Analytical characterization of sloppiness in neural networks: Insights from linear models

Jialin Mao [1], Itay Griniasty [2], Yan Sun [1], Mark K. Transtrum [3], James P. Sethna [4], and Pratik Chaudhari [1]

[1] *University of Pennsylvania, Philadelphia, Pennsylvania 19104, USA*

[2] *School of Mechanical Engineering, Tel Aviv University, Tel Aviv-Yafo 6997801, Israel*

[3] *Cross Stream Bioanalytics, Salt Lake City UT 84115, USA*

[4] *Department of Physics, Cornell University, Ithaca, New York 14850, USA*

(Received 14 May 2025; accepted 9 December 2025; published 21 January 2026)

Recent experiments have shown that training trajectories of multiple deep neural networks with different architectures, optimization algorithms, hyperparameter settings, and regularization methods evolve on a remarkably low-dimensional "hyperribbon-like" manifold in the space of probability distributions. Inspired by the similarities in the training trajectories of deep networks and linear networks, we analytically characterize this phenomenon for the latter. We show, using tools in dynamical systems theory, that the geometry of this low-dimensional manifold is controlled by (i) the decay rate of the eigenvalues of the input correlation matrix of the training data, (ii) the relative scale of the ground-truth output to the weights at the beginning of training, and (iii) the number of steps of gradient descent. By analytically computing and bounding the contributions of these quantities, we characterize phase boundaries of the region where hyperribbons are to be expected. We also extend our analysis to kernel machines and linear models that are trained with stochastic gradient descent.

DOI: 10.1103/fs4g-9nsz

I. INTRODUCTION

Recent experiments have shown that deep networks explore a low-dimensional manifold in the prediction space as they train to perform a task [1,3–5]. This manifold is dominated by its first few dimensions, with widths that decay geometrically. Remarkably, this manifold is shared between diverse networks, independent of their architecture, training algorithms and other specifications as depicted in Fig. 1. Low-dimensional structures also arise while fitting other nonlinear models [6,7], where these phenomena have been explained by the fact that the function being used for approximation has a limited flexibility. But deep networks are universal approximators, and they are capable of fitting arbitrary datasets. There must be another cause for the low-dimensional manifolds in deep networks.

In this paper, we argue that low-dimensional structures in training manifolds of deep networks arise not due to limited flexibility, but rather from the intrinsic low-dimensionality of the task. While nonlinear models are "sloppy," i.e., captured by manifolds in prediction space with geometrically decaying widths, due to their structural constraints, deep networks are sloppy because of the geometric properties of the task (examples in classical datasets are shown in Fig. 2). Inspired by the similarities in the training trajectories of deep and linear neural networks, we will exploit the analytical tractability of the latter to argue that it is indeed the task that results in the low-dimensionality of the training of deep networks. Let us first expand on the different facets of sloppiness.

A. Model manifold

Consider a dataset $\{(x_i, y_i)\}_{i=1}^n$ with inputs $x_i \in \mathbb{R}^d$ and labels $y_i \in \{1, \ldots, C\}$ which correspond to whether the corresponding input x_i belongs to one out of C different categories. A deep network-based classifier is a probabilistic model of this data. Given an input x, it assigns a probability to each possible output $y \in \{1, \ldots, C\}$, this is denoted by $p_w(y \mid x)$ where $w \in \mathbb{R}^p$ are the parameters/weights of the model. If $\vec{y} = (y_1, \ldots, y_n) \in \{1, \ldots, C\}^n$ is the set of all possible outputs for a dataset, we may write $P_w(\vec{y}) = \prod_{i=1}^n p_w(y_i \mid x_i)$ as the joint probability assigned to these outputs by the model. As we vary the weights w over some region in $\mathbb{R}^p$, the probabilistic model P_w spans a region that is a subset of the $n(C-1)$-dimensional simplex. Let us call this set the "model manifold". Statistical divergences, such as the Kullback–Leibler (KL) divergence, quantify distances between probabilistic models and thereby induce a natural geometry on the weight space. The Fisher information matrix (FIM) is the natural Riemannian metric on the model manifold [12].

B. Training manifold

Now consider Fig. 1 again, which was partially reproduced from Ref. [1]. The authors developed techniques to analyze the model manifold of deep networks. They showed that probabilistic models corresponding to different points on the training trajectories of multiple deep networks with different architectures, optimization algorithms, hyperparameter settings, and regularization methods evolve on a remarkably low-dimensional model manifold. To visualize and analyze this manifold, they used a technique known as intensive principal component analysis (inPCA) [13], closely related to multidimensional scaling [14] and principal components analysis (PCA) [15], to embed the models into a Minkowski space. While an exact isometric embedding is guaranteed when the dimensionality of the embedding space is equal to the number of models, they found that low-dimensional

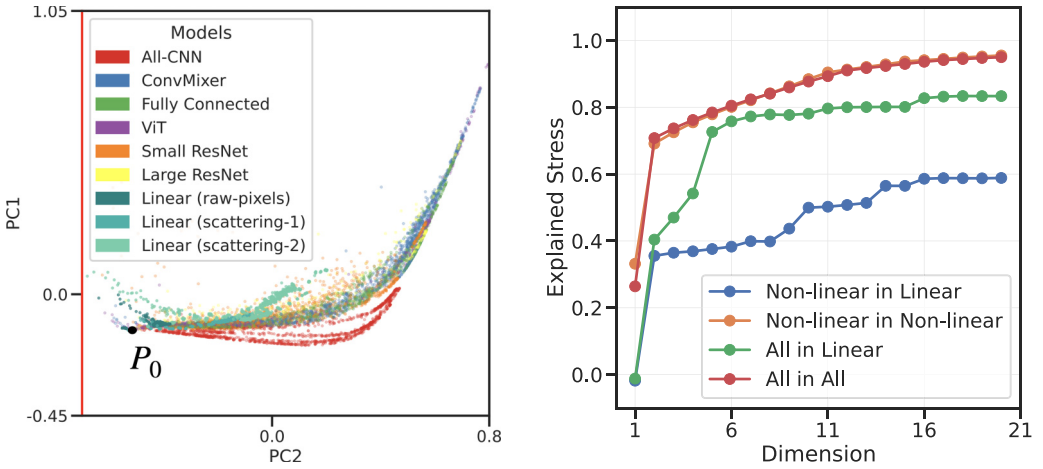


FIG. 1. Left: Manifold of models along training trajectories of networks with different configurations (architectures denoted by different colors, optimization algorithms, hyperparameters, and regularization mechanisms) is effectively low-dimensional for CIFAR-10. This is a partial reproduction using the data from Ref. [1]. Linear networks are trained upon images directly (dark green), after preprocessing using one layer ("Scattering-1" in lighter green) and two layers ("Scattering-2" in the lightest green) of a scattering transform [2]. With typical weight initializations all models begin training near P_0, where every sample is assigned equal probability to belong to every class (marked by hand to guide the reader). They progress to the truth P_* (not seen here) to different degrees. All nonlinear deep networks in this experiment achieve zero training error. While linear networks do not fit the data perfectly, the manifolds swept by linear networks are quite similar to those of deep networks. These common low-dimensional manifolds in probability space are the inspiration of this paper. Right: Quantitative analysis of the inPCA embedding in terms of the explained stress which characterizes how well pairwise distances between points are preserved after the embedding. When the inPCA embedding is computed using all the points on the left, the explained stress of the first two dimensions is about 71% (red), about the same when inPCA is computed using only nonlinear models (orange). We can compute inPCA using points corresponding to only linear models and embed all other models (green) or only the nonlinear models (blue) into this space. While there are clearly differences between the manifolds of linear and nonlinear models (blue curve is lower), it is remarkable that nonlinear models can be faithfully represented in the embedding constructed using linear models (green line is close to red and orange). Our analysis in this paper that focuses on linear models is therefore a meaningful insight into the manifolds of nonlinear models.

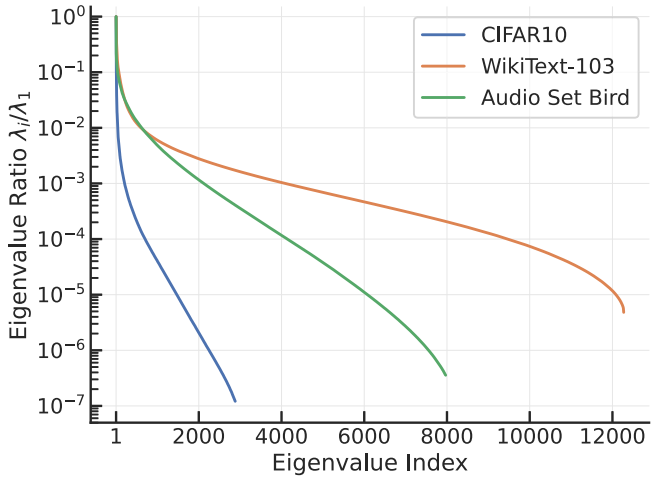


FIG. 2. Real-world data are sloppy. Eigenvalues of the empirical correlation matrices of feature representations of different types of data exhibit a sharp drop-off among the first few eigenvectors. This is characteristic of multiparameter models fit to data, but here it arises in real-world data. For CIFAR-10 [8], we use raw pixel values as the features. For WikiText-103 [9], the text is broken into token sequences of length 16, and BERT [10] embeddings from the last hidden layer are concatenated to form a feature vector. For sounds corresponding to the Bird category in AudioSet [11], each 1-s audio segment is treated as a sample, with spectrograms serving as feature vectors.

projections preserved pairwise distances between probabilistic models (statistical divergences such as the Bhattacharyya distance) very well. A two-dimensional embedding shown in Fig. 1 captures 71% of the "stress" (i.e., pairwise distance preservation). A mere 50 dimensions are sufficient to account for 98% of the stress, even when these architectures range from 0.6 million to 50 million weights. They observed that the stress explained by each successive dimension decays geometrically, indicating a highly concentrated spectrum. The points corresponding to one particular architecture in Fig. 1, which is the set of models explored during the training process, are a subset of the model manifold of that architecture. It is evident that this subset is very low-dimensional. Let us call it the "training manifold."

C. Sloppy models

The geometric decay in the stress explained by each dimension is strikingly reminiscent of structural patterns found in another class of over-parameterized models known as sloppy models [16]. Sloppiness was first identified in systems biology, where detailed mechanistic models—large systems of differential equations with numerous unknown rate constants—were constructed to describe protein or gene regulation networks [17]. In practice, inferring these parameters from data is nearly impossible. But one need not infer them exactly, analyses based on the Fisher Information Matrix and

the Cramér–Rao bound [18] showed that these parameters could vary by many orders of magnitude without substantially affecting the fit to experimental data, or predictions on new data. The signature of such "sloppy" models is that successive eigenvalues of the FIM computed at the fitted parameters decay geometrically. Sloppy models consistently exhibit good generalization—not only to out-of-sample data, but often to out-of-distribution conditions, such as predictions under novel drug regimens that differ significantly from those used for model calibration [19].

D. Sloppy data

Deep networks exhibit a lot of these same phenomena, e.g., an overwhelmingly large number of eigenvalues of the Hessian and FIM of typical networks are vanishingly small [20–22]. Symmetries in multilayer architectures entail that both these matrices have a large number of zero eigenvalues [23–25]. The authors of Ref. [26] showed that the eigenspectrum of the input correlation matrix is closely related to the eigenspectrum of the Hessian of the training loss and the FIM for general deep networks. They used this observation to argue that deep networks generalize—in spite having fewer data than the number of parameters—when the input correlation matrix has eigenvalues that decay geometrically. Let us call such data "sloppy." Almost pervasively, real-world data exhibits this sloppy structure [27]; for example, Fig. 2 shows that eigenvalues of the input correlation matrices of three different modalities (images, text, and sounds) span a very large range and are spread nearly uniformly on a logarithmic scale. This structure in the data has been argued to be the principle underlying effective continuum and universal theories in physics [28].

E. Hyperribbons

The FIM can exhibit geometric decay in its eigenvalues even when input data is one dimensional, i.e., when it could not be sloppy. Consider the situation when the targets are given by $y = f(x, w) \in \mathbb{R}$ for $x \in [-1, 1]$ and weights $w \in \mathbb{R}^p$. The truncated Taylor series of f may be written as

$$f(x, w) \approx f_n(x, w) = \sum_{k=0}^{n-1} \varphi_k(w) x^k$$

where $\varphi_k(w) = \frac{f^{(k)}(\xi, w)}{k!}$ for $\xi \in [-1, 1]$ and $f^{(k)}$ denotes the kth-derivative of f with respect to the data x. If the true function has a radius of convergence $R > 1$ and if w varies over some compact region, say, $\sum_k w_k^2 \leqslant 1$, then

$$|\varphi_k(w)| \leqslant cR^{-k}$$

for some constant c. The model manifold of f is contained within a distance $\mathcal{O}(R^{-n+1})$ of the model manifold of f_n. If we consider a dataset with n samples, the model manifold of f_n, which is the set of all possible predictions on these samples, is a subset of $\mathbb{R}^n$. In fact, it is a hyperellipsoid where the lengths of the principal axes decay geometrically. This is because the Jacobian of the predictions in $\mathbb{R}^n$ with respect to the weight space is a Vandermonde matrix which has geometrically decaying singular values [29,30]. The FIM, which is the outer-product of this Jacobian, inherits this structure, its eigenvalues also decay geometrically. The FIM is a purely local property, but it is closely related to the global geometry of model manifold, in our example. Such model manifolds look like "hyperribbons." There is one set of parameters that is tightly constrained by the data, another which can vary, say, twice as much without affecting predictions, and so on. This hyperribbon structure reflects the intrinsic limitations in model flexibility: the model simply cannot produce a wide range of predictions, despite its many parameters.

F. Contributions of this manuscript

These four concepts are clearly distinct from each other, but they share a suspiciously common pattern: (i) the stress captured by successive dimensions of an inPCA embedding of the training manifold of deep networks, (ii) eigenvalues of the FIM of sloppy models at typical fits, (iii) eigenvalues of the input correlation matrices of real-world data across diverse modalities, and (iv) cross-sectional widths of hyperribbon-like model manifolds—all exhibit geometric decay. The goal of this manuscript is to identify and clarify connections between these concepts.

The key question that will help frame our narrative is: why do different architectures, training and regularization methods, explore such a tiny subset of the prediction space? Deep networks are universal approximators [31]. With sufficient capacity and training, they can be induced to fit even anomalous or completely random data. The model manifold of a deep network-based classifier is therefore high-dimensional, it spans the entire $n(C-1)$-dimensional simplex. However, across large problems with $n(C-1) \sim 10^6$–10^8, the authors of Refs. [1,32] have found training manifolds to be remarkably low-dimensional, as low as 50 dimensions seem sufficient to embed the manifold faithfully. The same story holds for regression. This flexibility implies that the manifolds explored by deep networks during training cannot be hyperribbons in the same sense as those in sloppy models. While they may exhibit low-dimensional structure empirically, this structure cannot arise from a fundamental limitation in model flexibility. It requires a different theoretical explanation.

We show in Fig. 1 that the training manifold of linear models, e.g., linear predictors fitted upon raw pixels, or after projecting input images upon a fixed nonlinear basis computed using the scattering transform [2], is similar to that of deep networks. This empirical observation motivates our analysis. Figure 3 encapsulates some of the main results of this manuscript, discussed in Sec. II. We analyze how and when low-dimensional representations of high-dimensional training trajectories arise for linear regression problems (as we have found for the real-world models in Fig. 1). The key players of this analysis and their predominant roles are as follows.

(1) The sloppiness in the input data being fit, characterized by the logarithm of the ratio of successive eigenvalues of the input correlation matrix, we will call this the

$$\text{slope: } c,$$

that mimics the eigenvalue decay in Fig. 2. Sloppy data allows the first few dimensions of the hyperribbon of training trajectories to capture most of its variance.

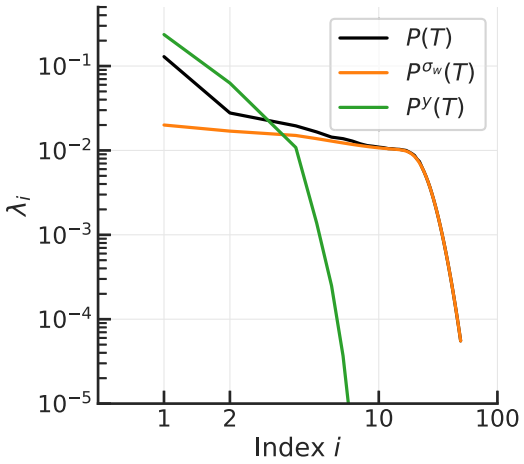


FIG. 3. Eigenvalues of PCA of points on training trajectories of linear regression and eigenvalues of different important contributions to it. There is a rapid decrease in the PCA eigenvalues (black), and thus these points admit a low-dimensional representation with high explained-variance in the first few dimensions, similar to linear and nonlinear models in Fig. 1. The PCA covariance matrix $P(T)$ after T gradient descent steps can be decomposed analytically into two important contributions: $P^{y}(T)$ that depends upon the regression targets and $P^{\sigma_w}(T)$ that depends upon the distribution of initial weights. The sharp initial decay in the eigenvalues of $P(T)$ is well-approximated by P^{y} while the decay in the tail is controlled by P^{σ_w}. Details in Fig. 5.

(2) The relative scale of the ground-truth weights to the initial weights which are drawn from zero-mean Gaussian distributions with isotropic variance σ_*^2 and σ_w^2, respectively,

$$\text{weight initialization strength: } \frac{\sigma_*}{\sigma_w}.$$

The spread of initial conditions leads to a spread in trajectories that will dominate the tail of the eigenspectrum.

(3) The

$$\text{number of weight updates: } T.$$

(4) The larger the T, the larger the volume of the output space explored by the different training trajectories and therefore, intuitively, the larger the dimensionality of the hyperribbon.

In the sequel, we will make more precise statements that relate these three parameters to training trajectories. Our analysis will proceed by analyzing the PCA matrix $P(T)$ of points along training trajectories obtained by initializing linear models at different points, and fitting them upon data with different degrees of sloppiness (different c). We will be able to decompose $P(T)$ into three pieces, the two most important being a piece P_1^{y} determined predominantly by the ground-truth targets, and a piece $P_1^{\sigma_w}$ determined by initialization, the relative importance of these terms is proportional to the weight initialization strength. By analytically computing and bounding these various contributions (Fig. 3), we characterize the "phase boundaries" of the region where low-dimensional hyperribbons are to be expected (Fig. 8). In Sec. III we will also extend our analysis to kernel machines and linear models that are trained with stochastic gradient descent.

II. TRAINING MANIFOLD FOR LINEAR REGRESSION

A. Target data set, training, and weights

Consider a dataset $\{(x_i', y_i)\}_{i=1}^n$ that consists of inputs $x_i' \in \mathbb{R}^{d-1}$ and outputs $y_i \in \mathbb{R}$. We will focus on the case with a scalar output in this paper for clarity of exposition, all results hold for multidimensional output. Let $x_i \equiv [x_i', 1]$ denote the input with a constant appended to it and consider a linear model $y_i = w^\top x_i$ with $w \in \mathbb{R}^d$ trained to minimize

$$C(w) = \frac{1}{2n}\sum_{i=1}^{n} r_i(w)^2. \tag{1}$$

Here the residuals $r_i(w) = \hat{y}_i - y_i$ for $i \in \{1, \ldots, n\}$ denote the difference between the predictions and the targets $y_i \in \mathbb{R}$. We will assume that targets correspond to unknown true weights $w^* \in \mathbb{R}^d$, i.e., $y_i = w^{*\top} x_i$. Discrete-time gradient descent to minimize this objective with a step size α (learning rate) can be written as $w_{t+1} = w_t - \alpha\, \partial_w C(w_t)$, starting from some $w_0 \in \mathbb{R}^d$ for all $t = 1, 2, \ldots$. We denote the n-dimensional vector of residuals computed at weights w_t by $r_t \equiv [r_1(w_t), \ldots, r_n(w_t)]^\top \in \mathbb{R}^n$. As the weights are updated by gradient descent, this vector r_t evolves as

$$r_{t+1} = (I - \alpha K) r_t = (I - \alpha K)^{t+1} r_0, \tag{2}$$

where $I \in \mathbb{R}^{n\times n}$ is the identity matrix and $K \in \mathbb{R}^{n\times n}$ is a symmetric positive semidefinite matrix with entries $K_{ij} = x_i^\top x_j/n$ for $i, j \in \{1, \ldots, n\}$. Note that $K = XX^\top/n$ where the ith row of $X \in \mathbb{R}^{n\times d}$ contains the input datum x_i. In other words, for linear models, the neural tangent kernel (NTK [33]) is simply the input-correlation matrix. Let the ith largest eigenvalue of K be denoted by $\lambda_i^K \geqslant 0$. The shorthand

$$K_d \equiv I - \alpha K$$

will be useful to simplify our expressions. It is the first-order approximation of $\exp(-\alpha K)$. We make the following assumptions in our analysis.

(i) *Input data and ground-truth targets*. We will assume that input data are sloppy with a decay rate $c > 0$ on the logarithmic scale, i.e.,

$$\lambda_i^K = \exp(-(i-1)c),$$

for all $i = 1, \ldots, n$, with $c \gg 1/n$. We will assume that the unknown true weights w^* are a random variable

$$\mathbb{R}^d \ni w^* \sim N(0, \sigma_*^2 I)$$

where σ_*^2 is a scalar. This will indirectly be an assumption on the norm of the targets y_i.

(ii) *Model*. The model is in the data scarce regime, i.e., the number of samples is smaller than the dimensionality of the input data $n < d$. Weights w are under-determined if $n < d$ and therefore an infinite set of solutions achieves $C(w) = 0$. We will assume that $\text{rank}(K) = n$. This is not unduly restrictive. All our analysis can also be conducted in the image space of K if K is rank deficient.

(iii) *Training method*. For a large part of the analysis we will be interested in training methods that resemble gradient descent. We will assume that the step size $\alpha < 1/\lambda_1^K$. This assumption ensures that $\|I - \alpha K\|_2 < 1$ and therefore $\|r_t\|_2 \to 0$ monotonically as $t \to \infty$ and therefore it is quite

standard in the analysis of gradient descent algorithms [34]. The most important parameter of the training process for us will be the total number of weight updates T.

(iv) *Weight initialization*. We will assume that weights are initialized to values w_0 sampled from a Gaussian distribution with zero mean and an isotropic variance

$$\mathbb{R}^d \ni w_0 \sim N(0, \sigma_w^2 I)$$

for a scalar σ_w^2. The ratio σ_*^2/σ_w^2 controls the distance in weight space that a training trajectory needs to travel from its initialization at w_0 to fit to the true weights w^*. We will see that, roughly speaking, the larger this ratio, the smaller the dimensionality of the hyperribbon and weaker the dependence on the other parameters of the problem such as the slope c or the number of weight updates T.

B. Principal component analysis of the training manifold

Consider N randomly initialized models with weights $\{w_0^{(i)}\}_{i=1}^N$. From assumption (iv) on weight initialization, we have $\mathbb{E}[w_0] = 0$ and $\mathbb{E}[w_0 w_0^\top] = \sigma_w^2 I$. The initial residual vectors satisfy

$$\mathbb{E}\big[r_0^{(i)}\big] = -y, \quad \mathbb{E}\big[r_0^{(i)} r_0^{(j)\top}\big] = n\delta_{ij}\sigma_w^2 K + yy^\top,$$

for all $i, j \in \{1, \ldots, N\}$, where $y = [y_1, \ldots, y_n]^\top \in \mathbb{R}^n$ is the vector of targets and $\delta_{ij} = \mathbf{1}_{\{i=j\}}$ is the δ function. The prediction space $\mathbb{R}^n$ has Euclidean geometry. We can therefore capture the geometry of the training manifold, which is a subset of $\mathbb{R}^n$, using principal component analysis (PCA) of points along trajectories $\{r_t^{(i)}\}_{t=0,\,i=1}^{T-1,\,N}$. The covariance matrix corresponding to PCA is

$$\begin{aligned} P(N,T) &= \frac{1}{NT}\sum_{i=1}^{N}\sum_{t=0}^{T-1}\big(r_t^{(i)} - \bar{r}\big)\big(r_t^{(i)} - \bar{r}\big)^\top, \\ &= \frac{1}{N}\left(\sum_{i=1}^{N}\frac{1}{T}\sum_{t=0}^{T-1} r_t^{(i)} r_t^{(i)\top}\right) - \bar{r}\,\bar{r}^\top, \end{aligned} \tag{3}$$

where the mean residual is $\bar{r} \equiv \bar{r}(N,T) = \frac{1}{NT}\sum_{i,t} r_t^{(i)}$ for $T \geqslant 1$. The mean residual evolves according to the equation $\bar{r}(N,T) = K_T \bar{r}(N,1)$, with

$$K_T \equiv \frac{1}{T}\sum_{t=0}^{T-1} K_d^t = \frac{1}{\alpha T} K^{-1}\big(I - K_d^T\big). \tag{4}$$

As the number of random initializations N goes to infinity, we can separate the PCA matrix P into two components,

$$\begin{aligned} P(T) &= \lim_{N\to\infty} P(N,T) \\ &= \frac{1}{T}\sum_{t=0}^{T-1} K_d^t\big(n\sigma_w^2 K + yy^\top\big)K_d^t - \underbrace{K_T yy^\top K_T}_{\equiv P_2(T)} \\ &\equiv \underbrace{P_1^{\sigma_w}(T) + P_1^{y}(T)}_{\equiv P_1(T)} - P_2(T). \end{aligned} \tag{5}$$

We have broken down the PCA matrix into its three components. The third term $P_2(T)$ has unit rank. The first term $P_1^{\sigma_w}(T) = \frac{n\sigma_w^2}{T}\sum_{t=0}^{T-1} K_d^{2t} K$ depends on the variance of initial weights σ_w^2, and the second term $P_1^y(T) = \frac{1}{T}\sum_{t=0}^{T-1} K_d^t yy^\top K_d^t$ again depends on the targets y, but it is not unit rank. We will analyze this expression further in the next section.

C. Geometry of the training manifold

The goal of this section will be to characterize the geometry of the training manifold, i.e., eigenvalues of the PCA matrix $P(T)$ in Eq. (5). To simplify the notation, we will denote eigenvalues of matrices P_2, TP_1^y, and $TP_1^{\sigma_w}$ by λ^{P_2}, $\lambda_i^{\sigma_w}$, and λ_i^y, respectively, for all $i = 1, \ldots, n$. The scaling with T on the latter two matrices is only for the sake of convenience in our exposition.

1. Eigenvalue of P_2

For any two real symmetric matrices $A, B \in \mathbb{R}^{n\times n}$, Weyl's inequality says that

$$\lambda_i^A + \lambda_n^B \leqslant \lambda_i^{A+B} \leqslant \lambda_i^A + \lambda_1^B, \tag{6}$$

for eigenvalues λ_i ordered in decreasing order. The third term $P_2(T)$ in Eq. (5) is an outer product of y with itself and has a single nonvanishing eigenvalue $\lambda^{P_2} = \|K_T y\|^2$. Therefore, eigenvalues of $P(T)$ are sandwiched by the eigenvalues of $P_1(T)$:

$$\max\big(\lambda_{i+1}^{P_1}, \lambda_i^{P_1} - \lambda^{P_2}\big) \leqslant \lambda_i^P \leqslant \lambda_i^{P_1}. \tag{7}$$

From Eq. (4),

$$\|K_T\|_2 \leqslant \frac{1 - \big(1 - \alpha\lambda_1^K\big)^T}{\alpha T \lambda_n^K} \leqslant \frac{1}{\alpha T \lambda_n^K}. \tag{8}$$

This suggests that $\lambda^{P_2} = \mathcal{O}(1/T^2)$. The approximation becomes tighter for long training times. We can therefore focus our analysis on understanding $P_1(T)$.

2. Weight initialization strength contribution

Since $K_d = I - \alpha K$ commutes with K, we can write $TP_1^{\sigma_w}(T)$ to be the geometric sum

$$TP_1^{\sigma_w}(T) = n\sigma_w^2 \sum_{t=0}^{T-1} K_d^{2t} K = \frac{n\sigma_w^2}{\alpha}(I + K_d)^{-1}\big(I - K_d^{2T}\big),$$

and calculate its eigenvalues explicitly as

$$\lambda_i^{\sigma_w} = \frac{n\sigma_w^2}{\alpha}\left(\frac{1 - \big(1 - \alpha\lambda_i^K\big)^{2T}}{2 - \alpha\lambda_i^K}\right). \tag{9}$$

Due to the inequality $1 - (1-x)^a \leqslant \min\{1, ax\}$, which holds for $|x| < 1$ and $a \geqslant 1$, under assumption (iii), the eigenvalue $\lambda_i^{\sigma_w}$ is bounded above by

$$\lambda_i^{\sigma_w} \leqslant \frac{n\sigma_w^2}{\alpha}\left(\frac{\min\big\{1, 2T\alpha\lambda_i^K\big\}}{2 - \alpha\lambda_i^K}\right). \tag{10}$$

Figure 4 uses Eq. (10) to explain how eigenvalues $\lambda_i^{\sigma_w}$ of different indices i depend upon the training duration T and the sloppy eigenspectrum of the input correlation matrix K. From Eq. (5), it is immediate that if the initialization variance σ_w^2 is small (relative to σ_*^2 which controls $\|y\|$), the

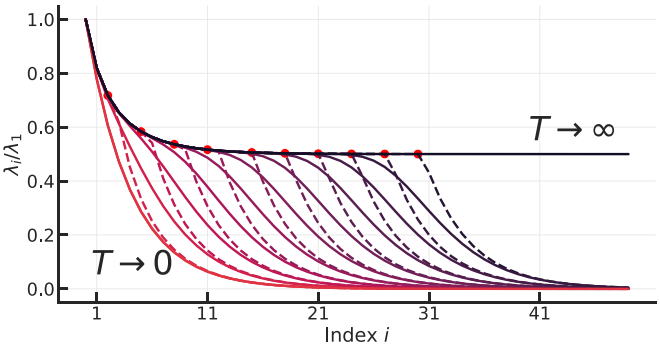


FIG. 4. Contributions to the eigenspectrum of the PCA matrix coming from weight initialization computed using the bound in Eq. (10) (dotted) and numerical computation of the corresponding term in Eq. (5) (bold) with $\sigma_w^2 = 1$, $\alpha = 1$, and $c = 0.5$. For very large training times $T \gg \lambda_1^K/\lambda_n^K$, the eigenvalue $\lambda_i^{\sigma_w}$ corresponds to the minimum in the numerator being 1 for any i. This means that $\lambda_i^{\sigma_w}$ decays to $\sim\sigma_w^2/2\alpha$ as the index i increases, at a rate determined by the decay of λ_i^K. From Eq. (10), for $T \ll \lambda_1^K/\lambda_n^K$ and $i < \ln(2T\alpha)/c$, the minimum in the numerator is 1, and we again have a similar decay. For larger values of i, the minimum comes from the other term and therefore $\lambda_i^{\sigma_w}$ decays to much smaller values $\sim\frac{1}{(2/\lambda_n^K-\alpha)}$. This is the lower envelope of the curves above. The limit $T \to 0$ corresponds to the eigenvalues of $\sigma_w^2 K$ and therefore reflects the sloppy decay in the input correlation matrix.

contribution of $P_1^{\sigma_w}(T)$ to the dimensionality of the hyperribbon is small for all times T. Our calculation shows that for all times T, the head of the eigenspectrum $P_1^{\sigma_w}$ decays rather quickly. For small times T, eigenvalues in the tail of $P_1^{\sigma_w}$ are quite small. The implication is that, everything else (i.e., P_1^y) being the same, models trained for long times have a higher-dimensional hyperribbon due to variations caused by the initialization of weights. For short times, the hyperribbon has a smaller dimensionality.

3. Target contribution

The second term corresponding to $P_1^y(T)$ in Eq. (5) resembles the so-called reachability Gramian in systems theory. It is well-known that $P = \lim_{T\to\infty} P_1^y(T)$ is the unique solution to the discrete algebraic Lyapunov equation [35]

$$K_d P K_d^\top - P + yy^\top = 0.$$

In systems theory, this concept is used for model reduction, i.e., to identify a low-dimensional dynamical system that captures time-varying data from a larger system. The rate of decay of the singular values of the reachability Gramian characterizes the quality of this approximation. Singular values of the Gramian decay quickly [36–39] when K_d has some nice properties (e.g., normal, well-conditioned), and $yy^\top$ is approximately low rank. This is precisely the setting of our paper. To study $P_1^y(T)$, which is a finite sum, we write it as the difference between two Gramians:

$$TP_1^y(T) = \sum_{t=0}^{\infty} K_d^t yy^\top K_d^t - \sum_{t=0}^{\infty} K_d^t \big(K_d^T yy^\top K_d^T\big) K_d^t$$
$$= \sum_{t=0}^{\infty} K_d^t \big(yy^\top - K_d^T yy^\top K_d^T\big) K_d^t.$$

We summarize our results in three lemmas whose proofs are given in the Appendix. The following lemma shows that the eigenspectrum of $P_1^y(T)$ decays quickly.

Lemma 1. We have

$$\frac{\lambda_{1+2i}^y}{\lambda_1^y} \leqslant \frac{4\rho^{-2i}}{(1+\rho^{-4i})^2} < 4\rho^{-2i},$$

where

$$\rho = \exp\left(\frac{\pi^2}{2\ln\big(8\lambda_1^K/\lambda_n^K - 4\big)}\right).$$

The following lemma now gives a lower bound on the eigenvalue λ_1^y.

Lemma 2. If we denote $\tilde{\lambda}_i = \sum_{t=0}^{T-1}(1-\alpha\lambda_i^K)^{2t}$, then

$$\|y\|^2 \leqslant \lambda_1^y \leqslant \tilde{\lambda}_n \|y\|^2,$$

where the norm of the targets concentrates around the value

$$\|y\|^2 \approx n\sigma_*^2 \sum_{i=1}^{n} \lambda_i^K. \tag{11}$$

Notice that $\tilde{\lambda}_i = \lambda_i^{\sigma_w}/(\sigma_w^2\lambda_i^K)$.

Lemma 1 suggests that the eigenspectrum of P_1^y decays as $\exp(-i\pi^2/nc)$. In contrast to the decay of $P_1^{\sigma_w}$, this rate is independent of the training time T. Suppose now that σ_w^2 is small relative to σ_*^2. Since $\lambda_1^y \geqslant \|y\|^2 = n\sigma_*^2 \mathrm{tr}(K)$, the head of the eigenspectrum of P_1^y can be much taller than that of $P_1^{\sigma_w}$, even if the decay of the two is similar. In other words, the head of eigenspectrum of $P_1(T) = P_1^{\sigma_w}(T) + P_1^y(T)$ is determined by P_1^y and the tail is determined by $P_1^{\sigma_w}$.

It might seem counterintuitive that sloppier data, i.e., large c, leads to a slower decay in the eigenspectrum of P_1^y. But notice sloppier data will also lead to a faster decay in the tail of the eigenspectrum of $P_1^{\sigma_w}$. Specifically, in Fig. 4 the threshold upon the index i after which the eigenspectrum of $P_1^{\sigma_w}(T)$ decreases quickly is $i^* < \ln(2T\alpha)/c$. This threshold scales as $1/c$. Therefore, if one trains for small times T, then the eigenspectrum of the sum $P_1(T) = P_1^{\sigma_w}(T) + P_1^y(T)$ still decays after i^*, essentially dominated by $P_1^{\sigma_w}$. In other words, for the same T, sloppier the data, smaller the threshold i^* after which the eigenspectrum of $P_1(T)$ decays.

We should note that although Lemma 1 does show that the eigenspectrum of P_1^y decays, it is a loose upper bound. In our experiments, the decay of the eigenspectrum is typically about twice as fast. This is because our upper bound depends on the displacement rank of the matrix, and the decay rate of the bound for a matrix with a displacement rank of 2 is twice as slow as that of a matrix with a displacement rank of 1. In our case, the displacement rank is the rank of the matrix $yy^\top - K_d^T yy^\top K_d^T$ in $P_1^y(T)$ which is 2, but in our numerical experiments we observe the decay rate to be closer to the bound with displacement rank of 1. This suggests that there

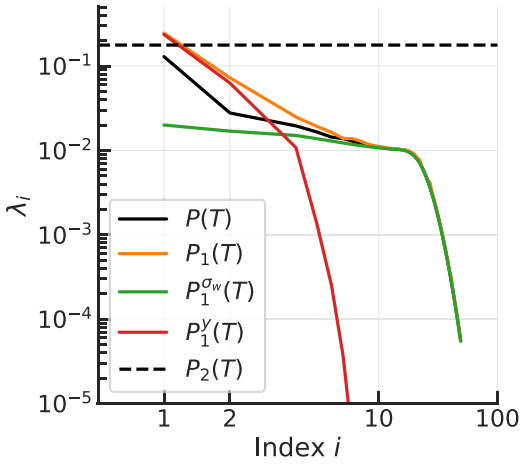


FIG. 5. Tail of the eigenspectrum of $P_1(T)$ is well-approximated by the contribution coming from the weight initializations $P_1^{\sigma_w}(T)$, while the head is well approximated by the contribution coming from the targets $P_1^{y}(T)$. These eigenvalues were computed for $d = 100$ dimensional data with slope $c = 0.2$ for the eigenvalues of the input correlation matrix, after fitting $n = 50$ samples for $T = 50$ iterations with initialization variance $\sigma_w^2 = 0.1$ and variance of the ground-truth weights being $\sigma_*^2 = 2$, this experiment uses $N = 100$ random initializations.

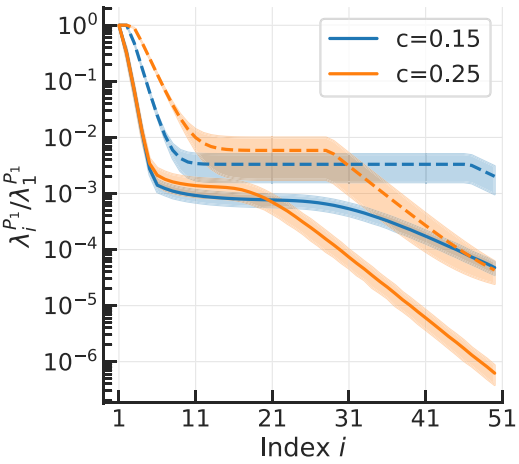


FIG. 6. Comparison of the bound in Lemma 3 (dashed) with eigenvalues of $P_1(T)$ computed directly from Eq. (5) (solid) for different values of sloppy decay rate c. We use 100 random initializations for each experiment, error bars in the above plot denote standard deviation across 100 numerical experiments. The proof of Lemma 3 works by computing the ideal point to apply Weyl's inequality. This enables us to separately calculate the decay in the head of the eigenspectrum and the tail, for both small and large training times T in spite of the fact that different parts of $P_1(T)$ in Eq. (5) dominate in different regimes.

could be better ways to exploit the displacement structure and improve our bounds. Note that if the labels y_i are aligned with an eigendirection of the data then $[yy^\top, K] = 0$, and the rank of the above matrix is 1. In this case, the training manifold geometry can be calculated analytically and is dominated by the decay of the initialization given in Eq. (9).

4. Combining the two parts to obtain the eigenspectrum of $\mathbf{P_1(T)}$

The following lemma combines the technical development in the previous two subsections.

Lemma 3. The eigenvalues of $P_1(T)$ in Eq. (5) satisfy

$$\frac{\lambda_i^{P_1}}{\lambda_1^{P_1}} \leqslant \min\left\{1, 4\rho^{-(i-1)} + \frac{\sigma_w^2}{\alpha\|y\|^2}\right\} \qquad \text{if } i \leqslant 2k^*,$$

$$\leqslant 4\rho^{-(k^*-1)} + \frac{\sigma_w^2}{\alpha\|y\|^2}\min\{1, 2\alpha T\lambda_{i-k^*+1}^K\} \qquad \text{else.}$$

for all $i = 1, \ldots, n$, where $k^* = \min\{\frac{\ln 2T\alpha}{2c}, \frac{n}{2}\}$.

Figure 6 compares the upper-bound on the eigenvalues derived from Lemma 3 against the eigenvalues computed directly from Eq. (5) for different values of sloppy decay rate c. This lemma correctly predicts the qualitative trends in the eigenspectrum, in the head of the spectrum where it decays quickly, in the intermediate plateau where eigenvalues do not decay, and in the tail where it again decays quickly. Figure 7 shows, using a numerical calculation, how the eigenspectrum of $P(N, T)$ in Eq. (3) changes for different training times T.

5. Phase diagrams

Figure 8 summarizes the development of this section using a phase diagram that describes the geometry of the hyperribbon in terms of the relevant parameters, the training time T, the slope c and the relative magnitude of the weight initialization σ_*/σ_w. Consider Fig. 8(a). For small initialization variance $\sigma_*/\sigma_w \gg 1$, the hyperribbon is very low-dimensional for most training times T and slope c.

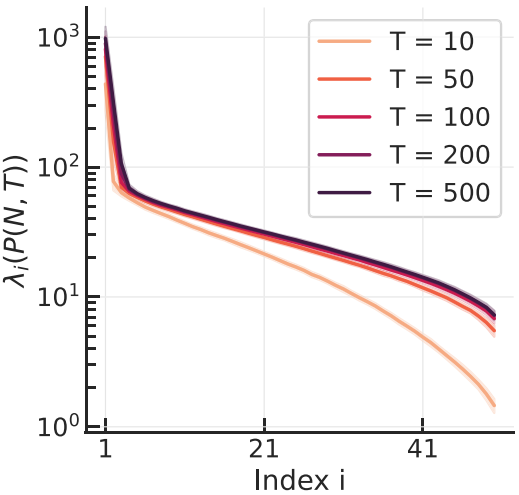
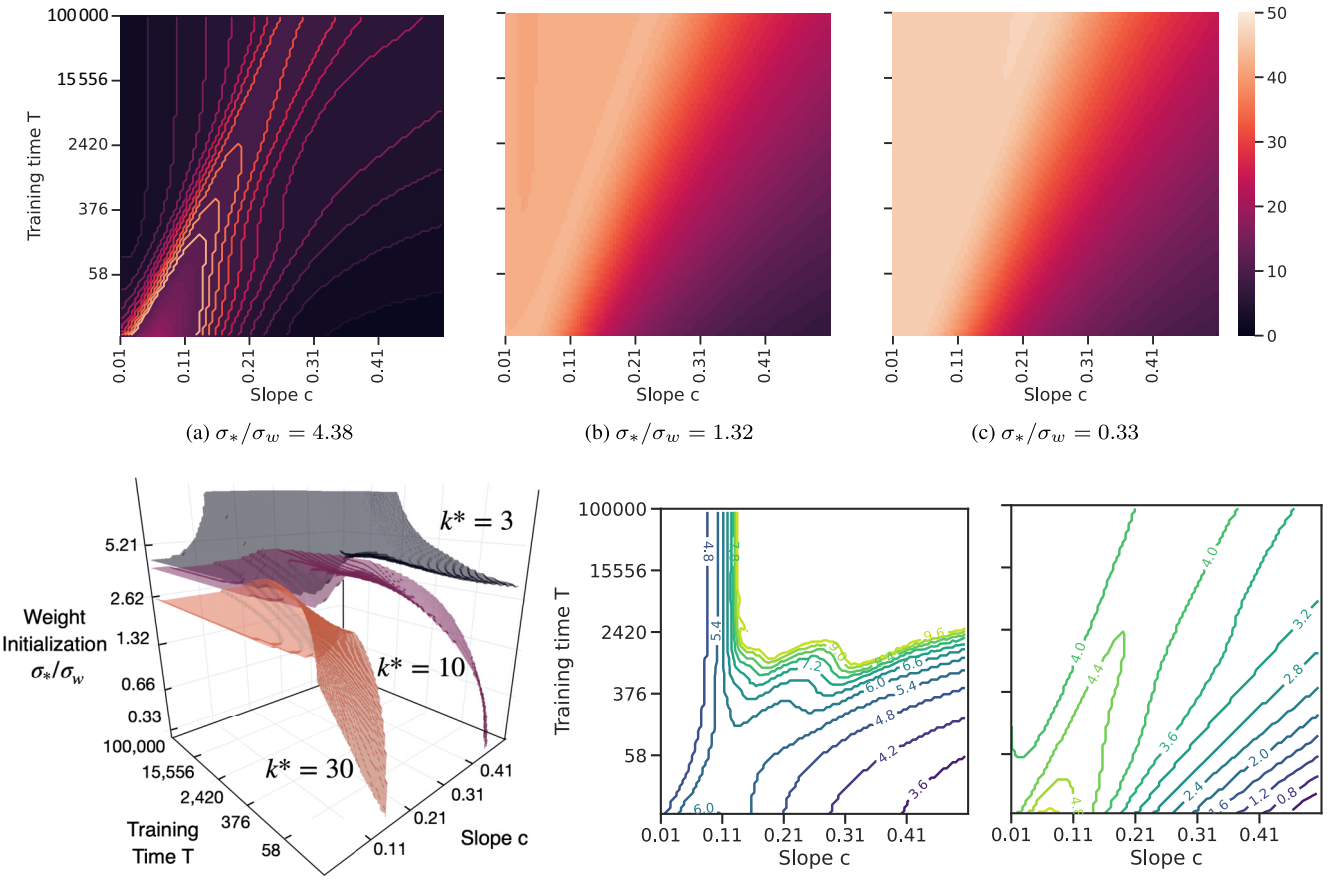


FIG. 7. Eigenvalues of $P(N, T)$ for different training times T from numerical experiments on linear models with $d = 100$ dimensional data with $n = 50$ training samples, slope $c = 0.1$, initialization variance $\sigma_w^2 = 1$ and learning rate $\alpha = 1/\lambda_1^K$. We use 100 random initializations for each experiment, error bars in the above plot denote standard deviation across 100 numerical experiments. As training time T increases, the eigenvalues in the tail increase in magnitude, this is because $P_1^{\sigma_w}(T)$. See Fig. 4. This also causes an increase in the largest eigenvalue in the head, due to the diminishing magnitude of $P_2(T)$ in Eq. (8).



(a) $\sigma_*/\sigma_w = 4.38$ (b) $\sigma_*/\sigma_w = 1.32$ (c) $\sigma_*/\sigma_w = 0.33$

(d) Transitions between phases of the hyper-ribbon (e) $k^* = 3$ dimensions (f) $k^* = 10$ dimensions

FIG. 8. Phase diagram for linear regression that describes the number of dimensions in the hyperribbon, i.e., the number of dimensions required to capture 95% of the variance of the points on the training manifold. This is studied with respect to three parameters: (i) the training time T, (ii) slope c, and (iii) the relative magnitude of weight initialization σ_*/σ_w. Panels (a)–(c) show a heat-map of the dimensionality of the hyperribbon for different training times T and slopes c for three different regimes of weight initialization. Panel (d) is a three-dimensional plot that depicts the boundaries of the different phases, defined by the dimensionality of the hyperribbon (3 dimensions in black, 10 in pink, and 30 in orange). Panels (e)–(f) show contours for different values of σ_*/σ_w for three- and ten-dimensional hyperribbons, respectively. See the narrative for an elaborate discussion.

The eigenspectrum is dominated by P_1^y in this case and its fast decay allows for lower-dimensional hyperribbons. There appears to be a straight line ($\log T \propto c$) along which the dimensionality is larger, due to relative magnitudes of $P_1^{\sigma_w}$ and P_1^y in Eq. (5). The matrix P_1^y results in a higher-dimensionality for large c while $P_1^{\sigma_w}$ is the cause of higher-dimensionality at relatively small values of c and large T. For small T, the relatively large magnitude of λ^{P_2} will reduce the initial part of the eigenspectrum (coming mostly from P_1^y), resulting in a higher dimensionality. If the initialization variance is small, short training times do not fit the data well. For large c, this causes the hyperribbon to be low-dimensional (roughly, because the condition number of optimization is large and different models end up being rather similar). The majority of experiments in Ref. [1] lie in this regime. For small c, this is evident as a higher-dimensional hyperribbon (roughly, because models are initialized in different subspaces of the data). For longer training times T, different models fit the data very well when c is small (again, because of a benign condition number). This is evident as a low-dimensional hyperribbon above the straight line.

Next consider Figs. 8(b) and 8(c). As the initialization variance increases, the apparent straight line $\log T \propto c$ that distinguishes low-dimensional hyperribbons from higher-dimensional ones, is still present. The upper-left region is increasingly higher-dimensional. For small slope c the hyperribbon is high-dimensional for all training times T. Because, models are initialized in very different subspaces of the data, and this is true for all three plots, except that it becomes more apparent as σ_*/σ_w decreases. For large c, for small times, the hyperribbon may be low-dimensional but we need much longer times to fit this data well. Reference [1] (Figs. 10, S.10,S.16) showed that when neural networks are initialized very far away from standard initializations, the hyperribbon is not low-dimensional. The dimensionality further increases when input data is not sloppy. Their experiments lie in regimes Figs. 8(b) and 8(c).

The training dynamics of deep linear networks has an initial "alignment" phase where weights rotate from their initial values toward the eventual solution, and a second phase where the dynamics predominantly changes the magnitude of the weights [4,40]. Linear networks considered here do not have the initial alignment phase, so our results in Fig. 8(a) hold for the deep linear networks in the second phase. In other words, we conjecture that the dimensionality of the training manifold in Fig. 8(a) is a lower-bound for the dimensionality of the manifold of an equivalent deep linear network. The contribution to the dimensionality coming from the initial alignment phase diminishes as the number of weight updates T increases.

Next consider Fig. 8(d). A three-dimensional plot that depicts the boundaries of the different phases, defined by the dimensionality of the hyperribbon (3 dimensions in black, 10 in pink, and 30 in orange). Some broad trends are apparent in the 3D plot, e.g., (i) large σ_*/σ_w leads to a low-dimensional hyperribbon, (ii) the geometry of the hyperribbon is very sensitive to other parameters when the slope c is small. As one goes from small T, large slope c and large σ_*/σ_w, to large times, small slope and small σ_*/σ_w, the dimensionality of the hyperribbon increases. The other panels in this figure are obtained by projecting this phase diagram upon different axes.

Figures 8(e) and 8(f) show contours for different values of σ_*/σ_w for three- and ten-dimensional hyperribbons, respectively. Focus on the contour marked 4.8, the two left and right wings of this contour together lead to a slice of Fig. 8(a) at a fixed dimension of three. Figure 8(e) indicates that there is a contiguous region in $(T, c, \sigma_*/\sigma_w)$-space with $\sigma_*/\sigma_w \gtrapprox 9$ where the hyperribbon has fewer than three dimensions. In Fig. 8(f) such contiguous regions occur at small values of σ_*/σ_w.

Altogether, Figs. 8(d)–8(f) shed light on thumb-rules for identifying the complexity of models that would be required to fit data in these different regimes. Given a dataset (a proxy for its complexity would be c), a training recipe (a proxy of which would be σ_*/σ_w) and training budget (a proxy of which would be T), the boundary of the phase diagram indicates the smallest model that one needs to achieve a good fit. For example, if our regime lies below the orange surface, then we need to fit a model with a larger number of parameters.

Sloppy models such as deep networks exhibit spectral bias where the training dynamics fits to the top eigen-components of the data at early times [41–43]. Low-dimensional training manifolds, identified experimentally in Ref. [1] and analytically in this paper, are not directly related to this phenomenon. The training manifold is low-dimensional due to the similarity of (the predictions of models along) trajectories initialized at different locations, e.g., when networks fit similar subspaces, not necessarily the top eigenspace, at similar rates. Low-dimensional training manifolds exist even when the network fits to the data well, e.g., in Fig. 8(a), the dimensionality is small for very large times, especially when $c \ll 1$. Spectral bias does imply that the training manifold is low-dimensional for a single architecture. But since different architectures have different spectral biases [44], this does not capture the low-dimensionality of the training manifolds for diverse architectures that was reported experimentally, especially in view of Ref. [1], Fig. 11 and Ref. [3] which showed that different architectures, optimization and regularization-based configurations fit the same easy images in the dataset first and the same challenging images toward the end of training.

III. VARIANTS OF THE LINEAR MODEL

We next extend our analysis to (i) stochastic optimization algorithms, (ii) ℓ_2 regularization of the weights, and (iii) kernel machines. In all three cases, we will see that the PCA matrix $P(N, T)$ in Eq. (3) undergoes minor changes and can be analyzed using Lemma 3 to show that the training manifold is low-dimensional. This section sheds further light into explaining the experimental results of Ref. [1]—just like we showed in the prequel that the manifold of networks with the same architecture (linear) trained with gradient descent from different initializations is low-dimensional, the manifolds obtained from different optimization algorithms, regularization techniques and architectures are also low-dimensional. In each of the three settings, we will also be able to interpret the decay of the eigenvalues of $P(N, T)$ to provide thumb-rules for hyperparameter selection.

1. Stochastic gradient descent

Let us now consider stochastic gradient descent for the linear predictive model. In this case, the weights $w \in \mathbb{R}^d$ are updated, not using the gradient on the entire objective as before $w_{t+1} = w_t - \alpha \partial_w C(w_t)$, but instead as

$$w_{t+1} = w_t - \frac{\alpha}{2\lfloor} \partial_w \left\{ \sum_{i=1}^{\lfloor} r^2_{\omega_i}(w_t) \right\},$$

where the random variable ω_i is uniformly distributed on $\{1, \ldots, n\}$ and $\lfloor$ is the batch size. We can model SGD as gradient descent perturbed by state-dependent Langevin noise

$$w_{t+1} = w_t - \alpha \partial_w C(w_t) + (\alpha/\sqrt{\lfloor})\xi_t, \tag{12}$$

where $\xi_t \sim N(0, D)$ where $D = X^\top X/n - \bar{x}^\top \bar{x}$ with $\bar{x} = n^{-1} \sum_{i=1}^n x_i$ is the covariance matrix of the inputs [22]. Under this dynamics of the weights, the residuals evolve as

$$r_{t+1} = (I - \alpha K) r_t + (\alpha/\sqrt{\lfloor}) X \xi_t.$$

The PCA matrix $P(N, T)$, as the number of random initializations N goes to infinity, becomes

$$P(T) = \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\left[r_t r_t^\top\right] - K_T y y^\top K_T,$$

where K_T is the same matrix as in Eq. (5). Notice here the randomness of r_t comes from both random initialization and noise from Langevin dynamics, and we are taking expectation with both sources of randomness, assuming they are

independent. We now have

$$\begin{aligned}\mathbb{E}\big[r_{t+1}r_{t+1}^\top\big] &= K_d\mathbb{E}\big[r_t r_t^\top\big]K_d^\top + 2(\alpha/\sqrt{\lfloor})K_d\mathbb{E}[r_t]\mathbb{E}[\xi_t]X^\top \\ &\quad + (\alpha^2/\lfloor)X\mathbb{E}\big[\xi_t\xi_t^\top\big]X^\top \\ &= K_d\mathbb{E}\big[r_t r_t^\top\big]K_d^\top + (\alpha^2/\lfloor)XX^\top XX^\top \\ &= K_d\mathbb{E}\big[r_t r_t^\top\big]K_d + (\alpha^2/\lfloor)K^2,\end{aligned}$$

where we recall that $K_d = I - \alpha K$.

If we define $P_\xi = (\alpha^2/\lfloor)\sum_{t=0}^\infty K_d^t K^2 K_d^t$, then P_ξ solves the Lyapunov equation $K_d P_\xi K_d^\top - P_\xi + (\alpha^2/\lfloor)K^2 = 0$, and it can be checked by induction that

$$\begin{aligned}\sum_{t=0}^{T-1}\mathbb{E}\big[r_t r_t^\top\big] &= TP_\xi + \sum_{t=0}^{T-1}K_d^t\big(\mathbb{E}\big[r_0 r_0^\top\big] - P_\xi\big)K_d^t \\ &= \sum_{t=0}^{T-1}K_d^t\big(\sigma_w^2 K + yy^\top\big)K_d^t + TP_\xi - \sum_{t=0}^{T-1}K_d^t P_\xi K_d^t,\end{aligned}$$

which has two additional terms compared to Eq. (5). Notice that K_d commutes with K^2 so P_ξ has the explicit expression

$$P_\xi = (\alpha^2/\lfloor)\big(I - K_d^2\big)^{-1}K^2 = (\alpha/\lfloor)(2I - \alpha K)^{-1}K.$$

Since P_ξ commutes with K_d, we can simplify

$$\begin{aligned}\sum_{t=0}^{T-1}K_d^t P_\xi K_d^t &= (\alpha/\lfloor)\sum_{t=0}^{T-1}K_d^{2t}(2I - \alpha K)^{-1}K \\ &= (1/\lfloor)(2I - \alpha K)^{-2}\big(I - K_d^{2T}\big).\end{aligned}$$

For $\alpha\lambda_i^K < 1$ the ith eigenvalue of the above matrix is $\simeq T\lambda_i^K/(2\lfloor)$. In Lemma 3, the upper bound on the eigenvalues of $P_1(T)$ for $i \leqslant 2k^*$ is perturbed by largest eigenvalue of the two additional terms. This is at most the largest eigenvalue of P_ξ, which is simply $\lambda_1(P_\xi) = (\alpha/\lfloor)(\lambda_1^K/(2 - \alpha\lambda_1^K)) \leqslant \alpha/\lfloor$ for our setting where $\lambda_1^K = 1$ and $\alpha < 1$ (which ensures the convergence of the infinite sum in P_ξ). In other words in Lemma 3, we will have

$$\frac{\lambda_i^{P_1}}{\lambda_1^{P_1}} \leqslant \min\left\{1, 4\rho^{-(i-1)} + \frac{1}{\|y\|^2}\left(\frac{\sigma_w^2}{\alpha} + \frac{\alpha}{\lfloor}\right)\right\}$$

for $i \leqslant 2k^*$.

This indicates an interesting relationship between the variance of weight initialization σ_w^2, the learning rate α and the batch size $\lfloor$. Suppose we wish to keep the volume of the ensemble of trajectories, as measured by the volume of the hyperribbon in residual space, the same. This is a reasonable because it indicates the propensity to identify good fits within the ensemble. Suppose we are in the regime where $\alpha \propto \lfloor$, which is very common while training neural networks. If we pick a batch size that is twice as large, the first term σ_w^2/α shrinks by a factor of two. To compensate—to keep the decay rate and effectively the dimensionality of the hyperribbon the same—we must pick a weight initialization variance that is twice as large.

2. Weight decay

The least-squares objective Eq. (1) is often "regularized" to be $C(w) + \frac{\lambda}{2}\|w\|_2^2$ to obtain a fit where the weights have a small ℓ_2-norm. This is important in the data scarce regime, i.e., $n < d$, where there may be multiple solutions to the nonregularized problem. The residual dynamics can be seen to be $r_{t+1} = (I - \alpha K_\lambda)r_t$ where $K_\lambda = XX^\top + \lambda I_{n\times n}$. All our calculations in Sec. II hold with K replaced by K_λ, i.e., with each $\lambda_i^{K_\lambda} = \lambda_i^K + \lambda$. For example, from Lemma 3, for $i \leqslant 2k^*$ we have

$$\frac{\lambda_i^{P_1}}{\lambda_1^{P_1}} \leqslant \min\left\{1, 4\rho^{-(i-1)} + \frac{1}{\|y\|^2}\left(\frac{\sigma_w^2}{\alpha} + \lambda\right)\right\},$$

which indicates that the decay rate of the eigenvalues is now a balance between the regularization parameter λ and the ratio σ_w^2/α.

3. Kernel machines

Consider predictions given by $\hat{y}_i = f(x_i)$ where the predictor f is not linear, but it belongs to a class of functions $\mathcal{F}$ with some regularity properties. A classical example of such a class of functions is called reproducing kernel Hilbert space (RKHS) which is a Hilbert space with the "reproducing kernel property." This property states that for any input datum x, there exists a function $\varphi_x \in \mathcal{F}$ such that the evaluation of $f \in \mathcal{F}$ at the input x can be written as an inner product $f(x) = \langle f, \varphi_x\rangle_{\mathcal{F}} = \int f(x')\varphi_x(x')\mathrm{d}x'$. The function

$$k(x, x') = \langle\varphi_x, \varphi_{x'}\rangle_{\mathcal{F}} \geqslant 0$$

is called the reproducing kernel. Gradient descent in RKHS [45] to minimize the objective in Eq. (1) corresponds to updates of the form

$$\forall x:\ f_{t+1}(x) = f_t(x) - \frac{\alpha}{n}\sum_{i=1}^n (f_t(x_i) - y_i)k(x_i, x).$$

Notice that the residuals $r_i = f_t(x_i) - y_i$ for all $i = 1, \ldots, n$ follow linear dynamics, same as those in Eq. (2), namely, $r_{t+1} = (I - \alpha K)r_t$ except that we now have $K_{ij} = k(x_i, x_j)/n$ for any i, j. This matrix is called the Gram matrix and it is positive semidefinite by Mercer's theorem [46]. In other words, all our development in the previous section holds for trajectories of kernel machines initialized from different initial conditions.

If the input correlation matrix is sloppy, then the Gram matrix is sloppy. This is easiest to see if x comes from a high dimensional distribution, i.e., $d \to \infty$ as $n \to \infty$ with a fixed d/n. Results in random matrix theory [47] state that the Gram matrix K can be well-approximated by the sample covariance matrix in such cases. And therefore, kernel machines are rather similar to linear models. If inputs x are drawn from a distribution with density $p(x)$ supported on $\mathbb{R}^d$, as $n \to \infty$, then the kth eigenvalue of the Gram matrix K converges to the $k$the eigenvalue of the integral operator $T[\varphi](x) = \int k(x, x')\varphi(x')p(x')\mathrm{d}x'$ [48]. The eigenspectrum of such integral operators has been studied, e.g., if $p(x)$ is Gaussian and $k(x, x') \propto \exp(-\|x - x'\|^2/d)$ is highly local, then eigenvalues of the Gram matrix K decay exponentially [49]. For translation invariant kernels, the decay is related to how quickly $p(x)$ goes to zero with increasing $\|x\|$ and to the Fourier transform of the kernel k [50]. In other words, the

Gram matrix K can also be sloppy even if the input correlation matrix is not.

Let us now consider the space of training trajectories corresponding to different kernel models. Let $\{K^{(m)}\}_{m=1}^{M}$ be M different kernel machines with trajectories in the residual space defined by $r_{t+1}^{(i,m)} = (I - \alpha K^{(m)})^{t+1} r_0^{(i,m)}$ for the ith initialization. This suggests that we should study the covariance matrix

$$P(N, M, T) = \frac{1}{NMT} \sum_{i,m,t} \left(r_t^{(i,m)} - \bar{r}\right)\left(r_t^{(i,m)} - \bar{r}\right)^{\top},$$

where the mean $\bar{r} = \frac{1}{MNT} \sum_{i,m,t} r_t^{(i,m)}$. Taking $N \to \infty$ as before, we get

$$P(M, T) \equiv \frac{1}{M} \sum_{m=1}^{M} P_1^{(m)}(T) - P_2(M, T),$$

where $P_2(M, T) = K_{T,M} y y^{\top} K_{T,M}$, with

$$K_{T,M} = \frac{1}{MT} \sum_{m=1}^{M} \sum_{t=0}^{T-1} K_d^{(m)t}.$$

The above equation is the analog of Eq. (5). Similar to our previous analysis, $P_2(M, T)$ has a single eigenvalue that vanishes for large T. We can thus obtain a result that is rather similar to the one in Lemma 3 with appropriate substitutions $\rho \to \rho^{(m)}$ and $\lambda_i^K \to \lambda_i^{K^{(m)}}$, assuming different kernels have different slopes $c^{(m)}$ but the same largest eigenvalue, i.e., the same learning rate α. Now by Weyl's inequality,

$$\lambda_{M(i-1)+1}\left(\sum_{m=1}^{M} P_1^{(m)}(T)\right) \leqslant \sum_{m=1}^{M} \lambda_i\left(P_1^{(m)}(T)\right),$$

and the fact that $\lambda_1(\sum_m P_1^m) \geqslant M\|y\|^2$, the right-hand side of the per-kernel version of Lemma 3 can be summed up over m. Altogether, the eigenvalues of $TP_1(M, T) \equiv TP_1$ satisfy

$$\frac{\lambda_i^{P_1}}{\lambda_1^{P_1}} \leqslant \min\left\{1, \frac{4}{M} \sum_m \rho^{(m)^{-\lfloor \frac{i-1}{M} \rfloor}} + \frac{\sigma_w^2}{\alpha\|y\|^2}\right\} \tag{13}$$

if $i \leqslant 2k^*$ and otherwise we have

$$\frac{\lambda_i^{P_1}}{\lambda_1^{P_1}} \leqslant \frac{1}{M}\left\{\sum_m 4\rho^{(m)^{-(k^*-1)}} + \frac{\sigma_w^2 \min\left\{1, 2\alpha T \lambda^{K^{(m)}}_{\lfloor \frac{i-1}{M} \rfloor - k^* + 2}\right\}}{\alpha\|y\|^2}\right\}$$

for all $i = 1, \ldots, n$, where $k^* = \min_m \ln(2T\alpha)/(2c^{(m)})$. This is a loose upper bound, because it uses the floor $\lfloor \frac{i-1}{M} \rfloor$ in the exponent of $\rho^{(m)}$. But due to the averaging over m, if different kernels have similar condition numbers, i.e., similar $\rho^{(m)}$, then the decay rate of eigenvalues $\lambda_i^{P_1}$ is shallower by a factor of M. But they do decay, and we should expect the hyperribbon to be low-dimensional. For the second expression when $i > 2k^*$, the second term (coming from $P_1^{\sigma_w}$) dominates the eigenspectrum. It has become worse due to the presence of $\lambda^{K^{(m)}}_{\lfloor \frac{i-1}{M} \rfloor - k^* + 2}$. But we can see that it still indicates a decay in the eigenspectrum at large i.

This narrative gives intuition into the experiments of Ref. [1] which are discussed in Sec. I, where the training manifold for different neural architectures was computed to find that the explained variance of the top few dimensions was quite high. Our discussion suggests that this can arise only if the "effective kernels" of all those networks have Gram matrices that decay quickly. If some of them do not decay quickly, then the explained variance would be low.

IV. DISCUSSION

1. Contextualization in terms of the broader literature on sloppy models

The central contribution of this paper is a technical result on characterizing the geometry of the manifold of predictions of linear models as they train on different types of data, for different durations, and using different types of architectures and training methods. We argued that there are broad similarities in the training trajectories of linear models and deep networks, not in the weight space where there are vast differences, but in the prediction space. Just like deep networks evolve on low-dimensional manifolds during training, linear models also evolve on low-dimensional manifolds. In the latter case however, tools in dynamics systems theory allow us to characterize the geometry very precisely. We showed that hyperribbon-like training manifolds in linear models are controlled by three factors: (i) sloppiness of the input data, (ii) strength of the weight initialization, and (iii) the number of gradient updates.

It informs our growing understanding of sloppy models and information geometry for the analysis of multiparameter models. Sloppiness is described better, not as the geometry of the set of predictions of the model for different values of its parameters, but rather as the restriction to the set of possible predictions on new data after fitting the model. The former is a property of the functional form of the model and the statistics of the input data. But the latter is also a property of the task—the questions we ask of the model. When the task depends on collective behavior of the system, rather than individual parameters, multiparameter models, including deep networks, are sloppy. For such models, further training on new data only makes small changes to the prediction space. This is yet another example of the emergence of simplicity from complexity in physics [16].

Let us note that we do not need geometric decay in the eigenvalues of the Gram matrix for our arguments in this paper to be valid, this assumption is adopted essentially to enable analytic computations. Figure 2 shows the eigenspectra for three different real datasets across diverse data modalities. Such eigenvalue plots are often plotted on a log-log scale (which gives the famous power laws [51,52]) or in terms of the density of the eigenvalues (in random matrix theory [20,21]). These are different ways of studying the same phenomenon, namely the decay in the eigenvalues. The important structure in natural data lies in the head of the eigenspectrum—decay in the head reflects how salient variations in the data diminish in importance. The log-log scale emphasizes trends in the mid/tail of the eigenspectrum and hides the trends in the head of the eigenspectrum. While this may be important for some problems, in the context of our specific problem and we argue machine learning in general, it is important to study the decay in the head of the eigenspectrum more precisely.

2. *Perspective on generalization in deep learning*

Deep networks often have more parameters than the number of training data. Their remarkable real-world performance therefore defies foundational assumptions about the role of model parsimony in generalization. There is a large amount of literature that studies this question, ranging from the "double descent" paradigm [53–55], arguments that relate wide minima to generalization [20,56], implicit regularization toward low-complexity solutions during training [57], analyses of generalization in the kernel regime [33,58–60], and compression in the mutual information between inputs, representations, and outputs [26,61–63]. Emergent low-dimensional geometries termed "hyperribbons" have been argued to lead to generalization in sloppy models [29,30]. Roughly, hyperribbons refer to the shape of the set of all possible predictions on the training samples made by models within a hypothesis class. This is conceptually similar to notions of capacity in statistical learning theory except that it focuses on the geometry of the set, not just the volume.

Together, these diverse viewpoints suggest that low-dimensional structure—whether in the loss landscape, the data, or the training dynamics—plays a central role in enabling generalization in deep learning. And this commonality perhaps hints at a unified theory of statistical generalization. There is some recent work [64] that takes a step toward such a theory. The authors use tools from contraction theory [65] to show that the generalization gap for a general deep network trained using gradient flow after time t is given by $r(0)^\top G(t) r(0)$ where $r(0)$ is the vector of residuals at initialization and $G(t)$ is a certain "effective Gram matrix" that conceptually captures the volume of the hypothesis space explored during training. This result is similar to information-theoretic generalization bounds [66]. It can be shown that if the Gram matrix in the present paper K is sloppy (which was also observed by Ref. [26]), then this effective Gram matrix is also sloppy. In the context of the present paper, this result therefore suggests that low-dimensional training manifolds lead to effective generalization. Deep networks are a very large hypothesis class, and therefore effective generalization is difficult without a sufficiently large number of samples. The only way deep networks could generalize as well as they do is if the training process does not explore the entire hypothesis space. Our paper shows that if input data is sloppy and if the variance of the weights at initialization is small, then the training manifold spans a very small subset of the space of predictions—and this is perhaps why generalization is possible.

ACKNOWLEDGMENTS

We thank Andrea Liu for her probing question about sloppiness and deep neural networks. J.M. and P.C. were supported by the Office of Naval Research (Grant No. N00014-22-1-2255) and the National Science Foundation (Grants No. IIS-2145164 and No. CCF-2212519). J.P.S. and I.G. were supported by NSF Grant No. DMR-2327094. I.G. was also supported by an Eric and Wendy Schmidt AI in Science Postdoctoral Fellowship, and is grateful to the Statistical Physics of Computation laboratory at EPFL for their hospitality. Y.S. was supported by grants from the Army Research Office (ARO) and the Air Force Office of Scientific Research (AFOSR). M.K.T. was supported by NSF Grants No. DMR-1753357 and No. ECCS-2223985.

DATA AVAILABILITY

The data that support the findings of this article are openly available [67]; embargo periods may apply.

APPENDIX: PROOFS

Proof of Lemma 1. We will denote $TP_1^y(T)$ as P in this proof for clearer exposition. We know that P solves the discrete algebraic Lyapunov equation

$$K_d P K_d^\top - P + \left(yy^\top - K_d^T yy^\top K_d^T\right) = 0.$$

See Ref. [37], Chapter 4.3.3], where such a P also solves the continuous Lyapunov equation $\tilde{K}_d P + P\tilde{K}_d{}^\top + BB^\top = 0$ with

$$\begin{aligned}\tilde{K}_d &= (K_d + I)^{-1}(K_d - I), \quad BB^\top \\ &= 2(K_d + I)^{-1}\left(yy^\top - K_d^T yy^\top K_d^T\right)(K_d + I)^{-1}.\end{aligned}$$

Notice that $\tilde{K}_d$ is normal and has a bounded spectrum:

$$\sigma(K_d) \subseteq [-b, -a],$$

with $0 < a < b < \infty$. Notice that $BB^\top$ has a rank of at most 2. From Ref. [39], Theorem 2.1 and Corollary 3.2] we can conclude that

$$\lambda_{1+2i}^P \leqslant 4\rho^{-2i}\lambda_1^P, \qquad \text{with } \rho = \exp\left(\frac{\pi^2}{2\log(4b/a)}\right).$$

Notice that $\lambda_i^{\tilde{K}_d} = -\frac{\alpha\lambda_i^K}{2-\alpha\lambda_i^K}$, so with our assumption (iii) which states that $\alpha < 1/\lambda_1^K$, we have

$$\frac{b}{a} = \frac{\lambda_1^K}{\lambda_n^K}\frac{\left(2-\alpha\lambda_n^K\right)}{\left(2-\alpha\lambda_1^K\right)} \leqslant \frac{2\lambda_1^K}{\lambda_n^K} - 1.$$

■

Proof of Lemma 2. The lower bound is obtained by seeing that

$$\lambda_1^y \geqslant \max_t \lambda_1\left(K_d^t yy^\top K_d^t\right) \geqslant \|y\|^2,$$

and the upper bound is given by

$$\lambda_1^y \leqslant \sum_t \lambda_1\left(K_d^t yy^\top K_d^t\right) = \tilde{\lambda}_n \|y\|^2.$$

Since the true weights w^* are drawn from an isotropic normal distribution with covariance $\sigma_*^2 I$, the norm of the targets concentrates around the trace of the data correlation matrix $\sigma_*^2 \sum_{i=1}^n \lambda_i^K$. Indeed, $\|y\|^2 = (V^\top w^*)^\top S^2 (V^\top w^*)$, where V and S, are given by the singular value decomposition of the data matrix $X = USV^\top$. Since entries of $V^\top w^*$ are independent random variables, $\|y\|^2$ is concentrated around the above

value by the Hanson-Wright inequality

$$\mathbb{P}\left(|\|y\|^2 - \sigma_*^2 \mathrm{tr}(K)| > t\right) \leqslant 2\exp\left[-a\min\left(\frac{t^2}{\sigma_*^4 \mathrm{tr}(K)}, \frac{t}{\sigma_*^2}\right)\right],$$

where $a > 0$ is a constant independent of t, K and σ_*. ■

Proof of Lemma 3. To keep the notation clear, in the proof we will refer to $P_1^{\sigma_w}(T)$ and $P_1^{y}(T)$ as P^{σ_w} and P^{y}, respectively.

Let us first discuss the case for small times T. Notice that if $T < \frac{\lambda_1^K}{2\lambda_n^K}$, then $2T\alpha < \alpha\frac{\lambda_1^K}{\lambda_n^K} < e^{c(n-1)} < e^{cn}$. In this case, we have $k^* = \frac{\ln(2T\alpha)}{2c} < \frac{n}{2}$, so we can obtain the following upper bound on $\lambda_i^{P_1}$ by Weyl's inequality:

$$\lambda_i^{P_1} \leqslant \begin{cases} \lambda_i^y + \lambda_1^{\sigma_w} & \text{for } i \leqslant 2k^*, \\ \lambda_{k^*}^y + \lambda_{i-k^*+1}^{\sigma_w} & \text{for } i > 2k^*. \end{cases}$$

For $i \leqslant 2k^*$, we have

$$\frac{\lambda_i^{P_1}}{\lambda_1^{P_1}} \leqslant \frac{\lambda_i^{P_1}}{\lambda_1^y} \leqslant 4\rho^{-(i-1)} + \frac{\lambda_1^{\sigma_w}}{\lambda_1^y} \leqslant 4\rho^{-(i-1)} + \frac{\sigma_w^2 \min\left\{1, 2T\alpha\lambda_1^K\right\}}{\alpha\|y\|^2},$$

where we have used $\lambda_1^{P_1} \geqslant \lambda_1^y$ since P^{σ_w} is positive definite. Note that under assumptions (i-ii), $\lambda_1^K = 1$ and $\alpha < 1$, therefore we do not need the minimum in the second term. For $i > 2k^*$, we have

$$\frac{\lambda_i^{P_1}}{\lambda_1^{P_1}} \leqslant \frac{\lambda_{i-k^*+1}^{\sigma_w}}{\lambda_1^y} + 4\rho^{-(k^*-1)} \leqslant \frac{\sigma_w^2 \min\left\{1, 2\alpha T\lambda_{i-k^*+1}^K\right\}}{\alpha\|y\|^2} + 4\rho^{-(k^*-1)},$$

where we have used Eq. (10) and assumption (ii).

For large times $T \geqslant \frac{\lambda_1^K}{2\lambda_n^K}$, we can choose the splitting point for Weyl's inequality to be $n/2$. We now have

$$\frac{\lambda_i^{P_1}}{\lambda_1^{P_1}} \leqslant \frac{\lambda_{i-n/2+1}^{\sigma_w}}{\lambda_1^y} + \frac{\lambda_{n/2}^y}{\lambda_1^y} \leqslant \frac{\lambda_{i-n/2+1}^{\sigma_w}}{\lambda_1^y} + 4\rho^{-(n/2-1)}.$$

If $T \geqslant \frac{\lambda_1^K}{2\alpha\lambda_n^K}$, then $2T\alpha\lambda_i^K > 1$ for all $i \geqslant 1$, so by Eq. (10)

$$\lambda_{i-n/2+1}^{\sigma_w} \leqslant \frac{\sigma_w^2}{2\alpha - \alpha^2\lambda_{i-n/2+1}^K}$$

and

$$\frac{\lambda_{i-n/2+1}^{\sigma_w}}{\lambda_1^y} \leqslant \frac{\sigma_w^2}{\alpha\|y\|^2}\frac{1}{\left(2 - \alpha\lambda_{i-n/2+1}^K\right)} \leqslant \frac{\sigma_w^2}{\alpha\|y\|^2}.$$

For $\frac{\lambda_1(K)}{2\lambda_n(K)} \leqslant T < \frac{\lambda_1(K)}{2\alpha\lambda_n(K)}$, we may still use the split at $k^* = n/4$ in the above calculation, and take the minimum of the above two bounds for $i > n/2$. ■

[1] J. Mao, I. Griniasty, H. K. Teoh, R. Ramesh, R. Yang, M. Transtrum, J. P. Sethna, and P. Chaudhari, The training process of many deep networks explores the same low-dimensional manifold, Proc. Natl. Acad. Sci. USA **121**, e2310002121 (2024).

[2] S. Mallat, Group invariant scattering, Commun. Pure Appl. Math. **65**, 1331 (2012).

[3] G. Hacohen, L. Choshen, and D. Weinshall, Let's agree to agree: Neural networks share classification order on real datasets, In *Proceedings of the International Conference on Machine Learning* (PMLR, 2020), pp. 3950–3960.

[4] A. Shamir, O. Melamed, and O. BenShmuel, The dimpled manifold model of adversarial examples in machine learning, arXiv:2106.10151.

[5] R. Ramesh, J. Mao, I. Griniasty, R. Yang, H. K. Teoh, M. Transtrum, J. P. Sethna, and P. Chaudhari, A picture of the space of typical learnable tasks, in *Proceedings of the International Conference of Machine Learning (ICML)* (2023).

[6] M. K. Transtrum, B. B. Machta, and J. P. Sethna, The geometry of nonlinear least squares with applications to sloppy models and optimization, Phys. Rev. E **83**, 036701 (2011).

[7] K. N. Quinn, H. Wilber, A. Townsend, and J. P. Sethna, Chebyshev approximation and the global geometry of model predictions, Phys. Rev. Lett. **122**, 158302 (2019).

[8] A. Krizhevsky, Learning multiple layers of features from tiny images, Ph.D. thesis, Computer Science, University of Toronto.

[9] S. Merity, C. Xiong, J. Bradbury, and R. Socher, Pointer sentinel mixture models, in *Proceedings of the International Conference of Learning and Representations (ICLR)* (2017).

[10] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, BERT: Pre-training of deep bidirectional transformers for language understanding, in *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, (2019), Vol. 1, pp. 4171–4186.

[11] J. F. Gemmeke, D. P. W. Ellis, D. Freedman, A. Jansen, W. Lawrence, R. C. Moore, M. Plakal, and M. Ritter, Audio set: An ontology and human-labeled dataset for audio events, in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (IEEE, Piscataway, NJ, 2017), pp. 776–780.

[12] S.-i. Amari, *Information Geometry and Its Applications*, Applied Mathematical Sciences Vol. 194 (Hikari, Tokyo, 2016).

[13] K. N. Quinn, C. B. Clement, F. De Bernardis, M. D. Niemack, and J. P. Sethna, Visualizing probabilistic models and data with intensive principal component analysis, Proc. Natl. Acad. Sci. USA **116**, 13762 (2019).

[14] M. A. A. Cox and T. F. Cox, Multidimensional scaling, *Handbook of Data Visualization* (Springer, Berlin, 2008), pp. 315–347.

[15] H. Hotelling, Analysis of a complex of statistical variables into principal components, J. Edu. Psychol. **24**, 417 (1933).

[16] K. N. Quinn, M. C. Abbott, M. K. Transtrum, B. B. Machta, and J. P. Sethna, Information geometry for multiparameter models: New perspectives on the origin of simplicity, *Reports on Progress in Physics* (IOP, Bristol, UK, 2022).

[17] K. S. Brown, Signal transduction, sloppy models, and statistical mechanics, Ph.D. Thesis, Cornell

University (2003), https://www.proquest.com/openview/b60abc9b1991b41e6b7b0b3151408691.
[18] R. W. Keener, *Theoretical Statistics: Topics for a Core Course, Springer Texts in Statistics* (Springer, New York, 2010).
[19] M. K. Transtrum and P. Qiu, Model reduction by manifold boundaries, Phys. Rev. Lett. **113**, 098701 (2014).
[20] P. Chaudhari, A. Choromanska, S. Soatto, Y. LeCun, C. Baldassi, C. Borgs, J. Chayes, L. Sagun, and R. Zecchina, Entropy-SGD: Biasing gradient descent into wide valleys, in *Proceedings of the International Conference of Learning and Representations (ICLR)* (2017).
[21] V. Papyan, The full spectrum of deepnet Hessians at scale: Dynamics with SGD training and sample size, arXiv:1811.07062.
[22] P. Chaudhari and S. Soatto, Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks, In *Proceedings of the International Conference of Learning and Representations (ICLR)* (2018).
[23] L. Dinh, R. Pascanu, S. Bengio, and Y. Bengio, Sharp minima can generalize for deep nets, in *Proceedings of the 34th International Conference on Machine Learning*, edited by D. Precup and Y. W. Teh (2017), Vol. 70, pp. 1019–1028.
[24] K. Sun and F. Nielsen, A geometric modeling of Occams razor in deep learning, Info. Geo. **8**, 233 (2025).
[25] S. Watanabe, Almost all learning machines are singular, In *Proceedings of the IEEE Symposium on Foundations of Computational Intelligence* (2007), pp. 383–388.
[26] R. Yang, J. Mao, and P. Chaudhari, Does the data induce capacity control in deep learning? in *Proceedings of the International Conference of Machine Learning (ICML)* (2022).
[27] R. Ramesh, A. Bisulco, R. W. DiTullio, L. Wei, V. Balasubramanian, K. Daniilidis, and P. Chaudhari, Many perception tasks are highly redundant functions of their input data, in Computational and Systems Neuroscience COSYNE, arXiv:2407.13841.
[28] B. B. Machta, R. Chachra, M. K. Transtrum, and J. P. Sethna, Parameter space compression underlies emergent theories and predictive models, Science **342**, 604 (2013).
[29] J. J. Waterfall, F. P. Casey, R. N. Gutenkunst, K. S. Brown, C. R. Myers, P. W. Brouwer, V. Elser, and J. P. Sethna, Sloppy-Model universality class and the Vandermonde matrix, Phys. Rev. Lett. **97**, 150601 (2006).
[30] J. Waterfall, Universality in multiparameter fitting: Sloppy models, Ph.D. thesis, Cornell University, 2006.
[31] G. Cybenko, Approximation by superpositions of a sigmoidal function, Math. Control Signals Syst. **2**, 303 (1989).
[32] R. Ramesh, J. Mao, I. Griniasty, R. Yang, H. K. Teoh, M. Transtrum, J. Sethna, and P. Chaudhari, A picture of the space of typical learnable tasks, in *Proceedings of the International Conference of Machine Learning (ICML)* (2023).
[33] A. Jacot, F. Gabriel, and C. Hongler, Neural tangent kernel: Convergence and generalization in neural networks, in *Proceedings of the Conference on Advances in Neural Information Processing Systems 31* (2018), pp. 8571–8580.
[34] L. Bottou, Stochastic gradient descent tricks, In *Neural Networks: Tricks of the Trade* (Springer, Berlin, 2012), pp. 421–436.
[35] Biswa Nath Datta, ed., Numerical solutions and conditioning of Lyapunov and Sylvester equations. In *Numerical Methods for Linear Control Systems* (Academic Press, San Diego, CA, 2004), chap. 8, pp. 245–303.
[36] T. Penzl, Eigenvalue decay bounds for solutions of Lyapunov equations: The symmetric case, Syst. Control Lett. **40**, 139 (2000).
[37] A. C. Antoulas, Approximation of large-scale dynamical systems, Soc. J. Ind. Appl. Math. 389 (2005).
[38] A. Townsend and H. Wilber, On the singular values of matrices with high displacement rank, Linear Algebra Appl. **548**, 19 (2018).
[39] B. Beckermann and A. Townsend, On the singular values of matrices with displacement structure, SIAM J. Matrix Anal. Appl. **38**, 1227 (2017).
[40] A. Atanasov, B. Bordelon, and C. Pehlevan, Neural networks as kernel learners: The silent alignment effect, in *Proceedings of the International Conference on Learning Representations* (2022).
[41] A. M. Saxe, J. L. McClelland, and S. Ganguli, Exact solutions to the nonlinear dynamics of learning in deep linear neural networks, in *Proceedings of the International Conference of Learning and Representations (ICLR)* (2014).
[42] P. L. Bartlett, P. M. Long, G. Lugosi, and A. Tsigler, Benign overfitting in linear regression, Proc. Natl. Acad. Sci. USA **117**, 30063 (2020).
[43] B. Bordelon, A. Canatar, and C. Pehlevan, Spectrum dependent learning curves in kernel regression and wide neural networks, in *Proceedings of the 37th International Conference on Machine Learning* (PMLR, 2020), pp. 1024–1034.
[44] M. Choraria, L. T. Dadi, G. Chrysos, J. Mairal, and V. Cevher, The spectral bias of polynomial neural networks, in *Proceedings of the International Conference of Learning and Representations (ICLR)* (2022).
[45] Y. Yao, L. Rosasco, and A. Caponnetto, On early stopping in gradient descent learning, Construct. Approx. **26**, 289 (2007).
[46] B. Schölkopf and A. J. Smola, *Learning with kernels* (The MIT Press, 2002).
[47] N. El Karoui, The spectrum of kernel random matrices, Ann. Stat. **38**, 1 (2010).
[48] V. Koltchinskii and E. Giné, Random matrix approximation of spectra of integral operators, Bernoulli **6**, 113 (2000).
[49] H. Zhu, C. K. I. Williams, R. Rohwer, and M. Morciniec, *Gaussian Regression and Optimal Finite Dimensional Linear Models* (Aston University, Birmingham, UK, 1998).
[50] H. Widom, Asymptotic behavior of the eigenvalues of certain integral equations, Trans. Am. Math. Soc. **109**, 278 (1963).
[51] D. L. Ruderman, Origins of scaling in natural images, Vision Res. **37**, 3385 (1997).
[52] D. J. Field, What is the goal of sensory coding? Neural Comput. **6**, 559 (1994).
[53] M. Belkin, D. Hsu, S. Ma, and S. Mandal, Reconciling modern machine-learning practice and the classical bias–variance trade-off, Proc. Natl. Acad. Sci. USA **116**, 15849 (2019).
[54] D. Chen, W.-K. Chang, and P. Chaudhari, Learning capacity: A measure of the effective dimensionality of a model, arXiv:2305.17332.
[55] M. K. Transtrum, G. L. W. Hart, T. J. Jarvis, and J. P. Whitehead, Aliasing and label-independent decomposition of risk: Beyond the bias-variance trade-off, arXiv:2408.08294.
[56] G. E. Hinton and D. Van Camp, Keeping the neural networks simple by minimizing the description length of the weights, In

Proceedings of the 6th Annual Conference on Computational Learning Theory (1993), pp. 5–13.

[57] D. Soudry, E. Hoffer, M. S. Nacson, S. Gunasekar, and N. Srebro, The implicit bias of gradient descent on separable data, J. Mach. Learn. Res. **19**, 1 (2018).

[58] A. Canatar, B. Bordelon, and C. Pehlevan, Spectral bias and task-model alignment explain generalization in Kernel regression and infinitely wide neural networks, Nat. Commun. **12**, 2914 (2021).

[59] N. Mallinar, J. Simon, A. Abedsoltan, P. Pandit, M. Belkin, and P. Nakkiran, Benign, tempered, or catastrophic: Toward a refined taxonomy of overfitting, in *Advances in Neural Information Processing Systems*, edited by S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Curran Associates, Red Hook, NY, 2022), Vol. 35, pp. 1182–1195.

[60] A. Amini, R. Baumgartner, and D. Feng, Target alignment in truncated kernel ridge regression, Adv. Neural Info. Process. Syst. **35**, 21948 (2022).

[61] G. K. Dziugaite and D. M. Roy, Computing nonvacuous generalization bounds for deep (stochastic) neural networks with many more parameters than training data, in *Proceedings of the Conference on Uncertainty in Artificial Intelligence (UAI)* (2017).

[62] A. Xu and M. Raginsky, Information-theoretic analysis of generalization capability of learning algorithms, in *Proceedings of the Conference on Neural Information Processing Systems* (NIPS) (2017).

[63] A. Achille and S. Soatto, Emergence of invariance and disentanglement in deep representations, J. Mach. Learn. Res. **19**, 1947 (2018).

[64] R. Yang and P. Chaudhari, An effective gram matrix characterizes generalization in deep networks, arXiv:2504.16450.

[65] W. Lohmiller and J.-J. E. Slotine, On contraction analysis for non-linear systems, Automatica **34**, 683 (1998).

[66] G. Neu, G. K. Dziugaite, M. Haghifam, and D. M. Roy, Information-theoretic generalization bounds for stochastic gradient descent, in *Proceedings of the* Conference on Learning Theory (2021), pp. 3526–3545.

[67] J. Mao, I. Griniasty, Y. Sun, M. Transtrum, J. Sethna, and P. Chaudhari, Code for "An Analytical Characterization of Sloppiness in Neural Networks Insights from Linear Models" Zenodo (2026), doi: 10.5281/zenodo.18160170.