K_n with small eigenvalues, the training process results in a small eventual generalization gap.

We approximate the gradient flow Eq. (3.1) by gradient descent with different learning rates. We calculate the averaged contraction factor $\bar{c}_n(t)$, the averaged perturbation factor $\bar{\epsilon}_n(t)$, and the matrix representation $F_n(t)$ of the gradient-covariance trace using Eqs. (3.9), (3.10) and (3.19) respectively. The propagator $\Omega_n(t)$ is approximated by an Euler discretization. The integrals for $\bar{\Delta}_n(t)$ in Eq. (3.8) and K_n in Eq. (3.22) are approximated by the trapezoidal method. For all

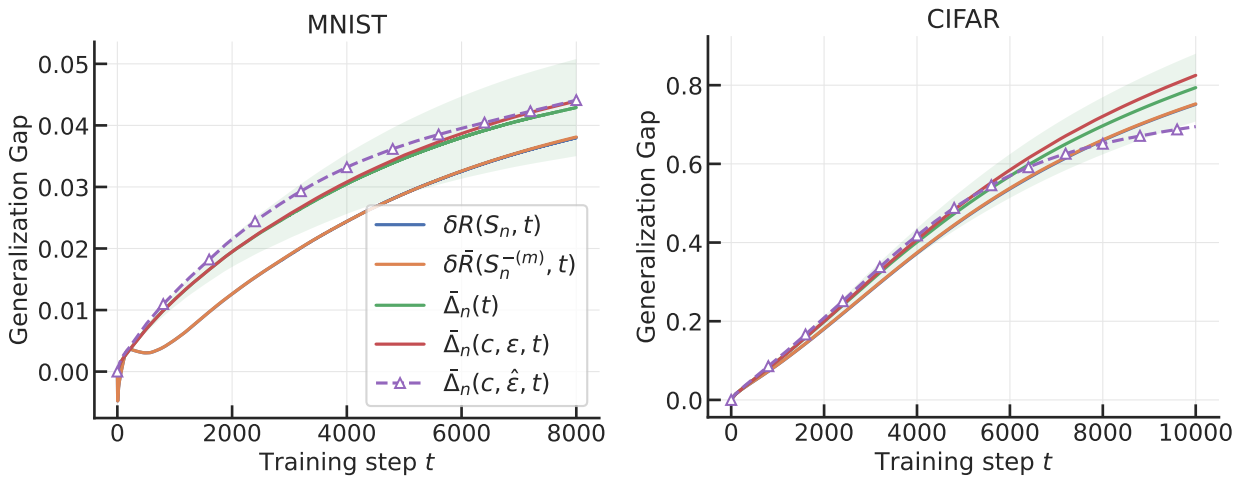

Figure 3.3: Left: A fully-connected network trained on MNIST-2, with $n = 50,000$ samples and statistics computed over 100 perturbed datasets obtained by omitting $m = 100$ samples. The final training loss is 0.007 after training using gradient descent. **Right:** A fully-connected network trained on CIFAR-2 with $n = 2000$ samples, $m = 10$ and 100 perturbed datasets. The final training loss is 0.052 after training using gradient descent. In both figures, the blue and orange curves are somewhat indistinguishable because they are on top of each other.

experimental validation, we create perturbed datasets by omitting m samples, using the expressions below. Sec. 3.3.6.1 provides more details of these experiments.

Contraction and perturbation factors for omit-m-samples experiments. Let $S_{(m)}$ denote the omitted subset of m samples, and let $\mathbb{E}_{(m)}$ denote the uniform average over all choices of that subset. In the omitting m-samples setting, by calculations analogous to those in Sec. 3.3.1, the batch-wise contraction and perturbation factors are

$$c_n^{-(m)}(t) = \frac{\nabla \bar{\ell}(w, S_{(m)}) \cdot \nabla \bar{\ell}(w, S_n^{-(m)})\big|_{w_n(t)}^{w_n^{-(m)}(t)}}{\Delta_n^{-(m)}(t)},$$

$$\epsilon_n^{-(m)}(t) = \nabla \bar{\ell}(w, S_{(m)}) \cdot \left(\nabla \bar{\ell}(w, S_n) - \nabla \bar{\ell}(w, S_n^{-(m)})\right)\Big|_{w_n(t)}.$$

The averaged contraction and perturbation factors are

$$\bar{c}_n(t) = \frac{\mathbb{E}_{(m)}\left[\nabla \bar{\ell}(w, S_{(m)}) \cdot \nabla \bar{\ell}(w, S_n^{-(m)})\big|_{w_n(t)}^{w_n^{-(m)}(t)}\right]}{\bar{\Delta}_n(t)},$$

$$\bar{\epsilon}_n(t) = \frac{\mathrm{tr}\hat{\Sigma}(t)}{n-1}, \quad \hat{\Sigma}_n(t) = \operatorname*{Cov}_{z \sim \mathrm{Unif}(S_n)} \nabla \ell(w_n(t), z),$$

where $\hat{\Sigma}_n(t)$ represents the covariance matrix of $\nabla\ell(w_n(t), z)$ for z sampled uniformly from the dataset S_n. Note that the averaged perturbation factor $\bar{\epsilon}_n(t)$ in the omit-m setting is independent of m.

Fig. 3.3 shows that the true generalization gap $\delta R(S_n, t)$, averaged loss difference $\bar{\Delta}_n(t)$, and the gap $\mathbb{E}_{(m)}\left[\delta R(S_n^{-(m)}, t)\right]$ (denoted by $\delta\bar{R}(\cdot)$ in the plot) are all close to each other throughout training. This indicates that the generalization gap can be approximated well by $\bar{\Delta}_n(t)$.

We calculate two numerical approximations of $\bar{\Delta}_n(t)$: the quantity $\bar{\Delta}_n(c, \epsilon, t)$ uses the true perturbation factor in Eq. (3.10) and $\bar{\Delta}_n(c, \hat{\epsilon}, t)$ with an approximate perturbation factor derived from Eq. (3.20) using the propagator. First, note that $\bar{\Delta}_n(c, \epsilon, t)$ is close to $\bar{\Delta}_n(t)$. This shows that the gradient descent approximation of Eq. (3.1) and the trapezoidal approximation of Eq. (3.11) are good. Second, the similarity of $\bar{\Delta}_n(c, \hat{\epsilon}, t)$ and $\bar{\Delta}_n(c, \epsilon, t)$ indicates that the Euler discretization of the propagator is working well. The CIFAR-2 experiment shows a similar qualitative agreement, albeit with slightly less accurate estimates of the generalization gap. Table 3.2 provides more detailed results where we characterize these quantities for a variety of neural architectures (fully-connected and convolutional) and datasets (MNIST and CIFAR, datasets where inputs are labeled randomly, as well as synthetically generated datasets of different kinds). Broadly across these experiments, our estimate $\bar{\Delta}_n(c, \hat{\epsilon}, t)$ tracks the true generalization error $\delta R(S_n, t)$ extremely well.

Paper	Architecture	Dataset	# samples	Training method	Bound on Test Data	Actual Value	Relative
Arora et al. (2019)	FC	MNIST-2	10,000	GD (second layer fixed)	0.05 (ℓ1 loss)	$<$ 0.01 (ℓ1 loss)	$>$4
Dziugaite and Roy (2017a)	FC	MNIST-2	55,000	SGD	0.161 (error)	0.018 (error)	7.9
Wang and Ma (2022)	FC	MNIST-2	55,000	SGD	0.25 (CE loss)		
Ours	FC	MNIST-2	50,000	GD	0.052 (CE loss)	0.045 (CE loss)	0.15
Ours	FC	MNIST-10	1,100	GD	0.47 (CE loss)	0.45 (CE loss)	0.05
Ours	LENET-5	MNIST-10	1,100	GD	0.24 (CE loss)	0.20 (CE loss)	0.18
Negrea et al. (2019)	CNN	MNIST-10	55,000	SGLD	0.25 (CE loss)	0.02 (error)	
Mou et al. (2018)	CNN	MNIST-10	55,000	SGLD	1.25 (CE loss)	0.02 (error)	
Ours	FC	MNIST-10	1,100	SGD	0.43 (CE loss)	0.42 (CE loss)	0.02
Ours	FC	CIFAR-2	2,000	GD	0.597 (CE loss)	0.580 (CE loss)	0.013
Arora et al. (2019)	FC	CIFAR-2	10,000	GD (second layer fixed)	0.6 (ℓ1 loss)	0.45 (ℓ1 loss)	0.33
Ours	WRN-10-2	CIFAR-2	2,000	GD	0.554 (CE loss)	0.546 (CE loss)	0.014
Ours	ViT-small	CIFAR-2	2,000	GD	0.529 (CE loss)	0.546 (CE loss)	0.032

Table 3.1: Comparison with previous results in terms of the relative accuracy of the estimate of the generalization error. CE loss indicates cross-entropy loss. (S)GD indicates (stochastic) gradient descent, SGLD is stochastic gradient Langevin dynamics. "Bound" in this table refers to the numerical value of the generalization bound. "Actual Value" is test loss or error on held-out test data. "Relative inaccuracy" equal "|Bound-Actual Value| / Actual Value".

Table 3.1 compares our estimate of the generalization gap with existing techniques in the literature. Different papers make different assumptions, apply to different models of neural networks, loss functions, training methods, and also use different techniques. So one must interpret this table with care, for example most methods in this table are upper bounds on the test loss while ours is an estimate. The relative inaccuracy of our estimate is quite small across a variety of settings. This indicates that the different approximations used in our analytical development and numerical implementation are adequate.

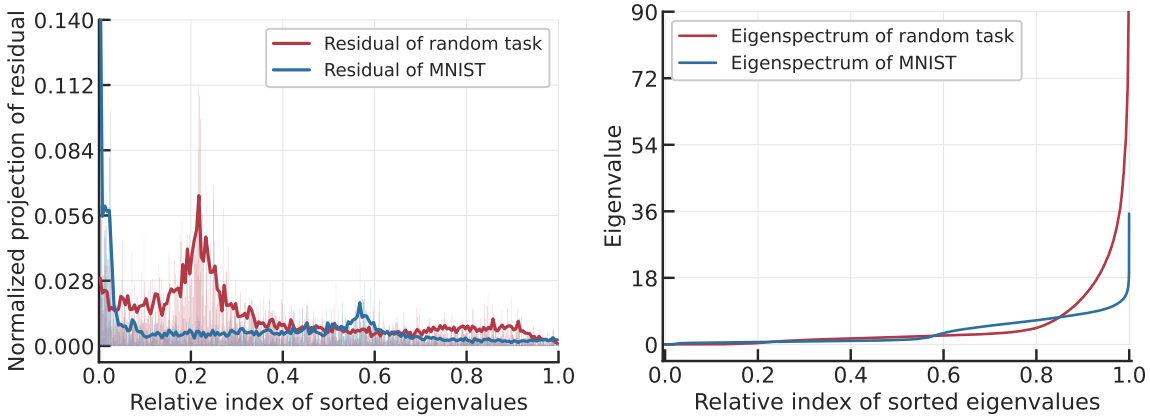

Figure 3.4: Statistics of the residual $\vec{r}_n$ and effective Gram matrix K_n for two different tasks. We use a fully-connected network trained on MNIST-2 with $n = 2,000$ samples and 100 perturbed datasets created by omitting $m = 10$ samples. The network has a final generalization gap of 0.118. The "random task" experiment is conducted under the same setting except that MNIST-2 images are assigned classes randomly. The network has a generalization gap of 1.372. Both tasks are trained to an endpoint with $\|\vec{r}_n\|_2 = 0.094$. The eigenspace corresponding to the smallest 5% of eigenvalues captures 87% of the residual magnitude for the benign task, but only 8% for the random task. The initial residual $\vec{r}_n(0)$ projects predominantly into the subspace of the effective Gram matrix K_n with small eigenvalues for MNIST, whereas for the randomly labeled task it projects on subspaces with larger eigenvalues. The X-axis represents different eigenvectors sorted in increasing order of their eigenvalues, normalized by the shape of the matrix. The left figure calculates the projection of the residual onto the corresponding eigenvector $|r^\top U_k|/\|r\|_2$. The magnitude of the eigenvalues themselves is shown on the right. The mean eigenvalue is 3.29 for MNIST and 5.08 for the random task.

Remark 3.15. Although the effective Gram matrix K_n is defined as the infinite-time limit of $K_n(0,t)$, exact interpolation may not be attainable while training models in practice. Since $\bar{\Delta}_n(t) = \vec{r}_n(0)^\top K_n(0,t)\,\vec{r}_n(0)$ holds at every finite time t, we instead evaluate the finite-time effective Gram matrix $K_n(0,t_{\text{end}})$ at the training endpoint t_{end}. Within each comparison in this subsection, we match the normalized endpoint residual norm $\|\vec{r}_n(t_{\text{end}})\|_2$ across models, ensuring that the effective Gram matrices are evaluated at comparable stages of training. When interpolation is attained asymptotically, $K_n(0,t_{\text{end}})$ approaches the limiting matrix K_n.

Theorem 3.12 suggests that if the initial residual $\vec{r}_n(0)$ projects on the subspace with small eigenvalues of K_n, then the eventual generalization gap $\bar{\Delta}_n(\infty)$ is small. To study this result further, we construct two tasks: the first task ("typical") is a MNIST classification problem with two classes (even digits vs. odd digits), while the second task ("random") consists of the same inputs but with random binary labels. A well-trained network has a small generalization gap on the typical task (here, it is 0.118). For the random task, even if the network fits the training set well, it does not obtain a good gap (1.372). Fig. 3.4 (left) indeed shows that the initial residual on the "typical" task lies predominantly in the subspace of K_n with small eigenvalues, and on the "random" task the initial residual has a much higher projection on the subspace with large eigenvalues. The eigenspace corresponding to the smallest 5% of eigenvalues captures 87% of the residual magnitude for the benign task, but only 8% for the random task. The mean eigenvalue of the numerically estimated effective Gram matrix is also larger for the random task (5.075) than for the typical task (3.285), contributing to the generalization gap. This shows that, much like kernel machines, if the task that we need to fit is simple, in the sense that the initial residual predominantly lies in the tail subspace of the gram matrix, then the eventual generalization gap is small.

3.3.6. Experimental Details and Additional Results

We describe the experimental setup and report the complete numerical results below. We also compute the effective Gram matrix for different datasets, architectures, and numbers of samples to show that our technique for estimating the generalization gap applies across these settings.

3.3.6.1 Experimental setup

Dataset We use the MNIST and CIFAR-10 datasets for the experiments in Sec. 3.3 and Sec. 3.3.6.3. For each sample size considered in our experiments, we draw a balanced subset from all ten classes and group them into two super-classes to construct a binary classification problem. (MNIST: even / odd; CIFAR10: airplane, bird, deer, frog, ship / automobile, cat, dog, horse, truck).

Architectures We use WRN-10-2(one residual block per group, channel widths 32–64–128, RMSNorm, global average pooling; 301744 parameters), ViT-small(4×4 patches, 96-dimensional embeddings, four Transformer blocks, four attention heads, MLP ratio 2, and pre-LayerNorm; 310562 parameters), and FC (a two-layer fully connected network, 128 hidden neurons). We use FC for both MNIST and CIFAR experiments, WRN-10-2 and ViT-small for only MNIST experiment.

Training We use full-batch gradient descent in all experiments except those in Sec. 3.3.7. Within each comparison, we train all models to the same endpoint residual norm $\|\vec{r}_n\|_2$ to ensure a fair comparison. As defined in Eq. (3.15), the residual vector is normalized by $\sqrt{n}$, making its norm comparable across models trained with different numbers of samples.

Synthetic datasets Datasets labeled Gaussian-α are created from Gaussian data with different covariance matrices, labeled by a teacher network with random weights.

- Create a covariance matrix A with i-th eigenvalue being $\exp(-\alpha i)$ normalized by the sum of eigenvalues. Construct $A = Q\,\mathrm{diag}(L)Q^\top$.
- Sample the input from the multivariate Gaussian distribution $N(0, A)$.
- Project the input onto the top 10 covariance eigendirections $Q_{1:10}$.
- Assign binary labels to the original inputs using a randomly initialized FC-200-30-2 teacher network.

The random task in Fig. 3.4 is constructed by assigning random binary labels in $\{0, 1\}$ to the original MNIST inputs while maintaining equal class sizes.

Constructing omit-m-samples perturbed datasets The theory in this chapter was developed assuming that the modified dataset S_n^{-i} with $(n-1)$ samples is created by omitting the i-th sample. For numerical stability and efficiency of the approximation, in the experiments, we create datasets by omitting a batch of m samples. Let (m) denote a subset of $[n]$ with size m. Let $S_{(m)} = \{z_i = (x_i, y_i)_{i\in(m)}\}$. We will conduct experiments using modified datasets $S_n^{-(m)} = S_n \setminus S_{(m)}$ obtained by removing $S_{(m)}$ from S_n. We therefore consider the weight trajectory $w_n^{-(m)}(t)$ of the differential equation $\dot{w} = -\nabla\bar{\ell}(w, S_n^{-(m)})$. The averaged loss difference is modified in the usual fashion $\bar{\Delta}_n(t) = \mathbb{E}_{(m)}\left[\Delta_n^{-(m)}\right]$ with the batch-wise loss difference

$$\Delta_n^{-(m)}(t) = \ell(w_n^{-(m)}(t), S_{(m)}) - \ell(w_n(t), S_{(m)}).$$

Note that $\mathbb{E}_{(m)}$ denotes the expectation taken over the uniform distribution on all possible choices of (m) in $[n]$. The formulae for the averaged contraction and perturbation factors $\bar{c}_n(t)$ and $\bar{\epsilon}_n(t)$ in this setting are shown in Sec. 3.3.5.

We should note that for the experiment, we cannot sample all omit-m trajectories. Hence, to reduce approximation variance, the m omitted samples are drawn evenly from the ten classes, with each trajectory omitting a different subset. To reduce approximation bias, we set m to less than 1% of the dataset. The values of m and n, together with the number of trajectories, are reported in each figure caption.

Omit-m-samples Compact Notation To facilitate the analysis of the omitting-m-samples setting, we extend our compact notation from Sec. 3.2 to denote the relevant batched quantities. We define the batched parameter shift as $\delta w_{(m)} \triangleq w_n^{-(m)} - w_n$. For the first-order derivatives, we denote the average gradient evaluated on the omitted batch $S_{(m)}$ as $g_{(m)} \triangleq \nabla\bar{\ell}(w_n, S_{(m)})$, and the average gradient over the reduced dataset as $g^{-(m)} \triangleq \nabla\bar{\ell}(w_n, S_n^{-(m)})$. Similarly, for the second-order derivatives, we distinguish the single-batch Hessian $H_{(m)} \triangleq \nabla^2\bar{\ell}(w_n, S_{(m)})$ from the leave-m-out dataset Hessian $H^{-(m)} \triangleq \nabla^2\bar{\ell}(w_n, S_n^{-(m)})$.

3.3.6.2 Approximation of the generalization gap

Table 3.2 records the generalization gaps, averaged loss difference, and its approximations for the experiments done in this chapter. In almost all cases, these quantities are close, indicating that the averaged loss difference approximates the generalization gap well. The small training losses show that the models are trained near interpolation. The column $\bar{\sigma}(K_n)$ reports the mean eigenvalue of the

effective Gram matrix.The column $\|\vec{r}_n(0)\|_2^2$ shows that the squared norm of the normalized initial residual remains nearly constant as the sample size varies (e.g., the MNIST-5 rows). This supports the $1/\sqrt{n}$ normalization in Eq. (3.15) when comparing experiments with different sample sizes.

Architecture	Dataset	# samples	$\bar{\Delta}_n(c,\hat{\epsilon},t)$	$\bar{\Delta}_n(c,\epsilon,t)$	$\bar{\Delta}_n(t)$	$\delta R(S_n,t)$	$\delta\bar{R}(S_n^{-(m)},t)$	Generalization error	$R_{\text{train}}(S_n,t)$	$\bar{\sigma}(K_n)$	$\|\vec{r}_n(t)\|_2$	$\|\vec{r}_n(0)\|_2$
FC-784-128-2	MNIST-2	50000	0.044	0.044	0.043	0.038	0.038	0.014	0.008	–	0.049	0.705
FC-784-128-2	MNIST-2	2000	0.130	0.132	0.130	0.117	0.117	0.054	0.030	3.285	0.095	0.702
FC-784-128-2	MNIST-2 (random label)	2000	0.501	1.532	1.441	1.372	1.379	0.503	0.046	5.075	0.095	0.713
FC-784-128-2	MNIST-2	50	0.405	0.315	0.312	0.265	0.283	0.184	0.135	1.746	0.215	0.694
FC-784-128-2	MNIST-2	100	0.274	0.243	0.241	0.262	0.271	0.176	0.119	1.445	0.215	0.699
FC-784-128-2	MNIST-2	500	0.221	0.192	0.191	0.190	0.194	0.114	0.099	1.453	0.215	0.704
FC-784-128-2	MNIST-2	1000	0.153	0.136	0.135	0.147	0.148	0.096	0.097	1.131	0.215	0.704
FC-784-128-2	MNIST-2	2000	0.091	0.088	0.087	0.081	0.081	0.067	0.094	0.744	0.215	0.702
FC-784-128-2	MNIST-2	10000	0.030	0.028	0.028	0.024	0.025	0.041	0.090	0.295	0.215	0.705
FC-784-128-2	MNIST-2	2000	0.126	0.126	0.125	0.112	0.112	0.054	0.036	2.612	0.110	0.702
FC-3072-100-2	CIFAR-2	2000	0.695	0.825	0.794	0.751	0.752	0.296	0.052	6.433	0.110	0.717
FC-3072-100-2	CIFAR-2	2000	0.075	0.070	0.070	0.063	0.064	0.300	0.516	0.184	0.583	0.717
WRN-10-2	CIFAR-2	2000	0.070	0.040	0.040	0.033	0.033	0.279	0.514	0.114	0.583	0.702
ViT-small	CIFAR-2	2000	0.024	0.017	0.041	0.036	0.036	0.278	0.512	0.034	0.583	0.722
FC-200-120-2	Gaussian-1	2000	0.042	0.041	0.041	0.064	0.063	0.122	0.246	0.160	0.380	0.719
FC-200-120-2	Gaussian-0.5	2000	0.023	0.022	0.022	0.045	0.045	0.111	0.245	0.099	0.380	0.720
FC-200-120-2	Gaussian-0.1	2000	0.034	0.032	0.032	0.053	0.054	0.121	0.245	0.134	0.380	0.725
FC-200-120-2	Gaussian-0.05	2000	0.056	0.054	0.054	0.070	0.070	0.126	0.246	0.200	0.380	0.727
FC-200-120-2	Gaussian-0.01	2000	0.106	0.101	0.101	0.116	0.117	0.162	0.255	0.297	0.380	0.726

Table 3.2: Statistics of effective Gram matrix approximation for a variety of different architectures and datasets. See Sec. 3.3.5 for the definitions of the generalization gaps $\delta R(S_n,t)$ and $\delta\bar{R}(S_n^{-(m)},t)$, the averaged loss difference $\bar{\Delta}_n(t)$, and its approximations $\bar{\Delta}_n(c,\hat{\epsilon},t)$ and $\bar{\Delta}_n(c,\epsilon,t)$. "Generalization error" refers to the averaged zero-one loss on the test dataset, reported as a value between zero and one. $R_{\text{train}}(S_n,t)$ denotes the training loss on S_n. For the last three columns, $\bar{\sigma}(K_n)$ denotes the mean eigenvalue of the effective Gram matrix, $\|\vec{r}_n(t)\|_2$ and $\|\vec{r}_n(0)\|_2$ denotes the norm of the normalized residual at the training endpoint and initialization respectively. All quantities except $\|\vec{r}_n(0)\|_2$ are evaluated at the specified training endpoint.

3.3.6.3 Additional numerical experiments

Comparing the statistics for different numbers of samples The effective Gram matrix $K_n \in \mathcal{Y}^n \times \mathcal{Y}^n$ lies in a different space when neural networks are trained with different numbers of samples n. Therefore, to compare quantities like $\sigma(K_n)$, $M(\vec{r}_n, E(K_n))$ and $M(K_n)$ for different n and the same $\mathcal{Y}$, we use a "relative index" as described in Table 3.3. We rescale the original index vector to have indices from zero to one. We should emphasize that by normalizing the residual by $\sqrt{n}$ in Eq. (3.15), the ℓ_2-norm of the initial residuals $\|\vec{r}_n(0)\|_2$ is similar when $\mathcal{Y}$ is the same, even if n is different. If $K_n = E(K_n)\,\mathrm{diag}(\sigma(K_n))E(K_n)^\top$, then

$$\vec{r}_n(0)^\top K_n \vec{r}_n(0) = \|\vec{r}_n(0)\|_2^2 \sum_i \sigma(K_n)_i P(\vec{r}_n(0), E(K_n))_i^2.$$

We therefore report both the residual alignment and $\bar{\sigma}(K_n) = n^{-1}\sum_i \sigma(K_n)_i$, a dimension-normalized measure of the overall spectral scale of K_n. Table 3.2 details the numerical values of these quantities for different datasets and architectures. We use the quantities defined in Table 3.3 to compare residual–Gram-matrix alignment across tasks.

Table 3.3: Quantities that we use to characterize the initial residual and effective Gram matrix.

Notation	Definition
$E(K), \sigma(K)$	The eigenspace and eigenspectrum of a symmetric matrix K with eigenvalue decomposition where $K = E(K)\,\mathrm{diag}(\sigma(K))E(K)^\top$, $\sigma(K)$ is the vector of eigenspectrum in ascending order.
$\bar{\sigma}(K)$	The mean of the eigenspectrum of a symmetric matrix K, $\bar{\sigma}(K) = \sum_i \sigma(K)_i/n$.
$U_k, U_{1:k}$	The k-th column of U, and the first k columns of U.
$P(r, U)$	Normalized projection of a vector r onto the space U (with orthonormal columns). the k-th element $P(r,U)_k = \left\|r^\top U_k\right\|/\|r\|_2$.
$M(r, U)$	"Explained magnitude" of a vector r in the space $U_{1:k}$ (with orthonormal columns) the k-th element $M(r,U)_k = \left\|r^\top U_{1:k}\right\|_2^2/\|r\|_2^2$.
$R(\mathrm{Idx})$	"Relative index" of the index vector $\mathrm{Idx} = [1, 2, \ldots, l]$, where $R(\mathrm{Idx}) = [1/n, 2/n, \ldots, 1]$.

Effective Gram matrix for different datasets Fig. 3.5 compares the normalized projection of residual and eigenspectra of the effective Gram matrix on synthetic datasets with different difficulty. We construct the following datasets. Gaussian-α with smaller α puts less weight on the top eigenvalues as showed in Yang et al. (2022). From the classical analysis of linear regression, we know that the task becomes more difficult as α decreases, since the labels capture less signal. We see that for difficult tasks, the residual projects more onto the subspace corresponding to larger eigenvalues, and the effective Gram matrix K_n has larger magnitude, which jointly lead to a larger predicted generalization gap by our theory. And indeed, the true generalization gap corroborates this trend.

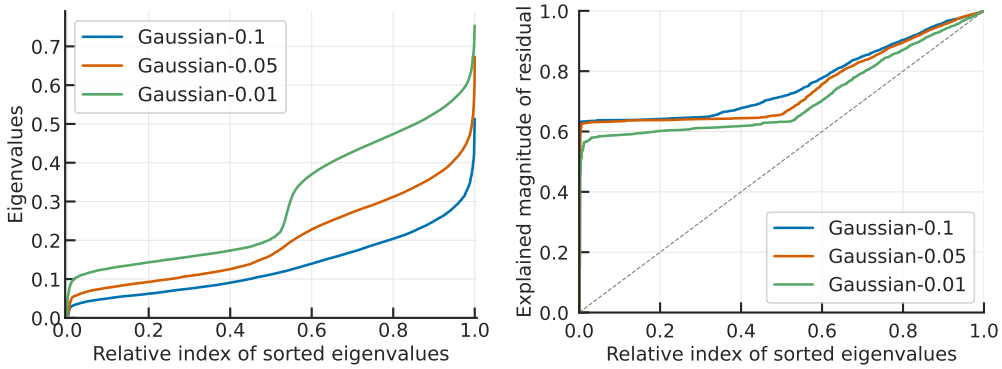

Figure 3.5: All models are trained to an endpoint with $\|\vec{r}_n\|_2 = 0.380$, using $n = 2000$, $m = 10$, and 100 trajectories. The true generalization gaps of Gaussian-α (α from large to small) are 0.053, 0.070, and 0.116, respectively. **Left:** Eigenspectra of the effective Gram matrix $\sigma(K_n)$ for Gaussian-α have larger magnitude when α is smaller ($\bar{\sigma}(K_n)$ is 0.134, 0.200, and 0.297, respectively, for α from large to small). **Right:** Explained magnitude of the initial residual trends towards the top-left when we increase α for a larger signal-to-noise ratio.

Fig. 3.6 compares the two-class tasks derived from MNIST and CIFAR-10. The initial residual for MNIST-2 projects more strongly onto eigenspaces of the effective Gram matrix with small eigenvalues, and the eigenvalues of K_n for CIFAR-2 are larger overall. This shows that both favorable

task–Gram-matrix alignment and a small spectral scale of K_n are important for a "benign training process" and a small eventual generalization gap.

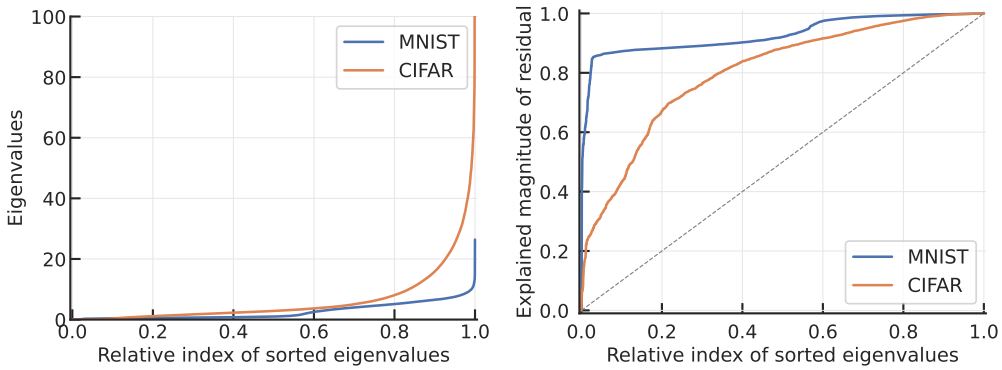

Figure 3.6: Residuals $\vec{r}_n(0)$ and effective Gram matrix K_n for a fully connected network trained on MNIST-2 and CIFAR-2 with $n = 2000$, $m = 10$, and 100 trajectories. Both models are trained to the same endpoint with $\|\vec{r}_n\|_2 = 0.110$. The generalization gaps for MNIST-2 and CIFAR-2 are 0.112 and 0.751, respectively. **Left:** Eigenvalues of the effective Gram matrix $\sigma(K_n)$ have quite different magnitudes ($\bar{\sigma}(K_n)$ is 2.612 for MNIST-2 and 6.433 for CIFAR-2). **Right:** Explained magnitude of the initial residual $M(\vec{r}_n(0), E(K_n))$ for CIFAR-2 has a larger overlap with the principal subspace of the effective Gram matrix than for MNIST-2. This is consistent with the larger generalization gap observed on CIFAR-2.

Effective Gram matrix for different architectures Fig. 3.7 compares the residual alignment and eigenvalues of the effective Gram matrix for CIFAR-2 trained with FC, WRN-10-2, and ViT-small. FC has the largest mean eigenvalue, whereas WRN-10-2 and ViT-small have more favorable overall spectral scales. The larger spectral scale of the FC effective Gram matrix leads to greater accumulation of the generalization gap, despite its more favorable residual alignment.

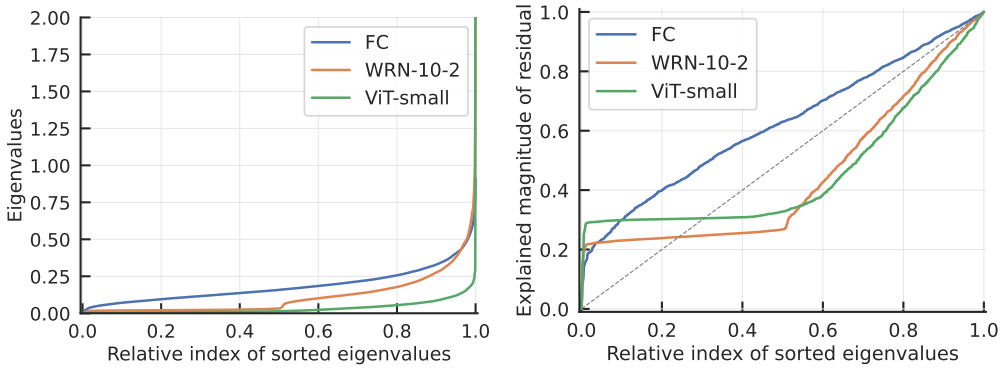

Figure 3.7: Residual $\vec{r}_n(0)$ and effective Gram matrix K_n for CIFAR-2 trained with FC (blue), WRN-10-2 (orange), and ViT-small (green), using $n = 2000$, $m = 10$, and 100 trajectories. All models are trained to the same endpoint with $\|\vec{r}_n\|_2 = 0.583$. The generalization gaps are 0.064, 0.033, and 0.036, respectively. **Left:** Eigenvalues $\sigma(K_n)$; the mean eigenvalues $\bar{\sigma}(K_n)$ are 0.184, 0.114, and 0.034 for FC, WRN-10-2, and ViT-small, respectively. **Right:** Explained magnitude $M(\vec{r}_n(0), E(K_n))$; the initial residuals for WRN-10-2 and ViT-small have larger overlap with the principal subspace of the effective Gram matrix than the residual for FC.

Effective Gram matrix for different number of samples Fig. 3.8 compares the training of datasets with different sizes. As n increases, the initial residual of MNIST-2 projects more strongly onto the tail subspaces of the effective Gram matrix K_n, while the overall spectral scale of K_n generally decreases. Both trends are consistent with the smaller generalization gap observed with more training samples.

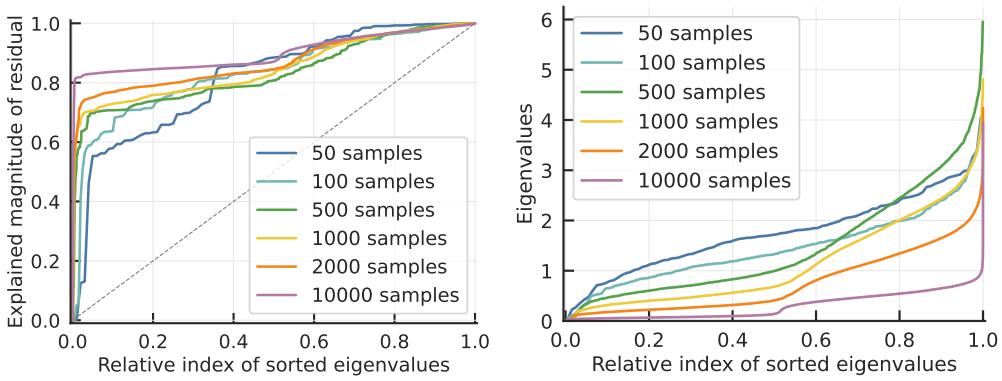

Figure 3.8: Residuals $\vec{r}_n(0)$ and effective Gram matrix K_n for FC trained on MNIST-2 with $n = 50, 100, 500, 1000, 2000$, and 10000. The models are trained to endpoints with the same residual norm $\|\vec{r}_n\|_2 = 0.215$. The generalization gaps are 0.280, 0.270, 0.190, 0.147, 0.083, and 0.023 for n from small to large. **Left:** Explained magnitude of the initial residual $M(\vec{r}_n(0), E(K_n))$ has a similar shape for all n, but the overlap with the principal subspace of the effective Gram matrix is larger for smaller n, which is corroborated by the numerical estimates of the generalization gap in our experiments. **Right:** The mean eigenvalue $\bar{\sigma}(K_n)$ generally decreases as n increases (1.746, 1.445, 1.453, 1.131, 0.743, and 0.295 for n from small to large).

3.3.7. Extension to stochastic gradient descent

Our theoretical framework can be adapted from full-batch gradient descent to stochastic gradient descent (SGD). Suppose that at the t-th iteration for the trajectory w_n trained on the full dataset S_n, SGD samples a mini-batch with index set $B_t \subset [n]$ uniformly without replacement to update the weights. For the perturbed trajectory $w_n^{-(m)}$ trained on the reduced dataset $S_n^{-(m)}$ (where (m) represents a subset of m omitted samples), the effective mini-batch becomes $B_t^{-(m)} = B_t \setminus (m)$. When the omitted batch (m) does not intersect with the sampled mini-batch ($B_t \cap (m) = \emptyset$), the exact same samples are used to update both w_n and $w_n^{-(m)}$ at time t. Consequently, the averaged contraction factor for the batched setting under SGD becomes:

$$\bar{c}_n(t) = \frac{\mathbb{E}_{(m)}\left[\nabla\bar{\ell}(w, S_{(m)}) \cdot \nabla\bar{\ell}(w, S_{B_t^{-(m)}})\big|_{w_n(t)}^{w_n^{-(m)}(t)}\right]}{\bar{\Delta}_n(t)} \tag{3.24}$$

This represents the force that pulls the two trajectories together under the stochastic gradient field $\nabla\bar{\ell}(w, S_{B_t^{-(m)}})$, instead of the full gradient field used in Eq. (3.9). The averaged perturbation factor $\bar{\epsilon}_n(t)$ remains structurally the same as that of full-batch gradient descent (as defined in Eq. (3.10)) for all batch sizes $|B_t| > m$. This is because the perturbation is larger for SGD than for GD whenever $|(m) \cap B_t|/|B_t| > m/n$, and smaller whenever $|(m) \cap B_t|/|B_t| < m/n$, which perfectly balances out in expectation. The evolution of the residual $\vec{r}_n(t)$ changes slightly under SGD because each

optimization step only uses the gradients for the samples in B_t. Thus, the columns of the matrix $P_n(t)$ (from Eq. (3.16)) corresponding to the unsampled indices are zero. Because of these dynamics, the effective Gram matrix for SGD is slightly modified, yielding an appropriately modified quadratic form for the generalization gap.

We can calculate all quantities in this chapter on trajectories obtained from SGD. We trained fully-connected networks on MNIST with 1100 samples using SGD. Our estimated generalization gap in terms of the cross-entropy loss is 0.429, which is very close to the true value of 0.416. Observe that this estimate (with relative error 0.02) is more accurate than the information-theoretic bounds in Table 3.1 such as those by Negrea et al. (2019) and Mou et al. (2018).

3.4. Calculations for analytically tractable models

In this section, we will focus on analytically tractable models, (i) high-dimensional linear regression, (ii) a linear regression problem with a two point-point domain that we began the introduction with, and (iii) a simplified variant of self-attention. We will obtain analytical expressions for the various quantities developed in the prequel. Our goal is to compute the contraction and perturbation factors, and the effective Gram matrix, to obtain a precise expression of the generalization gap.

3.4.1. High-dimensional linear regression

Consider the case with $\mathcal{X} = \mathbb{R}^p$ and $\mathcal{Y} = \mathbb{R}$. In the high-dimensional setting, we will set $p/n = \gamma$ to be the over-parameterization ratio with $p, n \to \infty$. Suppose data is generated from a linear model

$$y = \beta^\top x + \varepsilon, \text{ with } x \sim \mathcal{N}(0, I_p), \text{ and } \varepsilon \sim \mathcal{N}(0, \sigma^2),$$

where the true signal β satisfies $\|\beta\|_2 = \rho$. We denote the contraction and perturbation factor as $\bar{c}_\infty$ and $\bar{\epsilon}_\infty$ in the asymptotic limit.

The Marchenko-Pastur (MP) (Marčenko and Pastur, 1967) spectral density function for a shape parameter θ

$$f_{\mathrm{MP}}(\lambda; \theta) = \frac{1}{2\pi\theta\lambda}\sqrt{(\lambda_+ - \lambda)(\lambda - \lambda_-)} \cdot \mathbb{I}_{[\lambda_-, \lambda_+]}, \quad \theta \in (0, 1],$$

where the spectral bounds are $\lambda_\pm(\theta) = (1 \pm \sqrt{\theta})^2$. For the critical case where $\theta = 1$, the spectral bounds are $\lambda_- = 0$ and $\lambda_+ = 4$ and the density simplifies to

$$f_{\mathrm{MP}}(\lambda; 1) = \frac{1}{2\pi}\sqrt{\frac{4 - \lambda}{\lambda}} \cdot \mathbb{I}_{[0,4]}.$$

Let the k-th order MP moment

$$M_k(\tau, \theta) = \int_{\lambda_-}^{\lambda_+} \lambda^k e^{-\lambda\tau} f_{\mathrm{MP}}(\lambda; \theta) \mathrm{d}\lambda. \tag{3.25}$$

Lemma 3.16. The contraction factor $\bar{c}_\infty(t)$ and the perturbation factor $\bar{\epsilon}_\infty(t)$ exhibit the following decay rates over time t, dictated by the overparameterization ratio γ,

(i) Under-parameterized regime ($\gamma < 1$): Both factors converge to strictly positive constants. Specifically, $\lim_{t\to\infty} \bar{c}_\infty(t) = \frac{2(1-\gamma)^2}{2-\gamma}$ and $\lim_{t\to\infty} \bar{\epsilon}_\infty(t) = \sigma^2\gamma(1-\gamma)$.

(ii) Critical regime ($\gamma = 1$): Both factors exhibit polynomial decay. The contraction factor decays

as $\bar{c}_\infty(t) = \Theta(t^{-1})$, and the perturbation factor decays as $\bar{\epsilon}_\infty(t) = \Theta(t^{-1/2})$.

(iii) Over-parameterized regime ($\gamma > 1$): Both factors exhibit exponential decay up to polynomial prefactors. The contraction factor decays as $\bar{c}_\infty(t) = \Theta(te^{-\lambda_- t})$, and the perturbation factor decays as $\bar{\epsilon}_\infty(t) = \Theta(e^{-2\lambda_- t})$, where λ_- denotes $\lambda_-(1/\gamma)$.

Proof. The proof follows by evaluating the limits $\bar{c}_\infty$ and $\bar{\epsilon}_\infty$ using the moments M_0 and M_1 of the Marchenko-Pastur (MP) distribution.

In this proof, we adopt the following notaion. The leave-one-out Hessian is denoted as $H^{-i} = \sum_{j\neq i} x_j x_j^\top/(n-1)$, the residual is denoted as $r_i = w_n^\top x_i - y_i$, and the parameter shift is denoted as $\delta w_i = w_n^{-i} - w_n$.

The evolution of the parameter shift. The evolution of parameter shift δw_i follows

$$\begin{aligned}\frac{\mathrm{d}\delta w_i}{\mathrm{d}t} &= \frac{\mathrm{d}w_n^{-i}}{\mathrm{d}t} - \frac{\mathrm{d}w_n}{\mathrm{d}t} \\ &= -H^{-i}\delta w_i - \frac{1}{n}\left(\nabla\bar{\ell}(w_n, S_n^{-i}) - x_i r_i\right).\end{aligned}$$

The exact formulation of contraction and perturbation. We then give the exact formula for contraction and perturbation. The numerator of the contraction can be expressed in the following way,

$$\begin{aligned}\mathrm{Num}_n &= \frac{1}{n}\sum_{i=1}^n \nabla\ell(w, z_i)\cdot\nabla\bar{\ell}(w, S_n^{-i})\big|_{w_n(t)}^{w_n^{-i}(t)} \\ &= \frac{1}{n}\sum_{i=1}^n \left(r_i\left(x_i^\top H^{-i}\delta w_i\right) - \left(\frac{n}{n-1}\dot{r}_i + \frac{\|x_i\|^2}{n-1}r_i\right)\left(x_i^\top\delta w_i\right) + \left(x_i^\top\delta w_i\right)\left(x_i^\top H^{-i}\delta w_i\right)\right).\end{aligned}$$

The denominator of the contraction factor can be expressed as

$$\bar{\Delta}_n = \frac{1}{n}\sum_{i=1}^n r_i(x_i^\top\delta w_i) + \frac{1}{2n}\sum_{i=1}^n (x_i^\top\delta w_i)^2.$$

The perturbation factor can be expressed as

$$\bar{\epsilon}_n = \frac{\mathrm{tr}(\hat{\Sigma}_n)}{n-1} = \frac{1}{n-1}\left(\frac{1}{n}\sum_{i=1}^n r_i^2\|x_i\|^2 - \left\|\frac{1}{n}\sum_{j=1}^n r_j x_j\right\|^2\right).$$

The asymptotics of contraction and perturbation for large n. We now analyze the large n limit of the contraction and perturbation factors. We first define several terms that are essential components of the asymptotic formulation. We define $Q(t,s)$ to be the asymptotic residual product,

$$Q(t,s) \triangleq \lim_{n\to\infty}\frac{1}{n}\sum_{i=1}^n r_i(t)r_i(s). \tag{3.26}$$

We also use the following term in the derivation,

$$\partial_t Q(t,s) = \lim_{n\to\infty} \frac{1}{n}\sum_{i=1}^{n} \dot{r}_i(t) r_i(s). \tag{3.27}$$

We define the zeroth-order and first-order spectral moment as

$$D(\tau) \triangleq \lim_{n\to\infty} \frac{1}{n}\mathrm{Tr}(e^{-H\tau}), \qquad N(\tau) \triangleq \lim_{n\to\infty} \frac{1}{n}\mathrm{Tr}(He^{-H\tau}), \tag{3.28}$$

respectively.

By observing that for a matrix M with bounded operator norm w.r.t. n,

$$\frac{x_i^\top M \nabla \bar{\ell}(w_n, S_n^{-i})}{n} = \frac{\sum_{j\neq i} x_i^\top M x_j r_j}{n^2} = O(1/n),$$

while

$$\frac{x_i^\top M x_i r_i}{n} = O(1),$$

we can conclude that the modified evolution

$$\frac{\mathrm{d}\delta w_i}{\mathrm{d}t} = -H^{-i}\delta w_i + \frac{x_i r_i}{n}$$

gives the same contraction factor in the limit of $n \to \infty$. This shows that the gradient average term in the drift of the evolution of parameter shift is negligible.

We should also notice that in the perturbation factor,

$$\frac{1}{n}\sum_{i=1}^{n} r_i^2 \|x_i\|^2 = O(n), \quad \left\| \frac{1}{n}\sum_{j=1}^{n} r_j x_j \right\|^2 = \frac{1}{n} r^\top \left(\frac{1}{n} X X^\top \right) r = O(1),$$

which implies the negligibility of the second term. Hence the modified expression

$$\bar{\epsilon}_n = \frac{\mathrm{tr}(\hat{\Sigma}_n)}{n-1} = \frac{1}{n(n-1)} \left(\sum_{i=1}^{n} r_i^2 \|x_i\|^2 \right).$$

gives the same perturbation factor in the limit $n \to \infty$.

Substituting the previously established definitions from Eqs. (3.26) to (3.28), we can express the limits of the contraction factor's numerator and denominator, along with the perturbation factor,

as follows,

$$\begin{aligned}
\mathrm{Num}_\infty(t) &= \int_0^t N(t-s)Q(t,s)\mathrm{d}s - \int_0^t D(t-s)\partial_t Q(t,s)\mathrm{d}s \\
&\quad - \gamma \int_0^t D(t-s)Q(t,s)\mathrm{d}s + \int_0^t \int_0^t D(t-s_1)N(t-s_2)Q(s_1,s_2)\mathrm{d}s_1\mathrm{d}s_2, \\
\bar{\Delta}_\infty(t) &= \int_0^t D(t-s)Q(t,s)\mathrm{d}s + \frac{1}{2}\int_0^t \int_0^t D(t-s_1)D(t-s_2)Q(s_1,s_2)\mathrm{d}s_1\mathrm{d}s_2, \\
\bar{\epsilon}_\infty(t) &= \gamma Q(t,t).
\end{aligned}$$

The decay trend of the macroscopic limit of contraction and perturbation for large t. To analyze the asymptotic behavior of the contraction and perturbation factors, we derive analytical expressions for asymptotic residual product Q and spectral moments D, N (Eqs. (3.26) to (3.28)) using MP moments M_0 and M_1 (Eq. (3.25)).

By invoking the Marchenko-Pastur theorem (Marčenko and Pastur, 1967), we derive analytical expressions for D, N and Q based on the MP moments M_0 and M_1.

$$D(\tau) = \begin{cases} \gamma M_0(\tau,\gamma) & \text{if } \gamma < 1 \\ (\gamma-1) + M_0(\tau,1/\gamma) & \text{if } \gamma > 1 \\ M_0(\tau,1) & \text{if } \gamma = 1 \end{cases}, \qquad N(\tau) = \begin{cases} \gamma M_1(\tau,\gamma) & \text{if } \gamma < 1 \\ M_1(\tau,1/\gamma) & \text{if } \gamma > 1 \\ M_1(\tau,1) & \text{if } \gamma = 1 \end{cases},$$

$$Q(t,s) = \begin{cases} \sigma^2\left[(1-\gamma) + \gamma M_0(t+s,\gamma)\right] + \rho^2\gamma M_1(t+s,\gamma) & \text{if } \gamma < 1 \\ \sigma^2 M_0(t+s,1/\gamma) + \rho^2 M_1(t+s,1/\gamma) & \text{if } \gamma > 1 \\ \sigma^2 M_0(t+s,1) + \rho^2 M_1(t+s,1) & \text{if } \gamma = 1 \end{cases}.$$

We now proceed to analyze the asymptotic behavior of the contraction and perturbation factors as $t \to \infty$. We have the following results on the integrals of MP moments for $0 < \theta < 1$.

$$\int_0^\infty M_k(\tau,\theta)d\tau = \int_{\lambda_-}^{\lambda_+} \lambda^k \left(\int_0^\infty e^{-\lambda\tau}\mathrm{d}\tau\right) f_{\mathrm{MP}}(\lambda;\theta)d\lambda = \int_{\lambda_-}^{\lambda_+} \lambda^{k-1} f_{\mathrm{MP}}(\lambda;\theta)d\lambda$$

which implies

$$\int_0^\infty M_1(\tau,\theta)d\tau = 1, \quad \int_0^\infty M_0(\tau,\theta)d\tau = \frac{1}{1-\theta}.$$

We also have for both $k = 0, 1$, these exists some constant M such that,

$$|M_k(\tau,\theta)| \le Me^{-\lambda_-\tau}.$$

Now we are ready to estimate the decay pattern of the contraction and perturbation factors in different regimes.

1. **Under-parameterized regime, $\gamma < 1$.** The residual product can be decomposed as

$$Q(t,s) = Q_\infty + R(t,s)$$

where $Q_\infty = \sigma^2(1-\gamma)$, and $|R(t,s)| \le Me^{-\lambda_-(t+s)}$ for some constant $M > 0$. Furthermore, because the residual reaches a steady state, its derivative vanishes, i.e. $\lim_{t,s\to\infty} \partial_t Q(t,s) = 0$. We will now show that the integral in the contraction involving $R(t,s)$ vanishes. For a single integral convolved with an operator $\mathcal{O}$ bounded by $Ce^{-\lambda_-\tau}$,

$$\begin{aligned}\left|\int_0^t \mathcal{O}(t-s)R(t,s)ds\right| &\le \int_0^t \left(Ce^{-\lambda_-(t-s)}\right)\left(Me^{-\lambda_-(t+s)}\right)ds \\ &= \int_0^t CMe^{-2\lambda_- t}ds = CMte^{-2\lambda_- t}.\end{aligned}$$

For a double integral,

$$\begin{aligned}&\left|\iint \mathcal{O}(t-s_1)\mathcal{O}(t-s_2)R(s_1,s_2)ds_1ds_2\right| \\ &\le \iint \left(Ce^{-\lambda_-(t-s_1)}\right)\left(Ce^{-\lambda_-(t-s_2)}\right)\left(Me^{-\lambda_-(s_1+s_2)}\right)ds_1ds_2 \le C^2Mt^2e^{-2\lambda_- t}.\end{aligned}$$

Hence,

$$\begin{aligned}\lim_{t\to\infty}\mathrm{Num}_\infty(t) &= Q_\infty\int_0^\infty \mathcal{N}(\tau)d\tau - \gamma Q_\infty\int_0^\infty \mathcal{D}(\tau)d\tau + Q_\infty\left(\int_0^\infty \mathcal{D}(\tau)d\tau\right)\left(\int_0^\infty \mathcal{N}(\tau)d\tau\right) \\ &= Q_\infty\gamma, \\ \lim_{t\to\infty}\bar{\Delta}_\infty(t) &= Q_\infty\int_0^\infty \mathcal{D}(\tau)d\tau + \frac{1}{2}Q_\infty\left(\int_0^\infty \mathcal{D}(\tau)d\tau\right)^2 = Q_\infty\frac{2\gamma-\gamma^2}{2(1-\gamma)^2}.\end{aligned}$$

As a result, we have

$$\lim_{t\to\infty}\bar{c}_\infty(t) = \frac{2(1-\gamma)^2}{2-\gamma}$$

Similarly, for the perturbation factor, we derive the following limit,

$$\lim_{t\to\infty}\bar{\epsilon}_\infty(t) = \lim_{t\to\infty}\gamma Q(t,t) = \sigma^2\gamma(1-\gamma).$$

2. **Over-parameterized regime, $\gamma > 1$.** We first analyze the numerator. We should note that $D(t) \to D_\infty$, which is lower bounded, $N(t)$ follows exponential decay with respect to t, and $Q(t,s)$ follows exponential decay on $t+s$. Here is the asymptotic order of all the four terms in $\mathrm{Num}_\infty(t)$.

$$\int_0^t N(t-s)Q(t,s)ds \le C\int_0^t e^{-\lambda_-(t-s)}e^{-\lambda_-(t+s)}ds = C\int_0^t e^{-2\lambda_- t}ds = \Theta(te^{-2\lambda_- t})$$

$$\int_0^t D(t-s)|\partial_t Q(t,s)|ds \le C\int_0^t e^{-\lambda_-(t+s)}ds = \Theta(e^{-\lambda_- t})$$

$$\gamma\int_0^t D(t-s)Q(t,s)ds \le C\int_0^t e^{-\lambda_-(t+s)}ds = \Theta(e^{-\lambda_- t})$$

$$\iint D(t-s_1)N(t-s_2)Q(s_1,s_2)ds_1ds_2 \le C\iint e^{-\lambda_-(t-s_2)}e^{-\lambda_-(s_1+s_2)}ds_1ds_2$$
$$= Cte^{-\lambda_- t}\left(\frac{1-e^{-\lambda_- t}}{\lambda_-}\right) = \Theta(te^{-\lambda_- t})$$

Hence, we have

$$\mathrm{Num}_\infty(t) = \Theta(te^{-\lambda_- t}).$$

For the denominator $\bar{\Delta}_n(t)$, the first term vanishes, and the second term converges to

$$\frac{D_\infty^2}{2}\int_0^t\int_0^t Q(s_1,s_2)ds_1ds_2,$$

which is constant. Hence, we have

$$\bar{c}_\infty = \Theta(te^{-\lambda_- t}).$$

For the perturbation factor, by the exponential decay of M_0 and M_1, we have

$$\bar{\epsilon}_\infty(t) = \gamma Q(t,t) = \Theta(e^{-2\lambda_- t}).$$

3. **Critical regime, $\gamma = 1$.** For the Marchenko-Pastur distribution $f_{\mathrm{MP}}(\lambda,\theta)$, when $\theta = 1$, we have,

$$f_{\mathrm{MP}}(\lambda,1) = \frac{1}{2\pi}\sqrt{\frac{4-\lambda}{\lambda}} \sim \frac{1}{\pi\sqrt{\lambda}} \quad \text{for } \lambda \text{ close to } 0.$$

By applying Laplace's method, the asymptotic behavior of the spectral moments D, N is dominated by the density near the origin,

$$D(\tau) = \int_0^4 e^{-\lambda\tau}\mathrm{d}f_{\mathrm{MP}}(\lambda,1) \sim \int_0^\epsilon \frac{e^{-\lambda\tau}}{\pi\sqrt{\lambda}}\mathrm{d}\lambda = \Theta(\tau^{-1/2}),$$
$$N(\tau) = \int_0^4 \lambda e^{-\lambda\tau}\mathrm{d}f_{\mathrm{MP}}(\lambda,1) \sim \int_0^\epsilon \frac{\lambda e^{-\lambda\tau}}{\pi\sqrt{\lambda}}\mathrm{d}\lambda = \Theta(\tau^{-3/2}),$$

where the last equality is derived from a change of variable $u = \lambda\tau$. Similarly, we have $Q(\tau) = \Theta(\tau^{-1/2})$.
We will first analyze the decay of the denominator. The denominator $\bar{\Delta}_\infty(t)$ is dominated by the double integral term. Let $s_1 = tu_1$ and $s_2 = tu_2$ to scale the integration variables out of

the time domain, we have

$$\begin{aligned}
&\iint_0^t D(t-s_1)D(t-s_2)Q(s_1,s_2)\mathrm{d}s_1\mathrm{d}s_2 \\
\sim &\iint_0^t (t-s_1)^{-1/2}(t-s_2)^{-1/2}(s_1+s_2)^{-1/2}\mathrm{d}s_1\mathrm{d}s_2 \\
=&t^2\iint_0^1 (t-tu_1)^{-1/2}(t-tu_2)^{-1/2}(tu_1+tu_2)^{-1/2}\mathrm{d}u_1\mathrm{d}u_2 \\
=&t^2\cdot t^{-3/2}\iint_0^1 (1-u_1)^{-1/2}(1-u_2)^{-1/2}(u_1+u_2)^{-1/2}\mathrm{d}u_1\mathrm{d}u_2
\end{aligned}$$

By the change of variable $s_1 = tu_1$ and $s_2 = tu_2$, we get the first equality by scaling the integration variables out of the time domain. The second inequality can be achieved by factoring t out of the integrand. Because the remaining integral over u_1, u_2 converges to a strictly positive constant, the denominator diverges as,

$$\bar{\Delta}_\infty(t) = \Theta(t^{1/2})$$

Now we analyze the decay of the numerator. We apply the identical scaling procedure to the dominant double integral of the numerator $\mathrm{Num}_\infty(t)$. Crucially, the first order spectral moment $\mathcal{N}(\tau)$ decays much faster ($\Theta(\tau^{-3/2})$) due to the suppression of the zero-eigenvalue singularity:

$$\begin{aligned}
&\iint_0^t D(t-s_1)N(t-s_2)Q(s_1,s_2)\mathrm{d}s_1\mathrm{d}s_2 \\
\sim &\iint_0^t (t-s_1)^{-1/2}(t-s_2)^{-3/2}(s_1+s_2)^{-1/2}\mathrm{d}s_1\mathrm{d}s_2 \\
=&t^2\iint_0^1 (t-tu_1)^{-1/2}(t-tu_2)^{-3/2}(tu_1+tu_2)^{-1/2}\mathrm{d}u_1\mathrm{d}u_2 \\
=&t^2\cdot t^{-5/2}\iint_0^1 (1-u_1)^{-1/2}(1-u_2)^{-3/2}(u_1+u_2)^{-1/2}\mathrm{d}u_1\mathrm{d}u_2
\end{aligned}$$

Hence, we have

$$\mathrm{Num}_\infty(t) = \Theta(t^{-1/2}).$$

The calculations above yield the asymptotic decay order of the contraction factor

$$\bar{c}_\infty(t) = \frac{\Theta(t^{-1/2})}{\Theta(t^{1/2})} = \Theta(t^{-1}).$$

For the perturbation factor, we can similarly show that

$$Q(t,t) = C_1 t^{-1/2} + C_2 t^{-3/2} = \Theta(t^{-1/2}).$$

□

MP moments decay exponentially for $0 < \theta < 1$ but polynomially when $\theta = 1$. This spectral

structure induces different decay rates for the contraction and perturbation factors in each regime. This demonstrates that the contraction and the perturbation factors have the same decay trend (constant, polynomial, or exponential) across the different parameterization regimes. The trend also reflects in the training of neural networks in Fig. 3.1.

Remark 3.17. We can integrate Eq. (3.8) in the underparameterized regime to notice that, since $\bar{c}_\infty$ and $\bar{\epsilon}_\infty$ converge to strictly positive constants, the asymptotic averaged loss difference

$$\lim_{t\to\infty} \bar{\Delta}_\infty(t) = \frac{\sigma^2\gamma(2-\gamma)}{2(1-\gamma)}$$

In over-parameterized regime, since $\bar{c}_\infty$ and $\bar{\epsilon}_\infty$ both decay exponentially the asymptotic averaged loss difference remains bounded. In contrast, in the critical regime, the averaged loss difference diverges. These limits derived from our theory coincide perfectly with the generalization gap for high-dimensional linear regression and double descent results in the literature (Hastie et al., 2022; Belkin et al., 2019; Dobriban and Wager, 2018). Let us note that existing analyses of double descent focus on population risk, whereas our analysis concerns the generalization gap.

To corroborate the numerical example in Fig. 3.2, we next show that the parameter shift between the original and leave-one-out trajectories progressively aligns with the smaller eigenvalues of the Hessian during training in the high-dimensional setting.

Lemma 3.18. The Rayley quotient

$$\mathrm{R}(H, \delta w) = \frac{\delta w_i^\top H^{-i} \delta w_i}{\|\delta w_i\|^2},$$

exhibits the following decay rates over time t,

(i) Under-parameterized regime ($\gamma < 1$): The term $\mathrm{R}(H, \delta w)$ decays to strictly positive constant $1-\gamma$.
(ii) Critical regime ($\gamma = 1$): The term $\mathrm{R}(H, \delta w)$ decays polynomially at a rate of $\Theta(t^{-1})$.
(iii) Over-parameterized regime ($\gamma > 1$): The term $\mathrm{R}(H, \delta w)$ decays exponentially at a rate of $\Theta(te^{-\lambda_- t})$, where λ_- denotes $\lambda_-(1/\gamma)$.

Lemma 3.18 can be proved in a similar way as Lemma 3.16. The decay of the Rayleigh quotient $\mathrm{R}(H, \delta w)$ can be intuitively understood using the evolution of δw_i,

$$\frac{\mathrm{d}\delta w_i}{\mathrm{d}t} = -H^{-i}\delta w_i + \frac{x_i r_i}{n}$$

as derived in the proof of Lemma 3.16. This differential equation characterizes a dynamical system driven by an isotropic source term $\frac{x_i r_i}{n}$ and a selective damping force $-H^{-i}\delta w_i$. Components of the parameter shift δw_i that align with the large eigenvalues of the Hessian are rapidly dissipated, while components that align with the small eigenvalues accumulate. As training proceeds, the parameter shift projects more and more into the subspace of Hessian with small eigenvalues, which causes the Rayley quotient to decrease. This trend also reflects in the training of neural networks (Fig. 3.2). In comparison, the Rayley quotient of Hessian and gradient, defined as $\mathrm{R}(H, g_i) \triangleq \frac{g_i^\top H^{-i} g_i}{\|g_i\|^2} = \frac{x_i^\top H^{-i} x_i}{\|x_i\|^2}$ is unchanged during training, which is also reflected in Fig. 3.2.

3.4.2. An analytical effective Gram matrix for linear regression

We next consider a very simple example that we discussed in the Introduction. Suppose the sample space is supported on two points with orthonormal inputs, i.e., $\mathcal{Z} = \{(x_1, y_1), (x_2, y_2)\}$, with $x_1^\top x_2 = 0$, each with unit norm $\|x_1\|_2 = \|x_2\|_2 = 1$. Consider the dataset S_n with n even, where $z_i = (x_1, y_1)$ when $i \le n/2$ and $z_i = (x_2, y_2)$ when $i > n/2$. Even if this case is extremely simple, it is illuminating because we will be able to calculate the effective Gram matrix exactly and show that for "easy tasks", i.e., ones where the generalization gap at the end of training is small, the initial residual predominantly projects in the subspace of the effective Gram matrix with small eigenvalues.

Lemma 3.19. The contraction and perturbation factors for the above example are

$$\bar{c}_n(t) = \bar{c} \triangleq \frac{n-2}{2(n-1)}, \quad \bar{\epsilon}_n(t) = \frac{y_1^2 + y_2^2}{4(n-1)} e^{-t}.$$

The effective Gram matrix $K_n(0,t) = U\Lambda^K(t)U^\top$ where $\Lambda^K(t) = [\lambda_1^K(t), \dots, \lambda_n^K(t)]$, and

$$\lambda_1^K(t) = \lambda_2^K(t) = \Theta(e^{-\bar{c}t}), \quad \lambda_3^K(t) = \cdots = \lambda_n^K(t) = \Theta(1).$$

The initial residual $\vec{r}_n(0)$ lies in the subspace of $K_n = \lim_{t\to\infty} K_n(t)$ with zero eigenvalue, which guarantees that $\bar{\Delta}_n = 0$ as $t \to \infty$.

Proof. The gradients for the averaged loss $\bar{\ell}(w, S_n)$ and $\bar{\ell}(w, S_n^{-i})$ are

$$\begin{aligned}
\nabla\bar{\ell}(w, S_n) &= \frac{1}{2}\left(w^\top x_1 - y_1\right) x_1 + \frac{1}{2}\left(w^\top x_2 - y_2\right) x_2 \\
\nabla\bar{\ell}(w, S_n^{-1}) &= \frac{n-2}{2(n-1)}\left(w^\top x_1 - y_1\right) x_1 + \frac{n}{2(n-1)}\left(w^\top x_2 - y_2\right) x_2 \\
\nabla\bar{\ell}(w, S_n^{-2}) &= \frac{n}{2(n-1)}\left(w^\top x_1 - y_1\right) x_1 + \frac{n-2}{2(n-1)}\left(w^\top x_2 - y_2\right) x_2.
\end{aligned}$$

The averaged contraction factor can be calculated from Eq. (3.9),

$$\begin{aligned}
\bar{c}_n(t) &= \frac{\frac{1}{n}\sum_{i=1}^n \nabla\ell(w, z_i) \cdot \nabla\bar{\ell}(w, S_n^{-i})\big|_{w_n(t)}^{w_n^{-i}(t)}}{\bar{\Delta}_n(t)} \\
&= \frac{\frac{1}{2}\left(\nabla\ell(w, z_1) \cdot \nabla\bar{\ell}(w, S_n^{-1})\big|_{w_n(t)}^{w_n^{-1}(t)} + \nabla\ell(w, z_2) \cdot \nabla\bar{\ell}(w, S_n^{-2})\big|_{w_n(t)}^{w_n^{-2}(t)}\right)}{\frac{1}{2}\left(\frac{1}{2}(w^\top x_1 - y_1)^2\big|_{w_n(t)}^{w_n^{-1}(t)} + \frac{1}{2}(w^\top x_2 - y_2)^2\big|_{w_n(t)}^{w_n^{-2}(t)}\right)} \\
&= \frac{\frac{1}{2}\left(\frac{n-2}{2(n-1)}(w^\top x_1 - y_1)^2\big|_{w_n(t)}^{w_n^{-1}(t)} + \frac{n-2}{2(n-1)}(w^\top x_2 - y_2)^2\big|_{w_n(t)}^{w_n^{-2}(t)}\right)}{\frac{1}{2}\left(\frac{1}{2}(w^\top x_1 - y_1)^2\big|_{w_n(t)}^{w_n^{-1}(t)} + \frac{1}{2}(w^\top x_2 - y_2)^2\big|_{w_n(t)}^{w_n^{-2}(t)}\right)} = \frac{n-2}{2(n-1)} =: \bar{c}.
\end{aligned}$$

From Eq. (3.16) and Eq. (3.19) we can derive that

$$F_n(t) = I_n - \frac{P_n(t)}{n}, \qquad P_n(t) = \operatorname{diag}\left(\vec{1}\vec{1}^\top, \vec{1}\vec{1}^\top\right),$$

with $\vec{1} = [1, \dots, 1] \in \mathbb{R}^{n/2}$. Note that $P_n(t)$ is not full rank when $n > 2$. By Lemma 3.14, the convergence of $\lim_{t\to\infty} K_n(0,t)$ is largely controlled by the smallest eigenvalue of $P_n(t)$, which cannot be too small. Hence, to ensure convergence, we define a modified version of $P_n(t)$ with small perturbation $\varepsilon(t)$ on its singular subspace, i.e., $P_n^\varepsilon(t) = U\Lambda^\varepsilon U^\top$, with $\Lambda^\varepsilon = \operatorname{diag}(n/2, n/2, n\varepsilon(t)/2, \dots, n\varepsilon(t)/2)$, $U = [u_1, \dots u_n]$, where

$$u_1 = \sqrt{\frac{2}{n}}\,[1, \dots, 1, 0, \dots, 0], \quad u_2 = \sqrt{\frac{2}{n}}\,[0, \dots, 0, 1, \dots, 1].$$

In this case, when $\varepsilon(t) \equiv 0$, we have $P_n^\varepsilon(t) \equiv P_n(t)$. The dynamics $\mathrm{d}\vec{r}_n(t)/\mathrm{d}t = -P_n^\varepsilon(t)\vec{r}_n(t)/n$ give the same trajectory of $\vec{r}_n(t)$ as Eq. (3.16), since $\vec{r}_n(0) = n^{-1/2}[y_1, \dots, y_1, y_2, \dots, y_2] \in \operatorname{span}(u_1, u_2)$. The corresponding perturbed covariance matrix is $F_n^\varepsilon(t) = I_n - P_n^\varepsilon(t)/n$.

We set $\varepsilon(t) = \bar{\varepsilon}\left(1_{[0,1]}(t) + 1_{[1,\infty]}(t)/t^2\right)$ with $\bar{\varepsilon} \ll 1$. The propagator $\Omega_n^\varepsilon(t)$ of the evolution $\mathrm{d}\vec{r}_n(t)/\mathrm{d}t = -P_n^\varepsilon(t)\vec{r}_n(t)/n$ is

$$\Omega_n^\varepsilon(t) = \exp\left(-\frac{\int_0^t P_n^\varepsilon(s)\mathrm{d}s}{n}\right) = U\exp\left(-\frac{\int_0^t \Lambda^\varepsilon(s)\mathrm{d}s}{n}\right)U^\top.$$

Hence, the effective Gram matrix $K_n(0,t)$ can be calculated as

$$\begin{aligned} K_n(0,t) &= \frac{\int_0^t U\exp\left(-\frac{2\int_0^s \Lambda^\varepsilon(u)\mathrm{d}u}{n}\right)\left(I - \frac{\Lambda^\varepsilon(s)}{n}\right)U^\top \exp(-(t-s)\bar{c})\mathrm{d}s}{n-1} \\ &= U\Lambda^K(t)U^\top \end{aligned}$$

where $\Lambda^K(t) = [\lambda_1^K(t), \dots, \lambda_n^K(t)]$, and we have

$$\begin{aligned} \lambda_1^K(t) = \lambda_2^K(t) = \lambda(t) &\triangleq \frac{(1-\bar{c})^{-1}/2}{n-1}\left(e^{-\bar{c}t} - e^{-t}\right), \\ \lambda_3^K(t) = \cdots = \lambda_n^K(t) = \lambda'(t) &\triangleq \frac{\int_0^t \exp\left(-\int_0^s \varepsilon(u)\mathrm{d}u\right)\cdot\exp\left(-(t-s)\bar{c}\right)\cdot\left(1 - \frac{\varepsilon(s)}{2}\right)\mathrm{d}s}{n-1} \end{aligned}$$

For $\varepsilon(t) = \bar{\varepsilon}\left(1_{[0,1]}(t) + 1_{[1,\infty]}(t)/t^2\right)$, $\exp\left(-\int_0^s \varepsilon(u)\mathrm{d}u\right) \in [\exp(-2\bar{\varepsilon}), 1]$, $1 - \varepsilon(s)/2 \in [1 - \bar{\varepsilon}/2, 1]$, hence, for $\bar{\varepsilon}$ small enough, $\frac{1-\exp(-\bar{c}t)}{2\bar{c}(n-1)} \le \lambda'(t) \le \frac{1-\exp(-\bar{c}t)}{\bar{c}(n-1)}$. Hence, we have $\lambda(t) = \Theta(\exp(-\bar{c}t))$, $\lambda'(t) = \Theta(1)$. The generalization gap approaches 0 since $\lambda(t) \to 0$ as $t \to \infty$. □

Explicit weight trajectories. Here we also calculate the solution of $w_n(t)$ and $w_n^{-i}(t)$, although not required in the derivation of the effective gram matrix.

$$\begin{aligned} w_n(t) &= \int_0^t \exp\left(-(t-s)\left(\frac{1}{2}x_1x_1^\top + \frac{1}{2}x_2x_2^\top\right)\right)\left(\frac{1}{2}y_1x_1 + \frac{1}{2}y_2x_2\right)ds \\ &= (1 + \exp(-t/2))\cdot(y_1x_1) + (1 + \exp(-t/2))\cdot(y_2x_2) \end{aligned}$$

Similarly, we have

$$w_n^{-i}(t) = \left(1 - \exp\left(-\frac{n-2}{2(n-1)}t\right)\right) \cdot (y_k x_k) + \left(1 - \exp\left(-\frac{n}{2(n-1)}t\right)\right) \cdot (y_l x_l)$$

where $k = \lceil 2i/n \rceil$, $l \in \{1,2\}/\{k\}$. We can see that the solution $w_n(t)$ lies in $\text{span}(x_1, x_2)$. When trained on the dataset S_n^{-i}, the progress in the direction $x_{\lceil 2i/n \rceil}$ is slightly less than the other direction, which introduces the non-zero averaged loss difference $\bar{\Delta}_n(t)$ during training. We should also note that the calculation of the contraction and perturbation factors depends heavily on the sample-wise loss gradient $\nabla \ell(w, z_i)$ being supported on $\{x_1, x_2\}$. The clustering of per-sample gradients also happens in the training of neural networks, as shown in Fort and Ganguli (2019a).

Note that since $\mathcal{Z}$ is supported on only two points, it is intuitive that gradient flow trained on a dataset containing both types of samples generalizes and achieves zero loss for any distribution D supported on $\mathcal{Z}$. This coincides with the calculation above. If we assume a uniform distribution on $\mathcal{Z}$ instead, we have

$$\mathbb{E}[\bar{\Delta}_n(t)] = \mathbb{E}[y^\top K_n y] = \Theta(2^{-n}),$$

because S_n is supported on only one of the samples only with probability $2^{-(n-1)}$. This example therefore provides a tight prediction of the generalization gap.

Remark 3.20 (Comparison with weight-norm based generalization bound). Arora et al. (2019) suggest that one can bound the generalization gap in terms of the norm of the eventual weights. The Gram matrix of the linear regression described above is calculated in the proof of Lemma 3.19 to be $H^\varepsilon = P_n^\varepsilon(t)$ with $\varepsilon(t) = \bar{\varepsilon}$ for some constant $\bar{\varepsilon} \ll 1$. For technical reasons, this is a perturbed version of the differential equation in Eq. (3.16) and it guarantees positive definiteness of the drift operator without affecting the residual dynamics. By the calculations of Arora et al. (2019), the norm of weights can be bounded by $\sqrt{y^\top (H^\varepsilon)^{-1} y}$, this gives a generalization bound of

$$\sqrt{\frac{2y^\top (H^\varepsilon)^{-1} y}{n}} = \sqrt{\frac{2(y_1^2 + y_2^2)}{n}}.$$

for 1-Lipschitz loss. As we can see, this bound is far looser than the actual generalization error, which is $\Theta(2^{-n})$ for 1-Lipschitz loss from our calculation above. The key point to emphasize here is that by characterizing the evolution of the point-wise loss difference using the contraction factor, we can work directly in the prediction space instead of working in the weight space. This is the reason why our estimate of the generalization gap is more accurate.

Remark 3.21 (Comparison with information-theoretic generalization bound). Neu et al. (2021), Negrea et al. (2019), and Banerjee et al. (2022) derive mutual information-based generalization bounds for stochastic algorithms that depend on the sum of trace of the gradient covariance along the training trajectory $\sum_t \text{tr}\left(\hat{\Sigma}_n(t)\right)$. We can adapt their expression to deterministic algorithms like gradient descent using our theory. If we assume that that contraction factor $\bar{c}_n(t)$ is non-negative, we can bound $\bar{\Delta}_n(\infty)$ by $\int_0^\infty \bar{\epsilon}_n(t) \mathrm{d}t$. Notice that from Eq. (3.10), this would correspond to an estimate of the sum of the trace of the gradient covariances. If the contraction factor were identically zero, the expression for the effective Gram matrix in Theorem 3.12 gives a generalization estimate of $\frac{(y_1^2+y_2^2)}{4(n-1)}$. This is also a loose estimate. Evidently, the bound obtained using the sum of the trace of the gradient covariances is loose because these existing techniques do not

incorporate the contraction factor.

3.4.3. A single layer simplified self-attention network

We next study a nonlinear classifier with simplified self-attention operator. We show that trainable attention yields faster asymptotic decay of the averaged loss difference and perturbation than fixed attention and, under an additional positivity condition, a larger asymptotic contraction factor. This illustrates how trainable attention can improve generalization dynamics through the lens of our contraction and perturbation factors.

The data is generated as follows. Let L be the number of tokens contained in one datum. For an integer $1 \le M < L$, write

$$\pi_{\mathrm{sig}} = \frac{M}{L} \in (0,1) \qquad \text{and} \qquad \pi_{\mathrm{bg}} = 1 - \pi_{\mathrm{sig}}.$$

Let $\mathcal{C} := \{1,2\}$ denote the set of classes. Let $e_1, e_2, e_3 \in \mathbb{R}^{d_{\mathrm{emb}}}$ be orthonormal vectors and $d_{\mathrm{emb}} \ge 3$. An input x_j of class $j \in \mathcal{C}$ contains M copies of the signal token e_j and $L - M$ copies of the background token e_3, in an arbitrary order. Its label is $y_j = (-1)^{j+1}$. Assume that the training set contains $n/2$ inputs from each class with $n \ge 4$ even. For any input $x \in \mathbb{R}^{L \times d_{\mathrm{emb}}}$, let the predictor be

$$\begin{aligned} f(w,x) &= v^\top h_\vartheta(x) \\ \text{with } h_\vartheta(x) &= \frac{1}{L}\mathbf{1}^\top \operatorname{softmax}(xB_\vartheta x^\top)\, x, \\ B_\vartheta &= \kappa\left(\vartheta_1 e_1 e_1^\top + \vartheta_2 e_2 e_2^\top + \vartheta_b e_3 e_3^\top\right) \end{aligned}$$

for a constant $\kappa \ge 0$. The weights (trainable parameters) in this predictor are $w = (v, \vartheta_1, \vartheta_2, \vartheta_b)$. We are therefore considering a predictor that pools the outputs of self-attention (so-called "mean-pooled" operation). The quantity B_ϑ directly parameterizes the query-key map, since there are only three non-zero token-token inner-products regardless of the value of L and M. The value map is set to identity. The case $\kappa = 0$ indicates uniform attention (and a linear predictor), whereas $\kappa > 0$ gives trainable attention scores.

We use the squared loss $\ell(w,(x,y)) = (f(w,x)-y)^2/2$ as in the previous two examples, and initialize all training trajectories at $w = 0$. We are interested in three quantities: $\bar{\Delta}_{n,\kappa}$, $\bar{c}_{n,\kappa}$, and $\bar{\epsilon}_{n,\kappa}$, which are the averaged loss, contraction and perturbation respectively. We denote $\nu_\Delta(\kappa)$, $\nu_\epsilon(\kappa)$ as the decay rate of $\bar{\Delta}_{n,\kappa}$ and $\bar{\epsilon}_{n,\kappa}$. We denote $c(\kappa)$ as the limit infimum of $\bar{c}_{n,\kappa}(t)$.

$$\begin{aligned} \nu_\Delta(\kappa) &= \liminf_{\substack{n\to\infty \\ n \text{ even}}} \liminf_{t\to\infty} \left[-\frac{1}{t}\log\left|\bar{\Delta}_{n,\kappa}(t)\right|\right], \\ c(\kappa) &= \liminf_{\substack{n\to\infty \\ n \text{ even}}} \liminf_{t\to\infty} \bar{c}_{n,\kappa}(t), \\ \nu_\epsilon(\kappa) &= -\lim_{t\to\infty} \frac{1}{t}\log \bar{\epsilon}_{n,\kappa}(t). \end{aligned} \tag{3.29}$$

Lemma 3.22. Fix $\kappa > 0$. The decay rate of $\bar{\Delta}_{n,\kappa}$ and $\bar{\epsilon}_{n,\kappa}$ satisfy

$$\nu_\Delta(\kappa) > \nu_\Delta(0) = \pi_{\mathrm{sig}}^2, \qquad \nu_\epsilon(\kappa) > \nu_\epsilon(0) = \pi_{\mathrm{sig}}^2. \tag{3.30}$$

Suppose, in addition, that the averaged loss difference $\bar{\Delta}_{n,\kappa}(t)$ is eventually positive for all sufficiently large sample sizes, then

$$c(\kappa) > c(0) = \pi_{\mathrm{sig}}^2. \tag{3.31}$$

Proof. In this proof, we first use class symmetry to reduce the full-data flow to three scalar variables and show that trainable attention makes both endpoint kernel modes strictly larger than π_{sig}^2, which directly improves the perturbation rate. We then view removing one sample as an $O(n^{-1})$ perturbation of the class weights; uniform endpoint stability transfers the full-data spectrum to the leave-one-out (LOO) residual dynamics and yields the loss-gap bound. On an eventually positive gap, the exact contraction identity can be related to residual quantities, and the improved residual decay rate together with the LOO modal asymptotics yields stronger contraction.

Balanced trajectory and endpoint. We first reduce the balanced trajectory trained on the full dataset to three scalar coordinates and analyze its limit as $t \to \infty$. We now introduce the transformed residual,

$$\rho_i = y_i f(w, x_i) - 1, \qquad \ell_i = \frac{1}{2}\rho_i^2.$$

Permutation invariance, class balance, and the zero initialization imply $\vartheta_1 = \vartheta_2$ and $v \in \mathrm{span}\{e_1 - e_2\}$. Define the balanced coordinates

$$\zeta_{\mathrm{sig}} = \kappa\vartheta_1 = \kappa\vartheta_2, \quad \zeta_{\mathrm{bg}} = -\kappa\vartheta_b, \quad u = \frac{1}{2}\langle v, e_1 - e_2\rangle.$$

Then $v = u(e_1 - e_2)$ and $u(0) = 0$. The total masses for a signal query and a background query are

$$a = \frac{\pi_{\mathrm{sig}} e^{\zeta_{\mathrm{sig}}}}{\pi_{\mathrm{sig}} e^{\zeta_{\mathrm{sig}}} + \pi_{\mathrm{bg}}}, \quad b = \frac{\pi_{\mathrm{sig}} e^{\zeta_{\mathrm{bg}}}}{\pi_{\mathrm{sig}} e^{\zeta_{\mathrm{bg}}} + \pi_{\mathrm{bg}}}, \quad m = \pi_{\mathrm{sig}} a + \pi_{\mathrm{bg}} b. \tag{3.32}$$

Hence the pooled class-j feature is $h_j = me_j + (1-m)e_3$, and both classes have the common transformed residual

$$\rho = um - 1.$$

With $A = a(1-a)$ and $B = b(1-b)$, direct differentiation gives

$$\dot{u} = -\frac{1}{2}m\rho, \qquad \dot{\zeta}_{\mathrm{sig}} = -\frac{\kappa^2\pi_{\mathrm{sig}}}{2}Au\rho, \qquad \dot{\zeta}_{\mathrm{bg}} = -\kappa^2\pi_{\mathrm{bg}}Bu\rho. \tag{3.33}$$

For $j \in \mathcal{C}$, let x_j denote any class-j input and let $k_j = \nabla_w(y_j f(w, x_j))$. The balanced predictor Gram matrix has the form

$$G(t) = \begin{pmatrix} k_1^\top k_1 & k_1^\top k_2 \\ k_2^\top k_1 & k_2^\top k_2 \end{pmatrix} = \begin{pmatrix} d & \chi \\ \chi & d \end{pmatrix},$$

where

$$\begin{aligned} d &= m^2 + (1-m)^2 + \kappa^2 u^2(\pi_{\mathrm{sig}}^2 A^2 + \pi_{\mathrm{bg}}^2 B^2), \\ \chi &= -(1-m)^2 + \kappa^2 u^2 \pi_{\mathrm{bg}}^2 B^2. \end{aligned}$$

Its common and difference eigenvalues are therefore

$$q_{\mathrm{com}} = d + \chi = m^2 + \kappa^2 u^2(\pi_{\mathrm{sig}}^2 A^2 + 2\pi_{\mathrm{bg}}^2 B^2), \tag{3.34}$$

$$q_{\mathrm{diff}} = d - \chi = m^2 + 2(1-m)^2 + \kappa^2 u^2 \pi_{\mathrm{sig}}^2 A^2. \tag{3.35}$$

The residual vector $\boldsymbol{\rho} = (\rho_1, \rho_2)$ obeys $\dot{\boldsymbol{\rho}} = -G\boldsymbol{\rho}/2$. Since $\rho_1 = \rho_2$, the balanced trajectory lies in the common direction,

$$\dot{\rho} = -\frac{1}{2} q_{\mathrm{com}} \rho, \qquad \rho(0) = -1. \tag{3.36}$$

Equation (3.36) keeps $\rho < 0$. It follows from (3.33) that $u, \zeta_{\mathrm{sig}}, \zeta_{\mathrm{bg}} \geq 0$. Since the functions in (3.32) are increasing and equal π_{sig} at zero, $a, b \geq \pi_{\mathrm{sig}}$ and hence $m \geq \pi_{\mathrm{sig}}$. Consequently

$$q_{\mathrm{com}} \geq m^2 \geq \pi_{\mathrm{sig}}^2, \qquad |\rho(t)| \leq e^{-\pi_{\mathrm{sig}}^2 t/2}. \tag{3.37}$$

Moreover, $\rho = um - 1 < 0$ gives $0 \leq u < 1/\pi_{\mathrm{sig}}$. Since $A, B \leq 1/4$, the three equations in (3.33) now imply

$$|\dot{u}| + |\dot{\zeta}_{\mathrm{sig}}| + |\dot{\zeta}_{\mathrm{bg}}| \leq C_{\pi_{\mathrm{sig}},\kappa} |\rho|.$$

Thus the balanced path has finite length and converges to a finite endpoint $(u_*, \zeta_{\mathrm{sig},*}, \zeta_{\mathrm{bg},*})$. Equation (3.37) gives $\rho_* = 0$ and therefore $u_* m_* = 1$. For every $t > 0$, one has $u(t) > 0$; because $\kappa > 0$, this makes $\zeta_{\mathrm{sig}}(t), \zeta_{\mathrm{bg}}(t) > 0$. Hence

$$a_* > \pi_{\mathrm{sig}}, \qquad b_* > \pi_{\mathrm{sig}}, \qquad m_* > \pi_{\mathrm{sig}}.$$

Write $q_{\mathrm{com},*} = \lim_{t\to\infty} q_{\mathrm{com}}(t)$ and $q_{\mathrm{diff},*} = \lim_{t\to\infty} q_{\mathrm{diff}}(t)$. Evaluating (3.34)–(3.35) at this endpoint yields

$$q_{\mathrm{com},*} \geq m_*^2 > \pi_{\mathrm{sig}}^2, \qquad q_{\mathrm{diff},*} \geq m_*^2 > \pi_{\mathrm{sig}}^2.$$

Denote the corresponding full parameter endpoint by w_* and set $G_* = \lim_{t\to\infty} G(t)$. The endpoint is used only through these inequalities; no explicit root formula is needed.

The parameter tail is bounded by the tail integral of (3.37), so the endpoint convergence is exponential. In particular, $q_{\mathrm{com}} - q_{\mathrm{com},*}$ is integrable and

$$\rho(t) = -\exp\left[-\frac{1}{2}\int_0^t q_{\mathrm{com}}(s)\,\mathrm{d}s\right] = -A_\kappa e^{-\gamma t}(1 + o(1)), \qquad \gamma = \frac{q_{\mathrm{com},*}}{2}, \quad A_\kappa > 0. \tag{3.38}$$

Perturbation. With the balanced endpoint in hand, we compute the perturbation explicitly and determine its asymptotic decay rate. Let $w_{n,\kappa}(t)$ denote the zero-initialized full-data trajectory. On the balanced trajectory, write $g_j(t) = \nabla_w \ell_j(w_{n,\kappa}(t)) = \rho(t) k_j$ for $j \in \mathcal{C}$. Each sample-gradient value occurs $n/2$ times. Hence

$$\bar{g}(t) = \frac{1}{2}(g_1(t) + g_2(t)) = \frac{\rho}{2}(k_1 + k_2), \qquad g_1 - \bar{g} = \frac{\rho}{2}(k_1 - k_2), \qquad g_2 - \bar{g} = -\frac{\rho}{2}(k_1 - k_2).$$

Since $\|k_1 - k_2\|^2 = 2(d - \chi) = 2q_{\mathrm{diff}}$,

$$\bar{\epsilon}_{n,\kappa}(t) = \frac{\rho(t)^2}{2(n-1)} q_{\mathrm{diff}}(t). \tag{3.39}$$

Equations (3.38) and (3.39) give

$$\nu_\epsilon(\kappa) = q_{\mathrm{com},*} > \pi_{\mathrm{sig}}^2. \tag{3.40}$$

Large-sample leave-one-out stability. We next transfer the balanced endpoint analysis to the LOO flow by proving stability under the $O(n^{-1})$ class imbalance. For the two transformed class predictions, define

$$\Phi(w) = \begin{pmatrix} y_1 f(w, x_1) \\ y_2 f(w, x_2) \end{pmatrix}, \quad \boldsymbol{\varrho}(w) = \Phi(w) - \mathbf{1}_2, \quad J(w) = D\Phi(w), \quad \mathcal{G}(w) = J(w)J(w)^\top.$$

For $|\delta| < 1$, set

$$\Pi_\delta = \mathrm{diag}\left(\frac{1-\delta}{2}, \frac{1+\delta}{2}\right).$$

The balanced flow is $\delta = 0$, while deletion of one class-1 sample is $\delta = \delta_n = 1/(n-1)$. The following quantities with class imbalance δ satisfy

$$\dot{w}^{(\delta)} = -J(w^{(\delta)})^\top \Pi_\delta \boldsymbol{\varrho}_\delta, \qquad \dot{\boldsymbol{\varrho}}_\delta = -\mathcal{G}(w^{(\delta)}) \Pi_\delta \boldsymbol{\varrho}_\delta, \qquad \boldsymbol{\varrho}_\delta = \boldsymbol{\varrho}(w^{(\delta)}), \qquad w^{(\delta)}(0) = 0. \tag{3.41}$$

By this notation, the balanced trajectory is $w^{(0)} = w_{n,\kappa}$. For this representative deletion, write

$$w_{n,\kappa}^{(-1)} = w^{(\delta_n)}.$$

Along the balanced path, $\mathcal{G} = G$ and (3.34)–(3.35) give $\mathcal{G} \succeq \pi_{\mathrm{sig}}^2 I_2$. The closure of that path is compact, so there is a compact tube around it on which $\mathcal{G} \succeq (\pi_{\mathrm{sig}}^2/2) I_2$ and on which J and the Hessians of the components of $\boldsymbol{\varrho}$ are bounded. As long as a perturbed path remains in the tube, its weighted residual energy

$$E_\delta = \frac{1}{2} \boldsymbol{\varrho}_\delta^\top \Pi_\delta \boldsymbol{\varrho}_\delta$$

satisfies, uniformly for sufficiently small $|\delta|$,

$$\dot{E}_\delta = -(\Pi_\delta \boldsymbol{\varrho}_\delta)^\top \mathcal{G}(w^{(\delta)})(\Pi_\delta \boldsymbol{\varrho}_\delta) \le -\lambda_E E_\delta \tag{3.42}$$

for some $\lambda_E > 0$. Here $E_0(t) = \bar{\ell}(w_{n,\kappa}(t), S_n)$, while $E_{\delta_n}(t) = \bar{\ell}(w_{n,\kappa}^{(-1)}(t), S_n^{-1})$.

For sufficiently small $|s|$, let $w^{(s)}$ be the zero-initialized solution of (3.41) with weight Π_s, and set $J_s = J(w^{(s)})$ and $\xi_s = \partial_s w^{(s)}$. On the common time interval for which these paths remain in the tube, (3.42) gives $\|\boldsymbol{\varrho}_s(t)\| \le C e^{-\lambda_{\mathrm{tube}} t}$ for some $C, \lambda_{\mathrm{tube}} > 0$, uniformly in s. Differentiation in s gives

$$\dot{\xi}_s = -J_s^\top \Pi_s J_s \xi_s - \sum_{j=1}^{2} (\Pi_s \boldsymbol{\varrho}_s)_j \nabla^2 \varrho_j(w^{(s)}) \xi_s - J_s^\top \Pi_s' \boldsymbol{\varrho}_s, \qquad \Pi_s' = \tfrac{1}{2}\,\mathrm{diag}(-1, 1).$$

Here ϱ_j denotes the j-th component of $\boldsymbol{\varrho}$. The first term is dissipative and the remaining terms are

bounded by $Ce^{-\lambda_{\text{tube}}t}(\|\xi_s\|+1)$; hence

$$\frac{\mathrm{d}}{\mathrm{d}t}\|\xi_s(t)\|^2 \le Ce^{-\lambda_{\text{tube}}t}(\|\xi_s(t)\|^2+1), \qquad \xi_s(0)=0.$$

Gronwall's inequality bounds ξ_s uniformly in s and t, so

$$\sup_{t\ge 0}\|w^{(\delta)}(t)-w^{(0)}(t)\| \le |\delta| \sup_{|s|\le|\delta|,\, t\ge 0}\|\xi_s(t)\| \le C|\delta|. \tag{3.43}$$

For sufficiently small $|\delta|$, this closes the tube estimate by a first-exit argument. Moreover, $\|\dot w^{(\delta)}(t)\| \le Ce^{-\lambda_{\text{tube}}t}$, so the path converges to a finite interpolating endpoint and its parameter tail is exponentially small. For $\delta=\delta_n$, write

$$w_{n,*}^{(-1)} = \lim_{t\to\infty} w_{n,\kappa}^{(-1)}(t).$$

Taking $t\to\infty$ in (3.43) and using the smoothness of $\mathcal{G}$ gives, for every sufficiently large even n,

$$\|w_{n,*}^{(-1)}-w_*\| = O(\delta_n), \qquad \|\mathcal{G}_{n,*}^{(-1)}-G_*\| = O(\delta_n), \qquad \|\mathcal{G}_n^{(-1)}(t)-\mathcal{G}_{n,*}^{(-1)}\| = O(e^{-\lambda_{\text{end},n}t}), \tag{3.44}$$

where $\mathcal{G}_n^{(-1)}(t)=\mathcal{G}(w_{n,\kappa}^{(-1)}(t))$ and $\mathcal{G}_{n,*}^{(-1)}=\mathcal{G}(w_{n,*}^{(-1)})$. Here $\lambda_{\text{end},n}>0$ may depend on the fixed sample size n.

The rate of $\bar\Delta_n$. This stability controls the LOO residual decay and yields the asymptotic lower bound for the loss-gap rate. For the representative class-1 deletion, write

$$\Delta_{n,\kappa}^{(-1)}(t) = \ell_1(w_{n,\kappa}^{(-1)}(t)) - \ell_1(w_{n,\kappa}(t)).$$

By the definition of the averaged loss difference in Eq. (3.6), token permutation invariance and the class-swap isometry make every pointwise loss difference identical after relabeling, and hence $\bar\Delta_{n,\kappa}=\Delta_{n,\kappa}^{(-1)}$. Let $\Pi_n=\Pi_{\delta_n}$. By (3.44),

$$\begin{aligned}\lambda_{n,-} &= \lambda_{\min}\Big(\Pi_n^{1/2}\mathcal{G}_{n,*}^{(-1)}\Pi_n^{1/2}\Big),\\ 2\lambda_{n,-} &\longrightarrow \lambda_{\min}(G_*) = \min\{q_{\text{com},*}, q_{\text{diff},*}\}.\end{aligned} \tag{3.45}$$

The weighted residual equation and (3.38) therefore imply, for every small $\eta>0$ and all large t,

$$\|\boldsymbol{\varrho}_{\delta_n}(t)\| \le C_{n,\eta}e^{-(\lambda_{n,-}-\eta)t}, \qquad |\rho(t)| \le C_\eta e^{-(\gamma-\eta)t}.$$

Writing $\rho_{1,n}^{(-1)}$ for the first component of $\boldsymbol{\varrho}_{\delta_n}$ gives

$$\bar\Delta_{n,\kappa} = \Delta_{n,\kappa}^{(-1)} = \frac{1}{2}\left[(\rho_{1,n}^{(-1)})^2-\rho^2\right], \qquad |\bar\Delta_{n,\kappa}(t)| \le C_{n,\eta}e^{-2(\min\{\lambda_{n,-},\gamma\}-\eta)t}. \tag{3.46}$$

Taking successively the t-lower limit, $\eta\downarrow 0$, and the even-n lower limit, and using $2\gamma=q_{\text{com},*}$, yields

$$\nu_\Delta(\kappa) \ge \min\{q_{\text{com},*}, q_{\text{diff},*}\} > \pi_{\text{sig}}^2. \tag{3.47}$$

Contraction on a positive $\bar{\Delta}_n$. Assuming the loss gap is eventually positive, we use the LOO residual asymptotics to lower-bound the original contraction ratio. Fix a sufficiently large even n and work on the eventual positive tail assumed in Lemma 3.22. For the representative deletion, set

$$\phi_{n,\kappa}^{(-1)}(w) = \left\langle \nabla \ell_1(w), \nabla \bar{\ell}(w, S_n^{-1}) \right\rangle, \qquad N_{n,\kappa}^{(-1)}(t) = \phi_{n,\kappa}^{(-1)}(w_{n,\kappa}^{(-1)}(t)) - \phi_{n,\kappa}^{(-1)}(w_{n,\kappa}(t)),$$

and

$$c_{n,\kappa}^{(-1)}(t) = \frac{N_{n,\kappa}^{(-1)}(t)}{\Delta_{n,\kappa}^{(-1)}(t)} = \bar{c}_{n,\kappa}(t),$$

where the last equality follows from Eq. (3.9), token permutation invariance, and the class-swap isometry. Set

$$Q_n = \frac{\rho^2}{(\rho_{1,n}^{(-1)})^2}, \qquad h_n = \frac{q_{\mathrm{com}} - \delta_n q_{\mathrm{diff}}}{2},$$

where the quantities on the right are evaluated along the full balanced trajectory. Then $0 \le Q_n < 1$. The LOO and full evaluations of $\phi_{n,\kappa}^{(-1)}$ are, respectively, $-\rho_{1,n}^{(-1)}\dot{\rho}_{1,n}^{(-1)}$ and $h_n\rho^2$; hence (3.46) gives the exact bridge

$$\frac{\bar{c}_{n,\kappa}}{2} = \frac{c_{n,\kappa}^{(-1)}}{2} = -\frac{\dot{\rho}_{1,n}^{(-1)}}{\rho_{1,n}^{(-1)}} + \frac{Q_n}{1-Q_n}\left(-\frac{\dot{\rho}_{1,n}^{(-1)}}{\rho_{1,n}^{(-1)}} - h_n\right). \tag{3.48}$$

Endpoint stability gives $\mathcal{G}_{n,*}^{(-1)} = G_* + O(\delta_n) \succ 0$ and turns the weighted residual equation into

$$\frac{\mathrm{d}}{\mathrm{d}t}\left(\Pi_n^{1/2}\boldsymbol{\varrho}_{\delta_n}\right) = -\left[\Pi_n^{1/2}\mathcal{G}_{n,*}^{(-1)}\Pi_n^{1/2} + O(e^{-\lambda_{\mathrm{end},n}t})\right]\Pi_n^{1/2}\boldsymbol{\varrho}_{\delta_n}.$$

Thus $\lambda_{n,-}$ is the smallest normal rate. Moreover, $\mathcal{G}_{n,*}^{(-1)} \succ 0$ makes the endpoint residual Jacobian rank two; the analytic implicit-function theorem gives a codimension-two interpolation manifold, and the holomorphic stable-manifold theorem gives a two-dimensional analytic stable fiber through the LOO endpoint (Ilyashenko and Yakovenko, 2008, Thm. 7.1). Exponential endpoint convergence places the LOO orbit in this fiber, where $\Pi_n^{1/2}\boldsymbol{\varrho}$ is a local analytic coordinate. Writing $\lambda_{n,+} = \lambda_{\max}(\Pi_n^{1/2}\mathcal{G}_{n,*}^{(-1)}\Pi_n^{1/2})$, the convergent Poincaré–Dulac normal form on this fiber (Ilyashenko and Yakovenko, 2008, Thm. 5.5) has only the possible resonance $\lambda_{n,+} = k\lambda_{n,-}$ with $k \in \mathbb{Z}_{\ge 2}$. Its solutions are exponential-polynomials, so substitution into the residual's convergent Taylor series, followed by termwise differentiation, gives the simultaneous C^1 asymptotics

$$\rho_{1,n}^{(-1)}(t) = A_n t^{p_n^{\mathrm{poly}}} e^{-\mu_n t}(1+o(1)), \qquad -\frac{\dot{\rho}_{1,n}^{(-1)}(t)}{\rho_{1,n}^{(-1)}(t)} = \mu_n - \frac{p_n^{\mathrm{poly}}}{t} + o(1) \longrightarrow \mu_n \ge \lambda_{n,-},$$

for some $A_n \neq 0$ and $p_n^{\mathrm{poly}} \in \mathbb{Z}_{\ge 0}$. The positive gap ensures that the held-out residual is not identically zero, and μ_n is a positive integer combination of the two normal rates.

Together with (3.38), this gives

$$Q_n(t) = \frac{A_\kappa^2}{A_n^2} t^{-2p_n^{\mathrm{poly}}} e^{-2(\gamma-\mu_n)t}(1+o(1)).$$

Since $Q_n < 1$, necessarily $\mu_n \le \gamma$. If $\mu_n < \gamma$, then $Q_n \to 0$; if $\mu_n = \gamma$, then

$$-\frac{\dot\rho_{1,n}^{(-1)}}{\rho_{1,n}^{(-1)}} - h_n \longrightarrow \frac{\delta_n q_{\mathrm{diff},*}}{2} > 0.$$

Thus (3.48) yields, in either case,

$$\liminf_{t\to\infty} \bar c_{n,\kappa}(t) \ge 2\mu_n \ge 2\lambda_{n,-}.$$

Taking the outer lower limit and using (3.45) gives

$$c(\kappa) \ge \min\{q_{\mathrm{com},*}, q_{\mathrm{diff},*}\} > \pi_{\mathrm{sig}}^2. \tag{3.49}$$

Fixed-attention baseline. Finally, we solve the fixed-attention dynamics explicitly to obtain the baseline rates and complete the comparison. When $\kappa = 0$, attention remains uniform and

$$d_0 = \pi_{\mathrm{sig}}^2 + \pi_{\mathrm{bg}}^2, \qquad \chi_0 = -\pi_{\mathrm{bg}}^2, \qquad q_{\mathrm{com},0} = d_0 + \chi_0 = \pi_{\mathrm{sig}}^2.$$

Equation (3.39) immediately gives $\nu_\epsilon(0) = \pi_{\mathrm{sig}}^2$. For the LOO flow, direct diagonalization of the constant two-class residual operator gives

$$2\lambda_{n,-}^{(0)} = d_0 - \sqrt{\delta_n^2 d_0^2 + (1-\delta_n^2)\pi_{\mathrm{bg}}^4} < \pi_{\mathrm{sig}}^2,$$

and $2\lambda_{n,-}^{(0)} \to d_0 - \pi_{\mathrm{bg}}^2 = \pi_{\mathrm{sig}}^2$.

The initial LOO residual $(-1,-1)^\top$ is not an eigenvector: equality of the two row sums would require $(p_- - p_+)(d_0 - \chi_0) = 0$, where $p_- = (1-\delta_n)/2$ and $p_+ = (1+\delta_n)/2$, but both factors are nonzero. Since $\chi_0 \neq 0$, the two eigenvalues are distinct, so the initial vector has a nonzero slow-mode coefficient. The same nonzero off-diagonal entry ensures that the slow eigenvector is visible in the held-out coordinate. Because $2\lambda_{n,-}^{(0)} < \pi_{\mathrm{sig}}^2$, this visible LOO mode dominates the full-data residual. Thus, for some $B_n \neq 0$,

$$\rho_{1,n}^{(-1)}(t) = B_n e^{-\lambda_{n,-}^{(0)} t}(1+o(1)), \qquad \frac{\rho(t)^2}{(\rho_{1,n}^{(-1)}(t))^2} \longrightarrow 0, \qquad -\frac{\dot\rho_{1,n}^{(-1)}(t)}{\rho_{1,n}^{(-1)}(t)} \longrightarrow \lambda_{n,-}^{(0)}.$$

Hence the gap is eventually positive with rate $2\lambda_{n,-}^{(0)}$. The exact bridge (3.48), which is algebraic and also applies when $\kappa = 0$, gives the same limit for the contraction:

$$\liminf_{t\to\infty}\left[-\frac{1}{t}\log|\bar\Delta_{n,0}(t)|\right] = \lim_{t\to\infty} \bar c_{n,0}(t) = 2\lambda_{n,-}^{(0)}.$$

Letting $n \to \infty$ proves

$$\nu_\Delta(0) = c(0) = \nu_\epsilon(0) = \pi_{\mathrm{sig}}^2.$$

Combining this equality with (3.40), (3.47), and (3.49) proves (3.30) and (3.31). □

The above lemma shows that trainable attention learns to emphasize the class-specific signal tokens over the shared background tokens. The resulting representation separates the two classes more effectively, allowing the predictions to approach the correct labels more rapidly. Under the squared loss, the sample-wise gradients and their variation across samples therefore vanish faster, leading to faster perturbation decay. Meanwhile, by relating to the residual during training,we show that the predictor with $\kappa > 0$ makes the loss difference between the full-data and leave-one-out trajectories shrink faster, resulting in stronger contraction under the additional positivity condition. Thus, trainable attention improves both perturbation and contraction through the representation it learns. This example, albeit simplistic, shows that we can calculate the contraction and perturbation factors and the generalization gap for different types of architectures.

We should note that the predictor is invariant to permutations of token positions because self-attention is permutation equivariant and the subsequent mean pooling is permutation invariant. Consequently, all inputs with the same numbers of signal and background tokens have the identical results regardless of the ordering.

Related ideas, in which attention progressively concentrates on the most relevant signal tokens, also appear in Li et al. (2023) and Oymak et al. (2023). Under structured data models with informative and irrelevant tokens, these works show that gradient-based training can emphasize task-relevant information. Our example complements them by quantifying how this adaptive selection affects the contraction and perturbation components of generalization dynamics.

3.5. Related Work and Discussion

Our approach adapts and develops concepts from several established lines of literature to provide a tighter, data-dependent estimate of generalization.

Comparing trajectories in terms of their loss instead of their weights. Arora et al. (2019) also study the residual dynamics to obtain an estimate of the norm of eventual weights that are reachable by gradient descent. They derive a weight-norm-based bound using the PAC learning framework (Valiant, 1984) and Rademacher complexity (Bartlett and Mendelson, 2002; Bartlett et al., 2017). Similar ideas are adopted by Allen-Zhu et al. (2019) and Cao and Gu (2019) for analyzing SGD and online learning, respectively, in fully-connected neural nets. Liu et al. (2022) derive a weight-norm-bound under uniform conditions for the Lojasiewicz gradient inequality. Richards and Kuzborskij (2021) and Akbari et al. (2021) analyze the sensitivity of the weights found by the algorithm upon replacing one sample for uniformly Lipschitz losses. The key reason for loose generalization bounds in these analyses is that they are conducted in weight space, under uniform assumptions over the entire loss landscape. In contrast, we study the evolution of the residual in Eq. (3.16), which is more closely related to the difference in loss after training on perturbed datasets—as opposed to the difference in weights. Our analysis is conducted directly in the prediction space, which is why it provides a more precise characterization of generalization.

Relation to stability-based, trajectory-based and information-theoretic approaches. Stability based generalization bounds (Bousquet and Elisseeff, 2002; Hardt et al., 2016; Xu and Mannor, 2012) can be adapted for stochastic algorithms using information-theoretic approaches (Xu and Raginsky, 2017; Mou et al., 2018; Chu and Raginsky, 2023) to provide remarkably tight generalization bounds. Dadi and Cevher (2025) compare SGLD runs on different datasets to derive generalization bounds that do not grow with training time in non-convex settings. Nickl et al. (2023) use the memory-perturbation equation to estimate sample sensitivity and predict test loss through

approximate leave-one-out cross-validation during training. Steinke and Zakynthinou (2020) develop a conditional mutual information framework that relates generalization to how much a learned model reveals about which examples in a super-sample were used for training. Negrea et al. (2019), Neu et al. (2021), and Banerjee et al. (2022) develop information-theoretic generalization bounds in terms of the sum of the trace of the gradient covariance along the training trajectory—this is $\sum_t \mathrm{tr}\hat{\Sigma}_n(t)$ in our notation. This sum tells us about the size of the tube of trajectories in loss space that arises from training on slightly different datasets. The quality of the estimate of this tube completely determines the quality of the bound. Our expression in Eq. (3.11) provides a more general and tighter estimate of this tube than these prior works, because the damping factor $\exp\left(\int_s^t -\bar{c}_n(u)du\right)$ corrects for the size of the tube. A positive contraction factor leads to faster contraction of two trajectories that are being trained on slightly different datasets. Our analysis applies to deterministic algorithms, unlike previous work on information-theoretic bounds (Xu and Raginsky, 2017; Mou et al., 2018; Futami and Fujisawa, 2023), which only holds for randomized algorithms because the proof relies on the non-expansiveness of the KL-divergence of non-singular distributions.

Beyond information-theoretic approaches, several works derive generalization bounds from quantities computed along the training trajectory: Clerico et al. (2025) use pathwise curvature within a deterministic PAC-Bayes framework, Fu et al. (2023) use the decrease in the training loss and gradient covariance, and Chen et al. (2023b, 2025) use cumulative per-example gradient norms through a data-dependent loss path kernel. Johnson and Zhang (2023) bound the expected generalization gap of stochastic training using output inconsistency and instability. These approaches summarize a single training path by accumulating certain quantities along the trajectory. Trajectory-based analysis can also motivate algorithmic choices such as early stopping: Yao et al. (2007) compare empirical and population gradient-descent trajectories to derive stopping rules that balance approximation and sample-size errors.

Generalization bounds for kernel machines. Generalization loss in kernel ridge regression (Rakhlin and Liang, 2020; Mallinar et al., 2022) can be expressed in terms of quantities that resemble ours, namely, the alignment of the residuals with the Gram matrix $r(0)^\top K r(0)$ (Arora et al., 2019; Jacot et al., 2020). Woodworth et al. (2020) show that initialization scale controls the transition between kernel and rich regimes in a family of deep linear networks, influencing both implicit bias and generalization. Beyond kernel methods, Li et al. (2020b) analyze the training dynamics of two-layer ReLU networks and show that they can achieve better generalization than kernel methods under Gaussian inputs and orthogonal target weights. Our effective Gram matrix generalizes this idea to arbitrary deep neural networks and loss functions, going beyond two-layer neural networks and squared losses. However, unlike kernel ridge regression, where the Gram matrix is derived from a fixed kernel that directly recovers the target function, the effective Gram matrix K_n in our setting varies for different datasets and training regimes and does not necessarily coincide with any fixed kernel.

Contraction theory. Lohmiller and Slotine (1998, 2000) provide a uniform contraction rate and perturbation magnitude over time and space (α and $\bar{b}$ in Theorems 3.1 and 3.2). Davydov et al. (2022) extend contraction analysis to arbitrary norms and establish incremental input-to-state stability bounds that quantify how contraction and input perturbations jointly govern the distance between trajectories. In contrast, we derive these terms $c_n^{-i}(t)$ and $\epsilon_n^{-i}(t)$ in Lemma 3.7 directly from the evolution of $\bar{\Delta}_n^{-i}(t)$. This is a local quantity and describes only the contraction and

perturbation of $w_n(t)$ and $w_n^{-i}(t)$. It varies over time. The central motivation of this chapter is that the non-uniformity allows for a more refined analysis of gradient flow for neural networks. Indeed, the energy landscape may not be uniformly good in the entire weight space, but as we see in Fig. 3.1, it can be benign along most of the training trajectory. Our development in this chapter therefore diverges significantly from the generalization bounds derived in contraction theory (Kozachkov et al., 2023a; Charles and Papailiopoulos, 2018), and our results generalize these ideas, see Remark 3.8.

Cross-validation-based estimators of the generalization gap. These are widely used methods for estimating risk (Hastie et al., 2009). Blum et al. (1999) show that the cross-validated estimator performs no worse than the corresponding data-splitting estimator under certain conditions. Kale et al. (2011) show that the k-fold cross-validated estimator achieves at least a k-fold reduction in variance under certain stability conditions. Kumar et al. (2013) relax the stability condition. Austern and Zhou (2025) improve these results by proving central limit theorems and Berry-Esseen bounds for the cross-validation loss estimators under certain stability conditions. Patil et al. (2024) show that, in high-dimensional least squares, LOOCV consistently estimates prediction risk uniformly along the early-stopped gradient-descent trajectory. Chen et al. (2023a) build upon PAC-Bayes bounds developed by Yang et al. (2022) to obtain an estimate of the effective dimensionality of deep networks by computing the relationship between the leave-one-out estimator of the test loss and the number of training samples. Relatedly, Oberman (2026) uses leave-one-out stability to derive error bounds for transductive semi-supervised learning on data-augmentation graphs.

CHAPTER 4

PROSPECTIVE LEARNING: LEARNING FOR A DYNAMIC FUTURE

In the previous chapters we analyze the generalization of time-agnostic model trained on independent and identically distributed (IID) data, which based heavily on the prevailing theoretical framework of PAC learning or PAC-Bayesian as its stochastic version. However in real world applications, learning involves updating decision rules or policies, based on past experiences, to improve future performance. PAC learning has been extremely useful to develop algorithms that minimize the risk—typically defined as the expected loss—on unseen samples under certain assumptions. The assumption, that samples are IID within the training dataset and at test time, has served us well. But it is neither testable nor believed to be true in practice. The future is always different from the past: both distributions of data and goals of the learner may change over time. Moreover, those changes may cause the optimal hypothesis to change over time as well. There are numerous mathematical and empirical approaches that have been developed to address this issue, e.g., techniques for being invariant to (Blum-Smith and Villar, 2023), or adapting to, distribution shift (Quiñonero-Candela, 2009), modeling the future as a different task, etc. But we lack a first-principles framework to address problems where data distributions and goals may change over time in such a way that the optimal hypothesis is time-dependent. And as a consequence, machine learning-based AI today is brittle to changes in distribution and goals.

4.1. Introduction

This chapter develops a theoretical framework called "Prospective Learning" (PL). Instead of data arising from an unknown probability distribution like in PAC learning, prospective learning assumes that data comes from an unknown stochastic process, that the loss considers the future, and that the optimal hypothesis may change over time. A prospective learner uses samples received up to some time $t \in \mathbb{N}$ to output an infinite sequence of predictors, which it uses for making predictions on data at all future times $t' > t$. We discuss how prospective learning is related to existing problem formulations in the literature in Sec. 4.6.1.

Why should one care about prospective learning? Imagine a deployed machine learning system. The designer of this system desires to optimize—not the risk upon the past training data, or the risk on the immediate future data—but the risk on all data that the model will make predictions upon in the future. As data evolves, e.g., due to changing trends and preferences of the users, the optimal hypothesis to make predictions also changes. Time is the critical piece of information if the system designer is to achieve their goals. Both in the sense of how far back in time a particular datum was recorded, and in the sense of how far ahead in the future this system will be used to make predictions. The designer must take time into account to avoid retraining the model periodically, *ad infinitum*.

Biology is also rich with examples where systems seem to behave prospectively. The principle of allostasis, for example, states that regulatory processes of living things anticipate the needs of the organism and prepare to satisfy these needs before, rather than after, they arise (Sterling, 2012). For example, mitochondria increase their energy production to anticipate the demands of muscles (Wang et al., 2017), neural circuits anticipate changes in sensory stimuli and the task (i.e., predictive coding; Huang and Rao, 2011), and individual organisms optimize their actions with respect to anticipated changes in their environments (Seligman et al., 2013, 2016). These regulatory principles were learned early in evolutionary time so they must be important. In short, the world—including our internal

drives—changes all the time, and learning systems must anticipate (that is, prospect) these changes to thrive.

Contributions

- Sec. 4.2 defines Prospective Learning (PL) as an approach to address problems where the optimal hypothesis may evolve over time (due to shifts in distributions and/or goals of the learner). It also provides illustrative examples of PL.
- Sec. 4.6.1 puts prospective learning in context relative to existing ideas in the literature for addressing changes in the data distribution.
- Sec. 4.3 takes steps towards a theoretical foundation for prospective learning. We define strongly learnability (i.e., there exists a prospective learner whose risk is arbitrarily close to the Bayes optimal learner) and weakly learnability (i.e., there exists a prospective learner whose risk is better than chance) (Kearns and Valiant, 1994). Empirical risk minimization (ERM) without incorporating time, can result in failure to strongly, or even weakly, learn prospectively (Lugosi and Zeger, 1995).
- Sec. 4.4 introduces prospective empirical risk minimization, and proves that it can learn prospectively under certain assumptions on the stochastic process and loss.
- Sec. 4.5 demonstrates that our prospective learners can prospectively learn several canonical problems constructed using synthetic, MNIST (LeCun et al., 1990) and CIFAR-10 (Krizhevsky, 2009b) data. In contrast, a number of existing algorithms, including ERM, online and continual learning algorithms, fail. Sec. 4.5.4 demonstrates that current large language models, which use Transformer-based architectures trained using auto-regressive losses, fail to learn prospectively.

4.2. A definition of prospective learning

A prospective learner minimizes the expected cumulative risk of the future using past data. Such a learner is defined by the following key ingredients (see Fig. 4.1 (left) for schematic illustration).

Data. Let the input and output at time t be denoted by $x_t \in \mathcal{X}$ and $y_t \in \mathcal{Y}$ respectively. Let $z_t = (x_t, y_t)$. We will model the data as a stochastic process $Z \equiv (Z_t)_{t\in\mathbb{N}}$ defined on an appropriate probability space $(\Omega, \mathcal{F}, \mathbb{P})$. At time $t \in \mathbb{N}$, denote past data by $z_{\leq t} \equiv (z_1, \dots, z_t)$ and future data by $z_{>t} \equiv (z_{t+1}, \dots)$. We will find it useful to distinguish between the realization of the data, denoted by $z_{\leq t}$, and the corresponding random variable, $Z_{\leq t}$.

Hypothesis class. At each time t, a prospective learner selects an infinite sequence $h \equiv (h_1, \dots, h_t, h_{t+1}, \dots)$ which it uses to make predictions on data at any time in the future. Each element of this sequence $h_t : \mathcal{X} \mapsto \mathcal{Y}$ and therefore $h_t \in \mathcal{Y}^{\mathcal{X}}$.[3] The hypothesis class $\mathcal{H}$ is the space of such hypotheses, $h \in \mathcal{H} \subseteq (\mathcal{Y}^{\mathcal{X}})^{\mathbb{N}}$.[4] We will again use the shorthand $h_{\leq t} \equiv (h_1, \dots, h_t)$. We will sometimes talk about a "time-agnostic hypothesis" which will refer to a hypothesis such that $h_t = h_{t'}$ for all $t, t' \in \mathbb{N}$. Observe that this makes our setup different from the standard setup in PAC learning where the learner selects a single hypothesis in $\mathcal{Y}^{\mathcal{X}}$. One could also think of prospective learning as using a single time-varying hypothesis $h : \mathbb{N} \times \mathcal{X} \mapsto \mathcal{Y}$, i.e., the hypothesis takes both time and the datum as input to make a prediction.

[3] We will use some non-standard notation in this chapter. In particular, a hypothesis h will always refer to sequence of predictors $h \equiv (h_1, \dots, h_t, h_{t+1}, \dots)$. This helps us avoid excessively verbose mathematical expressions.

[4] When we say that "learner selects a hypothesis" in the sequel, it will always mean that the learner selects an infinite sequence from within the hypothesis class $\mathcal{H}$.

Learner. A prospective learner is a map from the data received up to time t, to a hypothesis that makes predictions on the data over all time (past and future): $(\mathcal{X} \times \mathcal{Y})^t \to (\mathcal{Y}^{\mathcal{X}})^{\mathbb{N}}$. The learner gives as output a hypothesis $h(z_{\leq t}) \in \mathcal{H}$. Unlike a PAC learner, a prospective learner can make different kinds of predictions at different times. This is a crucial property of prospective learning. In other words, after receiving data up to time t, the hypothesis selected by the prospective learner can predict on samples at any future time $t' > t$.

Prospective loss and risk. The future loss incurred by a hypothesis h is

$$\bar{\ell}_t(h, Z) = \limsup_{\tau \to \infty} \frac{1}{\tau} \sum_{s=t+1}^{t+\tau} \ell(s, h_s(X_s), Y_s), \tag{4.1}$$

where $\ell : \mathbb{N} \times \mathcal{Y} \times \mathcal{Y} \mapsto [0, 1]$ is a bounded loss function.[5] Prospective risk at time t is the expected future loss [6]

$$R_t(h) = \mathbb{E}\left[\bar{\ell}_t(h, Z) \,|\, z_{\leq t}\right] = \int \bar{\ell}_t(h, Z) \,\mathrm{d}\, \mathbb{P}_{Z \,|\, z_{\leq t}}\,, \tag{4.2}$$

where we assume that h is a random variable and $h \in \sigma(Z_{\leq t})$ where $\sigma(\cdot)$ denotes the filtration (an increasing sequence of sigma algebras) of the stochastic process Z. We have used the shorthand $\mathbb{E}[Y \,|\, x]$ for $\mathbb{E}[Y \,|\, X = x]$. Observe that we have conditioned the prospective risk of the hypothesis h upon the realized data $z_{\leq t}$. We can take an expectation over the realized data, to obtain the expected prospective risk

$$\mathbb{E}\left[R_t(h)\right] = \int R_t(h) \,\mathrm{d}\, \mathbb{P}_{Z_{\leq t}}\,.$$

Prospective Bayes risk is the minimum risk achievable by any hypothesis. In PAC learning, it is a constant that depends upon the (fixed) true distribution of the data and the risk function. In prospective learning, the optimal hypothesis can predict differently at different times. We therefore define the prospective Bayes risk at a time t as

$$R_t^* = \inf_{h \in \sigma(Z_{\leq t})} R_t(h), \tag{4.3}$$

which is the minimum achievable prospective risk by any learner that observes data $z_{\leq t}$. We define the Bayes optimal learner as any learner that achieves a Bayes optimal risk at every time $t \in \mathbb{N}$. In certain contexts, one might be interested in the limiting prospective Bayes risk as $t \to \infty$.

4.2.1. Different prospective learning scenarios with illustrative examples

We next discuss four prospective learning scenarios that are relevant to increasingly more general classes of stochastic processes. Our goal is to illustrate, using examples, how the definitions developed in the previous section capture these scenarios. We will assume that for all times t we have $X_t = 1$, $Y_t \in \{0, 1\}$. We will also focus on the time-invariant zero-one loss $\ell(t, \hat{y}, y) = \delta(\hat{y} \neq y)$ for all t, here δ is the Dirac delta function. Fig. 4.1 shows example realizations of the data for each scenario.

[5] The limsup is guaranteed to exist if ℓ is bounded. If the series converges, we can use lim instead.

[6] There are many real world scenarios where expected future loss may not be sufficient for good performance, e.g., for portfolio managements or inference by biological learners who optimize for a balance between value and risk. Moreover, the risk functional could, in general, also change over time. In this chapter, we will focus only on the expected future loss.

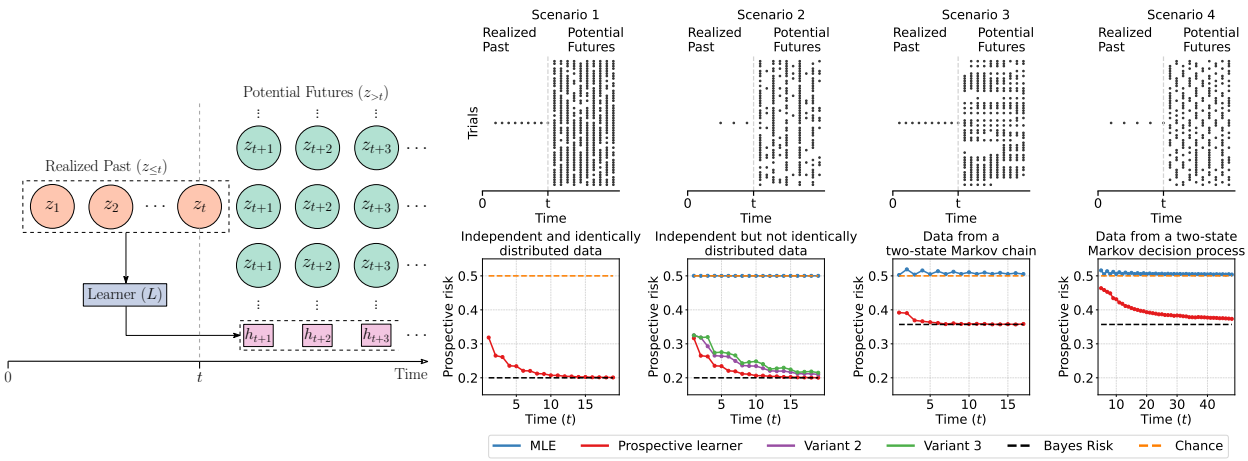

Figure 4.1: A schematic for prospective learning (left) and realizations of the examples for the four scenarios (top right); dots denote 1s and empty spaces denote 0s for $Y_t \in \{0, 1\}$ with $X_t = 1$ for all times t. Prospective risk of learners at different times is shown in the bottom panels and discussed in Sec. 4.2.1. **Theorem 4.1:** For Bernoulli probability $p = 0.2$, the maximum-likelihood estimator (MLE) in blue uses a time-agnostic hypothesis $h_t(X_t) = \mathbf{1}(\hat{p}_t > 0.5)$ where $\hat{p}_t = t^{-1} \sum_{s=1}^{t} y_s$, ties at $\hat{p}_t = 0.5$ are broken randomly. The risk of this learner converges to the Bayes risk. **Theorem 4.2:** For Bernoulli probability $p = 0.2$, the MLE estimator (blue) performs at chance levels. A prospective learner (red) that alternates between two predictors at even and odd times converges to Bayes risk. Variants of this learner that use less information from the stochastic process (purple does not know that the data distributions at even and odd times are tied, green does not know that the distribution shifts at every time-step) also converge to Bayes risk, but more slowly. **Theorem 4.3:** For $\theta = 0.1$ and $\gamma = 0.9$ in the discounted prospective risk, the MLE estimator (blue) again performs at chance levels. A prospective learner that computes an estimate of the transition probability of the two-state Markov chain to estimate $\mathbb{P}(Y_{t'} \mid y_t)$ for future times $t' > t$ converges to Bayes risk. **Theorem 4.4:** For $\theta_0 = \theta_1 = 0.1$, the MLE estimator (blue) performs at chance levels. A prospective learner that uses a variant of Q-learning (described in Theorem 4.4) converges to the prospective Bayes risk.

Scenario 4.1 (Data is independent and identically distributed). Formally, this consists of stochastic processes where $\mathbb{P}_{Z_{t'} \mid Z_{\leq t}} = \mathbb{P}_{Z_t}$ for all $t, t' \in \mathbb{N}$. As an example, consider $Y_t \sim \text{Bernoulli}(p)$ for some unknown parameter $p \in [0, 1]$. Prospective Bayes risk is equal to $\min(p, 1-p)$ in this case. A time-agnostic hypothesis, for example one that thresholds the maximum likelihood estimator (MLE) of the Bernoulli probability, converges to the limiting prospective Bayes risk.

Calculations. To examine whether learning can benefit from prospection even in this IID setting, let the MLE-based learner return the hypothesis $h_{t'} = h_t = \text{threshold}(\hat{p}_t > 0.5)$ for all $t' > t$, where $\hat{p}_t = \frac{1}{t} \sum_{s=1}^{t} y_s$. Alternatively, if we assume a prior distribution $\text{Beta}(\alpha, \beta)$ over p, then the resulting maximum *a posteriori* (MAP) estimate is $\hat{p}_t = \frac{\alpha + \sum_{s=1}^{t} y_s - 1}{\alpha + \beta + t - 2}$. The corresponding MAP-based learner returns the sequence $\hat{h}_{\geq t} = (\hat{h}_t, \hat{h}_t, \cdots)$, where $\hat{h}_t = \text{threshold}(\hat{p}_t > 0.5)$ for all future times beyond t. If the prior distribution has a small divergence with respect to the true posterior distribution, then the MAP-based learner converges faster to the Bayes optimal risk; for a poor choice of prior, the convergence is slower. However, in such situations, we can modify the MAP-based learner to use prospection and incorporate "time" to obtain faster convergence to the Bayes risk.

Let $y_1, \ldots, y_t$ be the IID sample sequence observed up to time t. We compute the rate of change

$\Delta p(t)$ of the MAP estimate at time t as

$$\Delta p(t) = \mathrm{MAP}(y_1, \ldots, y_t) - \mathrm{MAP}(y_1, \ldots, y_{t-1}). \tag{4.4}$$

Taking expectations on both sides of Eq. (4.4), and plugging in $\hat{p}_t = \mathrm{MAP}(y_1, \ldots, y_t)$ for the true parameter p, we construct the following estimate for $\Delta p(t)$:

$$\hat{\Delta} p(t) = \frac{(\alpha - 1) + tp}{(\alpha - 1) + (\beta - 1) + t} - \frac{(\alpha - 1) + (t-1)p}{(\alpha - 1) + (\beta - 1) + t - 1}.$$

Using this rate of change, we may forecast the estimate $p_{t'}$ at time $t' > t$ as

$$\hat{p}_{t'} = \hat{p}_t + \sum_{s=t}^{t'-1} \hat{\Delta} p(s).$$

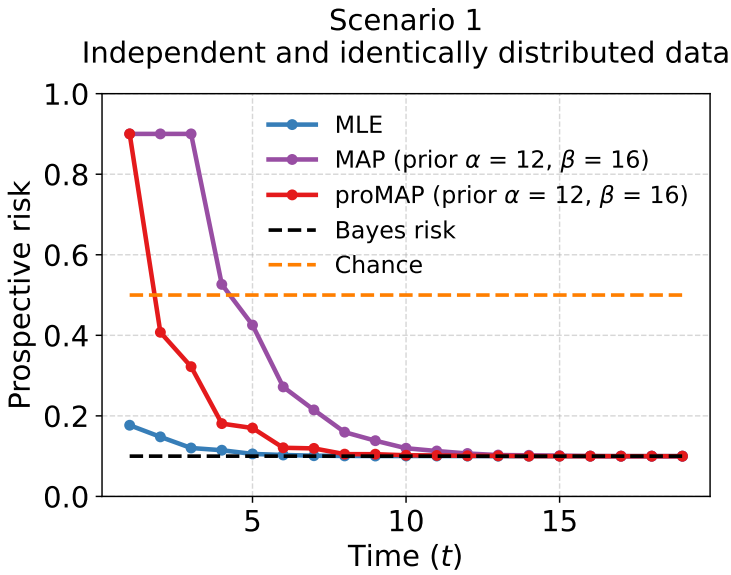

Figure 4.2: Prospective risk of MLE *(blue)*, MAP *(purple)*, prospective MAP *(red)* and random-chance *(orange)* based learners with respect to time. Both MAP and prospective MAP estimators assume a prior distribution of $\mathrm{Beta}(12, 16)$ over p.

We refer to this as the prospective MAP estimate. Based on it, we set the hypothesis to be $h_{t'} = \mathrm{threshold}(\hat{p}_{t'} > 0.5)$ for all future times beyond t. In Fig. 4.2, we plot the prospective risk of the MLE, MAP, and prospective MAP-based learners. Due to an unfavorable prior, the MAP-based learner converges slowly. However, the prospective MAP-based learner manages to leverage its forecasting to achieve a faster convergence rate despite having the same prior as the MAP. This shows that we can indeed benefit from prospection even in the IID case.

Scenario 4.2 (Data is independent but not identically distributed). This consists of stochastic processes where $\mathbb{P}_{Z_t \mid Z_{<t}} = \mathbb{P}_{Z_t}$ for all $t \in \mathbb{N}$. Consider $Y_t \sim \mathrm{Bernoulli}(p)$ if t is odd, and

$Y_t \sim \text{Bernoulli}(1-p)$ if t is even, i.e., data is drawn from two different distributions at alternate times. Prospective Bayes risk is again equal to $\min(1-p, p)$ in this case. A time-agnostic hypothesis can only perform at chance level. But a prospective learner, for example one that selects a hypothesis that alternates between two predictors at even and odd times, can converge to prospective Bayes optimal risk. We can also construct variants, e.g., when the relationship between the Bernoulli probabilities are not known (Variant 1 in Fig. 4.1), or when the learner does not know that the data distribution changes at every time step (Variant 2 in Fig. 4.1 where we implemented a generalized likelihood ratio test to determine whether the distribution changes). The risk of these variants also converges to prospective Bayes risk, but they need more samples because they use more generic models of the stochastic process. This scenario is closely related to (online) multitask/meta-learning (Finn et al., 2017b).

Scenario 4.3 (Data is neither independent nor identically distributed). Formally, this scenario consists of general stochastic processes. As an example, consider a Markov process $\mathbb{P}(Y_{t+1} = k \,|\, Y_t = k) = \theta$ with two states $k \in \{0, 1\}$ and $Y_1 \sim \text{Bernoulli}(\theta)$. The invariant distribution of this Markov process is $\mathbb{P}(0) = \mathbb{P}(1) = 1/2$. Prospective Bayes risk is also equal to $1/2$. For stochastic processes that have a invariant distribution, it is impossible to predict the next state infinitely far into the future and therefore it is impossible to prospect. The prospective Bayes risk is trivially chance levels. In such situations, the learner could consider losses that are discounted over time. For example, one could use a slightly different loss than the one in Eq. (4.1) to write

$$\bar{\ell}_t(h, Z) = (1-\gamma) \sum_{s=t+1}^{\infty} \gamma^{s-t-1} \ell(h_s(X_s), Y_s) \tag{4.5}$$

for some $\gamma \in [0, 1)$. The prospective Bayes risk for this example is calculated analytically below. For $\gamma = 0.9$, $\theta = 0.1$ and the zero-one loss, the limiting prospective Bayes risk is 0.357, as derived below.

Calculations. To derive this risk, consider the slightly more general Markov chain in which $P(Y_{t+1} = 0 \,|\, Y_t = 0) = \theta_0$ and $P(Y_{t+1} = 1 \,|\, Y_t = 1) = \theta_1$, with transition matrix

$$\Gamma = \begin{bmatrix} \theta_0 & 1-\theta_0 \\ 1-\theta_1 & \theta_1 \end{bmatrix}.$$

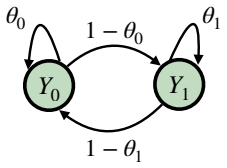

Figure 4.3: Markov chain describing the evolution of data.

The probability distribution at time t' is given by $\Gamma^{t'-t}(z_t, 1-z_t)^T$. The eigenvalues of the transition matrix are $\lambda_1 = 1$ and $\lambda_2 = \theta_0 + \theta_1 - 1$, with corresponding eigenvectors $(1, 1)^\top$ and $(\theta_0 - 1, 1 - \theta_1)^\top$.

Diagonalizing Γ gives

$$\begin{aligned}\Gamma^{t'-t} &= \begin{bmatrix}1 & \theta_0 - 1\\ 1 & 1-\theta_1\end{bmatrix}\begin{bmatrix}\lambda_1^{t'-t} & 0\\ 0 & \lambda_2^{t'-t}\end{bmatrix}\begin{bmatrix}1 & \theta_0 - 1\\ 1 & 1-\theta_1\end{bmatrix}^{-1}\\ &= \frac{1}{(2-\theta_0-\theta_1)}\begin{bmatrix}1-\theta_1+(1-\theta_0)\lambda_2^{t'-t} & (1-\theta_0)-\lambda_2^n\\ 1+\theta_1+(1-\theta_0)\lambda_2^{t'-t} & (1-\theta_0)+\lambda_2^n\end{bmatrix}.\end{aligned}$$

This implies that the probability distribution of the state at time t' is

$$\pi_{t'} = \frac{1}{(2-\theta_0-\theta_1)}\begin{bmatrix}1-\theta_1\\ 1-\theta_0\end{bmatrix} + \frac{\lambda_2^{t'-t}\left((1-\theta_0)(1-z_t)-(1-\theta_1)z_t\right)}{(2-\theta_0-\theta_1)}\begin{bmatrix}1\\ -1\end{bmatrix}.$$

Hence, the optimal sequence of hypotheses is $h^*_{\geq t+1} = (h^*_{t+1}, h^*_{t+2}, \dots)$, where

$$h^*_{t'} = \arg\max_{i\in\{0,1\}} \pi_{t'}(i),$$

with Bayes risk

$$R^*_t = \lim_{T\to\infty}\frac{1}{T}\sum_{s=t}^{T}\min_{i\in\{0,1\}}\pi_s(i) = \frac{1}{(2-\theta_0-\theta_1)}\min(1-\theta_0, 1-\theta_1).$$

The second term in the expression of $\pi_{t'}$ vanishes as $T\to\infty$. If $\theta_0 = 0.9$ and $\theta_1 = 0.5$, then $R^*_t = 1/6$.

If we restrict our attention to the case where $\theta_0 = \theta_1$, the discounted Bayes risk reduces to

$$\begin{aligned}(1-\gamma)\sum_{s=t+1}^{\infty}\gamma^{s-t-1}\ell(h^*_s) &= (1-\gamma)\sum_{s=t+1}^{\infty}\left(\frac{\gamma^{s-t-1}}{2} - \frac{\gamma^{s-t-1}\left|\lambda_2^{s-t}\right|}{2}\right)\\ &= (1-\gamma)\left(\frac{1}{2(1-\gamma)} - \frac{|\lambda_2|}{2(1-|\lambda_2|\gamma)}\right).\end{aligned}$$

Substituting $\theta_0 = \theta_1 = 0.1$, the discounted risk for $\gamma = 0.9$ is 0.357.

Now consider a learner which computes the MLE of the transition matrix $\Gamma_t^{t'-t}$. It calculates $\mathbb{P}(Y_{t'} \,|\, y_t) = \hat{p}_{t'}$ where $[1-\hat{p}_{t'}, \hat{p}_{t'}] = \Gamma_t^{t'-t}[1-y_t, y_t]^\top$ and uses the hypothesis $h_{t'}(X_{t'}) = \mathbf{1}(\hat{p}_{t'} > 0.5)$ (ties broken randomly). We can see in Fig. 4.1 that this learner converges to the prospective Bayes risk. This example shows that if we model the changes in the data, then we can perform prospective learning. This scenario is closely related to certain continual learning problems (Vogelstein et al., 2020; Ramesh and Chaudhari, 2022).

Scenario 4.4 (Future depends upon the current prediction). Problems where predictions of the learner affect future data are an interesting special case of Theorem 4.3. Prospective learning can also be used to address such scenarios. For $\theta_0, \theta_1 \in [0,1]$, consider a Markov decision process (MDP) $\mathbb{P}(Y_{t+1} = j \,|\, Y_t = j', h_{t+1}(1) = k) = \theta_k$ if $j = j'$ and $1-\theta_k$ otherwise. I.e., the prediction $h_{t+1}(X_{t+1}) = k$ (recall that $X_t = 1$ for all times) is the decision and the MDP remains in the same state with probability θ_k. Prospective Bayes risk for this example is the same as that of the example

in Theorem 4.3. This scenario is closely related to reinforcement learning (Sutton and Barto, 1998).

Calculations. Unlike the preceding scenarios, this is an active prospective learning problem in which the learner influences future realizations through its decisions. Our prospective learner is inspired by reinforcement learning, where the current state is Y_{t-1}, the action is h_t, and the next state is Y_t. The reward at the t^{th} time-step is

$$r(t, h_{t+1}, y_{t+1}) = \mathbf{1}\left\{h_{t+1} = y_{t+1}\right\},$$

as a result of selecting action h_{t+1} given that the previous output was y_t and the next output is y_{t+1}.

The learner estimates the transition matrix corresponding to the MDP for each decision $h_{t+1}(X_{t+1}) \in \{0, 1\}$ using a similar procedure to that of Theorem 4.3:

$$\forall k \in \{0,1\}: \quad \hat{\Gamma}_t(k) = \begin{bmatrix} \hat{\theta}_k & 1-\hat{\theta}_k \\ 1-\hat{\theta}_k & \hat{\theta}_k \end{bmatrix},$$

where $\hat{\theta} \equiv \hat{\theta}_t \in [0,1]^2$ after t time steps. Using this estimate of $\hat{\Gamma}_t$, the learner solves for the value function corresponding to the discounted prospective risk:

$$\hat{Q}_t(y_t, h_{t+1}) = \sum_{y \in \{0,1\}} \underbrace{\mathbb{P}(Y_{t+1} = y \,|\, Y_t = y_t, h_{t+1} = h)}_{=\hat{\Gamma}_t(h_{t+1})_{y_t, y}} \left(r(t, h_{t+1}, y) + \gamma \max_{\bar{h}} \hat{Q}_t(y, \bar{h}) \right).$$

The value function can be solved using value iteration until convergence. For a given $\hat{\Gamma}$, Banach's fixed point theorem guarantees that this procedure converges to the optimal value function in the tabular setting (Watkins and Dayan, 1992).

Once we have the Q-values $\hat{Q}_t$, we can use them to take actions. The optimal action at time t' is $\arg\max_h Q(y_{t'}, h)$. However, unlike reinforcement learning, we do not know $y_{t'}$ for $t' > t$ and must instead make a sequence of decisions using state y_t. The sequence of decisions made by the learner is $\hat{h}_{>t} = (\hat{h}_{t+1}, \dots)$, where

$$\hat{\pi}_{t'}^\top = \pi_t^\top \prod_{s=t+1}^{t'} \hat{\Gamma}_t(\hat{h}_{s-1}),$$

$\pi_t = (1 - y_t, y_t)^\top$, i.e., $\pi_{t'}$ is the estimated distribution over the state at time t', and

$$\hat{h}_{t'+1} = \arg\max_h \pi_{t'}^\top \hat{Q}_t(\cdot, h).$$

In Fig. 4.1, we use $\theta_0 = \theta_1 = 0.1$ and a discount factor $\gamma = 0.9$. This learner approaches the Bayes risk of 0.357 calculated in Theorem 4.3.

4.3. Theoretical foundations of prospective learning

Definition 4.5 (Strong Prospective Learnability). A family of stochastic processes is strongly prospectively learnable, if there exists a learner with the following property: there exists a time $t'(\epsilon, \delta)$ such that for any $\epsilon, \delta > 0$ and for any stochastic process Z from this family, the learner outputs a hypothesis h such that $\mathbb{P}\left[R_t(h) - R_t^* < \epsilon\right] \geq 1 - \delta$, for any $t > t'$.

This definition is similar to the definition of strong learnability in PAC learning with one key difference. Prospective Bayes risk R_t^* depends upon the realization of the stochastic process $z_{\leq t}$ up to time t. In PAC learning, it would only depend upon the true distribution of the data. Not all families of stochastic processes are strongly prospectively learnable. We therefore also define weak learnability with respect to a "chance" learner that predicts $\mathbb{E}[Y]$ and achieves a prospective risk R_t^0.[7]

Definition 4.6 (Weak Prospective Learnability). A family of stochastic processes is weakly prospectively learnable, if there exists a learner with the following property: there exists an $\epsilon > 0$ such that for any $\delta > 0$, there exists a time $t'(\epsilon, \delta)$ such that for any stochastic process Z from this family, $\mathbb{P}\left[R_t^0 - R_t(h) > \epsilon\right] \geq 1 - \delta$, for any $t > t'$.

In PAC learning for binary classification, strong and weak learnability are equivalent (Schapire, 1990) in the distribution agnostic setting, i.e., when strong and weak learnability is defined as the ability of a learner to learn any data distribution. But even in PAC learning, if there are restrictions on the data distribution, strong and weak learnability are not equivalent (Kearns, 1988). This motivates Theorem 4.8 below. Before that, we define a time-agnostic empirical risk minimization (ERM)-based learner. In PAC learning, ERM selects a hypothesis that minimizes the empirical loss on the training data. It outputs a time-agnostic hypothesis, i.e., using data, say, $z_{\leq t}$ standard ERM returns the same predictor for future data from any time $t' > t$. There is a natural application of ERM to prospective learning problems, defined below.

Definition 4.7 (Time-agnostic ERM). Let $\mathcal{H}$ be a hypothesis class that consists of time-agnostic predictors, i.e., $h_t = h_{t'}$ for any $t, t' \in \mathbb{N}$ for all predictors $h \in \mathcal{H}$. Given data $z_{\leq t}$, a learner that returns

$$\hat{h} = \arg\min_{h \in \mathcal{H}} \frac{1}{t} \sum_{s=1}^{t} \ell(s, h_s(x_s), y_s) \tag{4.6}$$

is called a time-agnostic empirical risk minimization (ERM)-based learner.

Time-agnostic ERM in prospective learning may use a time-dependent loss $\ell(s, h_s(x_s), y_s)$ upon the training data. This ERM is not very different from standard ERM in PAC learning (when instantiated with the hypothesis class that consists of sequences of predictors, that we are interested here). If data is IID (Theorem 4.1), then there is no information provided by time in the training samples. But if there are temporal patterns in the data, take Theorems 4.2 and 4.3 or Theorem 4.4 as examples, then time-agnostic ERM as defined here will return predictors that are different than those of standard ERM that uses a time-invariant loss.

Proposition 4.8. There exist stochastic processes for which time-agnostic ERM is not a weak prospective learner. There also exist stochastic processes for which time-agnostic ERM is a weak prospective learner but not a strong one.

[7] We can also define weak learnability with respect to the prospective risk of a particular learner, even one that is not prospective. This may be useful to characterize learning for stochastic processes which do not admit strong learnability.

Proof. Let $\mathcal{X} = \{-1, 1\}$ and $\mathcal{Y} = \{0, 1\}$. Consider two distributions P_1 and P_2 (Fig. 4.4):

$$P_1(X = x) = P_2(X = x) = \frac{1}{2} \quad \forall x,$$
$$P_1(Y = 1 \,|\, X = x) = \begin{cases} \theta & \text{if } x = 1 \\ 1 - \theta & \text{if } x = -1, \end{cases}$$
$$P_2(Y = 1 \,|\, X = x) = \begin{cases} 1 - \theta & \text{if } x = 1 \\ \theta & \text{if } x = -1. \end{cases}$$

In other words, the inputs have the same marginals but the labels are flipped between P_1 and P_2. Consider a stochastic process Z such that $Z_{2t+1} \sim P_1$ and $Z_{2t} \sim P_2$ where $t \in \mathbb{N}$.

Let $\mathcal{G}$ be any hypothesis class and let $\ell(s, \hat{y}, y) = \mathbf{1}(\hat{y} \neq y)$ be the time-invariant zero-one loss. The time-agnostic learner uses a sequence of hypotheses $h \equiv (h_t)$ where $h_t = h_{t'} \, \forall \, t, t' \in \mathbb{N}$ to make predictions at all times. The future loss is

$$\bar{\ell}_t(h, Z) = \lim_{\tau \to \infty} \frac{1}{2\tau} \sum_{s=t+1}^{t+2\tau} \ell(s, h_s(X_s), Y_s) = R_1(h) + R_2(h) = \frac{1}{2},$$

almost surely; here $R_1(h)$ and $R_2(h)$ are risks on data from distributions P_1 and P_2 at odd and even times, respectively. The last equation follows from the fact that $R_1(h) = 1 - R_2(h)$ because the labels are flipped. Prospective Bayes risk is zero if the hypothesis class $\mathcal{G}$ contains the Bayes optimal hypotheses for each of the two distributions. The future loss evaluates to $1/2$ for all realizations and so does the prospective risk. The prospective risk of a hypothesis sequence that makes random predictions (zero or one with equal probability at each instant) is also $1/2$. This stochastic process is not weakly prospective learnable.

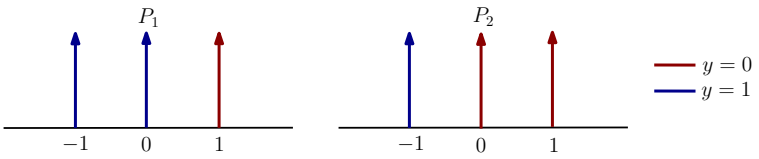

Figure 4.5: A simple stochastic process that is weakly but not strongly prospectively learnable.

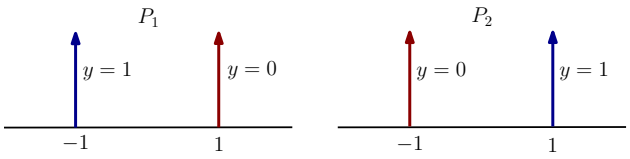

Figure 4.4: A simple stochastic process that is not weakly prospectively learnable.

Now consider the two distributions shown in Fig. 4.5,

$$P_1(X = x) = P_2(X = x) = \frac{1}{3} \quad \forall x,$$
$$P_1(Y = 1 \,|\, X = x) = \begin{cases} \theta & \text{if } x \leq 0 \\ 1 - \theta & \text{if } x = 1, \end{cases}$$
$$P_2(Y = 1 \,|\, X = x) = \begin{cases} 1 - \theta & \text{if } x \geq 0 \\ \theta & \text{if } x = -1. \end{cases}$$

Inputs are supported on the set $\{-1, 0, 1\}$ this time. Again consider a stochastic process Z such that $Z_{2t+1} \sim P_1$ and $Z_{2t} \sim P_2$ for $t \in \mathbb{N}$. For a time-agnostic learner, since its hypothesis h at each time step has to predict incorrectly at $x = 0$, we have $R_1(h) + R_2(h) \geq \frac{1}{3}$. The future loss is

$$\bar{\ell}_t(h, Z) = \lim_{\tau \to \infty} \frac{1}{2\tau} \sum_{s=t+1}^{t+2\tau} \ell(s, h(X_s), Y_s) = R_1(h) + R_2(h) \geq \frac{1}{3}.$$

almost surely. It follows that the prospective risk $R_t(h) \geq \frac{1}{3}$ for any hypothesis. Prospective Bayes risk is again zero and therefore this stochastic process is not strongly prospectively learnable. It is however weakly learnable.

A hypothesis that predicts $\hat{y} = \pm 1$ with equal probability has $R_t^0 = 0.5$. If the data contains samples for $x \in \{-1, 1\}$, ERM will select a hypothesis that minimizes the empirical risk which necessitates that $h(1) = 0$ and $h(-1) = 1$. Therefore $R_1(h) + R_2(h) \leq \frac{1}{3} + \epsilon$, since h predicts correctly at $x = \pm 1$, and incorrectly at $x = 0$ exactly one of the two distributions. The constant ϵ can be chosen to be $\propto t^{-1/2}$ after receiving data from t timesteps. The probability with which we do not get samples at $x = 1$ or at $x = -1$, is 2×3^{-t}. Therefore the probability that $R_1(h) + R_2(h) \leq \frac{1}{3} + \epsilon$ is at least $1 - 3^{-t+1}$ after t time steps. This learner is therefore better than the chance learner whose risk is R_t^0 and it is a weak prospective learner. This shows that there exist stochastic processes that are weakly prospective learnable using time-agnostic ERM but not strongly. □

We do not know yet whether (or when) strong and weak learnability are equivalent for prospective learning.

4.4. Prospective Empirical Risk Minimization (ERM)

In PAC learning, the hypothesis returned by ERM using the training data can predict arbitrarily well (approximate the Bayes risk arbitrarily well with arbitrarily high probability), with a sufficiently large sample size. This statement holds if (a) there exists a hypothesis in the hypothesis class whose risk matches the Bayes risk asymptotically, and (b) if risk on training data converges to that on the test data sufficiently quickly and uniformly over the hypothesis class (Blumer et al., 1989; Alon et al., 1997). Theorem 4.9 is an analogous result for prospective learning.

Theorem 4.9 (Prospective ERM is a strong prospective learner). Consider a finite family of stochastic processes $\mathcal{Z}$. If we have (a) consistency, i.e., there exists an increasing sequence of

hypothesis classes $\mathcal{H}_1 \subseteq \mathcal{H}_2 \subseteq \dots$ with each $\mathcal{H}_t \subseteq (\mathcal{Y}^{\mathcal{X}})^{\mathbb{N}}$ such that $\forall Z \in \mathcal{Z}$,

$$\lim_{t\to\infty} \mathbb{E}\left[\inf_{h\in\mathcal{H}_t} R_t(h) - R_t^*\right] = 0, \tag{4.7}$$

where $h \in \mathcal{H}_t$ is a random variable in $\sigma(Z_{\le t})$, and (b) uniform concentration of the limsup, i.e., $\forall Z \in \mathcal{Z}$,

$$\mathbb{E}\left[\max_{h\in\mathcal{H}_t}\left|\bar{\ell}_t(h,Z) - \max_{u_t\le m\le t}\frac{1}{m}\sum_{s=1}^{m}\ell(s,h_s(x_s),y_s)\right|\right] \le \gamma_t, \tag{4.8}$$

for some $\gamma_t \to 0$ and $u_t \to \infty$ with $u_t \le t$ (all uniform over the family of stochastic processes), then there exists a sequence i_t that depends only on γ_t such that a learner that returns

$$\hat{h} = \operatorname*{arg\,min}_{h\in\mathcal{H}_{i_t}} \max_{u_{i_t}\le m\le t}\frac{1}{m}\sum_{s=1}^{m}\ell(s,h_s(x_s),y_s), \tag{4.9}$$

is a strong prospective learner for this family. We define prospective ERM as the learner that implements Eq. (4.9) given train data $z_{\le t}$.

The proof uses the following auxiliary lemma. Define the shorthand

$$e_m(h) \equiv \frac{1}{m}\sum_{s=1}^{m}\ell(s,h_s(X_s),Y_s).$$

Lemma 4.10. For a stochastic process Z, for an increasing sequence of hypothesis $\mathcal{H}_1 \subseteq \mathcal{H}_2 \subseteq \dots$ such that the consistency condition in Eq. (4.7) is satisfied, for any u_t satisfying $u_t \le t$, $u_t \to \infty$, there exists $h^{(t)} \in \sigma(Z_{\le t})$, a $\mathcal{H}_t$-valued random variable, such that

$$\mathbb{E}\left[\sup_{u_t\le m\le\infty} e_m(h^{(t)}) \,|\, Z_{\le t}\right] - R_t^* \to 0 \tag{4.10}$$

almost surely.

Proof. By Eq. (4.7), there exists $h^{(t)} \in \sigma(Z_{\le t})$, a $\mathcal{H}_t$-valued random variable such that

$$\lim_{t\to\infty}\mathbb{E}\left[R_t(h^{(t)}) - R_t^*\right] = 0$$

Here, $h^{(t)} \in \sigma(Z_{\le t})$ means that $h^{(t)}$ is constant on the set $\{Z_{\le t} = z_{\le t}\}$. By the assumption on $h^{(t)}$, we have

$$R_t(h^{(t)}) \ge \inf_{h\in\mathcal{H}_t} R_t(h) \ge R_t^*$$

almost surely. We can choose a sub-sequence $\{j_k\}$, such that $\mathbb{E}\left[R_{j_k}(h^{(j_k)}) - R_{j_k}^*\right] \le 4^{-k}$. For all random variables $h^{(t)}$ satisfying the above assumption, the bounded convergence theorem implies

that

$$\begin{aligned}\mathbb{E}[R_t(h^{(t)}) - R_t^*] &= \mathbb{E}\left[\mathbb{E}\left[\limsup_{m\to\infty} e_m(h^{(t)}) \mid Z_{\le t}\right] - R_t^*\right] \\ &= \mathbb{E}\left[\mathbb{E}\left[\lim_{i\to\infty} \sup_{u_i \le m \le \infty} e_m(h^{(t)}) \mid Z_{\le t}\right]\right] - \mathbb{E}\left[R_t^*\right] \\ &= \lim_{i\to\infty} \mathbb{E}\left[\mathbb{E}\left[\sup_{u_i \le m \le \infty} e_m(h^{(t)}) \mid Z_{\le t}\right] - R_t^*\right]\end{aligned}$$

In particular, this implies that for any integer k there exists an integer i_k such that

$$\mathbb{E}\left[\mathbb{E}\left[\sup_{u_{i_k} \le m \le \infty} e_m(h^{(j_k)}) \mid Z_{\le j_k}\right] - R_{j_k}^*\right] \le \mathbb{E}\left[R_{j_k}(h^{(j_k)}) - R_{j_k}^*\right] + 4^{-k} \le 2 \times 4^{-k}.$$

By the definition of limsup, we have

$$\mathbb{E}\left[\sup_{u_i \le m \le \infty} e_m(h^{(t)}) \mid Z_{\le t}\right] \ge \mathbb{E}\left[\bar{\ell}_t(h^{(t)}, Z) \mid Z_{\le t}\right] \ge R_t^*$$

and therefore we can use Markov's inequality to get

$$\begin{aligned}&\sum_{k=0}^{\infty} \mathbb{P}\left(\mathbb{E}\left[\sup_{u_{i_k} \le m \le \infty} e_m(h^{(j_k)}) \mid Z_{\le j_k}\right] - R_{j_k}^* > 2^{(1/2)-k}\right) \\ \le &\sum_{k=0}^{\infty} \frac{1}{2^{(1/2)-k}} \mathbb{E}\left[\mathbb{E}\left[\sup_{u_{i_k} \le m \le \infty} e_m(h^{(j_k)}) \mid Z_{\le j_k}\right] - R_{j_k}^*\right] \\ \le &\sum_{k=0}^{\infty} \frac{2}{2^{(1/2)-k}} 4^{-k} < \infty.\end{aligned}$$

Therefore, by Borel-Cantelli lemma, with probability one, there exists a (random) integer k_0 such that for all $k \ge k_0$,

$$\mathbb{E}\left[\sup_{u_{i_k} \le m \le \infty} e_m(h^{(j_k)}) \mid Z_{\le j_k}\right] - R_{j_k}^* \le 2^{(1/2)-k}$$

We will now scale i_k, j_k by some integer k_t such that $i_{k_t}, j_{k_t} \le t$. This ensures that $u_{i_{k_t}} \le u_t$ and $\mathcal{H}_{i_{k_t}} \subseteq \mathcal{H}_t$. This scaling is necessary to relate the "empirical estimate" of the lim sup of $\hat{h}$ to the actual lim sup of $h^{(t)}$. To that end, define

$$k_t = \max\{k \in \mathbb{N} \cup \{0\} : \max\{i_k, j_k\} \le t\}.$$

Since $i_k \le \infty$ and $j_k \le \infty$, we also have $\lim_{t\to\infty} k_t = \infty$. Let $\alpha_t = 2^{(1/2)-k_t}$ and notice that $\alpha_t \to 0$. We can construct an integer-valued random variable t_0 such that for all $t \ge t_0$, we have $k_t \ge k_0$,

and therefore

$$\mathbb{E}\left[\sup_{u_{i_{k_t}} \le m \le \infty} e_m(h^{(j_{k_t})}) \,|\, Z_{\le j_{k_t}}\right] - R^*_{j_{k_t}} \le \alpha_t.$$

Now choose $h^{(t)} = h^{(j_{k_t})}$ for every $t \in \mathbb{N}$. Since $j_{k_t} \le t$, we have $\mathcal{H}_{j_{k_t}} \subseteq \mathcal{H}_t$ and $\sigma(Z_{\le j_{k_t}}) \subseteq \sigma(Z_{\le t})$. This implies that $h^{(j_{k_t})}$ is an $\mathcal{H}_t$-valued random variable and $h^{(j_{k_t})} \in \sigma(Z_{\le t})$. Also, since $i_{k_t} \le t$ and $\{u_t\}_{t=1}^{\infty}$ is non-decreasing, we have $u_{i_{k_t}} \le u_t$ for all $t \in \mathbb{N}$. Hence, with probability one, $\forall t \ge t_0$,

$$\mathbb{E}\left[\sup_{u_t \le m \le \infty} e_m(h^{(t)}) \,|\, Z_{\le j_{k_t}}\right] - R^*_{j_{k_t}} \le \mathbb{E}\left[\sup_{u_{i_{k_t}} \le m \le \infty} e_m(h^{(t)}) \,|\, Z_{\le j_{k_t}}\right] - R^*_{j_{k_t}} \le \alpha_t$$

Since $\alpha_t \to 0$, we have

$$\mathbb{E}\left[\sup_{u_t \le m \le \infty} e_m(h^{(t)}) \,|\, Z_{\le j_{k_t}}\right] - R^*_{j_{k_t}} \to 0 \text{ a.s.}$$

Again using the bounded convergence theorem,

$$\mathbb{E}\left[\sup_{u_t \le m \le \infty} e_m(h^{(t)}) \,|\, Z_{\le t}\right] - R^*_{t} \to 0 \text{ a.s.}$$

□

Proof of Theorem 4.9. We first show that for each $Z \in \mathcal{Z}$, if Eqs. (4.7) and (4.8) holds, then the risk of estimator in Eq. (4.9) converges in probability to the Bayes optimal, i.e.

$$\mathbb{P}\left(\left|R_t(\hat{h}^{(t)}) - R^*_t\right| < \epsilon\right) \to 1.$$

By taking

$$t = \max\left\{t : \mathbb{P}\left(\left|R_t(\hat{h}^{(t)}) - R^*_t\right| < \epsilon\right) \ge 1 - \delta \quad \forall t' \ge t, Z \in \mathcal{Z}\right\}$$

we can trivally show that the strong prospecitve learnability of estimator in Eq. (4.9) holds over the family $\mathcal{Z}$.

We now exploit the convergence of the empirical lim sup to the true lim sup. We construct a subsequence of integers i_t such that γ_{i_t} in Eq. (4.8) decays exponentially. Markov's inequality implies that

$$\begin{aligned}
&\sum_{t=0}^{\infty} \mathbb{P}\left(\max_{h \in \mathcal{H}_{i_t}} \left|\bar{\ell}_t(h, Z) - \max_{u_{i_t} \le m \le i_t} e_m(h)\right| > \sqrt{\gamma_{i_t}}\right) \\
\le &\sum_{t=0}^{\infty} \frac{1}{\sqrt{\gamma_{i_t}}} \mathbb{E}\left[\max_{h \in \mathcal{H}_{i_t}} \left|\bar{\ell}_t(h, Z) - \max_{u_{i_t} \le m \le i_t} e_m(h)\right|\right] \le \sum_{t=0}^{\infty} \sqrt{\gamma_{i_t}} < \infty.
\end{aligned}$$

By the Borel-Cantelli lemma, with probability one, there exists $t_1 \in \mathbb{N}$, random, and $t_1 \in \mathcal{F}$, such

that $\forall t \geq t_1$,

$$\max_{h \in \mathcal{H}_{i_t}} \left| \bar{\ell}_t(h, Z) - \max_{u_{i_t} \leq m \leq i_t} e_m(h) \right| \leq \sqrt{\gamma_{i_t}}.$$

Let $j_t = \max\{t' : i_{t'} \leq t\}$, then we have $i_{j_t} \to \infty$. Since $i_{j_t} \leq t$, we have

$$\bar{\ell}_t(h, Z) - \max_{u_{i_{j_t}} \leq m \leq t} e_m(h) \leq \bar{\ell}_t(h, Z) - \max_{u_{i_{j_t}} \leq m \leq i_{j_t}} e_m(h).$$

Construct a random variable t_2 such that $\forall t \geq t_2$, we have $j_t \geq t_1$. Hence, we have for all $t \geq t_2$,

$$\begin{aligned}
&\max_{h \in \mathcal{H}_{i_{j_t}}} \left\{ \bar{\ell}_t(h, Z) - \max_{u_{i_{j_t}} \leq m \leq t} e_m(h) \right\} \\
&\leq \max_{h \in \mathcal{H}_{i_{j_t}}} \left| \bar{\ell}_t(h, Z) - \max_{u_{i_{j_t}} \leq m \leq i_{j_t}} e_m(h) \right| \\
&\leq \sqrt{\gamma_{i_{j_t}}}.
\end{aligned}$$

Let $i_t = i_{j_t}$ and notice that since we have $i_t \to \infty$ we also have $i_t \leq t$. This gives a sub-sequence that depends only on γ_t. Then, with probability one, $\forall t \geq t_2$

$$\max_{h \in \mathcal{H}_{i_t}} \left\{ \bar{\ell}_t(h, Z) - \max_{u_{i_t} \leq m \leq t} e_m(h) \right\} \leq \sqrt{\gamma_{i_t}}.$$

Let $\{h^{(t)}\}_{t=1}^{\infty}$, where $h^{(t)} \in \sigma(Z_{\leq t})$ is a $\mathcal{H}_{i_t}$-valued random variable chosen as in Lemma 4.10 with $\mathcal{H}_t$ chosen to be $\mathcal{H}_{i_t}$ and u_t chosen to be u_{i_t}. Since $\hat{h}^{(t)} \in \sigma(Z_{\leq t})$, with probability one, for $t \geq t_2$,

$$\begin{aligned}
\bar{\ell}_t(\hat{h}^{(t)}, Z) &\leq \max_{u_{i_t} \leq m \leq t} e_m(\hat{h}^{(t)}) + \sqrt{\gamma_{i_t}} \\
&\leq \max_{u_{i_t} \leq m \leq t} e_m(h^{(t)}) + \sqrt{\gamma_{i_t}} \\
&\leq \sup_{u_{i_t} \leq m \leq \infty} e_m(h^{(t)}) + \sqrt{\gamma_{i_t}}.
\end{aligned}$$

Hence,

$$\mathbb{E}\left[\bar{\ell}_t(\hat{h}^{(t)}, Z) \,|\, Z_{\leq t} \right] - \mathbb{E}\left[\sup_{u_{i_t} \leq m \leq \infty} e_m(h^{(t)}) \,|\, Z_{\leq t} \right] \to 0$$

almost surely. By Lemma 4.10, we have,

$$\mathbb{E}\left[\bar{\ell}(\hat{h}^{(t)}, Z) \,|\, Z_{\leq t} \right] - R_t^* \to 0$$

almost surely. Hence, by the bounded convergence theorem,

$$0 = \mathbb{E}\left[\lim_{t \to \infty} \left(\mathbb{E}\left[\bar{\ell}(\hat{h}^{(t)}, Z) \,|\, Z_{\leq t} \right] - R_t^* \right) \right] = \lim_{t \to \infty} \mathbb{E}\left[\mathbb{E}\left[\bar{\ell}(\hat{h}^{(t)}, Z) \,|\, Z_{\leq t} \right] - R_t^* \right]$$

which implies that

$$\mathbb{P}\left(\left|R_t(\hat{h}^{(t)}) - R_t^*\right| \geq \delta\right) \leq \frac{1}{\delta}\,\mathbb{E}\left[\mathbb{E}\left[\bar{\ell}(\hat{h}^{(t)}, Z) \,|\, Z_{\leq t}\right] - R_t^*\right] \to 0.$$

We have therefore proved that $\hat{h}^{(t)}$ is a strong prospective learner. □

The proof above builds upon the work of Hanneke (2021a). The first condition, Eq. (4.7), is analogous to the consistency condition in PAC learning. In simpler words, it states that the Bayes risk can be approximated well using the chosen sequence of hypothesis classes $\{\mathcal{H}_t\}_{t=1}^{\infty}$. The second condition, Eq. (4.8), is analogous to concentration of measure in PAC learning, it requires that the limsup in Eq. (4.1) is close to an empirical estimate of the limsup (the second term inside the absolute value in Eq. (4.8)). At each time t, prospective ERM in Eq. (4.9) selects the best hypothesis $\hat{h} \in \mathcal{H}_t$[8] for future times $t' > t$, that minimizes an empirical estimate of the limsup using the training data $z_{\leq t}$. Prospective ERM can exploit the difference between the latest datum in the training set with time t and the time for which it makes predictions t' by selecting specific sequences inside the hypothesis class $\mathcal{H}_t$. For example, in Theorem 4.2 it can select sequences where alternating elements can be used to predict on data from even and odd times.

Remark 4.11 (How to implement prospective ERM?). An implementation of prospective ERM is therefore not much different than an implementation of standard ERM, except that there are two inputs: time s and the datum x_s. Suppose we use a hypothesis class where each predictor is a neural network, this could be a multi-layer perceptron or a convolutional neural network. The training set $z_{\leq t}$ consists of inputs x_s along with corresponding time instants s and outputs y_s. To implement prospective ERM, we modify the network to take (s, x_s) as input (using any encoding of time, we discuss one in Sec. 4.5) and train the network to predict the label y_s. In Eq. (4.9) we can set $u_{i_t} \equiv t$, doing so only changes the sample complexity. At inference time, this network is given the input $(t', x_{t'})$ to obtain the prediction $y_{t'}$. Note that if prospective ERM is implemented in this fashion, the learner need not explicitly calculate the infinite sequence of predictors.[9]

Corollary 4.12. There exist stochastic processes for which time-agnostic ERM is not a strong prospective learner, but prospective ERM is a strong learner.

Remark 4.13 (Why we need an increasing sequence of hypothesis classes $\mathcal{H}_1 \subseteq \mathcal{H}_2 \ldots$). We could have chosen $\mathcal{H}_t = \mathcal{H}_{t'}$ for all $t, t' \in \mathbb{N}$ to set up Theorem 4.9. But since the learner does not have a lot of data at early times, it should use a small hypothesis class. Just like PAC learning, the sequence $(\gamma_t)_{t\in\mathbb{N}}$ in Eq. (4.8) determines the convergence rate of a prospective learner. Therefore, using a monotonically increasing sequence of hypothesis classes is useful to ensure a good sample complexity.

Theorem 4.14. Consider a finite family of stochastic processes $\mathcal{Z}$. If there exists a countable hypothesis class $\mathcal{H}$ such that

$$\lim_{t\to\infty} \mathbb{E}\left[\inf_{h\in\mathcal{H}} R_t(h) - R_t^*\right] = 0, \tag{4.11}$$

[8] Note that this hypothesis class has infinite sequences, $\mathcal{H}_t \subseteq (\mathcal{Y}^{\mathcal{X}})^{\mathbb{N}}$.

[9] Hereafter, when we write ERM in empirical studies, we will mean a learner that approximates ERM via stochastic gradient descent.

for any stochastic process $Z \in \mathcal{Z}$, where $h \in \mathcal{H}$ is a random variable in $\sigma(Z_{\le t})$, then there exist $\mathcal{H}_t$, u_t, and γ_t such that the two conditions of Theorem 4.9 are satisfied for this family.

Proof. This construction follows closely with Hanneke (2021b, Section 4), and we omit some details here. For finite class $\mathcal{H}$ of sequence of hypothesis, we give a possible choice of u_t with

$$\lim_{t\to\infty} \mathbb{E}\left[\sup_{t'\ge t} \max_{h\in\mathcal{H}} \left|\bar{\ell}_t(h, Z) - \max_{u_t \le m \le t} e_m(h)\right|\right] = 0. \tag{4.12}$$

for all process Z in the finite family $\mathcal{Z}$. For a data sequence $\mathbf{z} = \{z_s = (x_s, y_s)\}_{s=0}^{\infty}$ and a hypothesis sequence $h \in \mathcal{H}$, we define

$$t_u^h(\mathbf{z}) = \min\left\{t \in \mathbb{N} : t \ge u, \forall t' \ge t \sup_{u\le m\le\infty} e_m(h) \le \max_{u\le m\le t'} e_m(h) + 2^{-u}\right\},$$

and

$$\begin{aligned} u_t^h(\mathbf{z}) &= \max\{u \in \{1, \dots, t\} : \ t \ge t_u^h(\mathbf{z})\}, \\ u_t^{\mathcal{H}}(\mathbf{z}) &= \min_{h\in\mathcal{H}} u_t^h(\mathbf{z}); \end{aligned}$$

note that $\mathcal{H}$ is finite and therefore the minimum exists. For the stochastic process Z, let

$$\begin{aligned} u_t^{\mathcal{H}}(\delta, Z) &= \max\left\{u \in \{1, \dots, t\} : \mathbb{P}_{\mathbf{z}\sim Z}(u_t^{\mathcal{H}}(\mathbf{z}) \ge u) = 1 - \delta\right\}, \\ u_t(Z) &= \max\left\{s \in \mathbb{N} \cup \{0\} : u_t^{\mathcal{H}}(2^{-u}, Z) \ge u\right\}, \\ u_t &= \min\left\{u_t(Z) : Z \in \mathcal{Z}\right\}. \end{aligned}$$

Since $\mathcal{Z}$ is finite, we have $u_t \to \infty$. And we also have $\forall Z \in \mathcal{Z}$,

$$\mathbb{P}\left(u_t^{\mathcal{H}}(Z) \ge u_t\right) \ge 1 - 2^{-u_t}.$$

By Borel-Cantelli Lemma, this construction gives a sequence u_t s.t. Eq. (4.12) holds, i.e. $\forall Z \in \mathcal{Z}$.

Suppose $\{\mathcal{H}_t\}_{t=1}^{\infty}$ is a sequence of non-empty finite sets of hypothesis sequences, and $\{\gamma_t\}_{t=1}^{\infty}$ is a sequence in $(0, \infty)$ with $\gamma_1 \ge 1$. Then for any finite family $\mathcal{Z}$, we can extend this construction to get a choice of u_t, $\mathcal{H}_t$ and γ_t such that Eq. (4.8) holds

$$\mathbb{E}\left[\max_{h\in\mathcal{H}_t} \left|\bar{\ell}_t(h, Z) - \max_{u_t \le m \le t} e_m(h)\right|\right] \le \gamma_t.$$

Since we have a u_t for a given hypothesis class $\mathcal{H}$, for each $i \in \mathbb{N}$, we can construct a sequence $\{u_{i,t}\}_{t=1}^{\infty}$ such that $\lim_{t\to\infty} u_{i,t} = \infty$, $u_{i,t} < t$ and $\forall Z \in \mathcal{Z}$,

$$\lim_{t\to\infty} \mathbb{E}\left[\sup_{t'\ge t} \max_{h\in\mathcal{H}_i} \left|\bar{\ell}_t(h, Z) - \max_{u_{i,t} \le m < n'} e_m(h)\right|\right] = 0.$$

Now let

$$j_t = \max\left\{ i \in \{1,\dots,t\} : \forall i' \le i, \sup_{t'' \ge t} \mathbb{E}\left[\sup_{t' \ge t''} \max_{h \in \mathcal{H}_{i'}} \left| \bar{\ell}_t(h, Z) - \max_{u_{i'}, t'' \le m \le t'} e_m(h) \right| \right] \le \gamma_{i'} \forall Z \in \mathcal{Z} \right\}$$

and

$$t_i = \min\left\{ t : j_t \ge i, u_{i,t} > u_{i-1,n_{i-1}} \right\}.$$

If $i_t = \max\{i : t_i < t\}$, then we have $i_t \to \infty$, and for $u_i = u_{i,t_i}$ we have

$$\mathbb{E}\left[\max_{h \in \mathcal{H}_{i_t}} \left| \bar{\ell}_t(h, Z) - \max_{u_{i_t} \le m \le t} e_m(h) \right| \right] \le \gamma_{i_t}.$$

Since $\mathcal{H}$ is countable, we can choose a sequence of finite hypothesis classes $\mathcal{H}_t$ such that $\cup_{t \in \mathbb{N}} \mathcal{H}_t = \mathcal{H}$. By choosing $u_t = u_{i_t}$, with $\mathcal{H}_t = \mathcal{H}_{i_t}$, and $\gamma_t = \gamma_{i_t}$, we now have a possible choice for u_t, $\mathcal{H}_t$ and γ_t in Theorem 4.9. □

This theorem provides a concrete example for which the assumptions of Theorem 4.9 are satisfied. In PAC learning, one first proves uniform convergence for a finite hypothesis class. This can then be used to, say, calculate the sample complexity of ERM, or extended to infinite hypothesis classes using constructions such as VC-dimension and covering numbers (Vapnik, 1998). The above theorem should be understood in the same spirit. It is a step towards characterizing the sample complexity of prospective learning.

4.4.1. Prospective ERM for Discounted Future Losses

The results above concern the averaged future loss in Eq. (4.1). As discussed in Theorem 4.3, prospecting meaningfully for some stochastic processes may instead require a discounted future loss, such as the one in Eq. (4.5). We next extend the guarantee in Theorem 4.9 to this setting. Let

$$\ell_t^{(\tau)}(h, Z; \gamma) = \left(\frac{1-\gamma}{1-\gamma^{\tau+1}} \right) \sum_{s=t+1}^{t+\tau} \gamma^{s-t-1} \ell(h_s(x_s), y_s),$$

where $\ell : \mathcal{Y} \times \mathcal{Y} \mapsto [0,1]$ is a bounded loss function and $0 < \gamma < 1$ is a constant. In general, we can use a probability measure $\mu^{(\tau)}$ supported on integers $\{1,\dots,\tau\}$ to write the loss as

$$\ell_t^{(\tau)}(h, Z; \mu^{(\tau)}) = \sum_{s=t+1}^{t+\tau} \mu_{s-t}^{(\tau)} \ell(h_s(x_s), y_s). \tag{4.13}$$

The averaged loss in Eq. (4.1) corresponds to $\mu_s^{(\tau)} = 1/\tau$ for all s. The discounted loss above corresponds to $\mu_s^{(\tau)} = \left(\frac{1-\gamma}{1-\gamma^{\tau+1}} \right) \gamma^{s-1}$.

In prospective learning, we are interested in the case when $\tau \to \infty$ and therefore define

$$\bar{\ell}_t(h, Z; \mu) = \limsup_{\tau \to \infty} \ell_t^{(\tau)}(h, Z; \mu^{(\tau)}),$$

where μ denotes the collection $\{\mu^{(\tau)}\}_{\tau=1}^{\infty}$. We can define $R_t(h; \mu)$ and $R_t^*(\mu)$ for this discounted

loss using expressions analogous to those in Eqs. (4.2) and (4.3). For clarity, we use the notation $R_t(h, 1/\tau) \equiv R_t(h)$ and $R_t^*(1/\tau) \equiv R_t^*$ for the prospective risk and prospective Bayes risk corresponding to the averaged loss, where $\mu_s^{(\tau)} = 1/\tau$.

Corollary 4.15 (Prospective ERM is a strong learner with discounted losses). Let the assumptions of Theorem 4.9 hold. If there exists a constant $c > 0$ such that $\forall Z \in \mathcal{Z}$,

$$R_t(h; \mu) - R_t^*(\mu) \leq c\,(R_t(h, 1/\tau) - R_t^*(1/\tau)) \quad \forall t \in \mathbb{N}, h \in \mathcal{H}_t, \tag{4.14}$$

i.e., if the gap in the risk for the discounted loss is dominated uniformly by the gap in the risk for the averaged loss over all realizations of the stochastic process, then prospective ERM implemented in Eq. (4.9) with the averaged loss is a strong prospective learner: its discounted risk $R_t(\hat{h}, \mu)$ converges to the discounted Bayes risk $R_t^*(\mu)$.

Proof. The assumption in Eq. (4.14) ensures that

$$\mathbb{P}\left(\left|R_t(\hat{h}, \mu) - R_t^*(\mu)\right| \geq c\epsilon\right) \leq \mathbb{P}\left(\left|R_t(\hat{h}, 1/\tau) - R_t^*(1/\tau)\right| \geq \epsilon\right)$$

for any ϵ and t. The right-hand side is shown to converge to zero in the proof of Theorem 4.9, and therefore the left-hand side also converges to zero. □

A result corresponding to Theorem 4.14 also holds for the discounted future loss. Moreover, Theorems 4.9 and 4.14 hold in the slightly more general setting in which the measure $\mu^{(\tau)}(\omega)$ is a random variable that depends upon the realization of the stochastic process $\omega \in \Omega$.

We next illustrate these guarantees for periodic processes and hidden Markov models.

4.4.2. Examples of Prospective ERM

Periodic processes. Suppose we have a stochastic process such that $Z_t \sim p_{(t \bmod T)}$ for some known period T, i.e., data is independent across time but not identically distributed, and the loss function $\ell(t, \cdot, \cdot)$ is time-invariant. Theorem 4.2 is a special case with $T = 2$. Assume that we can find a countable hypothesis class $\mathcal{G}$ that contains the Bayes plug-in estimator for each p_t with $t \in \{1, \ldots, T\}$. Then $\mathcal{H}_T = \{h : h_{t+T} = h_t \text{ and } h_t \in \mathcal{G}\ \forall t\}$ satisfies Eq. (4.11), and it is also countable. This implies consistency and uniform concentration of the limsup for sequences in some subclasses $\{\mathcal{H}_t\}_{t=1}^{\infty}$ that expand to $\mathcal{H}_T$. Note that even if we do not know the period, we can still implement prospective ERM using the hypothesis class $\cup_{T \in \mathbb{N}} \mathcal{H}_T$; this is a countable set. Prospective ERM is therefore a strong prospective learner if the period T is bounded.

Remark 4.16 (Implementing prospective ERM for periodic processes). If $\mathcal{G}$ has a finite VC-dimension, choosing $\mathcal{H}_t = \mathcal{H}_T$ for any $t > T$ as the increasing sequence of hypothesis classes in Theorem 4.9 guarantees the existence of $\lim_{m \to \infty} \frac{1}{m} \sum_{s=1}^{m} \ell(s, h_s(x_s), y_s)$. We can therefore choose $u_t = t$ in Eq. (4.8) and thereby select

$$\hat{h} = \operatorname*{arg\,min}_{h \in \mathcal{H}_t} \frac{1}{t} \sum_{s=1}^{t} \ell(s, h_s(x_s), y_s)$$

in Eq. (4.9). For $\mathcal{H}_t = \mathcal{H}_T$ selected above for the periodic process, this is identical to Eq. (4.6). In

other words, implementing prospective ERM for a periodic process boils down to solving T different time-agnostic ERM problems, each using data $\{z_{sT+k}\}_{s=0}^{\infty}$, $k \in \{1, ..., T\}$. Observe that this is precisely the prospective learner used for the example in Theorem 4.2 and Fig. 4.1.

Remark 4.17 (Sample complexity of prospective ERM for a periodic process). We can calculate the sample complexity by exploiting the relatedness of the different distributions in the periodic process. First assume $t > T$, i.e., at least one sample from each distribution is available. We again pick $\mathcal{H}_t = \mathcal{H}_T$ for all $t > T$ and assume that $\hat{h}_t \in \mathcal{G}$ for all times t. Let $C \equiv C(\epsilon/16, \mathcal{G}^T)$ denote the covering number of a hypothesis class of T-length sequences of hypotheses $\mathcal{G}^T = \{(h, \dots, h) : h \in \mathcal{G}\}$ using balls of radius $\epsilon/16$ with respect to loss ℓ. Then, using Baxter (2000b, Theorem 4), we can show that if

$$t \geq \max\left\{\frac{64}{\epsilon^2}\log\frac{4C(\epsilon, \mathcal{G}^T)}{\delta}, \frac{16T}{\epsilon^2}\right\}, \tag{4.15}$$

then for prospective ERM in Eq. (4.9), we have

$$\mathbb{E}\left[R_t(\hat{h})\right] \leq \lim_{t\to\infty} \mathbb{E}\left[\inf_{h\in\mathcal{H}_T} R_t(h)\right] + 2\epsilon$$

with probability at least $1 - \delta$. The sample complexity in Eq. (4.15) is dominated by the first term in the curly brackets; Baxter (2000b, Lemma 5) shows that $C(\epsilon, \mathcal{G}^T) \leq (C(\epsilon, \mathcal{G}))^T$. The sample complexity of prospective ERM therefore grows at most linearly with the period T, as one would expect.

Remark 4.18 (Exact sample complexity for periodic binary classification with one-dimensional Gaussian inputs). Let the period of the stochastic process be $T = 2$ with inputs $X_t \in \mathbb{R}$ and outputs $Y_t \in \{-1, 1\}$. Suppose $Y_t \sim$ Bernoulli(0.5). The distribution $\mathbb{P}(X_t \,|\, Y_t = y)$ is a Gaussian with mean $y\mu + \Delta(t \bmod T)$ and variance σ^2. In words, for even times t, the means of the Gaussians are shifted to the right by Δ. Consider the time-invariant squared error loss $\ell(s, \hat{y}, y) = (\hat{y} - y)^2$. Choose $\mathcal{G} = \{\mathbf{1}_A : A \in \{(-\infty, c), (c, \infty) : c \in \mathbb{R}\}\}$ to be the set of predictors for Fisher's linear discriminant (FLD); the prospective learner selects each element of its hypothesis from $\mathcal{G}$. The calculations in De Silva et al. (2023) for FLD can be used to show that if $\mathcal{H}_t = \{(h, h, \dots) : h \in \mathcal{G}\}$ for all times t, then a time-agnostic ERM has risk

$$\mathbb{E}\left[R_t(\hat{h})\right] \to 1/2\left[\Phi\left(\Delta/(2\sigma) - \mu/\sigma\right) + \Phi\left(-\Delta/(2\sigma) - \mu/\sigma\right)\right],$$

where Φ is the Gauss error function. But if $\mathcal{H}_t = \{(h_1, h_2, h_1, h_2, \dots) : h_1, h_2 \in \mathcal{G}\}$ for all times t, then prospective ERM can achieve Bayes risk

$$\mathbb{E}\left[R_t(\hat{h})\right] = \Phi\left(-t\mu/(2\sigma)(t/2(t/2+1))^{-1/2}\right) \to \Phi(-\mu/\sigma) = \mathbb{E}\left[R_t^*\right].$$

Hidden Markov models. Suppose Z is sampled from an HMM whose hidden states evolve according to a k^{th}-order time-homogeneous Markov process with a finite state space. Select a

hypothesis class $\mathcal{H}_k$ that consists of sequences $h \in \mathcal{H}_k$ such that each h satisfies

$$\forall t \in \mathbb{N} : h_t \in \mathcal{G} \text{ and,}$$
$$\forall t_1, t_2 \in \mathbb{N} : \text{if } h_{t_1+s} = h_{t_2+s} \; \forall s \in \{1 \dots, k\}, \text{then } h_{t_1+k+1} = h_{t_2+k+1}.$$

This hypothesis class contains sequences of predictors that depend only on the past k predictors. If we assume, as above, that $\mathcal{G}$ is countable, then so is $\mathcal{H}_k$. It also satisfies Eq. (4.11) because of the k^{th}-order Markov property. We can therefore implement prospective ERM using $\mathcal{H}_k$ as the hypothesis class. Observe that when the Markov process underlying the HMM is deterministic, this example models the output from an autoregressive language model that uses greedy decoding. The length of the context window is k, the hidden state of the HMM is the logit at each step, the next hidden state is a deterministic function of the previous k hidden states, and the output of the HMM Z_t is the next token. Our theory therefore shows that the output of such a model is prospectively learnable if the learner has access to the sequence of tokens.

4.5. Experimental Validation

This section demonstrates that we can implement prospective ERM on prospective learning problems constructed on synthetic data, MNIST and CIFAR-10. In practice, prospective ERM may approximately achieve the guarantees of Theorem 4.9. We will focus on the distribution changing, independently or not (Theorems 4.2 and 4.3). Recall that Theorem 4.1 is the same as the IID setting used in standard supervised learning problems. Theorem 4.4 is more involved and, therefore, we leave more elaborate experiments for future work. We discuss experiments that check whether large language models can do prospective learning in Sec. 4.5.4.

4.5.1. Experimental Setup

Learners and hypothesis classes. Task-agnostic online continual learning methods are the closest algorithms in the literature that can address situations when data evolves over time. We use the following three methods.

(i) **Follow-the-Leader** minimizes the empirical risk calculated on all past data and is a no-regret algorithm (Cesa-Bianchi and Lugosi, 2006). We note that while this is a popular online learning algorithm, we do not implement the algorithm in an online fashion.

(ii) **Online SGD** fine-tunes the network using new data in an online fashion. At every time-step, weights of the network are updated once using the last eight samples.

(iii) **Bayesian gradient descent** (Zeno et al., 2018) is an online continual learning algorithm designed to address situations where the identity of the task is not known during both training and testing, i.e., it implements continual learning without knowledge of task boundaries.

These three methods are not explicitly designed for prospective learning but they are designed to address the changing data distribution t.[10] We calculate the prospective risk of the predictor returned by these methods; note that they do not output a time-varying predictor and consequently, these methods output a time-agnostic hypothesis. As a result, when we evaluate the prospective

[10] There are many algorithms in the existing literature that the reader may think of as reasonable baselines. We have chosen a representative and relevant set here, rather than an exhaustive one. For example, online meta-learning approaches are close to online-SGD; since the learner fine-tunes on the most recent data. Algorithms in the literature on time-series (i) focus on predicting future data, say, $Y_{t'}$ given past data $y_{\leq t}$ without taking covariates $X_{t'}$ or some exogenous variables $X_{\leq t}$ into account, (ii) can usually only make predictions for a pre-specified future context window (Lim et al., 2021), and (iii) work for low-dimensional signals (unlike images).

risk of these methods, we use the same hypothesis for all future time. For all three methods, we use a multi-layer perceptron (MLP) for synthetic data and MNIST, and a convolutional neural network (CNN) for CIFAR-10.

Model architectures. For the time-agnostic retrospective algorithms, we use an MLP with two hidden layers of 256 units for the synthetic and MNIST tasks. For CIFAR-10, we use a small convolutional network with 0.12M parameters. It comprises three convolutional layers with kernel size 3 and 80 filters, interleaved with max-pooling, ReLU, and batch-normalization layers, followed by a fully connected classifier.

For **prospective ERM**, the sequence of predictors is built by incorporating time as an additional input to an MLP or CNN. For the MLP, we concatenate each input with its time embedding $\varphi(t)$. For the CNN, we add the time embedding to the output of the convolutional layers instead of concatenating it with the input image, as illustrated in Fig. 4.6. This construction allows the network to vary its hypothesis over time without explicitly maintaining the infinite sequence of predictors $h \equiv (h_1, \dots)$.

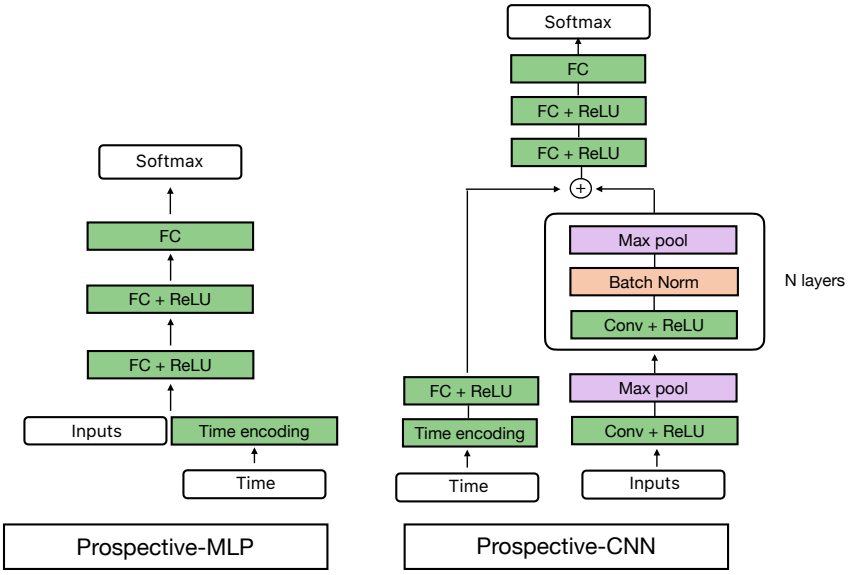

Figure 4.6: Schematic illustration of the prospective MLP and prospective CNN architectures.

Time embedding. We use an embedding function $\varphi : \mathbb{R} \to \mathbb{R}^d$ that maps the absolute time t to

$$\varphi(t) = (\sin(\omega_1 t), \dots, \sin(\omega_{d/2} t), \cos(\omega_1 t), \dots, \cos(\omega_{d/2} t)),$$

where $\omega_i = \pi/i$ for $i = 1, \dots, d/2$. We use $d = 50$ in our experiments. A predictor $h_t(\cdot)$ takes the time embedding $\varphi(t)$ and the input x_t to predict y_t.

This construction is related to the positional encoding of Vaswani et al. (2017), which uses frequencies $\omega_i = 1/10000^{2i/d}$. We instead encode absolute time and use the angular frequencies $\omega_i = \pi/i$. As shown in Fig. 4.7, the Transformer frequencies produce slowly changing features, with many dimensions nearly constant over the relevant time scale. In our experiments, MLPs and CNNs

using these frequencies performed poorly on the MNIST tasks in Scenarios 2 and 3. The figure uses $d = 128$ to illustrate the difference between the two choices of frequencies.

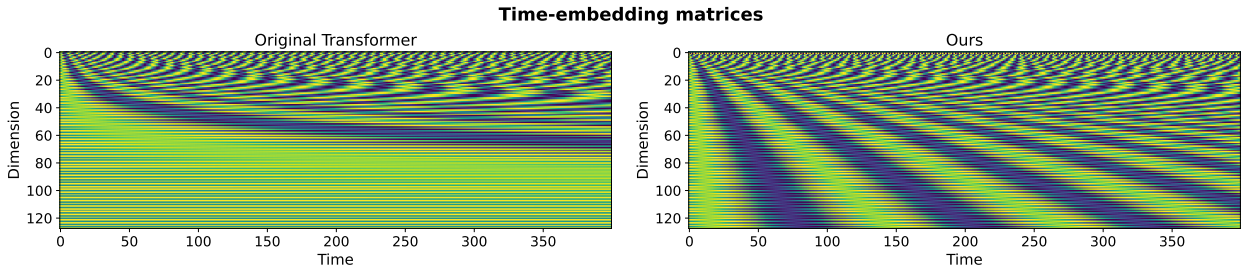

Figure 4.7: Time embeddings computed using the frequencies of Vaswani et al. (2017) *(left)* and the frequencies used in our experiments *(right)*.

Effect of the time-embedding placement. We also tried concatenating the time embedding with the input to the CNN, but this variant performed poorly in Scenarios 2 and 3. As Fig. 4.8 shows, adding the time embedding after the convolutional layers yields substantially lower prospective risk on CIFAR-10.

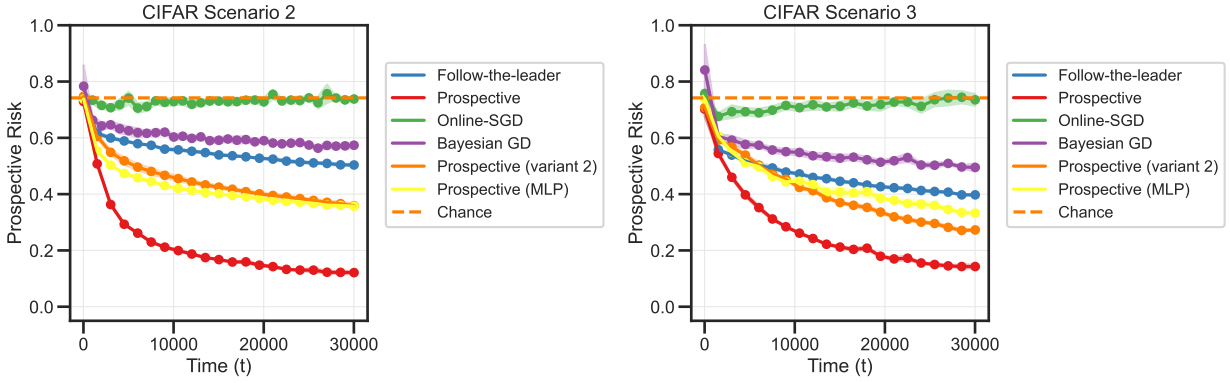

Figure 4.8: Prospective risk of the CNN architecture on CIFAR-10 for Scenarios 2 and 3. Performance is significantly worse when the time embedding is concatenated with the input image (variant 2).

Training protocol. Each learner receives a length-t sequence $z_{\leq t}$ drawn from the stochastic process and makes predictions at future times $t' > t$ up to a fixed horizon T. No samples after time t are used to update the learner. Given $z_{\leq t}$, a time-aware hypothesis class minimizes the empirical prospective risk

$$\hat{\ell}_t(h, Z) = \frac{1}{t} \sum_{s=1}^{t} \ell(s, h_s(x_s), y_s).$$

For an MLP or CNN, h_s corresponds to a network that takes time s as an additional input.

We use the zero-one error $\mathbf{1}\{\hat{y} \neq y\}$ to calculate prospective risk for all problems; all learners are trained using a standard surrogate of this objective, the cross-entropy loss. All networks are trained using stochastic gradient descent with Nesterov momentum and a cosine-annealed learning rate. We use an initial learning rate of 0.1 for the synthetic tasks and 0.01 for MNIST and CIFAR-10, with weight decay 10^{-5}. MNIST and CIFAR-10 images are normalized to have mean 0.5 and standard

deviation 0.25. Models are trained for 100 epochs, well beyond the point at which they achieve training accuracy 1. Except for online SGD and Bayesian gradient descent, learners corresponding to different times are trained completely independently.

Evaluation protocol. We estimate the prospective risk in Eq. (4.2) using a Monte Carlo estimate. For a given training sequence $z_{\leq t}$, the learned sequence of predictors $h \equiv (h_s)$ is evaluated on a future realization $z_{>t}$ according to

$$\hat{R}_t(h) = \frac{1}{T-t} \sum_{s=t+1}^{T} \ell(s, h_s(x_s), y_s).$$

We use $T = 50{,}000$ for MNIST and CIFAR-10 and $T = 10{,}000$ for the synthetic experiments. For each learning algorithm, we evaluate prospective risk at 15–40 different training times. At every training time, we report the mean and standard deviation over five random seeds.

Remark 4.19 (Why we do not use existing benchmark continual learning scenarios). The tasks constructed below resemble continual learning benchmark scenarios such as Split-MNIST or Split-CIFAR10 (Zenke et al., 2017) where data from different distributions are shown sequentially to the learner. However, there are three major differences. First, in these existing benchmark scenarios, data distributions do not evolve in a predictable fashion, and prospective learning would not be meaningful. Second, existing scenarios consider a fixed time horizon. We are keen on calculating the prospective risk for much longer horizons whereby the differences between different learners are easier to discern; our experiments go for as large as 30,000 time steps. Third, our tasks have the property that the Bayes optimal predictor changes over time.

4.5.2. Prospective learners for independent but not identically distributed data (Theorem 4.2)

We create tasks using synthetic data, MNIST and CIFAR-10 datasets to design prospective learning problems when data are independent but not identically distributed across time (Theorem 4.2).

Dataset and Tasks. For the synthetic data, we consider two binary classification problems ("tasks") where the input is one-dimensional. Inputs for both tasks are drawn from a uniform distribution on the set $[-2, -1] \cup [1, 2]$. Ground-truth labels correspond to the sign of the input for Task 1, and the negative of the sign of the input for Task 2. For MNIST and CIFAR-10 we consider 4 tasks corresponding to data from classes 1-5, 4-7, 6-9 and 8-10 in the original dataset, i.e., the first task considers classes 1-5 labelled 1-5 respectively, the second task considers classes 4-7 labelled 1-4, the third task considers classes 6-9 labeled 1-4 and the last task considers labels 8-10 labelled 1-3. In other words, images from class 1 in task 1, class 4 from task 2 and class 6 from task 3 are all assigned the label 1. For the prospective learning problem based on synthetic data, the task switches every 20 time steps. For MNIST and CIFAR-10, the data distribution cycles through the 4 tasks, and the distribution of data changes every 10 time-steps. All learners are trained and evaluated according to the protocol in Sec. 4.5.1.

Fig. 4.9 shows that **prospective ERM can learn problems when data are independent but not identically distributed (Theorem 4.2)**. For prospective learning problems constructed from synthetic data, the risk of prospective ERM converges to prospective Bayes risk over time. For the MNIST and CIFAR-10 prospective problems, the prospective learning risk drops precipitously. In contrast, online learning baselines discussed above achieve a far worse prospective risk. Observe that

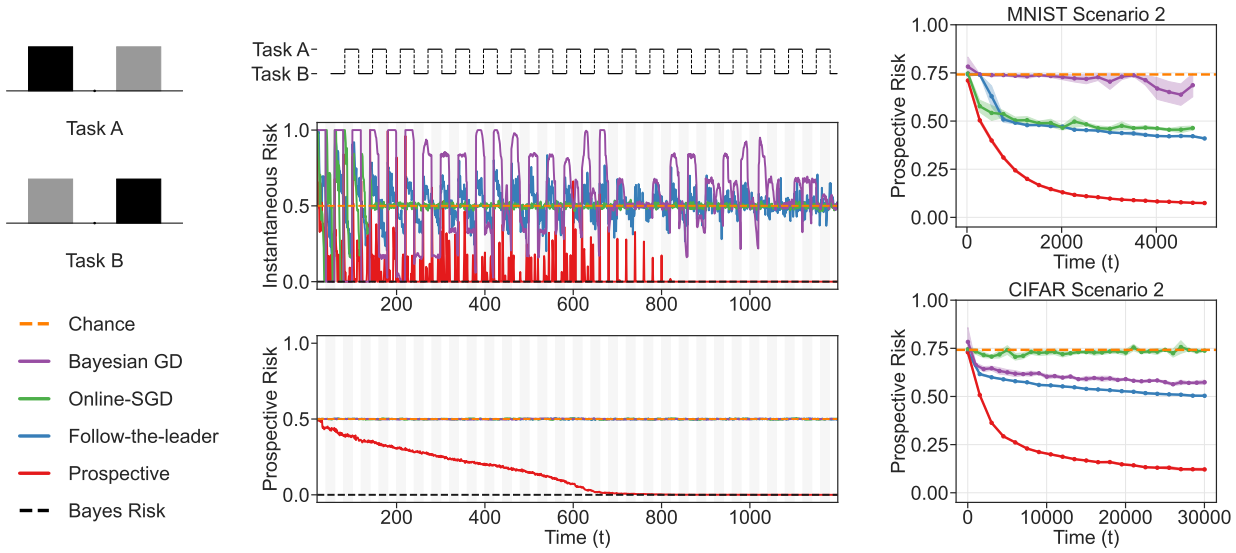

Figure 4.9: Prospective ERM can achieve good instantaneous and prospective risk in Theorem 4.2. Left: Instantaneous and prospective risks for problems constructed using synthetic data (see text) across 5 random seeds (which govern the sequence of samples and the weight initializations of neural networks). Instantaneous risk spikes when the task switches for many online learning baseline algorithms. In contrast, prospective ERM has minimal spikes at later times and both instantaneous and prospective risks eventually converge to zero. **Right:** Prospective risk for different baseline algorithms and prospective ERM for tasks constructed using MNIST and CIFAR-10 for Theorem 4.2. In all three cases, the risk of prospective ERM approaches Bayes risk while online learning baselines considered here do not achieve a low prospective risk. For comparison, the chance prospective risk is 0.5 for synthetic data and 0.742 for MNIST and CIFAR-10 tasks.

Follow-the-Leader (blue) performs as well, or better, as online SGD and Bayesian GD. This is not surprising, while ERM models corresponding to each time t were trained independently, networks corresponding to online SGD and Bayesian GD were training in an online fashion; in practice it is often difficult to tune online learning methods effectively (Li et al., 2020a).

4.5.3. Prospective learners when data are neither independent nor identically distributed (Theorem 4.3)

We next evaluate prospective ERM on three constructions in which the data are neither independent nor identically distributed: a hierarchical hidden Markov model, a Markov chain with periodic resets, and a stationary Markov chain.

4.5.3.1 Hierarchical Hidden Markov Model

Dataset and tasks. For synthetic data, we construct four binary classification problems with two-dimensional input data (see Fig. 4.10 and its caption for details). For CIFAR-10 and MNIST, we consider four tasks corresponding to the classes 1–5, 4–7, 6–9, and 8–10. Using these tasks, we construct problems where the data distribution is governed by a hierarchical hidden Markov model (Theorem 4.3). After every 10 time steps, a different Markov chain governs transitions among tasks: one Markov chain for Tasks 1 and 2 and another for Tasks 3 and 4, as shown in Fig. 4.10. These choices ensure that the stochastic process does not have a stationary distribution.[11]

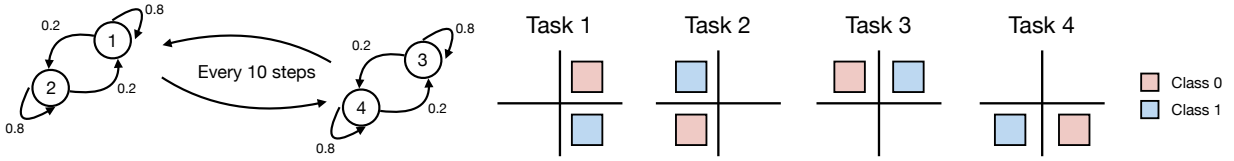

Figure 4.10: Left: For MNIST and CIFAR-10, we consider 4 tasks corresponding to the classes 1-5, 4-7, 6-9 and 8-10. Using these tasks, we construct Scenario 3 problems corresponding to a stochastic process which is a hierarchical hidden Markov model. After every 10 time-steps, a different Markov chain governs transitions among tasks (one Markov chain for tasks 1 and 2, and another for tasks 3 and 4). This ensures that the stochastic process does not have a stationary distribution. **Right:** For synthetic data, the 4 tasks are created using two-dimensional input data as shown pictorially above. The four parts of the input domain are $\{(x_1, x_2) : 1 \leq x_1, x_2 \leq 2\}$, $\{(x_1, x_2) : 1 \leq x_1 \leq 2, \text{ and } -2 \leq x_2 \leq -1\}$, $\{(x_1, x_2) : -2 \leq x_1, x_2 \leq -1\}$ and $\{(x_1, x_2) : -2 \leq x_1 \leq -1 \text{ and } 1 \leq x_2 \leq 2\}$. Colors indicate classes. The hierarchical hidden Markov model for transitions among the tasks is identical to the MNIST and CIFAR-10 setting shown on the left.

Learners and hypothesis classes. We compare prospective ERM with the time-agnostic and online baselines described in Sec. 4.5.1. All methods use an MLP for the synthetic and MNIST tasks and a CNN for the CIFAR-10 task, and are trained and evaluated according to the protocol in Sec. 4.5.1.

Results. We report the prospective risks of all learners in Fig. 4.11.

[11] As discussed in Theorem 4.3, prospective Bayes risk can be trivial when the stochastic process has a stationary distribution.

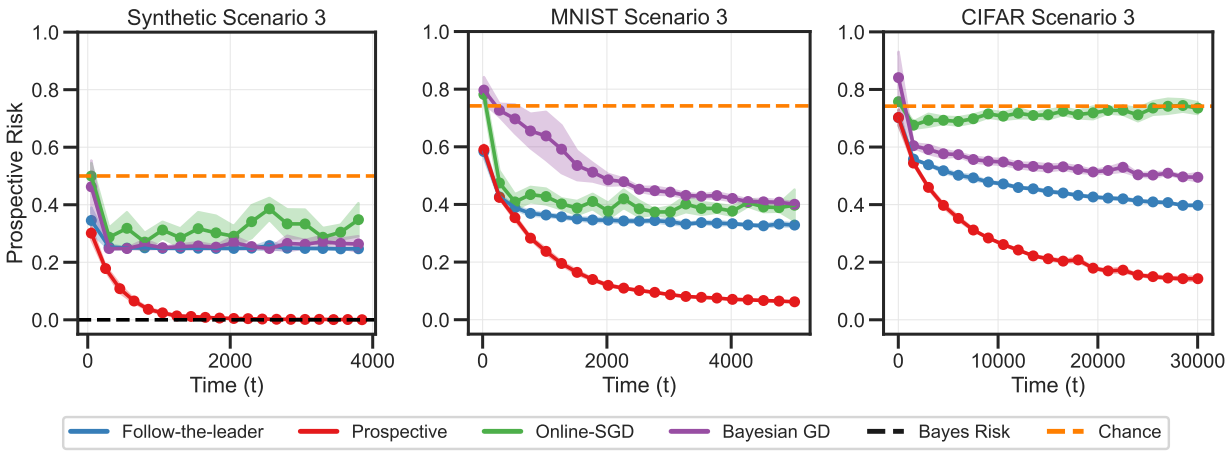

Figure 4.11: Prospective ERM can achieve good prospective risk in Theorem 4.3. Prospective risk across 5 random seeds (which govern the sequence of samples and the weight initializations of neural networks). In all three cases, the risk of prospective ERM approaches Bayes risk while a number of baseline algorithms do not achieve a low prospective risk. Stochastic processes in these problems corresponding to Scenario 3 do not have an invariant distribution. This is why a time-agnostic hypothesis (ERM) that is constructed by the baseline algorithms does not achieve a good prospective risk.

As Fig. 4.11 shows, **prospective ERM can prospectively learn problems when data are neither independent nor identically distributed (Theorem 4.3)**. The stochastic processes in these problems do not have a stationary distribution. This is why a time-agnostic hypothesis such as Follow-the-Leader does not achieve a good prospective risk, unlike prospective ERM.

4.5.3.2 Markov Chain with Periodic Resets

Dataset and tasks. For synthetic data, we consider the two binary classification problems described in Sec. 4.5.2. For CIFAR-10 and MNIST, we consider two tasks corresponding to the classes 1–5 and the classes 1–5 with each class y relabeled as $(y + 1) \pmod 5$. Using these tasks, we construct Scenario 3 problems governed by a hidden Markov model with two states. The tasks follow a Markov process with transition probability $P(S_{t+1} = k \mid S_t = k) = 0.1$, where S_t denotes the task at time t. After every 10 time steps, the state of the Markov chain is reset to the first task, ensuring that the resulting stochastic process does not have a stationary distribution.

Learners and hypothesis classes. We compare Follow-the-Leader and prospective ERM. Both methods use an MLP for the synthetic and MNIST tasks and a CNN for the CIFAR-10 task. Prospective ERM additionally receives the time embedding as input. All methods are trained and evaluated according to the protocol in Sec. 4.5.1.

Results. We report the prospective risks of both learners in Fig. 4.12.

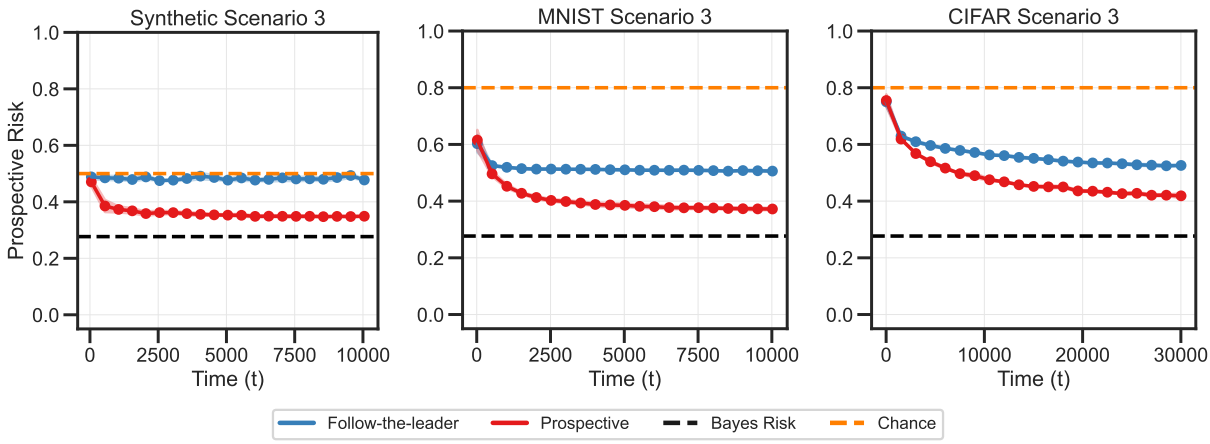

Figure 4.12: Prospective ERM can achieve good prospective risk in Theorem 4.3. We plot the prospective risk across 5 random seeds (which govern the sequence of samples and the weight initialization of the neural networks). In all three cases, the risk of prospective ERM approaches Bayes risk while Follow-the-Leader does not achieve a low prospective risk. Bayes risk for MNIST and CIFAR-10 problems is calculated by assuming that Bayes risk on individual tasks is zero.

As Fig. 4.12 shows, **prospective ERM can also prospectively learn Scenario 3 problems generated by a Markov chain with periodic resets**.

4.5.3.3 Stationary Markov Chain

Dataset and tasks. We use the same two task families as in the periodic-reset experiment. The tasks are governed by the same two-state Markov process, with $P(S_{t+1} = k \,|\, S_t = k) = 0.1$, but the state is not periodically reset. Consequently, the Markov chain equilibrates to its stationary distribution.

Learners and hypothesis classes. We again compare Follow-the-Leader and prospective ERM using the architectures and training protocol in Sec. 4.5.1. We evaluate both the empirical prospective risk and the empirical discounted prospective risk in Eq. (4.5), using a discount factor of 0.95 for the latter.

Results. We report the undiscounted and discounted prospective risks in Figs. 4.13 and 4.14, respectively.

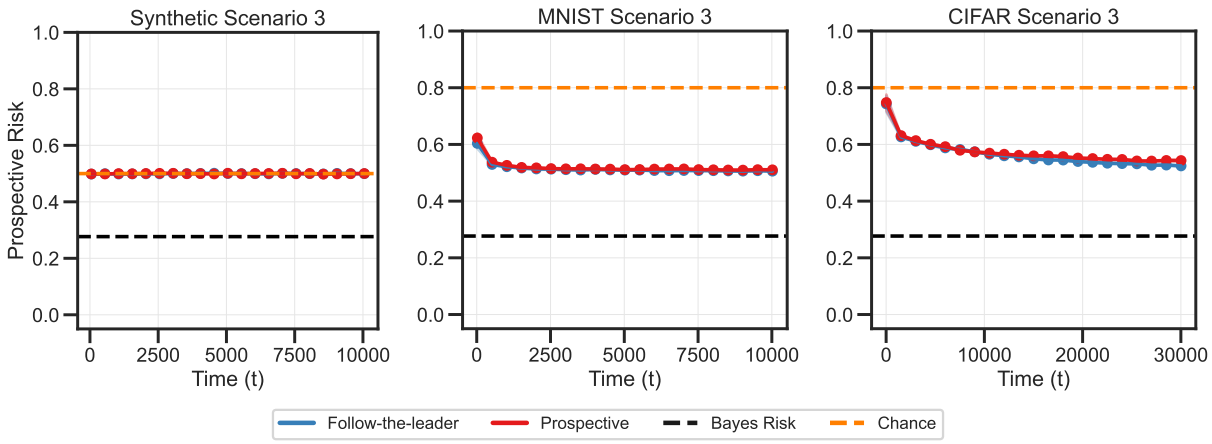

Figure 4.13: For a task defined on a stationary Markov process, the Bayes risk is trivial and can be achieved by a hypothesis that does not change over time. We plot the prospective risk across 5 random seeds (which govern the sequence of samples and the weight initialization of the neural networks). In all three cases, both Follow-the-Leader and prospective ERM approach the Bayes risk. The stationary distribution has an equal probability of seeing either task, and a fixed hypothesis can achieve Bayes risk on this problem.

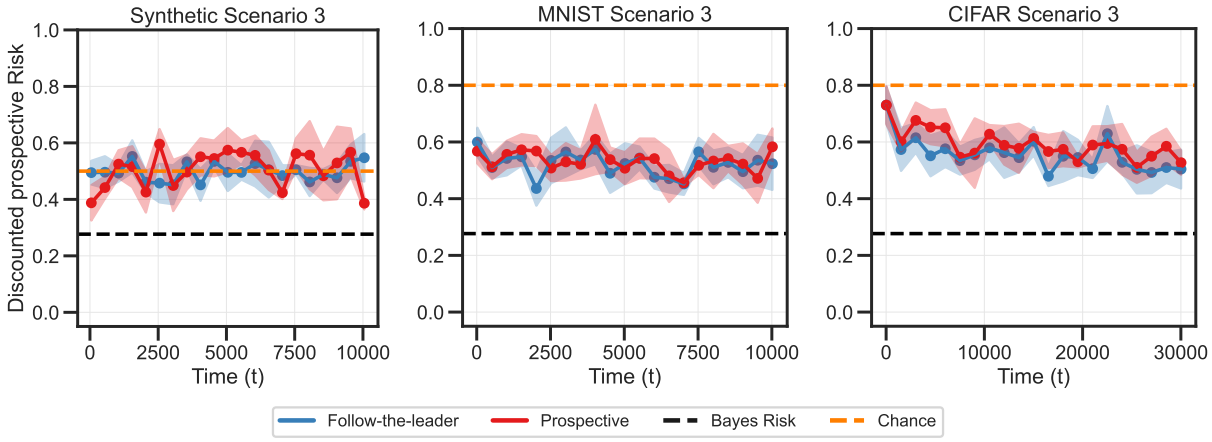

Figure 4.14: Both prospective ERM and Follow-the-Leader achieve similar discounted prospective risks with discount factor 0.95**.** We plot the discounted prospective risk across 5 random seeds. Both Follow-the-Leader and prospective ERM achieve similar discounted risks. The error bars are larger because the discount factor reduces the effective number of test samples.

As Fig. 4.13 shows, both learners approach the undiscounted prospective Bayes risk because the stationary distribution can be handled by a time-agnostic predictor. Their discounted prospective risks are also similar, as shown in Fig. 4.14.

4.5.4. Large Language Models May Not Be Good Prospective Learners

Experimental setup. It is an interesting question whether LLMs trained using autoregressive likelihoods with Transformer-based architectures can learn prospectively. To study this question, we use Llama-7B (Touvron et al., 2023) and Gemma-7B (Team et al., 2024) and evaluate their

prospective risk on Theorems 4.1 to 4.3. Each prompt contains samples from the stochastic process, represented by a subsequence of (Y_t) consisting of 0s and 1s, and an English-language description of the family of stochastic processes that generated them. The LLM is asked to complete the prompt with the next 20 most likely outcomes.

LLMs can be brittle and are known to generate different completions depending on whether a prompt is written in English, Thai, or Swahili (Deng et al., 2023), which makes prospective learning difficult to evaluate. We therefore do not describe prospective risk or prospective learnability in the prompt. We simply describe the data-generating process, provide samples from it, and ask the model for the most likely completion. We also experiment with several variants of these prompts.

We use greedy decoding, selecting the token with the highest probability at every step. We vary the number of time steps in the prompt from 1 to 100, corresponding to the amount of training data. For each time t, we generate 20 additional tokens and use them to estimate the prospective risk. Each point in Fig. 4.15 is the prospective risk on these 20 future samples, averaged over 100 realizations of the training data. The exact prompt templates are provided at the end of this subsection.

Prospective-learning results. We report the prospective risk of the LLMs on the three scenarios in Fig. 4.15.

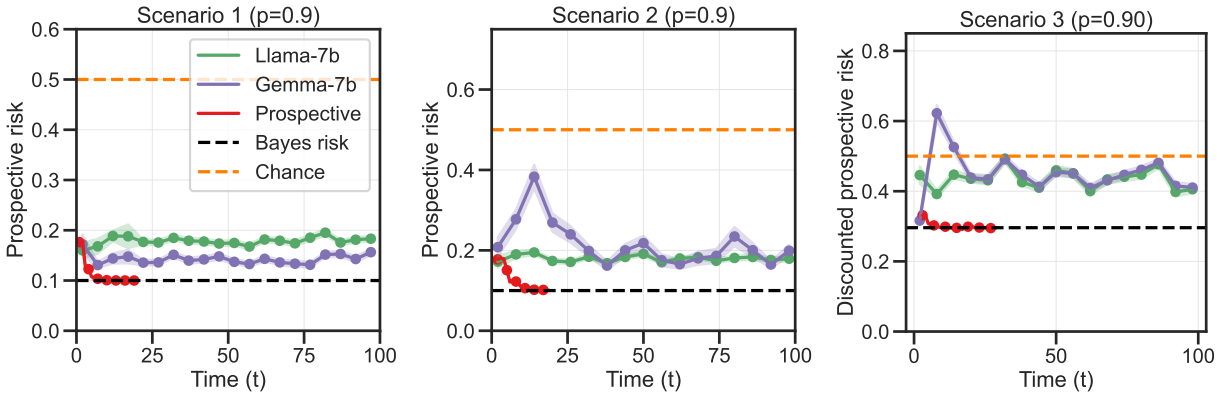

Figure 4.15: The prospective risk of LLMs on the three scenarios, averaged over realizations of the training data. Unlike a prospective maximum-likelihood learner, the LLMs do not improve with more data, suggesting that they do not prospect successfully under this experimental setup.

As Fig. 4.15 shows, Llama-7B and Gemma-7B do not achieve a lower prospective risk as the number of samples increases. It is particularly surprising that they do not achieve Bayes risk even on independent and identically distributed data. These experiments, however, do not definitively establish whether LLMs can learn prospectively.

Bernoulli-generation diagnostic. To provide context for these results, we prompt each LLM to generate a sequence of 0s and 1s sampled from a Bernoulli distribution with probability $p = 0.75$. We plot the probability of generating each outcome, over all sequences of length 10, in Fig. 4.16.

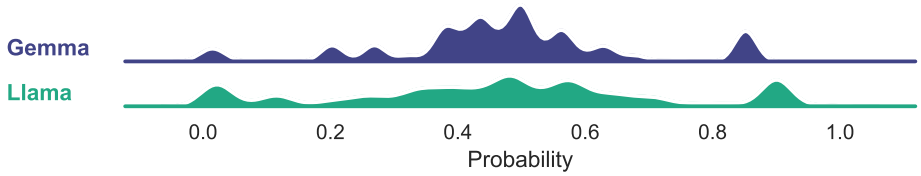

Figure 4.16: We prompt LLMs to generate the outcomes of 10 Bernoulli trials with $p = 0.75$. We plot the probability of generating token 1 over all possible sequences of 10 Bernoulli trials and find that the outcomes are generated with probabilities ranging from 0 to 1, with an average of 0.5. Ideally, token 1 should always be generated with probability $p = 0.75$. The LLMs therefore cannot simulate outcomes from a Bernoulli distribution under this experimental setup.

Ideally, the probabilities would be concentrated around 0.25 for token 0 and 0.75 for token 1. Instead, the LLMs appear unable to generate a sequence of Bernoulli trials with the specified probability. This result helps qualify the prospective-learning experiment: the models do not seem to learn prospectively, but under these prompting conditions they also cannot reliably sample from a simple Bernoulli distribution.[12]

Prompt templates. We use the following prompts to generate predictions with Llama-7B and Gemma-7B. The models always generate a sequence of 0s and 1s, so no response post-processing is required. We generate 20 samples using greedy decoding and execute the language models with weights in 16-bit precision. Variants such as adding spaces between the 1s and 0s produce qualitatively similar results.

Scenario 1

Consider the following sequence of outcomes generated from a single Bernoulli distribution.

11110101111110111111111111

The next 20 most likely outcomes are:

Scenario 2

Consider the following sequence of outcomes generated by two Bernoulli distributions, where all even outcomes are generated by a Bernoulli distribution with parameter 'p' and odd outcomes are generated from a Bernoulli distribution with parameter '1-p'.

10101010101010101010100010101010101010101

The next 20 most likely sequence of outcomes are:

[12]Responses from ChatGPT-turbo and GPT-4o were more verbose than those from Llama-7B and Gemma-7B. ChatGPT responded correctly to Scenario 1, perhaps because it used a scratchpad (Nye et al., 2021; Wei et al., 2022) to generate intermediate steps, but it did not achieve a small prospective risk for Scenarios 2 and 3. Gemini and GPT-4o refused to provide a complete response to Scenario 3 and instead outlined the sequence of steps, albeit correctly.

Scenario 3

Consider the following sequence of states generated by a Markov process with 2 states (0, 1):

10101101010100101010

The next 20 most likely outcomes are:

To make the LLM generate a sequence of Bernoulli trials with probability 0.75, we use the following prompt.

Bernoulli trials

Generate outcomes of 10 Bernoulli trials where 0 is generated with probability 0.25 and 1 with probability 0.75.

4.6. Related Learning Paradigms and Discussion

4.6.1. Related Learning Paradigms

When prospective learning (*De Silva et al., 2023) is first described, it can sound similar to several established learning frameworks. However, their formal assumptions, objectives, and evaluation protocols differ in important ways. Table 4.1 summarizes the typical distinctions, which we discuss in detail below. The central distinction is that a prospective hypothesis can make an inference or take an action arbitrarily far into the future. Certain forms of probabilistic forecasting (Gneiting and Katzfuss, 2014) also have this property, but differ from prospective learning in other respects.

Framework	Data Distribution	Loss	# Optimal hypotheses	Data Availability
PAC Learning	IID	Instantaneous	1	Batch
Transfer Learning	Change Point	Instantaneous	1–2	Batch
Meta Learning	Task IID	Instantaneous	# Tasks	Batch/Sequential
Lifelong Learning	Task IID	Instantaneous	# Tasks	Task Sequential
Online Learning	None	Instantaneous	1	Sequential
Forecasting	Stochastic Process	Fixed horizon	# Time steps	Batch/Sequential
Reinforcement Learning	Markov Decision Process	Time-varying	1	Sequential
Prospective Learning	Stochastic (Decision) Process	Instant./Time-varying	# Time steps	Any

Table 4.1: Comparison of different machine learning frameworks in terms of their typical assumptions and objectives. Task IID indicates that data within each task are IID and that tasks are IID draws from a meta-distribution. Loss indicates whether the objective is instantaneous, time-varying, or evaluated over a fixed horizon. The number of optimal hypotheses assumes that each hypothesis has a unique risk. Data availability indicates whether observations arrive in a batch, one task at a time, or one sample at a time.

PAC learning. PAC learning (Valiant, 2013; Vapnik, 1991) is a special case of prospective learning when the stochastic process is time-invariant, the data are IID, and the loss is fixed. It is also typically formulated for batch data. Whether the prospective viewpoint is useful for IID data remains an open question in general. In Theorem 4.1, we give a simple example in which prospection is beneficial even in the IID setting. More broadly, this viewpoint may suggest new algorithms for IID learning problems that do not have closed-form solutions.

Distribution shift and transfer learning. Violations of the IID assumption are often modeled as a distribution shift between training and test data (Quiñonero-Candela, 2009). Transfer

learning (Ben-David et al., 2010), including domain or covariate-shift adaptation (Daume, 2007; Sugiyama et al., 2007) and out-of-distribution learning (De Silva et al., 2023), typically considers two distributions, a source and a target. The distribution therefore changes once rather than potentially at every time step. Depending on whether the goal is to perform well on the target alone or on both source and target, there are one or two optimal hypotheses, and the occurrence of the shift is often known.

Methods such as propensity scoring (Agarwal et al., 2011; Fakoor et al., 2024) or domain adaptation (Daume, 2007; Ben-David et al., 2010) reweight or map the training and test data to recover an IID-like setting, while domain-invariance methods (Arjovsky, 2020; Blum-Smith and Villar, 2023) construct statistics that are invariant to the shift. The loss is typically unchanged between training and test data. If the set of marginals $\{\mathbb{P}(Z_t)\}$ contains only two elements, prospective learning can reduce to a classical distribution-shift problem. More generally, prospective learning allows data to be correlated across time, distributions to shift repeatedly, and the risk itself to change with time.

Multi-task, meta-, continual, and lifelong learning. A changing data distribution can be modeled as a sequence of tasks. Multi-task learning (Baxter, 2000a) is useful for settings such as Theorem 4.2 and Sec. 4.4.2, where there are finitely many tasks. Meta-learning (Finn et al., 2017a; Maurer and Jaakkola, 2005), including zero-shot (Romera-Paredes and Torr, 2015) and few-shot learning (Snell et al., 2017), typically assumes that data are *Task IID*: observations within a task are IID, and the tasks are themselves IID draws from a meta-distribution. This makes future task identities difficult to predict beyond selecting the most probable task. The learner typically knows that the task has changed, although this assumption is not universal. Classical meta-learning receives data in a batch, whereas online meta-learning receives tasks sequentially (Finn et al., 2019). Its goal is usually to perform well on the next unknown task, rather than across all future distributions as in prospective learning.

Continual or lifelong learning (Ramesh and Chaudhari, 2022; Vogelstein et al., 2020; Thrun, 1998) generally receives one batch per task and seeks to retain performance on previous tasks. Much of this literature focuses on task-incremental or class-incremental settings (Van de Ven and Tolias, 2019), where the learner knows when the task changes, although task-agnostic settings are also studied (De Lange and Tuytelaars, 2021). Prospective learning does not require known task boundaries. Data-incremental continual learning is therefore close to prospective learning, but the objectives differ: continual learning primarily seeks to control past error, whereas prospective learning evaluates performance across the future. As the experiments in Sec. 4.5 demonstrate, continual-learning methods can consequently be poor prospective learners.

Sequential decision making and online learning. Online learning (Shalev-Shwartz, 2011) generally imposes no distributional assumptions (Mohri et al., 2018), and the environment may be adversarial. Performance is measured relative to what the best fixed hypothesis could have achieved on observations seen so far, rather than by a generalization error over the future. Prospective learning instead models future observations through a stochastic process and evaluates the hypothesis on that future.

There is also a rich literature on learning from streaming data and sequential decision making. Existing methods may minimize error on samples from a stationary process (Gama et al., 2013), on a fixed held-out dataset, or on all past observations (Hayes et al., 2020); none of these objectives directly emphasizes prospection. Even for a finite-state stationary ergodic process, a consistent

estimator based on the finite past need not exist (Cover, 1975; Bailey, 1976; Ryabko, 1988; Ornstein, 1978). The same difficulty arises in regression when the true hypothesis is fixed (Morvai et al., 1996; Nobel, 2003). Consequently, the prospective Bayes risk may be nonzero even for finite-state stationary ergodic processes.

Hanneke (2021a) removes the restrictions of stationarity and ergodicity and characterizes input processes that admit consistent inductive, self-adaptive, or online learning (Rakhlin and Sridharan, 2008; Shalev-Shwartz et al., 2012). Inductive learning predicts at time $t' > t$ using data through time t; self-adaptive learning additionally observes the future inputs $X_{t+1:t'}$; and online learning predicts at time t' using observations through that time. Hanneke proves the existence of a learning rule that is consistent whenever the input process admits self-adaptive learning. If the input process is smooth, in the sense that its marginals have similar support over time, ERM can have sample complexity comparable to that in the IID setting (Block et al., 2024); Haghtalab et al. (2022) provide algorithmic guarantees for several online-estimation problems in this setting.

The true hypothesis in prospective learning can itself change over time. This differs from continual-learning settings in which a common hypothesis can solve the tasks at all times (Peng et al., 2023). We therefore define a hypothesis class as a set of predictor sequences, a subset of $(\mathcal{Y}^{\mathcal{X}})^{\mathbb{N}}$, and perform ERM in this space. Our guarantees concern strong learnability, namely convergence of ERM risk to prospective Bayes risk, rather than consistency of an individual prediction.

Forecasting. Forecasting, time-series analysis, and sequential analysis (Petropoulos et al., 2022; Ghosh and Sen, 1991; Cesa-Bianchi and Lugosi, 2006) assume that data follow a stochastic process, much like prospective learning in Theorems 4.2 and 4.3. The number of optimal hypotheses can therefore equal the number of time steps. Forecasting, however, typically evaluates loss at one or several prespecified horizons (Lim et al., 2021), and often assumes a parametric model, although nonparametric approaches also exist (Chen et al., 2018; Zeng et al., 2023). Probabilistic forecasting (Gneiting and Katzfuss, 2014) can predict arbitrarily far into the future by iteratively updating its forecasts, but this procedure can accumulate approximation and numerical errors, as occurs in sequential Monte Carlo methods.

Reinforcement learning. Reinforcement learning (Sutton and Barto, 1998) focuses on settings with a control component: the hypothesis selects an action that may influence future distributions. It therefore does not cover purely inferential instances such as Theorems 4.2 and 4.3. Classical reinforcement-learning theory also focuses on Markov decision processes (Strehl et al., 2009), whereas prospective learning permits more general stochastic decision processes, including non-Markov processes. Prospective learning may use either instantaneous or time-varying losses and permits batch or sequential data, while classical reinforcement learning is sequential, with offline reinforcement learning providing a batch alternative (Levine et al., 2020). Moreover, a classical optimal policy is usually time-invariant, although time-varying generalizations are being developed (Kumar et al., 2023). Reinforcement learning also typically uses multiple episodes, whereas prospective learning considers a single realization, related to recent work on single-episode reinforcement learning (Chen et al., 2022a).

Even without a control component, future risk matters when the optimal hypothesis varies with time. Minimizing only the current loss does not require the learner to represent how the data evolve, whereas minimizing prospective risk forces it to model the relevant modes of future variation. Under the conditions of Theorem 4.15, controlling the averaged prospective risk also controls a discounted

future loss. Prospective learning can thus be viewed as evaluating performance over an indefinitely long future horizon; it becomes more closely related to current-loss objectives when the data evolve slowly (Fakoor et al., 2024).

Architectures for prospective learning. Recurrent neural networks, including Long Short-Term Memory networks (Sherstinsky, 2020), are plausible architectures for prospective learning. Echo-state and liquid-state machines provide further examples. Other reservoir-computing methods (Lukoševičius and Jaeger, 2009) and Gaussian processes (Rasmussen and Williams, 2019) are also plausible. When an architecture satisfies the conditions of Theorem 4.9, it implements a prospective learner. These are therefore model classes that can instantiate the framework, rather than competing learning objectives.

Information-theoretic perspectives. Several works characterize classes of stochastic processes that can be predicted fruitfully. Bialek et al. (2001) define predictive information, closely related to the information-bottleneck principle (Tishby et al., 1999), and relate it to the degrees of freedom of a stochastic process. Shalizi and Crutchfield (2001) show that a causal-state representation called an ϵ-machine is a minimal sufficient statistic for prediction.

4.6.2. Discussion

Prospective learning, as we see it, is a paradigm of learning that characterizes many real-world scenarios which are currently modeled using much stronger (and less accurate) assumptions. These simplifying assumptions have certainly enabled progress in machine learning. But systems deployed built upon these approaches have proven to be extremely fragile in certain real-world settings. Today's machine learning-based systems fail to track changes in the data. They certainly do not model how biological organisms learn robustly and effectively over time. We believe characterizing which kinds of stochastic processes are prospectively learnable under which kinds of time-sensitive loss functions will be an important next theoretical step. Developing algorithms, from the perspective of prospective learning, which have theoretical guarantees in practice, will be another next step. Finally, building algorithms that scale and can therefore be deployed in real-world systems, will be important to demonstrate the utility of this approach. The precise real-world applications in which prospective learning based methods will outperform PAC learning, remains to be seen.

CHAPTER 5
CONCLUSIONS AND FUTURE DIRECTIONS

This thesis develops a data and training-dependent perspective on the generalization of deep neural networks. We show that the geometry of typical datasets can restrict highly overparameterized models to a small effective subspace, leading to a low effective dimensionality despite their large nominal capacity. We then characterize how the generalization gap evolves during optimization through the coupled effects of perturbation and contraction, and summarize these dynamics using an effective Gram matrix whose interaction with the residual predicts generalization. Finally, we extend the learning framework beyond the IID setting through prospective learning, where data distributions and optimal predictors may evolve over time. Together, these results highlight how data structure, optimization dynamics, and temporal variation jointly determine what a learning system can generalize to.

An important theme of this thesis is that the data distribution itself regularizes both the training and generalization of neural networks. The results in Chapters 2 and 3 suggest that generalization on typical tasks can arise from a low-dimensional geometric structure shared by the training and test data. In the era of foundation models and large language models, however, the data encountered during training and evaluation are increasingly heterogeneous, multimodal, and evolving (Bommasani et al., 2022). Such data may not be adequately described by a single IID distribution, a fixed collection of distributions, or a prescribed pattern of temporal evolution. Moreover, we should not expect a single global geometry to remain meaningful across all domains and modalities. Even if individual domains retain a locally low-dimensional structure, these local geometries need not be globally aligned and may not explain generalization to genuinely new tasks.

The absence of a single global geometry does not mean that heterogeneous data are unstructured. Rich domains are often constructed from reusable components, including entities, attributes, relations, and primitive operations. These components can be combined according to structured rules to generate a combinatorially large family of concepts and tasks. Related results show that hierarchical compositional functions can be efficiently represented when their constituent functions are locally low-dimensional (Mhaskar et al., 2017). Relational representations similarly organize knowledge in terms of entities, relations, and rules for their composition, thereby supporting combinatorial generalization (Battaglia et al., 2018). From this perspective, the data may be locally geometric but globally compositional.

The ability to reuse familiar concepts and operations in unfamiliar combinations is a hallmark of flexible human intelligence (Lake et al., 2017; Lake and Baroni, 2023). Humans can apply a known relation to new objects, transfer a familiar operation to a new context, and combine previously learned skills into a novel sequence. Such behavior goes beyond interpolation between nearby examples and forms an important component of reasoning. At the same time, standard neural networks can succeed when test examples are close to observed combinations but fail when systematic recombination is required (Lake and Baroni, 2018). This suggests that generalization over heterogeneous tasks may depend not only on geometric proximity, but also on whether the relevant primitives and their rules of composition can be identified from the observed data.

These considerations motivate compositional generalization as a controlled setting in which to study generalization beyond a shared global geometry. Rather than asking whether a test example is geometrically close to the training data, we ask whether a new task can be generated by recombining

operations and instance families that have already been observed. This also exposes a natural boundary of generalization: novel combinations may be learnable from existing components, whereas genuinely new primitives or composition rules may require additional supervision or interaction. The following section develops this perspective more formally.

5.1. Compositional generalization

Compositional generalization refers to the ability to recombine previously learned operations or skills and apply them to context or instances that were not observed during training. For example, a model may have learned how to extract numerical quantities from text and add them in problems involving prices, while also learning how to interpret temperature-related statements in a separate set of tasks. A compositional test problem may then require the model to apply the same extraction-and-addition procedure to a temperature instance. More formally, a model may have learned a particular sequence of operations on one class of inputs and learned the context information of another class of inputs, without ever observing their specific pairing.

To formalize this setting, let $T \in \mathcal{T}$ denote an operation template, which specifies the primitive operations required by a task and the order in which they are executed, and let $I \in \mathcal{I}$ denote the concrete instance on which the template acts. A sample is generated according to

$$x = \mathrm{Render}(T, I), \qquad y = f^\star(T, I), \tag{5.1}$$

where in the language model setting, both x and y can be question-answer pairs. We consider training and test distributions whose template and instance marginals are identical,

$$P_{\mathrm{tr}}(T) = P_{\mathrm{te}}(T), \qquad P_{\mathrm{tr}}(I) = P_{\mathrm{te}}(I), \tag{5.2}$$

but whose joint distributions differ,

$$P_{\mathrm{tr}}(T, I) \neq P_{\mathrm{te}}(T, I). \tag{5.3}$$

Thus, every template and every instance family appearing at test time has already been observed during training, while some template–instance pairings are held out. Compositional generalization measures whether the model can infer the correct behavior on these unseen pairings by reusing the functional mechanisms learned from the observed combinations.

By analogy with IID generalization, we seek a data-dependent quantity that relates the error on an unseen composition to the residual error remaining after training. Let $z = (T, I)$, let $s_\theta(z)$ denote a task-relevant scalar output of the model, and define the residual

$$r(z) = s_{\hat{\theta}}(z) - s^\star(z). \tag{5.4}$$

Under a local linearization around the trained solution, assume that the residual can be represented in the local tangent space as

$$r(z) = J(z)w, \qquad J(z) = \nabla_\theta s_{\hat{\theta}}(z), \tag{5.5}$$

where w represents a local parameter-space direction associated with the remaining prediction error. For a training set

$$S = \{z_i\}_{i=1}^n, \tag{5.6}$$

define the empirical Jacobian Gram matrix

$$G_S = \frac{1}{n} \sum_{i=1}^{n} J(z_i)^\top J(z_i). \tag{5.7}$$

We then define the regularized local compositional leverage of a test composition z as

$$u_\lambda(z; S) = J(z) \left(G_S + \lambda I\right)^{-1} J(z)^\top. \tag{5.8}$$

The quantity $u_\lambda(z; S)$ measures how strongly the prediction at z depends on functional directions that are weakly constrained by the training samples. It is small when the Jacobian of an unseen template–instance pairing lies primarily in directions repeatedly activated by the training data, as would occur when the model reuses previously learned template and instance modules. In contrast, it is large when the new pairing requires an interaction or routing direction that was not sufficiently constrained by the observed combinations.

Define the empirical squared residual and the regularized residual energy by

$$\widehat{R}_S = \frac{1}{n} \sum_{i=1}^{n} r(z_i)^2, \tag{5.9}$$

and

$$\mathcal{E}_{S,\lambda} = w^\top \left(G_S + \lambda I\right) w = \widehat{R}_S + \lambda \|w\|_2^2. \tag{5.10}$$

The pointwise squared error at a test composition z satisfies

$$e(z) = r(z)^2 \leq u_\lambda(z; S) \mathcal{E}_{S,\lambda}. \tag{5.11}$$

This is derived by rewriting the residual at a test composition z as

$$\begin{aligned} r(z) &= J(z) w \\ &= J(z) A_{S,\lambda}^{-1/2} A_{S,\lambda}^{1/2} w \\ &= \left(A_{S,\lambda}^{-1/2} J(z)^\top\right)^\top \left(A_{S,\lambda}^{1/2} w\right), \end{aligned} \tag{5.12}$$

where $A_{S,\lambda} = G_S + \lambda I$, and applying the Cauchy–Schwarz inequality

$$\begin{aligned} r(z)^2 &\leq \left\| A_{S,\lambda}^{-1/2} J(z)^\top \right\|_2^2 \left\| A_{S,\lambda}^{1/2} w \right\|_2^2 \\ &= J(z) A_{S,\lambda}^{-1} J(z)^\top \, w^\top A_{S,\lambda} w. \end{aligned} \tag{5.13}$$

This formulation separates two requirements for successful compositional generalization. First, training must leave only a small residual within the explored functional space. Second, the functional directions required by unseen compositions must be well covered by those activated on the observed training combinations. Thus, compositional generalization depends not only on the effective size of the functional space explored during training, but also on how well this space covers the functional mechanisms required by the compositional test distribution.

Post-training may provide a useful setting for studying how compositional structure emerges in large language models. SFT learns from a fixed distribution of correct demonstrations and may therefore be most effective when the required reasoning skills and their interfaces can be explicitly covered by representative traces. By contrast, RL samples from the model's current output distribution and uses reward to reshape that distribution. It may therefore be particularly useful when the model already assigns non-negligible probability to the required local skills and reasoning transitions, but does not reliably select or recombine them on unseen template–instance pairings. However, when the relevant trajectories lie far outside the model's current support, sparse rewards may provide little useful signal, and additional SFT or curriculum design may first be necessary. A promising future direction is to characterize this transition in terms of functional coverage: SFT may introduce or strengthen the directions required by the task, whereas RL may reorganize their probability and routing. Comparing how the two stages change the leverage of unseen compositions could clarify when supervised learning or reward-based exploration is more effective for compositional generalization.

BIBLIOGRAPHY

Alessandro Achille, Matteo Rovere, and Stefano Soatto. Critical learning periods in deep networks. In *International Conference on Learning Representations*, 2018.

Deepak Agarwal, Lihong Li, and Alexander Smola. Linear-time estimators for propensity scores. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 93–100, 2011.

Ali Akbari, Muhammad Awais, Manijeh Bashar, and Josef Kittler. How does loss function affect generalization performance of deep learning? application to human age estimation. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 141–151. PMLR, 2021.

Zeyuan Allen-Zhu, Yuanzhi Li, and Yingyu Liang. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in Neural Information Processing Systems*, volume 32, pages 6158–6169, 2019.

Noga Alon, Shai Ben-David, Nicolò Cesa-Bianchi, and David Haussler. Scale-sensitive dimensions, uniform convergence, and learnability. *Journal of the ACM*, 44(4):615–631, July 1997.

Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.

Shun-ichi Amari, Hyeyoung Park, and Tomoko Ozeki. Geometrical singularities in the neuromanifold of multilayer perceptrons. *Advances in neural information processing systems*, 1:343–350, 2002.

Amiran Ambroladze, Emilio Parrado-Hernández, and John Shawe-Taylor. Tighter pac-bayes bounds. *Advances in neural information processing systems*, 19:9, 2007.

Martin Arjovsky. *Out of Distribution Generalization in Machine Learning*. PhD thesis, New York University, 2020.

Sanjeev Arora, Simon S. Du, Wei Hu, Zhiyuan Li, and Ruosong Wang. Fine-grained analysis of optimization and generalization for overparameterized two-layer neural networks. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Morgane Austern and Wenda Zhou. Asymptotics of Cross-Validation. *Annales de l'Institut Henri Poincaré, Probabilités et Statistiques*, 61(4):2804–2865, 2025.

David Harold Bailey. Sequential schemes for classifying and predicting ergodic processes. *Stanford University ProQuest Dissertations Publishing*, 1976.

Arindam Banerjee, Tiancong Chen, Xinyan Li, and Yingxue Zhou. Stability based generalization bounds for exponential family Langevin dynamics. In *Proceedings of the 39th International Con-*

ference on Machine Learning, volume 162 of *Proceedings of Machine Learning Research*, pages 1412–1449. PMLR, 17–23 Jul 2022.

Peter Bartlett and Shahar Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. In *JMLR*, volume 3, pages 463–482. 2002.

Peter L. Bartlett, Dylan J. Foster, and Matus Telgarsky. Spectrally-normalized margin bounds for neural networks. In *Advances in Neural Information Processing Systems*, volume 30, pages 6240–6249, 2017.

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.

Peter W. Battaglia, Jessica B. Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, Caglar Gulcehre, Francis Song, Andrew Ballard, Justin Gilmer, George Dahl, Ashish Vaswani, Kelsey Allen, Charles Nash, Victoria Langston, Chris Dyer, Nicolas Heess, Daan Wierstra, Pushmeet Kohli, Matt Botvinick, Oriol Vinyals, Yujia Li, and Razvan Pascanu. Relational inductive biases, deep learning, and graph networks. *arXiv:1806.01261 [cs, stat]*, October 2018.

J. Baxter. A Model of Inductive Bias Learning. *Journal of Artificial Intelligence Research*, 12: 149–198, March 2000a. ISSN 1076-9757.

Jonathan Baxter. A model of inductive bias learning. *Journal of Artificial Intelligence Research*, 12:149—-198, 2000b.

Mikhail Belkin, Daniel Hsu, Siyuan Ma, and Soumik Mandal. Reconciling modern machine-learning practice and the classical bias–variance trade-off. *Proceedings of the National Academy of Sciences*, 116(32):15849–15854, 2019.

Mikhail Belkin, Daniel Hsu, and Ji Xu. Two models of double descent for weak features. *SIAM Journal on Mathematics of Data Science*, 2(4):1167–1180, 2020.

Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

William Bialek, Ilya Nemenman, and Naftali Tishby. Predictability, complexity, and learning. *Neural computation*, 13(11):2409–2463, 2001.

Adam Block, Alexander Rakhlin, and Abhishek Shetty. On the performance of empirical risk minimization with smoothed data, 2024.

Avrim Blum, Adam Kalai, and John Langford. Beating the hold-out: Bounds for k-fold and progressive cross-validation. In *Proceedings of the Twelfth Annual Conference on Computational*

Learning Theory (COLT), pages 203–208, Santa Cruz, CA, USA, July 7–9 1999. ACM.

Ben Blum-Smith and Soledad Villar. Machine learning and invariant theory. *arXiv preprint arXiv:2209.14991*, 2023.

Anselm Blumer, Andrzej Ehrenfeucht, David Haussler, and Manfred K Warmuth. Learnability and the vapnik-chervonenkis dimension. *Journal of the ACM (JACM)*, 36(4):929–965, 1989.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri Chatterji, Annie Chen, Kathleen Creel, Jared Quincy Davis, Dora Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren Gillespie, Karan Goel, Noah Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark Krass, Ranjay Krishna, Rohith Kuditipudi, Ananya Kumar, Faisal Ladhak, Mina Lee, Tony Lee, Jure Leskovec, Isabelle Levent, Xiang Lisa Li, Xuechen Li, Tengyu Ma, Ali Malik, Christopher D. Manning, Suvir Mirchandani, Eric Mitchell, Zanele Munyikwa, Suraj Nair, Avanika Narayan, Deepak Narayanan, Ben Newman, Allen Nie, Juan Carlos Niebles, Hamed Nilforoshan, Julian Nyarko, Giray Ogut, Laurel Orr, Isabel Papadimitriou, Joon Sung Park, Chris Piech, Eva Portelance, Christopher Potts, Aditi Raghunathan, Rob Reich, Hongyu Ren, Frieda Rong, Yusuf Roohani, Camilo Ruiz, Jack Ryan, Christopher Ré, Dorsa Sadigh, Shiori Sagawa, Keshav Santhanam, Andy Shih, Krishnan Srinivasan, Alex Tamkin, Rohan Taori, Armin W. Thomas, Florian Tramèr, Rose E. Wang, William Wang, Bohan Wu, Jiajun Wu, Yuhuai Wu, Sang Michael Xie, Michihiro Yasunaga, Jiaxuan You, Matei Zaharia, Michael Zhang, Tianyi Zhang, Xikun Zhang, Yuhui Zhang, Lucia Zheng, Kaitlyn Zhou, and Percy Liang. On the opportunities and risks of foundation models, 2022.

Aleksandar Botev, Hippolyt Ritter, and David Barber. Practical Gauss-Newton Optimisation for Deep Learning. In *Proceedings of the 34th International Conference on Machine Learning*, pages 557–565. PMLR, 2017.

Olivier Bousquet and André Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

Benjamin Bowman and Guido Montúfar. Spectral bias outside the training set for deep networks in the kernel regime. In *Advances in Neural Information Processing Systems*, volume 35, 2022.

Kevin S Brown, Colin C Hill, Guillermo A Calero, Christopher R Myers, Kelvin H Lee, James P Sethna, and Richard A Cerione. The statistical mechanics of complex signaling networks: Nerve growth factor signaling. *Physical biology*, 1(3):184, 2004.

Yuan Cao and Quanquan Gu. Generalization bounds of stochastic gradient descent for wide and deep neural networks. In *Advances in Neural Information Processing Systems*, volume 32, pages

10836–10846, 2019.

Nicolo Cesa-Bianchi and Gabor Lugosi. *Prediction, Learning, and Games*. March 2006.

Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

Zachary Charles and Dimitris Papailiopoulos. Stability and generalization of learning algorithms that converge to global optima. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 745–754. PMLR, 10–15 Jul 2018.

Pratik Chaudhari and Stefano Soatto. Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. *Proc. of International Conference of Learning and Representations (ICLR), Apr 30-May 3, 2018*, 2018.

Pratik Chaudhari, Anna Choromanska, Stefano Soatto, Yann LeCun, Carlo Baldassi, Christian Borgs, Jennifer Chayes, Levent Sagun, and Riccardo Zecchina. Entropy-SGD: Biasing gradient descent into wide valleys. In *Proc. of the International Conference on Learning Representations (ICLR)*, 2017.

Annie Chen, Archit Sharma, Sergey Levine, and Chelsea Finn. You only live once: Single-life reinforcement learning. *Advances in Neural Information Processing Systems*, 35:14784–14797, 2022a.

Daiwei Chen, Weikai Chang, and Pratik Chaudhari. Learning capacity: A measure of the effective dimensionality of a model. *arXiv preprint arXiv:2305.17332*, 2023a.

Minshuo Chen, Haoming Jiang, Wenjing Liao, and Tuo Zhao. Nonparametric regression on low-dimensional manifolds using deep ReLU networks: Function approximation and statistical recovery. *Information and Inference: A Journal of the IMA*, 11(4):1203–1239, 2022b. arXiv:1908.01842.

T Chen, Yulia Rubanova, J Bettencourt, and D Duvenaud. Neural ordinary differential equations. *Neural Information Processing Systems*, 31:6572–6583, June 2018.

Yilan Chen, Wei Huang, Hao Wang, Charlotte Loh, Akash Srivastava, Lam Nguyen, and Lily Weng. Analyzing generalization of neural networks through loss path kernels. In *Advances in Neural Information Processing Systems*, volume 36, pages 70948–70982. Curran Associates, Inc., 2023b.

Yilan Chen, Zhichao Wang, Wei Huang, Andi Han, Taiji Suzuki, and Arya Mazumdar. Generalization bound of gradient flow through training trajectory and data-dependent kernel. In *Advances in Neural Information Processing Systems*, volume 38, pages 10148–10189, 2025.

Lénaïc Chizat and Francis Bach. On the global convergence of gradient descent for over-

parameterized models using optimal transport. In *Advances in Neural Information Processing Systems*, volume 31, pages 3036–3046. Curran Associates, Inc., 2018.

Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. *Advances in Neural Information Processing Systems*, 32:2937–2947, 2019.

Yifeng Chu and Maxim Raginsky. A unified framework for information-theoretic generalization bounds. In *Advances in Neural Information Processing Systems*, volume 36, pages 79260–79278, 2023.

Eugenio Clerico, Tyler Farghly, George Deligiannidis, Benjamin Guedj, and Arnaud Doucet. Generalisation under gradient descent via deterministic pac-bayes. In *Proceedings of The 36th International Conference on Algorithmic Learning Theory*, volume 272 of *Proceedings of Machine Learning Research*, pages 349–389. PMLR, 24–27 Feb 2025.

Thomas M. Cover. Open problems in information theory. *IEEE USSR Joint Workshop on Information Theory*, 1975.

Leello Tadesse Dadi and Volkan Cevher. Generalization of noisy SGD in unbounded non-convex settings. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pages 11862–11883. PMLR, 13–19 Jul 2025.

Felix Dangel, Frederik Kunstner, and Philipp Hennig. BackPACK: Packing more into backprop. In *International Conference on Learning Representations*, 2020.

H Daume, III. Frustratingly Easy Domain Adaptation. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 2007.

Alexander Davydov, Saber Jafarpour, and Francesco Bullo. Non-Euclidean Contraction Theory for Robust Nonlinear Stability. *IEEE Transactions on Automatic Control*, 67(12):6667–6681, 2022.

Matthias De Lange and Tinne Tuytelaars. Continual prototype evolution: Learning online from non-stationary data streams. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 8250–8259, 2021.

Ashwin *De Silva, Rahul *Ramesh, Pratik **Chaudhari, and Joshua T **Vogelstein. Prospective Learning: Principled Extrapolation to the Future. In *Proc. of Conference on Lifelong Learning Agents (CoLLAs)*, 2023.

Ashwin De Silva, Rahul Ramesh, Carey Priebe, Pratik Chaudhari, and Joshua T Vogelstein. The value of out-of-distribution data. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 7366–7389. proceedings.mlr.press, 2023.

Yue Deng, Wenxuan Zhang, Sinno Jialin Pan, and Lidong Bing. Multilingual jailbreak challenges in large language models. *arXiv preprint arXiv:2310.06474*, 2023.

Guneet S Dhillon, Pratik Chaudhari, Avinash Ravichandran, and Stefano Soatto. A baseline for few-shot image classification. In *Proc. of International Conference of Learning and Representations (ICLR)*, 2020.

Laurent Dinh, Razvan Pascanu, Samy Bengio, and Yoshua Bengio. Sharp minima can generalize for deep nets. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1019–1028. PMLR, August 2017.

Edgar Dobriban and Stefan Wager. High-dimensional asymptotics of prediction: Ridge regression and classification. *The Annals of Statistics*, 46(1):247–279, 2018.

Simon S. Du, Xiyu Zhai, Barnabás Póczos, and Aarti Singh. Gradient descent provably optimizes over-parameterized neural networks. In *International Conference on Learning Representations*, 2019.

Gintare Karolina Dziugaite and Daniel M. Roy. Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data. In *Proceedings of the 33rd Conference on Uncertainty in Artificial Intelligence*, 2017a.

Gintare Karolina Dziugaite and Daniel M. Roy. Computing Nonvacuous Generalization Bounds for Deep (Stochastic) Neural Networks with Many More Parameters than Training Data. In *Proc. of the Conference on Uncertainty in Artificial Intelligence (UAI)*, 2017b.

Gintare Karolina Dziugaite and Daniel M Roy. Data-dependent PAC-Bayes priors via differential privacy. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 8440–8450, 2018.

Rasool Fakoor, Pratik Chaudhari, Stefano Soatto, and Alexander J Smola. Meta-Q-Learning. In *Proc. of International Conference of Learning and Representations (ICLR)*, 2020.

Rasool Fakoor, Jonas Mueller, Zachary C. Lipton, Pratik Chaudhari, and Alexander J. Smola. Time-Varying Propensity Score to Bridge the Gap between the Past and Present. In *ICLR*, 2024.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017a.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135, 2017b.

Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online Meta-Learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1920–1930. PMLR, 2019.

Stanislav Fort and Surya Ganguli. Emergent properties of the local geometry of neural loss landscapes. *arXiv preprint arXiv:1910.05929*, 2019a.

Stanislav Fort and Surya Ganguli. Emergent properties of the local geometry of neural loss landscapes. *arXiv:1910.05929 [cs, stat]*, October 2019b.

Jingwen Fu, Zhizheng Zhang, Dacheng Yin, Yan Lu, and Nanning Zheng. Learning trajectories are generalization indicators. In *NeurIPS 2023*, June 2023.

Futoshi Futami and Masahiro Fujisawa. Time-Independent Information-Theoretic Generalization Bounds for SGLD. In *Advances in Neural Information Processing Systems*, volume 36, pages 8173–8185, 2023.

Joao Gama, Raquel Sebastiao, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. *Machine learning*, 90:317–346, 2013.

B K Ghosh and P K Sen. *Handbook of Sequential Analysis (Statistics: A Series of Textbooks and Monographs)*. CRC Press, 1 edition, April 1991. ISBN 9780824784089.

Tilmann Gneiting and Matthias Katzfuss. Probabilistic forecasting. *Annu. Rev. Stat. Appl.*, 1(1): 125–151, January 2014.

Guy Gur-Ari, Daniel A. Roberts, and Ethan Dyer. Gradient Descent Happens in a Tiny Subspace. *arXiv:1812.04754 [cs, stat]*, December 2018.

Ryan N Gutenkunst, Joshua J Waterfall, Fergal P Casey, Kevin S Brown, Christopher R Myers, and James P Sethna. Universally sloppy parameter sensitivities in systems biology models. *PLoS computational biology*, 3(10):e189, 2007.

Hassan Hafez-Kolahi, Zeinab Golgooni, Shohreh Kasaei, and Mahdieh Soleymani. Conditioning and processing: Techniques to improve information-theoretic generalization bounds. In *Advances in Neural Information Processing Systems*, volume 33, pages 3492–3503. Curran Associates, Inc., 2020.

Nika Haghtalab, Tim Roughgarden, and Abhishek Shetty. Smoothed analysis with adaptive adversaries. In *2021 IEEE 62nd Annual Symposium on Foundations of Computer Science (FOCS)*, pages 942–953, 2022.

Steve Hanneke. Learning whenever learning is possible: Universal learning under general stochastic processes. *J. Mach. Learn. Res.*, 22:130:1–130:116, 2021a.

Steve Hanneke. Learning whenever learning is possible: Universal learning under general stochastic processes. *The Journal of Machine Learning Research*, 22(1):5751–5866, 2021b.

Moritz Hardt, Ben Recht, and Yoram Singer. Train faster, generalize better: Stability of stochastic gradient descent. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1225–1234, New York, NY, USA, June 20–22 2016. PMLR.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction*. Springer Series in Statistics. Springer, New York, 2nd edition, 2009. ISBN 978-0387848570.

Trevor Hastie, Andrea Montanari, Saharon Rosset, and Ryan J. Tibshirani. Surprises in high-dimensional ridgeless least squares interpolation. *The Annals of Statistics*, 50(2):949–986, 2022.

Tyler L Hayes, Kushal Kafle, Robik Shrestha, Manoj Acharya, and Christopher Kanan. Remind your neural network to prevent catastrophic forgetting. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VIII 16*, pages 466–483. Springer, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Flat minima. *Neural Computation*, 9(1):1–42, 1997.

Yanping Huang and Rajesh P N Rao. Predictive coding. *Wiley interdisciplinary reviews. Cognitive science*, 2(5):580–593, September 2011.

Y. Ilyashenko and S. Yakovenko. *Lectures on Analytic Differential Equations*, volume 86 of *Graduate Studies in Mathematics*. American Mathematical Society, 2008.

Alberto Isidori. *Nonlinear Control Systems*. Communications and Control Engineering. Springer-Verlag, London, 3rd edition, 1995.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, volume 31, pages 8571–8580. Curran Associates, Inc., 2018.

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

Rie Johnson and Tong Zhang. Inconsistency, instability, and generalization gap of deep neural network training. In *Advances in Neural Information Processing Systems*, volume 36, pages 9479–9505. Curran Associates, Inc., 2023.

Satyen Kale, Ravi Kumar, and Sergei Vassilvitskii. Cross-validation and mean-square stability. In *Innovations in Computer Science (ICS)*, pages 487–495, Beijing, China, January 7–9 2011.

Tsinghua University Press.

Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. Universal Statistics of Fisher Information in Deep Neural Networks: Mean Field Approach. *arXiv:1806.01316 [cond-mat, stat]*, October 2019.

Noureddine El Karoui. The spectrum of kernel random matrices. *Annals of Statistics*, 38:1–50, 2010.

Kenji Kawaguchi, Zhun Deng, Kyle Luh, and Jiaoyang Huang. Robustness implies generalization via data-dependent generalization bounds. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 10866–10894. PMLR, 17–23 Jul 2022.

M Kearns and L Valiant. Cryptographic limitations on learning Boolean formulae and finite automata. *Journal of the ACM*, 1994.

Michael Kearns. Thoughts on Hypothesis Boosting, 1988.

Leo Kozachkov, Patrick Wensing, and Jean-Jacques Slotine. Generalization as dynamical robustness–the role of riemannian contraction in supervised learning. *Transactions on Machine Learning Research*, 2023a.

Leo Kozachkov, Patrick M. Wensing, and Jean-Jacques E. Slotine. Generalization as dynamical robustness—the role of riemannian contraction in supervised learning. *Transactions on Machine Learning Research*, 2023b.

A. Krizhevsky. *Learning Multiple Layers of Features from Tiny Images*. PhD thesis, Computer Science, University of Toronto, 2009a.

Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, April 2009b.

Ravi Kumar, Cheng Li, and Matus Telgarsky. Near-optimal bounds for cross-validation via loss stability. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, pages 27–35, Atlanta, GA, USA, June 16–21 2013. PMLR.

Saurabh Kumar, Henrik Marklund, Ashish Rao, Yifan Zhu, Hong Jun Jeon, Yueyang Liu, and Benjamin Van Roy. Continual learning as computationally constrained reinforcement learning. *arXiv preprint arXiv:2307.04345*, 2023.

Frederik Kunstner, Philipp Hennig, and Lukas Balles. Limitations of the empirical Fisher approximation for natural gradient descent. In *Advances in Neural Information Processing Systems*, pages 4156–4167, 2019.

Ilja Kuzborskij and Christoph Lampert. Data-dependent stability of stochastic gradient descent. In

Proceedings of the 35th International Conference on Machine Learning, volume 80 of *Proceedings of Machine Learning Research*, pages 2815–2824. PMLR, 10–15 Jul 2018.

Brenden Lake and Marco Baroni. Generalization without systematicity: On the compositional skills of sequence-to-sequence recurrent networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2873–2882. PMLR, 10–15 Jul 2018.

Brenden M. Lake and Marco Baroni. Human-like systematic generalization through a meta-learning neural network. *Nature*, 623(7985):115–121, 2023.

Brenden M. Lake, Tomer D. Ullman, Joshua B. Tenenbaum, and Samuel J. Gershman. Building machines that learn and think like people. *Behavioral and Brain Sciences*, 40, 2017.

John Langford and Rich Caruana. (Not) bounding the true error. *Advances in Neural Information Processing Systems*, 2:809–816, 2002.

John Langford and Matthias Seeger. Bounds for averaging classifiers. 2001.

Thomas Laurent and James von Brecht. Deep linear networks with arbitrary loss: All local minima are global. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2902–2907. PMLR, 2018.

Yann LeCun, Bernhard E Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne E Hubbard, and Lawrence D Jackel. Handwritten digit recognition with a back-propagation network. In *Advances in Neural Information Processing Systems*, pages 396–404, 1990.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv [cs.LG]*, May 2020.

Hao Li, Pratik Chaudhari, Hao Yang, Michael Lam, Avinash Ravichandran, Rahul Bhotika, and Stefano Soatto. Rethinking the hyper-parameters for fine-tuning. In *Proc. of International Conference of Learning and Representations (ICLR)*, 2020a.

Hongkang Li, Meng Wang, Sijia Liu, and Pin-Yu Chen. A Theoretical Understanding of Shallow Vision Transformers: Learning, Generalization, and Sample Complexity. In *The Eleventh International Conference on Learning Representations*, 2023.

Yuanzhi Li and Yingyu Liang. Learning overparameterized neural networks via stochastic gradient descent on structured data. In *Advances in Neural Information Processing Systems*, volume 31, pages 8157–8166. Curran Associates, Inc., 2018.

Yuanzhi Li, Tengyu Ma, and Hongyang R. Zhang. Learning Over-Parametrized Two-Layer Neural Networks beyond NTK. In *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 2613–2682. PMLR, 09–12 Jul 2020b.

Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. Temporal Fusion Transformers for interpretable multi-horizon time series forecasting. *International journal of forecasting*, 37(4): 1748–1764, October 2021.

Fusheng Liu, Haizhao Yang, Soufiane Hayou, and Qianxiao Li. From optimization dynamics to generalization bounds via lojasiewicz gradient inequality. *Transactions on Machine Learning Research*, 2022. Accepted; to appear.

Winfried Lohmiller and Jean-Jacques E. Slotine. On contraction analysis for non-linear systems. *Automatica*, 34(6):683–696, 1998. ISSN 0005-1098.

Winfried Lohmiller and Jean-Jacques E. Slotine. Nonlinear process control using contraction theory. *AIChE Journal*, 46(3):588–596, March 2000.

Gábor Lugosi and K Zeger. Nonparametric estimation via empirical risk minimization. *IEEE transactions on information theory / Professional Technical Group on Information Theory*, 41 (3):677–687, May 1995.

Gábor Lugosi and Gergely Neu. Generalization bounds via convex analysis. In *Proceedings of the 35th Conference on Learning Theory (COLT)*, volume 178 of *Proceedings of Machine Learning Research*, pages 3523–3544. PMLR, 2022.

Mantas Lukoševičius and Herbert Jaeger. Reservoir computing approaches to recurrent neural network training. *Comput. Sci. Rev.*, 3(3):127–149, August 2009.

Wesley Maddox, T. Garipov, Pavel Izmailov, Dmitry P. Vetrov, and Andrew Gordon Wilson. A simple baseline for bayesian uncertainty in deep learning. In *NeurIPS*, 2019.

Neil Mallinar, James B. Simon, Amirhesam Abedsoltan, Parthe Pandit, Mikhail Belkin, and Preetum Nakkiran. Benign, tempered, or catastrophic: A taxonomy of overfitting. In *Advances in Neural Information Processing Systems*, volume 35, pages 29912–29925. Curran Associates, Inc., 2022.

Riccardo Marino and Patrizio Tomei. *Nonlinear Control Design: Geometric, Adaptive, and Robust.* Prentice Hall, London, 1995. ISBN 978-0133426359.

James Martens and Roger Grosse. Optimizing Neural Networks with Kronecker-factored Approximate Curvature. *arXiv:1503.05671 [cs, stat]*, May 2016.

V A Marčenko and L A Pastur. Distribution of eigenvalues for some sets of random matrices. *Mathematics of the USSR-Sbornik*, 1(4):457, apr 1967.

Andreas Maurer and Tommi Jaakkola. Algorithmic stability and meta-learning. *Journal of Machine Learning Research*, 6(6), 2005.

David A. McAllester. Pac-bayesian model averaging. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory (COLT)*, pages 164–170. ACM, 1999a.

David A McAllester. PAC-Bayesian model averaging. In *Proceedings of the Twelfth Annual Conference on Computational Learning Theory*, pages 164–170, 1999b.

Colin McDiarmid. On the method of bounded differences. In *Surveys in Combinatorics, 1989 (Norwich, 1989)*, volume 141 of *London Mathematical Society Lecture Note Series*, pages 148–188. Cambridge University Press, 1989.

Song Mei and Andrea Montanari. The generalization error of random features regression: Precise asymptotics and the double descent curve. *Communications on Pure and Applied Mathematics*, 75(4):667–766, 2022.

Song Mei, Theodor Misiakiewicz, and Andrea Montanari. Mean-field theory of two-layers neural networks: dimension-free bounds and kernel limit. In *Proceedings of the 32nd Conference on Learning Theory*, volume 99 of *Proceedings of Machine Learning Research*, pages 2388–2464. PMLR, 2019.

Hrushikesh Mhaskar, Qianli Liao, and Tomaso Poggio. When and why are deep networks better than shallow ones? In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pages 2343–2349, San Francisco, California, USA, 2017.

Mehryar Mohri, Afshin Rostamizadeh, and Ameet Talwalkar. *Foundations of Machine Learning*. November 2018.

Gusztav Morvai, Sidney Yakowitz, and Laszlo Gyorfi. Nonparametric inference for ergodic, stationary time series. *The Annals of Statistics*, 24(1):370–379, 1996.

Wenlong Mou, Liwei Wang, Xiyu Zhai, and Kai Zheng. Generalization bounds of sgld for non-convex learning: Two theoretical viewpoints. In *Proceedings of the 31st Conference On Learning Theory*, volume 75 of *Proceedings of Machine Learning Research*, pages 605–638. PMLR, 06–09 Jul 2018.

Jeffrey Negrea, Mahdi Haghifam, Gintare Karolina Dziugaite, Ashish Khisti, and Daniel M. Roy. Information-theoretic generalization bounds for sgld via data-dependent estimates. In *Advances in Neural Information Processing Systems*, volume 32, 2019.

Gergely Neu, Gintare Karolina Dziugaite, Mahdi Haghifam, and Daniel M. Roy. Information-theoretic generalization bounds for stochastic gradient descent. In *Proceedings of the 34th Annual Conference on Learning Theory*, volume 134 of *Proceedings of Machine Learning Research*, pages 3526–3546. PMLR, 2021.

Behnam Neyshabur, Ryota Tomioka, and Nathan Srebro. Norm-based capacity control in neural networks. In *Conference on Learning Theory*, pages 1376–1401, 2015.

Behnam Neyshabur, Srinadh Bhojanapalli, and Nathan Srebro. A pac-bayesian approach to spectrally-normalized margin bounds for neural networks. In *International Conference on Learning Representations*, 2018.

Peter Nickl, Lu Xu, Dharmesh Tailor, Thomas Möllenhoff, and Mohammad Emtiyaz Khan. The memory-perturbation equation: Understanding model's sensitivity to data. In *Advances in Neural Information Processing Systems*, volume 36, pages 26923–26949. Curran Associates, Inc., 2023.

A.B. Nobel. Average reward reinforcement learning: Foundations, algorithms, and empirical results. *IEEE Transactions on Information Theory*, 49(1):83–98, 2003.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

Adam M. Oberman. Fast Rates for Semi-Supervised Learning via Data-Augmentation Graph Regularization. *arXiv preprint arxiv:2607.07513*, 2026.

D. S. Ornstein. Guessing the next output of a stationary process. *Israel Journal of Mathematics*, 30(3):292—-296, 1978.

Samet Oymak, Ankit Singh Rawat, Mahdi Soltanolkotabi, and Christos Thrampoulidis. On the Role of Attention in Prompt-tuning. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 26724–26768. PMLR, 23–29 Jul 2023.

Vardan Papyan. The full spectrum of deepnet hessians at scale: Dynamics with sgd training and sample size. *arXiv preprint arXiv:1811.07062*, 2018.

Vardan Papyan. Measurements of three-level hierarchical structure in the outliers in the spectrum of deepnet hessians. *arXiv preprint arXiv:1901.08244*, 2019.

Emilio Parrado-Hernández, Amiran Ambroladze, John Shawe-Taylor, and Shiliang Sun. PAC-Bayes bounds with data dependent priors. *The Journal of Machine Learning Research*, 13(1):3507–3531, 2012.

Pratik Patil, Yuchen Wu, and Ryan Tibshirani. Failures and successes of cross-validation for early-stopped gradient descent. In *Proceedings of The 27th International Conference on Artificial Intelligence and Statistics*, volume 238 of *Proceedings of Machine Learning Research*, pages 2260–2268. PMLR, 2024.

Liangzu Peng, Paris Giampouras, and Rene Vidal. The ideal continual learner: An agent that never forgets. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

Jeffrey Pennington and Yasaman Bahri. Geometry of Neural Network Loss Surfaces via Random Matrix Theory. page 9.

Ankit Pensia, Varun Jog, and Po-Ling Loh. Generalization error bounds for noisy, iterative algorithms. In *2018 IEEE International Symposium on Information Theory (ISIT)*, pages 546–550. IEEE, 2018.

Fotios Petropoulos, Daniele Apiletti, Vassilios Assimakopoulos, Mohamed Zied Babai, Devon K Barrow, Souhaib Ben Taieb, Christoph Bergmeir, Ricardo J Bessa, Jakub Bijak, John E Boylan, Jethro Browell, Claudio Carnevale, Jennifer L Castle, Pasquale Cirillo, Michael P Clements, Clara Cordeiro, Fernando Luiz Cyrino Oliveira, Shari De Baets, Alexander Dokumentov, Joanne Ellison, Piotr Fiszeder, Philip Hans Franses, David T Frazier, Michael Gilliland, M Sinan Gönül, Paul Goodwin, Luigi Grossi, Yael Grushka-Cockayne, Mariangela Guidolin, Massimo Guidolin, Ulrich Gunter, Xiaojia Guo, Renato Guseo, Nigel Harvey, David F Hendry, Ross Hollyman, Tim Januschowski, Jooyoung Jeon, Victor Richmond R Jose, Yanfei Kang, Anne B Koehler, Stephan Kolassa, Nikolaos Kourentzes, Sonia Leva, Feng Li, Konstantia Litsiou, Spyros Makridakis, Gael M Martin, Andrew B Martinez, Sheik Meeran, Theodore Modis, Konstantinos Nikolopoulos, Dilek Önkal, Alessia Paccagnini, Anastasios Panagiotelis, Ioannis Panapakidis, Jose M Pavía, Manuela Pedio, Diego J Pedregal, Pierre Pinson, Patrícia Ramos, David E Rapach, J James Reade, Bahman Rostami-Tabar, Michal Rubaszek, Georgios Sermpinis, Han Lin Shang, Evangelos Spiliotis, Aris A Syntetos, Priyanga Dilini Talagala, Thiyanga S Talagala, Len Tashman, Dimitrios Thomakos, Thordis Thorarinsdottir, Ezio Todini, Juan Ramón Trapero Arenas, Xiaoqian Wang, Robert L Winkler, Alisa Yusupova, and Florian Ziel. Forecasting: theory and practice. *International journal of forecasting*, 38(3):705–871, July 2022. ISSN 0169-2070.

Joaquin Quiñonero-Candela, editor. *Dataset Shift in Machine Learning*. Neural Information Processing Series. Cambridge, Mass, 2009. ISBN 978-0-262-17005-5.

Alexander Rakhlin and Tengyuan Liang. Just interpolate: Kernel 'ridgeless' regression can generalize. *Annals of Statistics*, 48(3):1329–1347, 2020.

Alexander Rakhlin and Karthik Sridharan. Lecture notes on online learning, 2008.

Rahul Ramesh and Pratik Chaudhari. Model Zoo: A Growing "Brain" That Learns Continually. In *Proc. of International Conference of Learning and Representations (ICLR)*, 2022.

Carl Edward Rasmussen and Christopher K I Williams. *Gaussian processes for machine learning*. Adaptive Computation and Machine Learning Series. MIT Press, London, England, June 2019.

Dominic Richards and Ilja Kuzborskij. Stability & generalisation of gradient descent for shallow neural networks without the neural tangent kernel. In *Advances in Neural Information Processing Systems*, volume 34, pages 24688–24700, 2021.

Bernardino Romera-Paredes and Philip Torr. An embarrassingly simple approach to zero-shot learning. In *International Conference on Machine Learning*, pages 2152–2161. PMLR, June 2015.

D Russo and J Zou. How much does your data exploration overfit. *Controlling bias via information usage. ArXiv e-prints*, 2015.

Boris Yakovlevich Ryabko. Prediction of random sequences and universal coding. *Problems of Information Theory*, 24(2):87–96, 1988.

Levent Sagun, Leon Bottou, and Yann LeCun. Singularity of the hessian in deep learning. *arXiv:1611:07476*, 2016.

Robert E Schapire. The strength of weak learnability. *Machine learning*, 5:197–227, 1990.

Martin E P Seligman, Peter Railton, Roy F Baumeister, and Chandra Sripada. *Homo Prospectus*, volume 384. Oxford University Press, New York, NY, US, June 2016. ISBN 9780199374489.

Martin EP Seligman, Peter Railton, Roy F Baumeister, and Chandra Sripada. Navigating into the future or driven by the past. *Perspectives on psychological science*, 8(2):119–141, 2013.

Shai Shalev-Shwartz. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2011.

Shai Shalev-Shwartz et al. Online learning and online convex optimization. *Foundations and Trends® in Machine Learning*, 4(2):107–194, 2012.

Cosma Rohilla Shalizi and James P Crutchfield. Computational Mechanics: Pattern and Prediction, Structure and Simplicity. *Journal of statistical physics*, 104(3):817–879, August 2001.

Alex Sherstinsky. Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network. *Physica D*, 404(132306):132306, March 2020.

Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017. ISSN 1049-5258.

Jost Tobias Springenberg, Alexey Dosovitskiy, Thomas Brox, and Martin Riedmiller. Striving for Simplicity: The All Convolutional Net. *arXiv:1412.6806 [cs]*, April 2015.

Thomas Steinke and Lydia Zakynthinou. Reasoning about generalization via conditional mutual information. In *Proceedings of the 33rd Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 3437–3452. PMLR, 2020.

Peter Sterling. Allostasis: a model of predictive regulation. *Physiology & behavior*, 106(1):5–15, April 2012.

Alexander L Strehl, Lihong Li, and Michael L Littman. Reinforcement learning in finite MDPs: PAC analysis. *Journal of machine learning research: JMLR*, 10(84):2413–2444, 2009. ISSN 1532-4435,1533-7928.

Masashi Sugiyama, Matthias Krauledat, and Klaus-Robert Müller. Covariate Shift Adaptation by Importance Weighted Cross Validation. *Journal of machine learning research: JMLR*, 8(May): 985–1005, 2007. ISSN 1532-4435,1533-7928.

Ke Sun and Frank Nielsen. Lightlike Neuromanifolds, Occam's Razor and Deep Learning. *arXiv:1905.11027 [cs, stat]*, February 2020.

Richard S Sutton and Andrew G Barto. *Introduction to Reinforcement Learning*. Camgridge: MIT Press, March 1998.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Sebastian Thrun. Lifelong learning algorithms. In *Learning to Learn*, pages 181–209. 1998.

Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer, 1998.

Naftali Tishby, Fernando C. Pereira, and William Bialek. The information bottleneck method. In *Proc. of the 37-Th Annual Allerton Conference on Communication, Control and Computing*, pages 368–377, 1999.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Mark K. Transtrum, Benjamin B. Machta, and James P. Sethna. The geometry of nonlinear least squares with applications to sloppy models and optimization. *Physical Review E*, 83(3):036701, March 2011. ISSN 1539-3755, 1550-2376.

Hiroyasu Tsukamoto, Soon-Jo Chung, and Jean-Jacques Slotine. Contraction theory for nonlinear stability analysis and learning-based control: A tutorial overview. *Annual Reviews in Control*, 52:135–148, October 2021.

Leslie Valiant. *Probably Approximately Correct: Nature's Algorithms for Learning and Prospering in a Complex World*. Basic Books, June 2013. ISBN 9780465032716.

Leslie G. Valiant. A theory of the learnable. *Communications of the ACM*, 27(11):1134–1142, 1984.

Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

Vladimir Vapnik. Principles of risk minimization for learning theory. *Advances in neural information processing systems*, 4, 1991.

Vladimir Vapnik. *Statistical Learning Theory*. 1998.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Joshua T Vogelstein, Jayanta Dey, Hayden S Helm, Will LeVine, Ronak D Mehta, Tyler M Tomita, Haoyin Xu, Ali Geisa, Qingyang Wang, Gido M van de Ven, Chenyu Gao, Weiwei Yang, Bryan Tower, Jonathan Larson, Christopher M White, and Carey E Priebe. A simple lifelong learning approach. *arXiv [cs.AI]*, April 2020.

Mingze Wang and Chao Ma. Generalization error bounds for deep neural networks trained by sgd. *arXiv preprint arXiv:2206.03299*, 2022.

Xianhua Wang, Xing Zhang, Di Wu, Zhanglong Huang, Tingting Hou, Chongshu Jian, Peng Yu, Fujian Lu, Rufeng Zhang, Tao Sun, et al. Mitochondrial flashes regulate ATP homeostasis in the heart. *Elife*, 6:e23908, 2017.

Joshua J. Waterfall, Fergal P. Casey, Ryan N. Gutenkunst, Kevin S. Brown, Christopher R. Myers, Piet W. Brouwer, Veit Elser, and James P. Sethna. Sloppy-Model Universality Class and the Vandermonde Matrix. *Physical Review Letters*, 97(15):150601, October 2006.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8:279–292, 1992.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

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 *Proceedings of Thirty Third Conference on Learning Theory*, volume 125 of *Proceedings of Machine Learning Research*, pages 3635–3673. PMLR, 09–12 Jul 2020.

Yikai Wu, Xingyu Zhu, Chenwei Wu, Annie Wang, and Rong Ge. Dissecting Hessian: Understanding Common Structure of Hessian in Neural Networks. *arXiv:2010.04261 [cs, stat]*, June 2021.

Aolin Xu and Maxim Raginsky. Information-theoretic analysis of generalization capability of learning algorithms. In *Advances in Neural Information Processing Systems*, volume 30, pages 2524–2533, 2017.

Huan Xu and Shie Mannor. Robustness and generalization. *Machine Learning*, 86(3):391–423, 2012.

Rubing Yang, Jialin Mao, and Pratik Chaudhari. Does the data induce capacity control in deep learning? In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 25348–25368. PMLR, 2022.

Yuan Yao, Lorenzo Rosasco, and Andrea Caponetto. On early stopping in gradient descent learning. *Constructive Approximation*, 26(2):289–315, 2007.

Dong Yin, Ashwin Pananjady, Max Lam, Dimitris Papailiopoulos, Kannan Ramchandran, and Peter Bartlett. Gradient diversity: A key ingredient for scalable distributed learning. In *International Conference on Artificial Intelligence and Statistics*, pages 1998–2007. PMLR, 2018.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *British Machine Vision Conference 2016*. British Machine Vision Association, 2016.

Ailing Zeng, Muxi Chen, Lei Zhang, and Qiang Xu. Are Transformers effective for time series forecasting? *Proceedings of the ... AAAI Conference on Artificial Intelligence. AAAI Conference on Artificial Intelligence*, 37(9):11121–11128, June 2023.

Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995, 2017.

Chen Zeno, Itay Golan, Elad Hoffer, and Daniel Soudry. Task agnostic continual learning using online variational bayes. *arXiv preprint arXiv:1803.10123*, 2018.