Associative Memory in Iterated Overparameterized Sigmoid Autoencoders

Yibo Jiang [1] **Cengiz Pehlevan** [2]

Abstract

Recent work showed that overparameterized autoencoders can be trained to implement associative memory via iterative maps, when the trained input-output Jacobian of the network has all of its eigenvalue norms strictly below one. Here, we theoretically analyze this phenomenon for sigmoid networks by leveraging recent developments in deep learning theory, especially the correspondence between training neural networks in the infinite-width limit and performing kernel regression with the Neural Tangent Kernel (NTK). We find that overparameterized sigmoid autoencoders can have attractors in the NTK limit for both training with a single example and multiple examples under certain conditions. In particular, for multiple training examples, we find that the norm of the largest Jacobian eigenvalue drops below one with increasing input norm, leading to associative memory.

1. Introduction

The mechanisms behind memory have been a long interest of neuroscientists. Hopfield's seminal work proposed that associative memory can be implemented by attractor neural dynamics (Hopfield, 1982), and has been the dominant model that shapes thinking in this domain (Hertz, 2018). Recently, Radhakrishnan et al. (2019; 2018) proposed an alternative mechanism and showed that overparameterized autoencoders trained with gradient descent could also implement associative memory in an iterative fashion. These networks are reported to be easy to train and to not suffer from spurious attractors, unlike Hopfield networks (Amit & Treves, 1989; Hertz, 2018). The potential benefits of this approach make a theoretical account of it necessary, which we aim to provide here.

[1]John A. Paulson School of Engineering and Applied Sciences, Harvard University, Cambridge, MA, USA [2]Center for Brain Science, Harvard University, Cambridge, MA, USA. Correspondence to: Cengiz Pehlevan <cpehlevan@seas.harvard.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

We study auto-encoders in a limit of neural networks that makes theoretical analysis possible. Specifically, as the width of the hidden layers of a feedforward neural network is taken to infinity with a particular initialization scheme, its training dynamics simplifies and can be described by ridgeless kernel interpolation with a kernel called the Neural Tangent Kernel (NTK) (Jacot et al., 2018). Working in the NTK limit, we examine the input-output Jacobian matrices of the trained networks as they control the stability of trained fixed points. We focus on networks with $\mathrm{sigmoid}$ activation functions, but our results can be extended to other sigmoidal functions such as erf and $\tanh$ using similar techniques. We make a distinction between the cases of a single training example and multiple training examples, which exhibit different memory behaviors.

Our main contributions and results are summarized below:

- First, we analyze autoencoders in the NTK limit trained on a single training example. We argue that the trained Jacobian will stay close to initialization under certain conditions. Therefore, if the initial Jacobian has all eigenvalue norms smaller than 1, this training example will be an **attractor**.
- Next, we specialize to 2-layer networks. We show that when the norms of training examples are small, attractor formation can **fail** due to the presence of eigenvalue 1 in the spectrum of trained Jacobian matrices.
- We show that a 2-layer network can have **attractors** when the norms of training examples are large as the induced NTK relies more on the non-linear, saturated region of the activation function. This suggests that the network transitions from a regime where attractor formation fails to a regime where it succeeds as the input norm grows.
- We verify the predictions of our theoretical results in simulations.

Many previous works on generalization (Allen-Zhu et al., 2019; Cao & Gu, 2019) and training (Jacot et al., 2018; Du et al., 2018; Allen-Zhu et al., 2018) in the NTK limit focus on input data on the unit sphere which is violated in practice (Krizhevsky et al., 2012). We highlight how the input norm can set trained neural networks in different learning regions with respect to the trained input-output Jacobian.

Associative memory behavior induced by training overparameterized autoencoders could provide insights into the

implicit bias and generalization of neural networks. Autoencoders are trained to learn identity maps, however the existence of attractors indicates failure to learn such maps and thus failure to generalize. Recently, Zhang et al. (2019) observed that fully connected networks tend to learn a constant function, a global attractor, when trained on a single example. Our single training example results explain this behavior. However, we also show that attractor formation is dependent on input norm when multiple examples are present, demonstrating that training with a single example is not sufficient to explain the implicit bias of neural networks.

2. Related Work

Our results use ideas related to NTK and signal propagation in deep networks with random weights. Attractor behavior can also be associated with the implicit bias of deep learning. We review relevant literature from these domains.

Neural Tangent Kernel: We first review literature on neural networks optimization, especially the NTK theory which we will use extensively through this paper. Training of neural networks poses a challenging non-convex optimization problem. Analysis simplifies if focused on the linearized training dynamics of gradient flow using the NTK theory (Jacot et al., 2018). The basic idea is that, if initialized properly, in the infinite width limit, parameters of the network stay close to initialization (Chizat et al., 2019). Thus, NTK stays relatively constant throughout training. Because NTK governs the training dynamics, positive-definite kernel ensures global convergence of optimization. Subsequent papers expanded this idea to finite network widths, gradient descent or stochastic gradient descent as opposed to gradient flow, and different loss functions for regression and classification (Du et al., 2018; Allen-Zhu et al., 2018; Zou & Gu, 2019; Lee et al., 2019). Also related is research pointing that neural networks at initialization in the infinite width limit behave as Gaussian Processes (Lee et al., 2017; Matthews et al., 2018).

Signal Propagation in Deep Networks with Random Weights: In a set of ideas that we will make use of later, Poole et al. (2016) developed a mean-field formalism to study layer-to-layer propagation of activation variances and covariances in deep networks with random weights. This line of work (Poole et al., 2016; Schoenholz et al., 2016) identifies a phase transition between ordered and chaotic regime, where nearby input points converge or diverge as they propagate through the layers, induced by different variances of weight and bias initializations. Using random matrix theories developed for Gram matrices of neural networks (Pennington & Worah, 2017), Pennington et al. (2017; 2018) calculate singular value spectra for input-output Jacobians at initialization and identify its relation to ordered/chaotic training regime. These results cannot be applied to our problem, as they are in a different setting assuming a large depth limit such that the variances for all layers are at the fixed points of the layer-to-layer iterative maps.

Generalization and Implicit Bias: Associative memory behavior of neural networks can provide insight into generalization of deep learning and implicit bias of gradient descent. Here, we review some recent works in this area, focusing on overparameterized networks and NTK. A pair of recent papers (Hayou et al., 2019; Xiao et al., 2019) demonstrate how to use the theories developed for signal propagation to understand NTK regression better and thus trainability and generalization of neural networks. Methods derived from statistical physics also give insight (Cohen et al., 2019; Bordelon et al., 2020). However, there is a gap in terms of generalization between NTK regression and training neural networks (Allen-Zhu & Li, 2019; Arora et al., 2019) (see however (Lee et al., 2019)), which prompts research on generalization in deep learning beyond NTK (Allen-Zhu et al., 2019; Bai & Lee, 2019). Another line of research on generalization looks at the implicit bias of gradient descent. Gradient descent on logistic regression can lead to the max-margin solution (Soudry et al., 2018; Ji & Telgarsky, 2018), while optimization on mean squared regression has a shortest path solution (Oymak & Soltanolkotabi, 2018).

3. Preliminaries

In this section, we set up our notation and review background material.

3.1. Neural Networks

The output of a neural network is defined by $f(x) = \tilde{}^{(L)}(x)$, where the functions $\tilde{}^{(\ell)}(\cdot) : \mathbb{R}^{n_0} \to \mathbb{R}^{n_\ell}$ (*pre-activations*) and $^{(\ell)}(\cdot) : \mathbb{R}^{n_0} \to \mathbb{R}^{n_\ell}$ (*activations*) follow the recursive relation:

$$^{(0)}(x) \equiv x$$

$$\tilde{}^{(\ell+1)}(x) \equiv \frac{1}{\sqrt{n_\ell}} W^{(\ell)} \ ^{(0)}(x); \qquad ^{(\ell)}(x) \equiv \ (\tilde{}^{(\ell)}(x));$$

where is an element-wise activation function and weights are initialized by sampling from an i.i.d. standard Gaussian. We are mostly interested in

$$\text{sigmoid} = \frac{1}{1+e^{\ x}} \tag{1}$$

as the activation function. We drop the bias term for simplicity. We expect our results to be qualitatively the same with a bias term for sigmoid as the behavior is governed by the activation's shape. Throughout the paper, we will use $f_0(x)$ and $f_1(x)$ to denote neural networks at initialization and training to zero loss respectively. As shown later, the

attractor behavior is mostly governed by the shape of the sigmoidal activations.

Inputs: Given n training points $\{x_i\}_1^n$, we define the following two matrices.

$$\hat{X} = \begin{pmatrix} | & \cdots & | \\ x_1 & \cdots & x_n \\ | & \cdots & | \end{pmatrix}; \quad f(\hat{X}) = \begin{pmatrix} | & \cdots & | \\ f(x_1) & \cdots & f(x_n) \\ | & \cdots & | \end{pmatrix};$$

where $\hat{X} \in \mathbb{R}^{n_0 \times n}$ is the data matrix and each column of $f(\hat{X}) \in \mathbb{R}^{n_0 \times n}$ is the output of the corresponding training example. We further assume that all input examples share the same norm (i.e. $\forall i \ \|x_i\|_2 = r$).

Jacobian Matrix: Given the network setup, the input-output Jacobian matrix can be computed as:

$$J(x) = \frac{1}{\sqrt{n_L}} W^{(L)} \prod_{k=1}^{L-1} D^{(k)} \frac{1}{\sqrt{n_{k-1}}} W^{(k-1)}$$

where

$$D^{(k)} = \operatorname{diag}(\dot{\sigma}(\tilde{x}^{(k)}(x))).$$

Here $\dot{\sigma}$ denotes first derivative, and diag takes in a vector and outputs a diagonal matrix with the vector at the diagonal. We will use $J_0(x)$ and $J_1(x)$ to denote Jacobian at initialization and training to zero loss respectively.

Autoencoder: An autoencoder network is trained via gradient flow to optimize the following loss function:

$$\arg\min_f \frac{1}{2n} \sum_{i=1}^{n} \|f(x_i) - x_i\|_2^2,$$

where f is the network defined above.

3.2. Neural Tangent Kernel

In the large-width regime, the neural network can be approximated by a linearization with respect to its parameters (Lee et al., 2019):

$$f(x; \theta_t) \approx f_0(x) + \partial f(x)|_{\theta=\theta_0} (\theta_t - \theta_0),$$

where θ_0 and θ_t are vectors of the network parameters at initialization and time t. The first term remains unchanged throughout training. Moreover, we can view $\partial f(x)|_{\theta=\theta_0}$ as a feature map in Hilbert space and derive the following matrix kernel,

$$(\Theta_0^{(L)}(\hat{x}; x))_{dd'} = \langle \partial f_d(\hat{x})|_{\theta=\theta_0}, \partial f_{d'}(x)|_{\theta=\theta_0} \rangle,$$

where $\langle \cdot, \cdot \rangle$ denotes inner product. Indeed, in the infinite width limit (Jacot et al., 2018), $\Theta_0^{(L)}$ converges in probability (stochasticity induced by random initialization) to a deterministic limiting kernel, $\Theta_0^{(L)}(\hat{x}; x) \to \Theta_1^{(L)}(\hat{x}; x) I_{n_L}$, where $\Theta_1^{(L)}$ is a scalar kernel and the training dynamics is entirely governed by it.

On the other hand, it can also be shown that in the NTK limit ($n_1, \ldots, n_L \to \infty$ sequentially) and when the weights are initialized by sampling from an i.i.d. standard Gaussian distribution, output functions at each layer tend to be i.i.d. centered Gaussian Processes at initialization (Lee et al., 2017), and the covariance matrix of layer ℓ can be defined recursively by

$$\Sigma^{(1)}(\hat{x}; x) = \frac{1}{n_0} \hat{x}^T x;$$
$$\Sigma^{(\ell+1)}(\hat{x}; x) = \mathbb{E}_{g \sim \mathcal{N}(0, \Sigma^{(\ell)})}[\sigma(g(x))\sigma(g(\hat{x}))].$$

Under the same limit, $\Theta_1^{(L)}$ can also be recursively defined (Jacot et al., 2018):

$$\Theta_1^{(1)}(\hat{x}; x) = \Sigma^{(1)}(\hat{x}; x);$$
$$\Theta_1^{(\ell+1)}(\hat{x}; x) = \Theta_1^{(\ell)}(\hat{x}; x) \dot{\Sigma}^{(\ell+1)}(\hat{x}; x) + \Sigma^{(\ell+1)}(\hat{x}; x);$$

where

$$\dot{\Sigma}^{(\ell+1)}(\hat{x}; x) = \mathbb{E}_{g \sim \mathcal{N}(0, \Sigma^{(\ell)})}[\dot{\sigma}(g(x))\dot{\sigma}(g(\hat{x}))].$$

For gradient flow training with a least squares loss to zero training error, there is a closed form solution for $f_1(x)$ using NTK in the infinite width limit (Jacot et al., 2018). In the case of autoencoder, we get that,

$$f_1(x) = \hat{X} f_0(\hat{X}) K^{-1} k_x + f_0(x); \qquad (2)$$

and

$$J_1(x) = \hat{X} f_0(\hat{X}) K^{-1} \frac{\partial k_x}{\partial x} + J_0(x); \qquad (3)$$

where

$$K_{ij} = \Theta_1^{(L)}(x_i, x_j); \quad (k_x)_i = \Theta_1^{(L)}(x_i, x); \quad \left(\frac{\partial k_x}{\partial x}\right)_i = \frac{\partial \Theta_1^{(L)}(x_i, x)}{\partial x}.$$

3.3. Iterative Maps, Attractors, Associative Memory and Jacobian

We define attractors with respect to an iterative map. Notice that an autoencoder f is a map from $\mathbb{R}^{n_0}$ to $\mathbb{R}^{n_0}$. Therefore, we can apply f iteratively to input x. Formally, we define this sequence for any input x, $\{f^k(x)\}_{k \in \mathbb{N}}$ where $f^k = \underbrace{f(\cdots f}_{k}(x)\cdots))$.

Definition 1. A fixed point x^* of map f ($f(x^*) = x^*$) is an attractor if there exists an open neighborhood $\mathcal{N}$ of x^* such that for any x in this neighborhood, $\{f^k(x)\}_{k \in \mathbb{N}}$ converges to x^* as $k \to \infty$. The set of all such points is called basin of attraction of x^*.

Fixed points attractors can be used to implement associative memory (Hopfield, 1982). A memory clue sets the initial condition of the network dynamics, positioning the network state in a basin of attraction, and the actual memory is recapitulated by the attractor dynamics converging to the corresponding fixed point.

There is a well-know condition for a fixed point to be an attractor (Rudin et al., 1964).

Proposition 1. A fixed point x is an attractor of a differentiable map f if all eigenvalues of the Jacobian of f at x are strictly less than 1 in absolute value.

Therefore, for a point x to be an attractor, we need two conditions: (1) $f(x) = x$ and (2) all the eigenvalues of $J(x)$ have norm strictly smaller than 1. Condition (1) can be justified in the NTK limit (Jacot et al., 2018; Du et al., 2018; Allen-Zhu et al., 2018) as long as K is positive definite. This is theoretically true if all data points live on a sphere, the network has non-polynomial Lipschitz activation function and $L \geq 2$ (c.f. Proposition 2 in (Jacot et al., 2018)). In practice, it is easy to achieve $f(x) \approx x$ in overparameterized networks. Therefore, our focus is on the second condition.

4. Theoretical Results

In this section, we present our theoretical results about the attractor behavior of iterated overparametrized autoencoders. We focus on two distinct settings:

1. In the first setting, we consider a neural network of any depth in the NTK limit with a single training example.
2. In the second setting, we focus on a two-layer network in the NTK limit and multiple training examples.

We first give two key results about the Jacobian norm at network initialization that will be used later. We use the operator norm (induced l_2-norm) of Jacobian as a proxy to control the eigenvalue, because the norm is given by the largest singular value, which upper bounds the norm of the largest eigenvalue.

4.1. A Bound on the Norm of the Jacobian at Network Initialization

In this section, we first show that with high probability, the operator norm of the initial Jacobian for sigmoid networks drops with increasing depth. To prove this, we use a mathematical technique called the *net argument (Tao, 2012) (Theorem 1) valid for any activation function. We then argue that for sigmoid networks the upper bound of the initial Jacobian norm is concentrated around 0.5 for large n_0. This explains the formation of attractors when the trained Jacobian norm stays close to initilization (c.f. Section 4.2).

We first prove a proposition to be used in the *net argument.

Given a unit vector $z^{(0)}$ and input $\hat{x}$, we recursively define $z^{(L)} \equiv J(\hat{x})z^{(0)}$:

$$\tilde{z}^{(\ell)} = \frac{1}{\sqrt{n_{(\ell-1)}}} W^{(\ell-1)} z^{(\ell-1)}; \quad z^{(\ell)} = D^{(\ell)} \tilde{z}^{(\ell)}.$$

The following proposition provides the distribution of $z^{(i)}$.

Proposition 2. For a fixed unit vector $z^{(0)}$, fixed input data $\hat{x}$ and a network of depth L at random initialization, with a Lipschitz nonlinearity σ, and in the limit $n_1, \ldots, n_{L-1} \to \infty$, $J(\hat{x})z^{(0)}$ has the following recursion with $\hat{z}_i^{(\ell)} = \hat{z}^{(\ell)}$:

$$\hat{z}^{(1)} = \sigma'(a)b \quad (a, b) \sim \mathcal{N}\left(0, \begin{bmatrix} \frac{\|\hat{x}\|_2^2}{n_0} & \frac{\hat{x}^T z^{(0)}}{n_0} \\ \frac{\hat{x}^T z^{(0)}}{n_0} & \frac{\|z^{(0)}\|_2^2}{n_0} \end{bmatrix}\right);$$

$$\hat{z}^{(\ell+1)} = \sigma'(a)b \quad (a, b) \sim \mathcal{N}\left(0, \begin{bmatrix} E[(\hat{\alpha}^{(\ell)})^2] & E[\hat{\alpha}^{(\ell)}\hat{z}^{(\ell)}] \\ E[\hat{\alpha}^{(\ell)}\hat{z}^{(\ell)}] & E[(\hat{z}^{(\ell)})^2] \end{bmatrix}\right);$$

$$\tilde{z}_i^{(L)} = \hat{z}^{(L)} \quad \mathcal{N}\left(0, E[(\hat{z}^{(L-1)})^2]\right);$$

where

$$\hat{\alpha}^{(1)} = \sigma(a) \quad a \sim \mathcal{N}\left(0, \frac{\|\hat{x}\|_2^2}{n_0}\right);$$

$$\hat{\alpha}^{(\ell+1)} = \sigma(a) \quad a \sim \mathcal{N}\left(0, E[(\hat{\alpha}^{(\ell)})^2]\right).$$

Proof. See Appendix A. □

Using Proposition 2, we apply the *net argument (Tao, 2012) (Appendix A) to obtain a bound on the Jacobian operator norm.

Theorem 1. For any data point x_i, $i \in [1, \ldots, n]$, with probability at least $1 - O(n)e^{-\Omega(n_0)}$,

$$\|J(x_i)\|_{op} \leq c\sqrt{n_0}$$

where c is a constant and

$$\gamma = \sup_{x_i \in \hat{X},\, \|z^{(0)}\|_2 = 1} E[(\hat{z}^{(L-1)})^2 | z^{(0)}, x_i]$$

Proof. See Appendix A. □

This bound on singular values of $J(x)$ provides a way to understand how the Jacobian at network initialization changes with respect to activation functions and the number of layers. Specifically, for sigmoid activation function, we know that $\sigma'(x) \in (0, \frac{1}{4}]$ and for any arbitrary unit vector $z^{(0)}$,

$$E[(\hat{z}^{(L-1)})^2 | z^{(0)}, x] = E[\sigma'(a)^2 b^2] \leq \frac{E[(\hat{z}^{(L-2)})^2 | z^{(0)}, x]}{16}.$$

Thus,

$$E[(\hat{z}^{(L-1)})^2 | z^{(0)}, x] \leq \frac{1}{n_0 16^{L-1}} \|z^{(0)}\|_2^2 = \frac{1}{n_0 16^{L-1}};$$

and with probability at least $1 - O(n)e^{-O(n_0)}$,

$$\|J(x_i)\|_{op} \leq \frac{c}{4^{L-1}} \quad \forall i \in [n]. \tag{4}$$

Therefore, with high probability, the initial Jacobian norm decreases with increasing layers. Based on this result, we choose to study a 2-layer sigmoid network because it would give an upper bound on the initial Jacobian norm.

Next, we argue that the largest initial Jacobian norm for a two-layer sigmoid network is concentrated around $1/2$. For a given training point $\hat{x}$,

$$J(\hat{x}) = \frac{1}{\sqrt{n_1}} W^{(1)} D^{(1)} \frac{1}{\sqrt{n_0}} W^{(0)}.$$

From Proposition 2, we can reach maximum $E[(\hat{z}^{L-1})^2 | z^{(0)}; \hat{x}])$ for any fixed $z^{(0)}$ when $\hat{x} = 0$ because every gradient of the hidden layer is at its max and the covariance between x and $z^{(0)}$ is zero, which would informally suggest that the upper bound of initial Jacobian norm is likely achieved at $\hat{x} = 0$ (Another way to view this comes from Proposition 2 where for any given unit vector $z^{(0)}$, the norm of output vector $\hat{z}_i^{(2)}$ has a higher mean when $\hat{x} = 0$). This gives us the simplification:

$$J(\hat{x}) = \frac{1}{4} \frac{1}{\sqrt{n_0}} \frac{1}{\sqrt{n_1}} W^{(1)} W^{(0)}.$$

Observe that $W = \frac{1}{\sqrt{n_1}} W^{(1)} W^{(0)}$ is a Gaussian random matrix with $n_1 \to \infty$ where each entry is i.i.d. and the largest singular value of $\frac{1}{\sqrt{n_0}} W$ is concentrated at 2 for large n_0 (Vershynin, 2012). Consequently, the largest initial Jacobian norm for a sigmoid network is concentrated around $1/2$.

4.2. Training a Multilayer Network with a Single Example

In this section, we consider the special case when there is only one training example x_1. We show that under certain conditions the Jacobian stays close to initialization, and, combined with the result from the previous section, the trained network can form attractors.

We start by analyzing the NTK solution. Note that in this case (2) can be simplified to:

$$f_1(x) = \frac{\Phi_1^{(L)}(x; x_1)}{\Phi_1^{(L)}(x_1; x_1)}(x_1 - f_0(x_1)) + f_0(x).$$

As we describe below, $\Phi_1^{(L)}$ tends to a constant kernel as $L \to \infty$ for some network initialization (Hayou et al., 2019) and therefore the trained Jacobian equals the initial one.

To see this, first, we pay close attention to the covariance matrix $\Sigma^{(L)}$, which is the building block of $\Phi_1^{(L)}$. We define $q_{ab}^{(\ell)} = \Sigma^{(\ell)}(x_a; x_b)$ and $c_{ab}^{(\ell)} = q_{ab}^{(\ell)} / \sqrt{q_{aa}^{(\ell)} q_{bb}^{(\ell)}}$. It can be shown that $c = 1$ is a fixed point of $c_{ab}^{(\ell)}$ as $\ell \to \infty$ (Poole et al., 2016). For sigmoidal networks, the stability of c is governed by

$$\chi_1 = \frac{\partial c_{ab}^{(\ell)}}{\partial c_{ab}^{(\ell-1)}}\Big|_{c=1} = E[(\sqrt{q^*} z)^2], \quad z \sim N(0, 1).$$

where q^* is what $q_{aa}^{(\ell)}$ converges to. If $\chi_1 < 1$, c is a stable fixed point, suggesting that all points become equally similar as they progress through layers. A network under such initialization is said to be in the ordered region (Schoenholz et al., 2016). For such networks, $\Phi_1^{(\ell)}$ converges to a constant kernel with increasing layers (Xiao et al., 2019; Hayou et al., 2019). As $L \to \infty$, $\partial \Phi_1^{(L)}(x; x_1)|_{x=x_1} \to 0$ and,

$$J_1(x_1) \xrightarrow{L \to \infty} J_0(x_1),$$

as long as $\Phi_1^{(L)}(x_1; x_1)$ does not converge to 0 as $L \to \infty$. We note that this argument applies to other weight and bias variance scaling factors at initialization than the special case (1 and 0 respectively) we focus on.

Specializing to sigmoid networks, we first observe that they are in the ordered region because $\dot{\Sigma}(x) \in (0, \frac{1}{4}]$, giving an upper bound on χ_1 as $1/16$. Further the lower bound of $\Phi_1^{(L)}(x_1; x_1)$ is $1/4$ (Lemma 5 in Appendix B). Therefore, we expect the Jacobian to be constant during training in the large depth limit. In practice, the trained Jacobian stays close to initialization for 2 - 3 layer sigmoid networks as shown in Section 5.2.

Our analysis can explain the empirical results of (Zhang et al., 2019) that fully connected networks trained with single example tend to learn constant functions, leading to memorization. To see this, remember that for a sigmoid network the norm of the initial Jacobian falls with increasing the number of layers, eq(4). Other sigmoidal activation functions may show the same behavior. In the same limit, the Jacobian remains constant during training, implying $f_1(x)$ is approximately constant around x_1.

The large-depth analysis presented in this section does not carry over to multiple training examples. The limiting constant kernel is singular and fails in training the network to zero loss (Jacot et al., 2018). Instead, we will study 2-layer sigmoid networks for multiple training examples.

4.3. Training a 2-Layer Network with Multiple Examples: Linear Region

Next, we consider our second setting of a 2-layer network trained with multiple examples. In this section, we argue that for small input norm, a network in the NTK regime behaves like a linear network resulting in a Jacobian with eigenvalue 1, and thus may not form associative memory.

(a) (b)

Figure 1: (a) Linear Region Illustration for Sigmoid: The top graph is a sigmoid function while the bottom one is a Gaussian distribution. The majority of Gasussian distribution falls in the linear region of sigmoid. (b) Sigmoid vs Rescaled Erf.

To see this, first note that for a 2-layer network and any two training points x_i and x_j, NTK can be written as

$$\Theta^{(2)}_1(x_i, x_j) = \Theta^{(1)}(x_i, x_j)\dot{\Sigma}^{(2)}(x_i, x_j) + \Sigma^{(2)}(x_i, x_j),$$

where $\Theta^{(1)}(x_i, x_j) = \frac{x_i^T x_j}{n_0}$ and both $\Sigma^{(2)}$ and $\dot{\Sigma}^{(2)}$ are expectations governed by $\Theta^{(1)}$. Because $\Theta^{(1)}$ is a covariance matrix defined by inner products, small input norms means small variance for the Gaussian process and the expectation is mostly concentrated to the linear region of the activation. This justifies a linear approximation to activation function. Figure 1a shows this sigmoid, which can be approximated linearly around $x = 0$ as $\sigma(x) \approx \frac{1}{4}x + \frac{1}{2}$.

Most of activation functions have similar linear behavior though the range in which this approximation is accurate may differ. Thus, we focus on an arbitrary linear activation function $\alpha x + \beta$ which leads to an initial Jacobian $J_0(x) = \frac{1}{\sqrt{n_1}}\frac{1}{\sqrt{n_0}}W^{(1)}W^{(0)}$ with initial weights $W^{(1)}$ and $W^{(0)}$. We will return to the discussion of sigmoid at the end of this section. For simplicity, we also assume that X is full rank and $n_0 \geq n$ as memory tasks tend to have high dimension inputs like images and audios (Radhakrishnan et al., 2019).

We start examining the Jacobian with the easiest case $\sigma(x) = x$ and $n = n_0$, which leads to $J_1(x) = I_{n_0}$.

Lemma 2. Suppose there is a 2-layer network. If the activation function is $\sigma(x) = x$, $n = n_0$ and the data matrix is full rank. Then at NTK limit, $J_1(x) = I_{n_0}$.

Proof. See Appendix C.1. □

In fact, the multiplicity of eigenvalue 1 is directly related to n under certain conditions.

Lemma 3. Suppose there is a 2-layer network with activation function $\sigma(x) = x$ and given initial weights $W^{(1)} \in \mathbb{R}^{n_0 \times n_1}$, $W^{(0)} \in \mathbb{R}^{n_1 \times n_0}$. If the data matrix is full rank with $n \leq n_0$, then, at the NTK limit ($n_1 \to \infty$), $J_1(x)$ has eigenvalue 1 with multiplicity at least n. If at the NTK limit, α is chosen such that $\|J_0(x)\|_{op} < 1$, then the multiplicity is exactly n and 1 is the largest eigenvalue norm.

Proof. See Appendix C.1. □

Remark 1. Lemma 3 suggests that a network trained with a single example at convergence can have Jacobian eigenvalue 1 for $\sigma(x) = x$ regardless of initial Jacobians. This result may seem to contradict Section 4.2. However, in this case, the argument in Section 4.2 is violated because the diagonal value of $\Sigma^{(2)}_1$ is also converging to zero as when $n = 1$, the linear activation has a shrinking effect on the layer outputs.

The result can be naturally extended to $\sigma(x) = x + \beta$ with $\beta > 0$.

Lemma 4. Suppose there is a 2-layer network with activation function $\sigma(x) = x + \beta$, given initial weights $W^{(1)} \in \mathbb{R}^{n_0 \times n_1}$, $W^{(0)} \in \mathbb{R}^{n_1 \times n_0}$ and every data point has the same norm r (i.e. $\forall i \in [n] \quad \|x_i\|_2 = r$). If the data matrix is full rank with $n \leq n_0$, then, at the NTK limit $n_1 \to \infty$, $J_1(x)$ has eigenvalues 1 with multiplicity at least $n - 1$. If at the NTK limit, α and β are chosen such that

$$\|J_0(x)\|_{op} = 1 \quad ; \quad \left\|\frac{1}{\sqrt{n_1}}W^{(1)}1_{n_1}\right\|_2 < \frac{n_0}{2r\beta^2};$$

where $0 < \alpha \leq 1$, then the multiplicity is exactly $n - 1$ and 1 is the largest eigenvalue norm.

Proof. See Appendix C.1. □

Collectively we proved some sufficiency conditions for the largest trained Jacobian eigenvalue to be 1. We emphasize that if the largest eigenvalue is 1, and not strictly below 1, the network can fail to form attractors.

Now, we return to sigmoid networks and discuss the implications of Lemma 4. In the linear region, we have $\alpha = \frac{1}{4}$ and $\beta = \frac{1}{2}$. As mentioned in Section 4.1, in the NTK limit, $\|J_0(x)\|_2 \approx \frac{1}{2}$, implying $\alpha \approx \frac{1}{2}$. On the other hand, $\|\frac{1}{\sqrt{n_1}}W^{(1)}1_{n_1}\|_2$ is the norm of a standard Gaussian vector which follows the chi distribution, and it is concentrated around $\sqrt{n_0}$. Then, the conditions in Lemma 4 hold with high probability if $r < 2\sqrt{n_0}$. Attractor formation can fail in sigmoid networks for small input norms, which we observe in simulations (Section 5).

4.4. Training a 2-Layer Network with Multiple Examples: Beyond Linear Region and a Transition to Attractor Formation

Section 4.3 suggests that small α leads to linear behavior and networks may fail to form associative memory. In this

section, we explore larger norm inputs where a network in the NTK regime utilizes the non-linear region of the sigmoid. We will argue that in the large norm limit all Jacobian eigenvalues' norms are below 1. Taking into account our result that attractor formation may fail for small values of r, we identify a transition with increasing r from a regime where memory formation does not occur to a regime where it occurs. Our results can be adapted for other sigmoidal functions.

For simplicity, we further assume that there are no parallel inputs in the training data (x_i and x at the same time). This is not a hard constraint and an analysis with parallel inputs is given in Appendix D.3.

Our strategy is to calculate the Jacobian in the large norm limit using Equation (3). We first note for larger r, $X_{ij} \approx f_0(X)_{ij}$ because all the hidden units in the network is between 0 and 1. Therefore Equation 3 can be approximated by

$$J_1(x) \approx \hat{X}\hat{K}^{-1}\frac{\partial \hat{K}_X}{\partial x} + J_0(x) \tag{5}$$

The items of interest here are $\hat{K}$ and $\frac{\partial \hat{K}_X}{\partial x}$, for which we need to calculate the NTK and its gradient. We first define an approximation of the NTK and then use it to estimate the Jacobian.

Approximation of the NTK: Approximating sigmoid function (σ_s) by erf function $\sigma_s(x) \approx \frac{1}{2}\mathrm{erf}\left(\frac{1}{2}x\right) + \frac{1}{2}$ (shown in Figure 1b) allows us to use a known closed form solution for NTK (Lee et al., 2019; Williams, 1997). With this approximation, 2-layer NTK can be written as follows (full derivation can be found in Appendix D.1):

$$\Sigma_1^{(2)}(\hat{x}; x) \approx \frac{1}{2}\sqrt{\frac{\hat{x}^T x}{(2n_0 + x^T x)(2n_0 + \hat{x}^T\hat{x}) - (\hat{x}^T x)^2)}} + \frac{1}{2}\arcsin\sqrt{\frac{\hat{x}^T x}{(x^T x + 2n_0)(\hat{x}^T\hat{x} + 2n_0)}} + \frac{1}{4}. \tag{6}$$

In order to calculate the gradient of NTK, we define:

$$T(\Sigma; \Sigma_1; \Sigma_2)(x_i; x_j) \approx \mathbb{E}_{f \sim N(0;\Sigma)}[\dot{\Sigma}_1(f(x_i))\dot{\Sigma}_2(f(x_j))]$$

Using this definition, the NTK's gradient for a 2-layer network with arbitrary activation function is given by,

$$\frac{\partial \Sigma_1^{(2)}(\hat{x}; x)}{\partial x} = \frac{1}{n_0^2}\hat{x}^T x\, T(\Sigma_1; \cdot; \cdot)\hat{x} + \frac{2}{n_0}T(\Sigma_1; \perp; \cdot)\hat{x} + \frac{1}{n_0^2}\hat{x}^T x\, T(\Sigma_1; \perp; \cdot)x + \frac{1}{n_0}T(\Sigma_1; \Sigma; \cdot)x.$$

Now we can use these expressions to calculate $\hat{K}$ and $\frac{\partial \hat{K}_X}{\partial x}$ in Equation 5.

Calculation of $\hat{K}$ and $\frac{\partial \hat{K}_X}{\partial x}$: We first look at Equation (6), using the fact that because all the data points have the same norm $x_i^T x_j = r^2\,\Sigma_{i;j}$ with $|\Sigma_{i;j}| < 1$ (since there are no parallel inputs):

$$\hat{K}_{ij} = \frac{1}{4} + \frac{1}{2}\arcsin\frac{r^2\,\Sigma_{i;j}}{r^2 + 2n_0} + \frac{1}{2}\sqrt{\frac{r^2\,\Sigma_{i;j}}{(2n_0 + r^2)^2 - r^4\,\Sigma_{i;j}^2}}.$$

To get insight into the behavior of $\hat{K}_{ij}$, let's consider diagonal and off-diagonal entries separately in the large r limit. First the off-diagonals:

$$\hat{K}_{ij} \approx \frac{1}{4} + \frac{1}{2}\arcsin(\Sigma_{i;j}) + \frac{1}{2}\sqrt{\frac{\Sigma_{i;j}}{1 - \Sigma_{i;j}^2}}.$$

Next we look at the diagonals:

$$\hat{K}_{ii} \approx \frac{r}{4}\sqrt{n_0}.$$

The diagonal grows linearly with r and dominates over the off-diagonals. The kernel tends to a scaled identity matrix in the $r \to \infty$ limit. Thus,

$$\|\hat{K}^{-1}\|_{op} \approx \frac{4}{r\sqrt{n_0}}. \tag{7}$$

Next, we look at $\frac{\partial \hat{K}_X}{\partial x}$. As we are interested in trained Jacobian at training examples, without loss of generality, we focus on $J_1(x_1)$. We already know that $\hat{K}$ tends to converge to a diagonal matrix with large r. One could use this result to suggest that $\frac{\partial \Sigma_1^2(x_i;x)}{\partial x}\big|_{x_j} \to 0$ if $i \neq j$. In fact, with larger r, it can be shown that (Appendix D.2):

$$\frac{\partial \hat{K}_X}{\partial x}_{x_1} \approx \left[\frac{\partial \Sigma_1^{\perp}(x_1;x)}{\partial x}\big|_{x_1}; 0; \ldots; 0\right]^T \tag{8}$$

and

$$\left\|\frac{\partial \hat{K}_X}{\partial x}\right\|_{op} \approx \frac{1}{8}\sqrt{n_0}. \tag{9}$$

Trained Jacobian and attractor formation: Finally, we can use our results in Equations (7) and (9) to calculate the trained Jacobian using Equation (5) with large r.

$$\|J_1(x)\|_{op} \approx \left\|\hat{X}\hat{K}^{-1}\frac{\partial \hat{K}_X}{\partial x} + J_0(x)\right\|_{op} \leq \|\hat{X}\hat{K}^{-1}\|\left\|\frac{\partial \hat{K}_X}{\partial x}\right\|_{op} + \|J_0(x)\|_{op} \approx r\frac{4}{r\sqrt{n_0}}\frac{1}{8}\sqrt{n_0} + \|J_0(x)\|_{op} = \frac{1}{2} + \|J_0(x)\|_{op}.$$

To go from the second to the third row of the equation above, we used the fact that each column X_0 has norm r, and in the limit K^{-1} is a scaled identity matrix and $\frac{\partial x_1}{\partial x}$ has only one non-zero row and that row is a scaled x_1 (Appendix D.2).

To see the implications of this result, we observe that as $r \to 1$, $J_0(x) \to 0$ because the pre-activation of the two layer network will be in nitely big and the units saturate leading to zero gradients in the Jacobian matrix, we can conclude in this limit

$$\|J_1(x)\|_{op} \to 1/2.$$

Combined with our previous result that attractor formation may fail for small values of r, this suggests that there is a transition from a region where associative memory formation fails to region where it succeeds with increasing r. We show this transition in simulations in Section 5 and verify that the largest eigenvalue of the Jacobian falls toward $1/2$ as asymptotically.

4.5. Beyond NTK

In practice, large values of r often require a larger width to stay in NTK. If the large width condition is violated, trained weights will not stay close to initialization. However, in this case, a signi cant deviation of weights from initialization caused by gradient descent may saturate hidden units, leading to zero gradients in the Jacobian matrix. As shown in Figure 3, for xed hidden size, larger input radius leads to near zero eigenvalues when the NTK conditions no longer hold. Therefore, the network can behave as if only the last layer is trained with saturated hidden features and close to zero Jacobian norm since all the $D^{(\ell)}$ matrices have mostly zero diagonal entries. Note that, in overparameterized networks, this type of optimization can often result in zero training loss as well due to the over-parameterization of the last layer.

5. Simulations

5.1. Experiment Setup

Training and iterative convergence criteria: Training is stopped when the training loss of the auto-encoder drops below a threshold, which we chose to be 10^{-7}. Iterative convergence happens when a non- xed point converges to a xed point measured by mean-squared-error after passing through the trained autoencoder iteratively. This threshold was 10^{-2}.

Implementation details: We used vanilla gradient descent with learning rate 1, similar to (Jacot et al., 2018). The code is implemented with Pytorch. For each setting, we ran experiments 100 times to get 100 sets of samples. For all experiments except experiments on MNIST, the samples are randomly generated.

Figure 2: Difference of the largest eigenvalue norms at initialization and after training, i.e. $|\|\lambda_1(J_0)\| - \|\lambda_1(J_1)\||$.

5.2. Single Training Example

We rst present experiments for a single training example. The sigmoid network is chosen to have hidden dimension 1000 with input dimension 32. Figure 2 shows that the difference between the largest eigenvalue norms at initialization and after training decreases with more number of layers.

5.3. Multiple Training Examples

Linear Region: In this section, we rst illustrate the eigenvalue distribution in the linear region by sampling unit vectors as data ($r = 1$). Here, we trained 2 layer sigmoid networks with input dimension 10 and hidden size 1000 for 2, 5 and 8 training points. As suggested by Lemma 4, there should be $n - 1$ eigenvalues with norm around 1. This is supported by Figure E.1 in the Appendix, where 10%, 40% and 70% of the eigenvalues are near 1. Here, the presence of eigenvalue 1 indicates that the network operates in the linear region.

Beyond Linear Region: We demonstrate how the largest eigenvalue norm varies with the input radius. We test with various hidden dimensions to achieve the NTK limit on input dimension 32 with the number of training data 5, 20, and 40. Only experiments that can be trained to have loss below 10^{-7} or t into single Titan V GPU are included.

The general trend, as shown in Figure 3, is that as we move away from the linear region, the largest eigenvalue norm will drop, and as we keep increasing the hidden layer size, it will move close to the $1/2$ limit as suggested by our analysis. It is also worth noting that the input radius needs to be increased with large number of training examples to get training loss below 10^{-7} under reasonable iterations, implying the capacity of the networks is controlled by input radius and agreeing with the intuition and theoretical results from (Allen-Zhu et al., 2018), that there needs to be some level of separation between data points to train the network.

(a) number of training points: 5 (b) number of training points: 20 (c) number of training points: 40

Figure 3: Largest eigenvalue norm vs input norm: input dimension 32.

(a) Random vectors (b) MNIST

Figure 4: Convergence Success Rate vs Input Radius: 5 training examples

5.4. Basin of Attraction

We test basin of attraction by adding Gaussian noise to training examples and check if the modi ed examples can converge to the original ones via iterative maps under 50 iterations. And the convergence rate is the number of samples that could be successfully recovered. The standard deviation of the Gaussian noise is called the noise radius. The network has two layers with hidden size 10000 and input dimension 32. Not surprisingly, Figure 4a shows that larger input norm gives greater basin of attraction. More experiments can be found in Appendix E.2.

5.5. MNIST Data

We also test basin of attraction experiments on MNIST dataset to check if we can recover real training examples. The images are prepossessed by subtracting means and rescaled to have different input norms for testing. Similar to the setting before, Figure 4b also shows that larger input norm gives greater basin of attraction. Notice that because MNIST images have large input dimension, they need larger radius to move out of the linear region. More experiments can be found in Appendix E.3.

5.6. Sigmoidal Activation

Finally, we show that our results can be extended to different sigmoidal activation functions as well. We chose two-layer network with hidden size 10000 and input dimension 32 and 20 training examples. As before, only settings that led to a training loss below 10^{-7} are included. Figure 5 clearly shows that all the activation functions share similar curves. Notice that both tanh and erf have large eigenvalue when r is small. This is not a contradiction to our Lemma 3 as their $\sigma = \sigma_1(0)$ is too large to satisfy the conditions in Lemma 3. To further verify this result, we also include how the histogram of eigenvalue norm changes for those activations in the Appendix E.4.

Figure 5: Input Radius and Eigenvalue Norm Curve for Different Activation Functions

6. Conclusion

In this paper, we theoretically and empirically show that training overparameterized sigmoid autoencoders can lead to attractors for a single training example and multiple training examples in the non-linear region, with the help of theories developed in the NTK limit. We identi ed a behavior change governed by the input radius. Some future directions include generalizing our results to other activations, identifying other factors that can determine whether autoencoders can learn to have attractors or not, and how the formations of attractors are related to generalization in deep learning.

Acknowledgements

C. Pehlevan acknowledges support by the Harvard Data Science Initiative.

References

Allen-Zhu, Z. and Li, Y. What can resnet learn ef ciently, going beyond kernels? In *Advances in Neural Information Processing Systems*, pp. 9015–9025, 2019.

Allen-Zhu, Z., Li, Y., and Song, Z. A convergence theory for deep learning via over-parameterization. *arXiv preprint arXiv:1811.03962*, 2018.

Allen-Zhu, Z., Li, Y., and Liang, Y. Learning and generalization in overparameterized neural networks, going beyond two layers. In *Advances in neural information processing systems*, pp. 6155–6166, 2019.

Amit, D. J. and Treves, A. Associative memory neural network with low temporal spiking rates. *Proceedings of the National Academy of Sciences*, 86(20):7871–7875, 1989.

Arora, S., Du, S. S., Hu, W., Li, Z., Salakhutdinov, R. R., and Wang, R. On exact computation with an in nitely wide neural net. In *Advances in Neural Information Processing Systems*, pp. 8139–8148, 2019.

Bai, Y. and Lee, J. D. Beyond linearization: On quadratic and higher-order approximation of wide neural networks. *arXiv preprint arXiv:1910.01619*, 2019.

Bordelon, B., Canatar, A., and Pehlevan, C. Spectrum dependent learning curves in kernel regression and wide neural networks. *arXiv preprint arXiv:2002.02561*, 2020.

Cao, Y. and Gu, Q. Generalization bounds of stochastic gradient descent for wide and deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 10835–10845, 2019.

Chizat, L., Oyallon, E., and Bach, F. On lazy training in differentiable programming. In *Advances in Neural Information Processing Systems*, pp. 2933–2943, 2019.

Cohen, O., Malka, O., and Ringel, Z. Learning curves for deep neural networks: a gaussian eld theory perspective. *arXiv preprint arXiv:1906.05301*, 2019.

Du, S. S., Lee, J. D., Li, H., Wang, L., and Zhai, X. Gradient descent nds global minima of deep neural networks. *arXiv preprint arXiv:1811.03804*, 2018.

Hayou, S., Doucet, A., and Rousseau, J. Mean- eld behaviour of neural tangent kernel for deep neural networks. *arXiv preprint arXiv:1905.13654*, 2019.

Hertz, J. A. *Introduction to the theory of neural computation*. CRC Press, 2018.

Hop eld, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

Jacot, A., Gabriel, F., and Hongler, C. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in neural information processing systems*, pp. 8571–8580, 2018.

Ji, Z. and Telgarsky, M. Risk and parameter convergence of logistic regression. *arXiv preprint arXiv:1803.07300*, 2018.

Kolar, M. and Liu, H. Marginal regression for multitask learning. In *Arti cial Intelligence and Statistics*, pp. 647–655, 2012.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classi cation with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Lee, J., Bahri, Y., Novak, R., Schoenholz, S. S., Pennington, J., and Sohl-Dickstein, J. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

Lee, J., Xiao, L., Schoenholz, S., Bahri, Y., Novak, R., Sohl-Dickstein, J., and Pennington, J. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in neural information processing systems*, pp. 8570–8581, 2019.

Matthews, A. G. d. G., Rowland, M., Hron, J., Turner, R. E., and Ghahramani, Z. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 2018.

Miller, K. S. On the inverse of the sum of matrices. *Mathematics magazine*, 54(2):67–72, 1981.

Oymak, S. and Soltanolkotabi, M. Overparameterized nonlinear learning: Gradient descent takes the shortest path? *arXiv preprint arXiv:1812.10004*, 2018.

Pennington, J. and Worah, P. Nonlinear random matrix theory for deep learning. In *Advances in Neural Information Processing Systems*, pp. 2637–2646, 2017.

Pennington, J., Schoenholz, S., and Ganguli, S. Resurrecting the sigmoid in deep learning through dynamical isometry: theory and practice. In *Advances in neural information processing systems*, pp. 4785–4795, 2017.

Pennington, J., Schoenholz, S. S., and Ganguli, S. The emergence of spectral universality in deep networks. *arXiv preprint arXiv:1802.09979*, 2018.

Poole, B., Lahiri, S., Raghu, M., Sohl-Dickstein, J., and Ganguli, S. Exponential expressivity in deep neural networks through transient chaos. In *Advances in neural information processing systems*, pp. 3360–3368, 2016.

Radhakrishnan, A., Yang, K., Belkin, M., and Uhler, C. Memorization in overparameterized autoencoders. *arXiv preprint arXiv:1810.10333*, 2018.

Radhakrishnan, A., Belkin, M., and Uhler, C. Overparameterized neural networks can implement associative memory. *arXiv preprint arXiv:1909.12362*, 2019.

Rudin, W. et al. *Principles of mathematical analysis*, volume 3. McGraw-hill New York, 1964.

Schoenholz, S. S., Gilmer, J., Ganguli, S., and Sohl-Dickstein, J. Deep information propagation. *arXiv preprint arXiv:1611.01232*, 2016.

Soudry, D., Hoffer, E., Nacson, M. S., Gunasekar, S., and Srebro, N. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

Tao, T. *Topics in random matrix theory*, volume 132. American Mathematical Soc., 2012.

Vershynin, R. Introduction to the non-asymptotic analysis of random matrices. *Compressed Sensing*, pp. 210–268, 2012.

Williams, C. K. Computing with infinite networks. In *Advances in neural information processing systems*, pp. 295–301, 1997.

Xiao, L., Pennington, J., and Schoenholz, S. S. Disentangling trainability and generalization in deep learning. *arXiv preprint arXiv:1912.13053*, 2019.

Zhang, C., Bengio, S., Hardt, M., and Singer, Y. Identity crisis: Memorization and generalization under extreme overparameterization. *arXiv preprint arXiv:1902.04698*, 2019.

Zou, D. and Gu, Q. An improved analysis of training overparameterized deep neural networks. In *Advances in Neural Information Processing Systems*, pp. 2053–2062, 2019.

A. Proofs for Sec 4.1

Proposition 2. For a xed unit vector $z^{(0)}$, xed input data $\hat{x}$ and a network of depth L at random initialization, with a Lipschitz nonlinearity , and in the limit $n_1, \ldots, n_{L-1} \to \infty$, $J(\hat{x})z^{(0)}$ has the following recursion with $\tilde{z}_i^{(\ell)} = \hat{z}^{(\ell)}$:

$$\hat{z}^{(1)} = \dot{\sigma}(a)b \quad (a, b) \sim \mathcal{N}\left(0, \begin{bmatrix} \frac{\|\hat{x}\|_2^2}{n_0} & \frac{\hat{x}^T z^{(0)}}{n_0} \\ \frac{\hat{x}^T z^{(0)}}{n_0} & \frac{\|z^{(0)}\|_2^2}{n_0} \end{bmatrix}\right);$$

$$\hat{z}^{(\ell+1)} = \dot{\sigma}(a)b$$

$$(a, b) \sim \mathcal{N}\left(0, \begin{bmatrix} E[(\hat{\alpha}^{(\ell)})^2] & E[\hat{\alpha}^{(\ell)}\hat{z}^{(\ell)}] \\ E[\hat{\alpha}^{(\ell)}\hat{z}^{(\ell)}] & E[(\hat{z}^{(\ell)})^2] \end{bmatrix}\right);$$

$$\tilde{z}_i^{(L)} = \hat{z}^{(L)} \quad \mathcal{N}\left(0, E[(\hat{z}^{(L-1)})^2]\right);$$

where

$$\hat{\alpha}^{(1)} = (a) \quad a \sim \mathcal{N}\left(0, \frac{\|\hat{x}\|_2^2}{n_0}\right);$$

$$\hat{\alpha}^{(\ell+1)} = (a) \quad a \sim \mathcal{N}\left(0, E[(\hat{\alpha}^{(\ell)})^2]\right):$$

Proof. We will prove this by induction for $\ell = 1, \ldots, L-1$.

Basic Step

$$z_i^{(1)} = \dot{\sigma}\left(\frac{1}{\sqrt{n_0}}(W_i^{(0)})^T\hat{x}\right) \frac{1}{\sqrt{n_0}}(W_i^{(0)})^T z^{(0)}$$

Notice that $W_i^{(0)} \sim \mathcal{N}(0, I_{n_0})$. Thus, we have the following:

$$a = \frac{1}{\sqrt{n_0}}(W_i^{(0)})^T\hat{x} \sim \mathcal{N}\left(0, \frac{\|\hat{x}\|_2^2}{n_0}\right)$$

$$b = \frac{1}{\sqrt{n_0}}(W_i^{(0)})^T z^{(0)} \sim \mathcal{N}\left(0, \frac{\|z^{(0)}\|_2^2}{n_0}\right)$$

a and b are not independent:

$$E[ab] = E[(\frac{1}{\sqrt{n_0}}(W_i^{(0)})^T\hat{x})(\frac{1}{\sqrt{n_0}}(W_i^{(0)})^T z^{(0)})] = \frac{1}{n_0}\hat{x}^T E[(W_i^{(0)})(W_i^{(0)})^T]z^{(0)} = \frac{1}{n_0}\hat{x}^T I_{n_0} z^{(0)} = \frac{\hat{x}^T z^{(0)}}{n_0}$$

Note that the result is independent of the index i, we can de ne $\hat{z}^{(1)} = z_i^{(1)}$. Therefore, the base step has been proven.

Inductive Step

$$z_i^{(\ell+1)} = \dot{\sigma}(\frac{1}{\sqrt{n_\ell}}(W_i^{(\ell)})^T \tilde{\sigma}^{(\ell)}(\hat{x})) \frac{1}{\sqrt{n_\ell}}(W_i^{(\ell)})^T z^{(\ell)}$$

Then,

$$a = \frac{1}{\sqrt{n_\ell}}(W_i^{(\ell)})^T \tilde{\sigma}^{(\ell)}(\hat{x}) \sim \mathcal{N}\left(0, \frac{1}{n_\ell}\sum_{i=0}^{n_\ell}((\tilde{\sigma}^{(\ell)}(\hat{x})_i)^2\right)$$

With $n_1, \ldots, n_\ell \to \infty$, $\mathrm{Var}(a) = E[(\hat{\alpha}^{(\ell)})^2]$. Similarly,

$$b = \frac{1}{\sqrt{n_\ell}}(W_i^{(\ell)})^T z^{(\ell)}$$

$$b \sim \mathcal{N}(0, E[(\hat{z}^{(\ell)})^2]) \quad \text{if } n_1, \ldots, n_\ell \to \infty$$

On the other hand,

$$E[ab] = E[(\frac{1}{\sqrt{n_\ell}}(W_i^{(\ell)})^T \sim^{(\ell)}(x))(\frac{1}{\sqrt{n_\ell}}(W_i^{(\ell)})^T z^{(\ell)})] = \frac{1}{n_\ell}(\sim^{(\ell)}(x))^T E[(W_i^{(\ell)})(W_i^{(\ell)})^T] z^{(\ell)} = \frac{1}{n_\ell}(\sim^{(\ell)}(x))^T z^{(\ell)}$$
$$= E[\hat{}^{(\ell)}\hat{z}^{(\ell)}] \quad \text{if } n_1, :::, n_\ell \to 1$$

The recursive de nition is now proven up to layer ℓ 1. Now let's look at the last layer.

$$\hat{z}_i^{(L)} = \frac{1}{\sqrt{n_{L-1}}} W_i^{(L-1)} z^{(L-1)}$$

By similar arguments as before, it is easy to show that with $n_1, :::, n_{L-1} \to 1$, $\hat{z}_i^{(L)} \sim N(0; E[(\hat{}^{(L-1)})^2])$. This concludes the proof. □

Theorem 1. For any data point x_i, $i \in [1; :::; n]$, with probability at least $1 - O(n)e^{-O(n_0)}$,

$$kJ(x_i)k_{op} \le c\sqrt{n_0}$$

where c is a constant and

$$= \sup_{x_i \in \hat{X};\ kz^{(0)}k_2=1} E[(\hat{}^{(L-1)})^2 | z^{(0)}; x_i]$$

Proof. For a xed unit vector $z^{(0)}$ and xed input $\hat{x}$, we know that based on Proposition 2 $\hat{z}_i^{(L)} \sim N(0; E[(\hat{}^{(L-1)})^2 | z^{(0)}; \hat{x}])$. De ne z as

$$z = \frac{1}{E[(\hat{}^{(L-1)})^2 | z^{(0)}; \hat{x}]} k\hat{z}^{(L)}k_2^2 = \chi^2_{n_0}$$

First, notice that we can have the following tail bound for chi-square distribution (for instance, (Kolar & Liu, 2012))

$$Pr[|z/n_0 - 1| \ge \epsilon] \le \exp(-\frac{3}{16} n_0 \epsilon^2)$$

when $\epsilon \in [0; 1/2)$. In this case, let $\epsilon = \frac{1}{3}$. Consider a subset of coordinates M with cardinality $|M| = O(n_0)$ (Allen-Zhu et al., 2018). Taking the ball B of this subspace with $= 1/3$, we know what

$$|B| \cdot 7^{|M|} = e^{|M| \ln 7} = e^{O(n_0)}$$

Then, taking the union bound for all unit vectors in B, we know that

$$\forall z_0 \in B, \quad \bigcup_{z_0} Pr[|z/n_0 - 1| \ge \frac{1}{3}]$$
$$\le \exp(-\frac{1}{48} n_0) \exp(O(n_0)) \le \exp(-O(n_0))$$

Therefore, by the ϵ-net argument (Tao, 2012), for any unit vector u with only non-zero entries in M, we have with probability $1 - \exp(-O(n_0))$,

$$kJ(\hat{x})uk_2^2 \le 2n_0 \cdot kuk_2^2 = C^2 kuk_2^2$$

For any arbitrary vector v, we can decompose it in the following way: $v = u_1 + u_2 + ::: + u_K$ with $K = O(1)$ where each u_i comes from a different non-overlapping coordinate set M_i.

$$kJ(\hat{x})vk_2 \le C \sum_{i=1}^{K} ku_i k_2 \le C\sqrt{K}(\sum_{i=1}^{K} ku_i k_2^2)^{1/2}$$
$$\le O(1)Ckvk:$$

Thus, with probability at least $1 - O(1)\exp(-O(n_0))$,

$$kJ(\hat{x})k_{op} \le O(1)C = O(1)\sqrt{2n_0}$$
$$= c\sqrt{n_0};$$

where c is a constant. Taking the union bound over all the data points concludes the proof. □

B. Proofs for Sec 4.2

Lemma 5. Under the setting in Section 3.1 with sigmoid as the activation function,

$$\Sigma_1^{(L)}(x,x) \quad \frac{1}{4}$$

Proof. For any ℓ, we have

$$\begin{aligned}\Sigma_1^{(\ell+1)}(x,x) &= \Sigma_1^{(\ell)}(x,x) - \Lambda^{(\ell+1)}(x,x) + \Psi^{(\ell+1)}(x,x) \\ &= E_{g\sim N(0,\Sigma^{(\ell)})}[\dot{\sigma}(g(x))^2] \\ &= E_{g\sim N(0,\Sigma^{(\ell)})}\left(\sigma(g(x)) - \frac{1}{2}\right)^2 + \frac{1}{4} \\ &\quad \frac{1}{4}\end{aligned}$$

where $(f(x)) \quad \frac{1}{2}$ moves sigmoid function to the origin such that it is an odd function. □

C. Proofs for Sec 4.3

C.1. Main Lemmas

Lemma 2. Suppose there is a 2-layer network. If the activation function is $\sigma(x) = x$, $n = n_0$ and the data matrix is full rank. Then at NTK limit, $J_1(x) = I_{n_0}$.

Proof.

$$J_1(x) = \frac{2}{n_0}(\hat{X} \quad f_0(\hat{X}))\left(\frac{2}{n_0}\hat{X}^T\hat{X}\right)^{-1}\hat{X}^T + J_0(x)$$

Notice that $J_0(x) = \sqrt{\frac{1}{n_1}}\sqrt{\frac{1}{n_0}}W^{(1)}W^{(0)}$ and $f_0(\hat{X}) = \sqrt{\frac{1}{n_1}}\sqrt{\frac{1}{n_0}}W^{(1)}W^{(0)}\hat{X} = J_0(x)\hat{X}$.

$$\begin{aligned}J_1(x) &= J_0(x) \quad f_0(\hat{X})(\hat{X}^T\hat{X})^{-1}\hat{X}^T + \hat{X}(\hat{X}^T\hat{X})^{-1}\hat{X}^T \\ &= \sqrt{\frac{1}{n_1}}\sqrt{\frac{1}{n_0}}W^{(1)}W^{(0)} \quad \sqrt{\frac{1}{n_1}}\sqrt{\frac{1}{n_0}}W^{(1)}W^{(0)}\hat{X}(\hat{X}^T\hat{X})^{-1}\hat{X}^T + \hat{X}(\hat{X}^T\hat{X})^{-1}\hat{X}^T \\ &= \sqrt{\frac{1}{n_1}}\sqrt{\frac{1}{n_0}}W^{(1)}W^{(0)} \quad \sqrt{\frac{1}{n_1}}\sqrt{\frac{1}{n_0}}W^{(1)}W^{(0)}I_{n_0} + I_{n_0} \\ &= I_{n_0}\end{aligned}$$

□

Lemma 3. Suppose there is a 2-layer network with activation function $\sigma(x) = x$ and given initial weights $W^{(1)} \in R^{n_0 \quad n_1}$, $W^{(0)} \in R^{n_1 \quad n_0}$. If the data matrix is full rank with $n \quad n_0$, then, at the NTK limit ($n_1 \to \infty$), $J_1(x)$ has eigenvalue 1 with multiplicity at least n. If at the NTK limit, α is chosen such that $\|J_0(x)\|_{op} < 1$, then the multiplicity is exactly n and 1 is the largest eigenvalue norm.

Proof. Based on the proof of last section, we know that

$$\begin{aligned}J_1(x) &= J_0(x) \quad f_0(\hat{X})(\hat{X}^T\hat{X})^{-1}\hat{X}^T + \hat{X}(\hat{X}^T\hat{X})^{-1}\hat{X}^T \\ &= J_0(x) \quad J_0(x)\hat{X}(\hat{X}^T\hat{X})^{-1}\hat{X}^T + \hat{X}(\hat{X}^T\hat{X})^{-1}\hat{X}^T\end{aligned}$$

In this case,

$$\hat{X}(\hat{X}^T\hat{X})^{-1}\hat{X}^T = V \quad V^T$$

where V is an orthgonal matrix and

$$= \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & \cdots & 1 & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}$$

So

$$\begin{aligned} J_1(x) &= J_0(x)(I_{n_0} - V V^T) + V V^T \\ &= \frac{1}{\sqrt{n_1}} \frac{1}{\sqrt{n_0}} W^{(1)} W^{(0)} (I_{n_0} - V V^T) + V V^T \end{aligned}$$

Interestingly, $(I_d - V V^T)$ and $V V^T$ contain orthogonal eigenvectors. For convenience, let $\{v_i\}_{i=1}^n$ be the set of eigenvectors of $V V^T$ with eigenvalue 1. Furthermore, let $V_k = \text{span}(\{v_i\}_{i=1}^n)$ and $V_? = \text{span}(\{v_i\}_{i=1}^n)^?$. Because we are in the linear region, $J_1(x)$ and $J_0(x)$ do not depend on x. We'll use J_1 to refer $J_1(x)$ and J_0 as $J_0(x)$.

For any vector $v^k \in V_k$,

$$J_0(I_{n_0} - V V^T) v^k = 0$$

and

$$J_1 v^k = v^k$$

Thus, all vectors in $\{v_i\}_{i=1}^n$ are eigenvetors of J_1 with eigenvalue 1 regardless of the choice of α.

On the other hand, let v be any complex vector such that

$$v = \text{Re}(v) + i\,\text{Im}(v)$$

If v is an eigenvector of J_1 with eigenvalue $\lambda = a + ib$, then

$$\begin{aligned} J_1 \text{Re}(v) &= a\text{Re}(v) - b\text{Im}(v) \\ J_1 \text{Im}(v) &= b\text{Re}(v) + a\text{Im}(v) \end{aligned}$$

Let's first decompose $\text{Re}(v)$ and $\text{Im}(v)$.

$$\begin{aligned} \text{Re}(v) &= v_r^? + v_r^k \\ \text{Im}(v) &= v_i^? + v_i^k \end{aligned}$$

where $v_r^?, v_i^? \in V_?$ and $v_r^k, v_i^k \in V_k$.

$$\begin{aligned} J_1(v_r^? + v_r^k) &= J_0 v_r^? + v_r^k = (a v_r^? - b v_i^?) + (a v_r^k - b v_i^k) \\ J_1(v_i^? + v_i^k) &= J_0 v_i^? + v_i^k = (b v_r^? + a v_i^?) + (b v_r^k + a v_i^k) \end{aligned}$$

By adding and subtracting two equations,

$$J_0(v_r^? + v_i^?) + v_r^k + v_i^k = (a+b) v_r^? + (a-b) v_i^? + (a+b) v_r^k + (a-b) v_i^k$$

$$J_0(v_r^? - v_i^?) + v_r^k - v_i^k = (a-b) v_r^? - (a+b) v_i^? + (a-b) v_r^k - (a+b) v_i^k$$

When α is chosen such that $\|J_0\| < 1$,

$$\begin{aligned} \|(a+b) v_r^? + (a-b) v_i^?\|_2 &< \|v_r^? + v_r^k\|_2 \\ \|(a-b) v_r^? - (a+b) v_i^?\|_2 &< \|v_r^? - v_r^k\|_2 \end{aligned}$$

Then,

$$(a^2 + b^2)\|v_r^? k_2^2 + (a^2 + b^2)\|v_i^? k_2^2 < \|v_r^? k_2^2 + \|v_i^? k_2^2$$
$$|\ |^2 = a^2 + b^2 < 1$$

This suggests that any complex eigenvector with components from $V_?$ would have eigenvalue with norm smaller than 1.

□

Lemma 4. Suppose there is a 2-layer network with activation function $(x) = x + $, given initial weights $W^{(1)} 2 R^{n_0 \ n_1}$, $W^{(0)} 2 R^{n_1 \ n_0}$ and every data point has the same norm r (i.e. $8i 2 [n] \ kx k_2 = r$). If the data matrix is full rank with $n \ n_0$, then, at the NTK limit $n_1 ! 1$, $J_1(x)$ has eigenvalues 1 with multiplicity at least $n \ 1$. If at the NTK limit, and are chosen such that

$$kJ_0(x)k_{op} = 1 \quad ; \quad p\frac{1}{n_1} W^{(1)} 1_{n_1} \ _2 < \frac{n_0}{2r^2};$$

where $0 < \ 1$, then the multiplicity is exactly $n \ 1$ and 1 is the largest eigenvalue norm.

Proof. First of all, let B be an all-one matrix

$$\begin{aligned}
J_1(x) &= \hat{X} \quad f_0(\hat{X}) \quad K^{-1} \frac{@k}{@x} + J_0(x) \\
&= \hat{X} \quad f_0(\hat{X}) \quad \frac{2^2}{n_0} \hat{X}^T \hat{X} + {}^2 B \quad ^1 \quad \frac{2^2}{n_0} \hat{X}^T \quad + J_0(x) \\
&= \hat{X} \quad f_0(\hat{X}) \quad \hat{X}^T \hat{X} + \frac{n_0}{2^2} B \quad ^1 \hat{X}^T + J_0(x) \\
&= J_0(x) + \hat{X} \quad \hat{X}^T \hat{X} + \frac{n_0}{2^2} B \quad ^1 \hat{X}^T \quad (p\frac{}{n_1 n_0} W^{(1)} W^{(0)} \hat{X} + p\frac{1}{n_1} W^{(1)} 1_{n_1} 1_n^T) \ \hat{X}^T \hat{X} + \frac{n_0}{2^2} B \quad ^1 \hat{X}^T \\
&= J_0(x) + \hat{X} \quad \hat{X}^T \hat{X} + \frac{n_0}{2^2} B \quad ^1 \hat{X}^T \quad (J_0(x) \hat{X} + p\frac{1}{n_1} W^{(1)} 1_{n_1} 1_n^T) \ \hat{X}^T \hat{X} + \frac{n_0}{2^2} B \quad ^1 \hat{X}^T
\end{aligned}$$

Because in the linearized region $J_1(x)$ and $J_0(x)$ do not depend on x. We'll use J_1 to refer $J_1(x)$ and J_0 as $J_0(x)$. For simplicity, we'll also use $c = \frac{n_0}{2^2}$.

Based on Lemma 6,

$$\hat{X} \quad \hat{X}^T \hat{X} + cB \quad ^1 \hat{X}^T = V \quad V^T$$

where $= \text{diag}(\underbrace{1; ...; 1}_{n \ 1}, \hat{}, \underbrace{0; ...; 0}_{n_0 \ n})$ where $0 < \hat{} < 1$. Now,

$$J_1 = J_0(I_{n_0} \quad V \ V^T) + V \ V^T \quad p\frac{1}{n_1} W^{(1)} 1_{n_1} 1_n^T \ \hat{X}^T \hat{X} + cB \quad ^1 \hat{X}^T$$

From Corollary 7, we know that the following two vectors are eigenvectors of W^T with eigenvalue $\hat{}$,

$$\hat{X}(\hat{X}^T \hat{X} + cB) \ ^1 1_n \qquad \hat{X}(\hat{X}^T \hat{X}) \ ^1 1_n$$

Furthermore,

$$\hat{X}(\hat{X}^T \hat{X} + cB) \ ^1 1_n = \hat{} \hat{X}(\hat{X}^T \hat{X}) \ ^1 1_n$$

And

$$\hat{\alpha} = \frac{1}{1 + cg}$$

where

$$g = \text{trace}(B(\hat{X}^T\hat{X})^{-1}) = \|\hat{X}(\hat{X}^T\hat{X})^{-1}1_n\|_2^2$$

Let $\hat{v}$ be a rescaled unit vector of $\hat{X}(\hat{X}^T\hat{X})^{-1}1_n$, then

$$J_1\hat{v} = J_0(1-\hat{\alpha})\hat{v} + \hat{\alpha}\hat{v} - \sqrt{g}\hat{\alpha}\frac{1}{\sqrt{n_1}}W^{(1)}1_{n_1}\hat{v}$$

$$\|J_1\hat{v}\|_2 = \|J_0(1-\hat{\alpha})\hat{v} + \hat{\alpha}\hat{v} - \sqrt{g}\hat{\alpha}\frac{1}{\sqrt{n_1}}W^{(1)}1_{n_1}\hat{v}\|_2$$

$$\|J_0\|_{op}\|(1-\hat{\alpha})\hat{v}\|_2 + \|\hat{\alpha}\hat{v}\|_2 + \sqrt{g}\hat{\alpha}\frac{1}{\sqrt{n_1}}W^{(1)}1_{n_1}\hat{v}\|_2$$

$$= (1-\hat{\alpha})\|J_0\|_{op} + \hat{\alpha} + \sqrt{g}\hat{\alpha}\|\frac{1}{\sqrt{n_1}}W^{(1)}1_{n_1}\|_2$$

$$< (1-\hat{\alpha})(1-\delta) + \hat{\alpha} + \sqrt{g}\hat{\alpha}\sqrt{\frac{2n_0}{2r^2}}$$

$$(1-\hat{\alpha})(1-\delta) + \hat{\alpha} + g\hat{\alpha}\sqrt{\frac{2n_0}{2\delta^2}} \qquad \text{(Lemma 9)}$$

$$= \frac{(1-\delta)\ cg + 1 + \frac{g\sqrt{2n_0}}{2\delta}}{1+cg} = 1$$

Therefore, J_1 will shrink every vectors orthogonal to the eigenvectors in V with eigenvalue 1. By the same arguments in the proof of Lemma 3, we can conclude the proof. □

C.2. Useful Lemmas

Lemma 6. Suppose $X \in R^{k\times m}$ is a full-rank matrix with $k \geq m$ and $m \geq 2$. Let c be an arbitrary positive constant and B an all-one matrix. Consider the following real symmetric matrix,

$$X(X^TX + cB)^{-1}X^T$$

It can be characterized by having eigenvalue 1 with multiplicity $m-1$, eigenvalue 0 with multiplicity $k-m$ and another eigenvalue α such that $0 < \alpha < 1$.

Proof. By (Miller, 1981), if P and $P+Q$ are invertible, and Q has rank 1, then let $g^0 = \text{trace}(QP^{-1})$, we know that $g^0 \neq -1$, and

$$(P+Q)^{-1} = P^{-1} - \frac{1}{1+g^0}P^{-1}QP^{-1}$$

First of all, it is easy to see that $(X^TX + cB)^{-1}$ is invertible. This is because X^TX is positive de nite and cB is positive semi-de nite.

Since B is a rank one matrix,

$$(X^TX + cB)^{-1} = \underbrace{(X^TX)^{-1}}_{I_1} - \underbrace{\frac{c}{1+cg}(X^TX)^{-1}B(X^TX)^{-1}}_{I_2}$$

where $g = \text{trace}(B(X^TX)^{-1})$.

Let's consider the singular value decomposition of X^T, $X^T = U\ \ V^T$

$X^TX = U\ \ ^2U^T$ and $(X^TX)^{-1} = U\ \ ^{-2}U^T$. So

$$XI_1X^T = X(X^TX)^{-1}X^T = V\ \ U^TU\ \ ^{-2}U^TU\ \ V^T = V\ \ _mV^T$$

where $_m = \text{diag}(\underbrace{1,\ldots,1}_{m},\underbrace{0,\ldots,0}_{k-m})$

$$X^T I_2 X = \frac{c}{1+cg} M = \frac{c}{1+cg} X(X^T X)^{-1} B (X^T X)^{-1} X^T$$

The first thing to notice is that $B = 11^T$ where 1 is a vector of ones. Therefore,

$$M = X(X^T X)^{-1} B (X^T X)^{-1} X^T = X(X^T X)^{-1} 11^T (X^T X)^{-1} X^T = aa^T$$

where $a = X(X^T X)^{-1} 1$.

This implies that M is a rank one matrix with singular value $\|a\|^2$. But we also know the following:

$$\begin{aligned}\|a\|^2 = a^T a &= \text{trace}(aa^T) = \text{trace}(M) \\ &= \text{trace}(X(X^T X)^{-1} B (X^T X)^{-1} X^T) = \text{trace}(X^T X (X^T X)^{-1} B (X^T X)^{-1}) \\ &= \text{trace}(B(X^T X)^{-1}) = g > 0\end{aligned}$$

The last strict inequality comes from the fact that X is full rank so that $X(X^T X)^{-1}$ has no zero singular value. Furthermore,

$$\begin{aligned}X I_1 X^T a &= X(X^T X)^{-1} X^T a \\ &= X(X^T X)^{-1} X^T X (X^T X)^{-1} 1 = X(X^T X)^{-1} 1 \\ &= a\end{aligned}$$

Because a is not a zero vector, it is also one of the eigenvectors of $X I_1 X^T$ with eigenvalue 1.

And the eigenvalue of $X^T I_2 X$ is the following:

$$0 < \frac{cg}{1+cg} < 1$$

The inequalities comes from the fact that c is also non-negative. We'll denote $= \frac{cg}{1+cg}$. So

$$X I_2 X^T = \hat{a}\hat{a}^T$$

where $\hat{a}$ is a rescaled to have unit length.

Now that we have examined two parts separately. Let's put them together. For convenience, we'll also denote $X(X^T X + cB)^{-1} X^T = X I_1 X^T - X I_2 X^T = M_1 - M_2$.

Based on the eigen decomposition of M_1,

$$M_1 = \sum_{k=1}^{m} u_k u_k^T$$

with lost of generality, let's also denote $\hat{a} = u_1$. Now,

$$\begin{aligned}M_1 - M_2 &= \sum_{k=1}^{m} u_k u_k^T - u_1 u_1^T \\ &= (1-)u_1 u_1^T + \sum_{k=2}^{m} u_k u_k^T\end{aligned}$$

Because $0 < < 1$, $X(X^T X + cB)^{-1} X^T$ has eigenvalue 1 with multiplicity $m - 1$, eigenvalue 0 with multiplicity $k - m$ and another eigenvalue such that $0 < < 1$. □

Corollary 7. Following the setup in Lemma 6, we could also know that $X(X^T X)^{-1} 1$ is an eigenvector with with eigenvalue and

$$X(X^T X + cB)^{-1} X^T \cdot X(X^T X)^{-1} 1 = X(X^T X + cB)^{-1} 1 = \cdot X(X^T X)^{-1} 1$$

Corollary 8. Suppose $X \in \mathbb{R}^{k \times m}$ is a full-rank matrix with $k \geq m$ and $m \geq 2$. Let c be an arbitrary non-negative constant and B an all-one matrix.

$$\|X(X^TX + cB)^{-1}X^T\|_{op} = 1$$

Remark 2. c can also takes on negative values as long as $|c|$ is not close to 1.

Lemma 9. Suppose $X \in \mathbb{R}^{k \times m}$ is a full-rank matrix with $k \geq m$ and B an all-one matrix. If

$$\|X_{:,i}\|_2 = r \qquad \forall i \in [m]$$

Then,

$$\text{trace}(B(X^TX)^{-1}) \geq \frac{1}{r^2}$$

Proof. First of all,

$$\text{trace}(B(X^TX)^{-1}) \geq \text{trace}(\mathbf{1}^T(X^TX)^{-1}\mathbf{1})$$
$$\geq \|\mathbf{1}\|_2^2 \frac{1}{\|X^TX\|_{op}} = \frac{m}{\|X^TX\|_{op}}$$

On the hand,

$$\|X^TX\|_{op} = \|X^T\|_{op}^2 \leq \|X^T\|_f^2 \leq \text{trace}(X^TX) \leq r^2 m$$

Therefore,

$$\text{trace}(B(X^TX)^{-1}) \geq \frac{1}{r^2}$$

□

D. Proofs for Sec 4.4

D.1. Derivation for the Approximated NTK

The closed form NTK of erf (Lee et al., 2019; Williams, 1997) can be written with the following two components:

$$T(\infty; \text{erf}, \text{erf})(x, \hat{x}) = \frac{2}{\pi}\arcsin\frac{\Sigma(x, \hat{x})}{\sqrt{(\Sigma(x, x) + 0.5)(\Sigma(\hat{x}, \hat{x}) + 0.5)}}$$
$$T(\infty; \dot{\text{erf}}, \dot{\text{erf}})(x, \hat{x}) = \frac{4}{\pi}\det(I + 2\Sigma)^{-\frac{1}{2}} = \frac{4}{\pi}\frac{1}{\sqrt{(1 + 2\Sigma(x, x))(1 + 2\Sigma(\hat{x}, \hat{x})) - 4\Sigma(x, \hat{x})^2}}$$

Here, we can approximate sigmoid function σ_s by erf function:

$$\sigma_s(x) \approx \hat{\sigma}_s(x) = \frac{1}{2}\text{erf}(\frac{1}{2}x) + \frac{1}{2}$$

Then,

$$T(\infty; \sigma_s, \sigma_s)(x, \hat{x}) = \mathbb{E}_{u,v \sim \mathcal{N}(0, \Sigma)}[\sigma_s(u)\sigma_s(v)] = \mathbb{E}[\frac{1}{4}\text{erf}(\frac{1}{2}u)\text{erf}(\frac{1}{2}v)] + \mathbb{E}[\frac{1}{4}\text{erf}(\frac{1}{2}u) + \frac{1}{4}\text{erf}(\frac{1}{2}v)] + \frac{1}{4}$$
$$= \frac{1}{4}\mathbb{E}[\text{erf}(\frac{1}{2}u)\text{erf}(\frac{1}{2}v)] + \frac{1}{4}$$
$$= \frac{1}{4}T(\frac{1}{4}\Sigma; \text{erf}, \text{erf})(x, \hat{x}) + \frac{1}{4}$$

$$\boxed{T(\infty; \sigma_s, \sigma_s)(x, \hat{x}) = \frac{1}{4} + \frac{1}{2\pi}\arcsin\frac{\Sigma(x, \hat{x})}{\sqrt{(\Sigma(x, x) + 2)(\Sigma(\hat{x}, \hat{x}) + 2)}}}$$

and

$$T(\ ;\dot{\sigma};\dot{\sigma})(x;\hat{x}) = E_{u;v\ N\ (0;\ )}[\dot{\sigma}(u)\dot{\sigma}(v)] = \frac{1}{16}E[\operatorname{erf}(\frac{1}{2}u)\operatorname{erf}(\frac{1}{2}v)]$$
$$= \frac{1}{16}T(\frac{1}{4}\ ;\operatorname{erf};\operatorname{erf})(x;\hat{x})$$

$$\boxed{T(\ ;\dot{\sigma};\dot{\sigma})(x;\hat{x}) = \frac{1}{2}\,p\frac{1}{(2+\ (\ x;x)(2+\ (\ \hat{x};\hat{x}))\quad (\ x;\hat{x})^2)}}$$

Based on the de nition of NTK, we can derive the following forΣ^g

$$\Sigma_1^1(\hat{x};x) = \quad {}^1(\hat{x};x) = \frac{1}{n_0}\hat{x}^T x$$
$$\Sigma_1^2(\hat{x};x) = \quad \Sigma_1^1(\hat{x};x)T(\ \Sigma_1^1\ ;\dot{\sigma};\dot{\sigma})(x;\hat{x}) + T(\ \Sigma_1^1\ ;\ \dot{\sigma};\ \dot{\sigma})(x;\hat{x})$$

Let's look at the rst part

$$\Sigma_1^1(\hat{x};x)T(\ \Sigma_1^1\ ;\dot{\sigma};\dot{\sigma})(x;\hat{x}) = \frac{1}{2}\,q\frac{1}{(2+\frac{1}{n_0}x^T x)(2+\frac{1}{n_0}\hat{x}^T\hat{x})\quad(\frac{1}{n_0}\hat{x}^T x)^2)}[\frac{1}{n_0}\hat{x}^T x]$$
$$= \frac{1}{2}\,p\frac{\hat{x}^T x}{(2n_0+x^T x)(2n_0+\hat{x}^T\hat{x})\quad(\hat{x}^T x)^2)}$$

and the second part

$$T(\ \Sigma_1^1\ ;\ \dot{\sigma};\ \dot{\sigma})(x;\hat{x}) = \frac{1}{4}+\frac{1}{2}\arcsin\quad q\frac{\frac{1}{n_0}\hat{x}^T x}{(\frac{1}{n_0}x^T x+2)(\frac{1}{n_0}\hat{x}^T\hat{x}+2)}$$
$$= \frac{1}{4}+\frac{1}{2}\arcsin\quad p\frac{\hat{x}^T x}{(x^T x+2n_0)(\hat{x}^T\hat{x}+2n_0)}$$

D.2. Detailed Discussion of$\frac{@\Sigma^L x}{@x}$

Without loss of generality, we will focus on$\frac{@\Sigma^L x}{@x}j_{x_1}$,

$$\frac{@\Sigma^L x}{@x} = \begin{bmatrix}\frac{\Sigma_1^L(x_1;x)}{@x}\\ :::\\ \frac{\Sigma_1^L(x_n;x)}{@x}\end{bmatrix}$$

where

$$\frac{@\ \Sigma_1^L(\hat{x};x)}{@x} = \underbrace{\frac{@T(\ \Sigma_1^1\ ;\ \dot{\sigma};\ \dot{\sigma})(\hat{x};x))}{@x}}_{\Sigma_1^g(\hat{x};x)} + \underbrace{\frac{@\ \Sigma_1^1(\hat{x};x))T(\ \Sigma_1^1\ ;\dot{\sigma};\dot{\sigma})(\hat{x};x))}{@x}}_{\Sigma_2^g(\hat{x};x)}$$

Let's look at each row separately, and break this down into two parts.

$\Sigma_1^g(\hat{x};x)$

After deriving the derivative, we get this:

$$\Sigma_1^g(\hat{x};x) = \frac{1}{2}\,p\frac{1}{1\quad A^2}\frac{\hat{x}\ (\hat{x}^T\hat{x}+2n_0)(x^T x+2n_0)\qquad x\ (\hat{x}^T\hat{x}+2n_0)x^T\hat{x}}{(\hat{x}^T\hat{x}+2n_0)(x^T x+2n_0)^{\frac{3}{2}}}$$

$$A = p\frac{\hat{x}^T x}{(x^T x+2n_0)(\hat{x}^T\hat{x}+2n_0)}$$

Since we are only interested in $J_{\hat{f}_1}(x_1)$ and each row of $\frac{\partial \hat{f}_1}{\partial x}$, we'll examine $l_1^g(x_i; x_1)$.

$$l_1^g(x_i; x_1) = \frac{1}{2}\sqrt{\frac{r^2 + 2n_0}{(r^2 + 2n_0)^2 - (r^2 \cdot_{i1})^2}} \frac{x_i(r^2 + 2n_0)^2 - x_1 \cdot_{i1}(r^2 + 2n_0)r^2 \cdot_{i1}}{(r^2 + 2n_0)^3}$$
$$= \frac{1}{2}\sqrt{\frac{1}{(r^2 + 2n_0)^2 - (r^2 \cdot_{i1})^2}} \frac{x_i(r^2 + 2n_0) - x_1 r^2 \cdot_{i1}}{r^2 + 2n_0}$$

It is easy to see that $l_1^g(x_i; x_1) \to 0$ as r grows regardless of $\cdot_{i1}$.

$l_2^g(\hat{x}; x)$

We know that

$$l_2^g(\hat{x}; x) = \frac{1}{2} \frac{\hat{x} \cdot (x^T x + 2n_0)(\hat{x}^T \hat{x} + 2n_0) - (\hat{x}^T x)^2 \cdot \hat{x} - x^T x \cdot (2n_0 + \hat{x}^T \hat{x})x - (\hat{x}^T x)\hat{x}}{(x^T x + 2n_0)(\hat{x}^T \hat{x} + 2n_0) - (\hat{x}^T x)^2}^{\frac{3}{2}}$$

Again, let's examine $l_2^g(x_i; x_1)$.

$$l_2^g(x_i; x_1) = \frac{1}{2} \frac{(r^2 + 2n_0)^2 x_i - r^2 \cdot_{i1}(2n_0 + r^2)x_1}{(r^2 + 2n_0)^2 - r^4 \cdot_{i1}^2}^{\frac{3}{2}}$$

$$\|l_2^g(x_i; x_1)\|_2^2 = \frac{1}{4} \cdot^2 \frac{r^2 \cdot (r^2 + 2n_0)^4 + r^4 \cdot_{i1}^2(2n_0 + r^2)^2 - 2r^2 \cdot_{i1}^2(2n_0 + r^2)^3}{(r^2 + 2n_0)^2 - r^4 \cdot_{i1}^2}^3$$

$$= \frac{1}{4} \cdot^2 \frac{\begin{array}{l} 16n_0^4 + r^2 \cdot n_0^3(32 - 16 \cdot_{i1}^2) + r^2 \cdot n_0^2(24 - 20 \cdot_{i1}^2) + r^2 \cdot n_0(8 - 8 \cdot_{i1}^2) + r^2(1 - \cdot_{i1}^2) \end{array}}{r^4(1 - \cdot_{i1}^2) + 4n_0 r^2 + 4n_0^2}^3$$

Based on the equation for $\|l_2^g(x_i; x_1)\|_2^2$, we know that if $\cdot_{i1}^2 \neq 1$, $\|l_2^g(x_i; x_1)\|_2^2$ eventually decays to zero with larger r. But $\|l_2^g(x_i; x_1)\|_2^2$ converges to a constant if $\cdot_{i1}^2 = 1$. For simplicity, in this section, we do not assume there is any parallel input. Therefore, we can see that all the other terms will go to zero except $l_2^g(x_1; x_1)$. It is worth noting that if $\cdot_{i1}$ is close to one, the norm will see a spike before going down to zero. But in practice, the data is more than likely to be well separated with small $|\cdot_{ij}|$. The discussion here is illustrated in Figure. D.1.

Combining the above analysis on the two components of gradient, it is easy to see that with large r,

$$\frac{\partial \hat{x}}{\partial x} J_{x_1} \approx \begin{bmatrix} \frac{\partial l_1^g(x_1; x)}{\partial x} J_{x_1} \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

$$\left\| \frac{\partial \hat{x}}{\partial x} J_{x_1} \right\|_{op} \approx \left\| \frac{\partial l_1^g(x_1; x)}{\partial x} J_{x_1} \right\|_2 \approx \left\| \frac{1}{2} \frac{2n_0(r^2 + 2n_0)}{(4n_0 r^2 + 4n_0^2)^{\frac{3}{2}}} x_1 \right\|_2 = \frac{1}{2} \frac{2n_0 r(r^2 + 2n_0)}{(4n_0 r^2 + 4n_0^2)^{\frac{3}{2}}} \to \frac{1}{8}\sqrt{n_0}$$

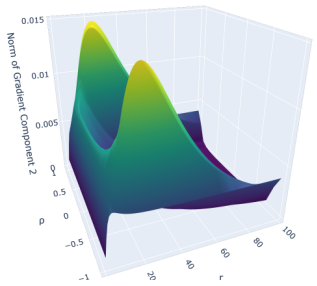


Figure D.1: , r vs Norm of Gradient Component 2

D.3. Parallel Inputs Analysis

In the previous section, we assume that there are no parallel inputs. But this assumption is not necessary. In fact, given training data $\{x_i\}_1^n$, w.l.o.g, let's impose $x_1 = -x_2$. Based on the results we have in Section 4.4, we can still derive a similar approximation for the NTK regression solution.

- $\tilde{K}$

  Fisrt of all,

$$K_{ij}^1 = \mathcal{T}(\Theta_1^1; \ s; \ s)(x_i; x_j) = \frac{1}{4} + \frac{1}{2} \arcsin \frac{r^2 \ _{i;j}}{(r^2 + 2n_0)}$$

  If $_{i;j} = 1$, then K_{ij}^1 is going to converge to $\frac{1}{2}$ as r grows bigger. But if $_{i;j} = -1$, this term is going to zero.

  Therefore, $\tilde{K}$ can be approximated by this block diagonal matrix.

$$\tilde{K} \approx \begin{bmatrix} B_1 & \cdots & 0 \\ 0 & B_2 & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & B_2 \end{bmatrix}$$

  where

$$B_1 = \begin{matrix} I_k + \frac{1}{2} & -I_k \\ -I_k & I_k + \frac{1}{2} \end{matrix} \qquad B_2 = I_k + \frac{1}{2} \qquad I_k = \frac{1}{2} - \frac{r^2}{4n_0^2 + 4n_0 r^2} \approx \frac{r}{4\sqrt{n_0}}$$

  The inverse of $\tilde{K}$, is the following, as r grows large:

$$\tilde{K}^{-1} \approx \begin{bmatrix} B_1^{-1} & \cdots & 0 \\ 0 & \frac{1}{I_k + \frac{1}{2}} & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & \frac{1}{I_k + \frac{1}{2}} \end{bmatrix} \approx \begin{bmatrix} B_1^{-1} & \cdots & 0 \\ 0 & \frac{4\sqrt{n_0}}{r} & 0 \\ \vdots & \vdots & \vdots \\ 0 & \cdots & \frac{4\sqrt{n_0}}{r} \end{bmatrix}$$

  where

$$B_1^{-1} = \frac{1}{I_k + \frac{1}{4}} \begin{matrix} I_k + \frac{1}{2} & I_k \\ I_k & I_k + \frac{1}{2} \end{matrix}$$

- $\frac{\partial k_x}{\partial x}$

  Based on the discussion from Section 4.4,

$$\frac{\partial k_x}{\partial x}|_{x_1} \approx \begin{bmatrix} \frac{L_1^1(x_1;x)}{\partial x}|_{x_1} \\ -\frac{L_1^1(x_1;x)}{\partial x}|_{x_1} \\ \vdots \\ 0 \end{bmatrix} = \begin{bmatrix} J_k x_1 \\ -J_k x_1 \\ \vdots \\ 0 \end{bmatrix}$$

where

$$J_k = \frac{1}{2}\frac{2n_0(r^2+2n_0)}{(4n_0r^2+4n_0^2)^{\frac{3}{2}}} \approx \frac{1}{8}\frac{1}{\sqrt{n_0}}\frac{1}{r}$$

Finally,

$$\hat{X} - f_0(\hat{X}) \quad \tilde{K}^{\,1}\frac{@k_x}{@x} \approx \hat{X}\tilde{K}^{\,1}\frac{@k_x}{@x} \approx \hat{X}\begin{bmatrix} B_1^{\,1} & \cdots & 0 & \begin{bmatrix} J_k x_1 \\ -J_k x_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ 0 & \frac{1}{l_k+\frac{1}{2}} & 0 & \\ \vdots & \vdots & \vdots & \\ 0 & \cdots & \frac{1}{l_k+\frac{1}{2}} & \end{bmatrix}$$

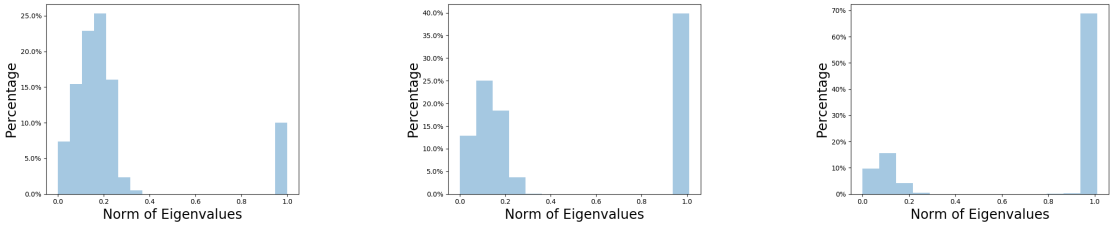
$$= \hat{X}\begin{bmatrix} \frac{\frac{1}{2}J_k}{l_k+\frac{1}{4}}x_1 \\ -\frac{\frac{1}{2}J_k}{l_k+\frac{1}{4}}x_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} = 2\frac{\frac{1}{2}J_k}{l_k+\frac{1}{4}}x_1x_1^T$$

Thus,

$$\| \quad \hat{X} - f_0(\hat{X}) \quad \tilde{K}^{\,1}\frac{@k_x}{@x}\|_{op} \approx \frac{r^2 J_k}{l_k+\frac{1}{4}} = \frac{r^2\frac{1}{8}\frac{1}{\sqrt{n_0}}\frac{1}{r}}{\frac{1}{4}\frac{1}{\sqrt{n_0}}} = \frac{1}{2}$$

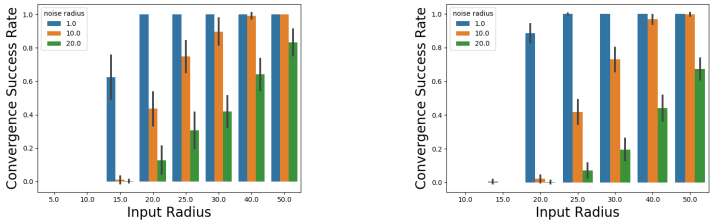
By similar argument, as $r \to \infty$, we have

$$\|J_1(x)\|_{op} \leq \frac{1}{2}$$



(a) 2 training points (b) 5 training points (c) 8 training points

Figure E.1: Eigenvalue distribution of 2-layer sigmoid network trained with input dimension 10



(a) 5 training points (b) 20 training points (c) 40 training points

Figure E.2: Convergence success rate vs input norm: random data with input dimension 32

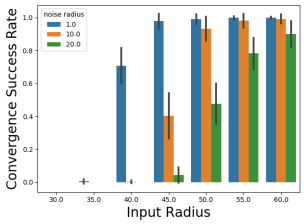
E. Additional Simulations

E.1. Multiple Points: Linear Region

In this section, we first illustrate the eigenvalue distribution in the linear region. Here, we trained 2 layer $\mathrm{sigmoid}$ networks with input dimension 10 and hidden size 1000 for 2, 5 and 8 training points. As suggested by Lemma 4, there should be $n-1$ eigenvalues with norm around 1. This is supported by Figure E.1, as there are 10%, 40% and 70% eigenvalues around that region.

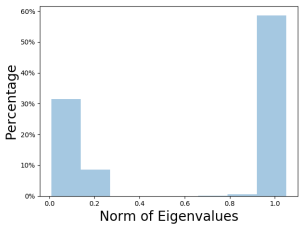
E.2. Basin of Attraction

We test basin of attraction by adding Gaussian noises to training examples and check if the modified examples can converge to the original ones via iterative maps under 50 iterations. The standard deviation of the Gaussian noise is called the noise radius. The network has 2 layers with hidden size 10000 and input dimension 32. Figure E.3 details experiments for 5, 20 and 40 examples. Not surprisingly, the basin of attraction is larger when there are fewer training examples and larger input norms since a level of separation between data is required.

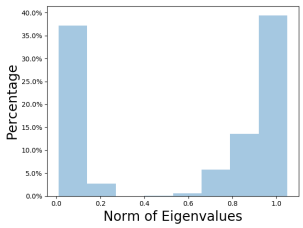


(a) 5 training points (b) 20 training points

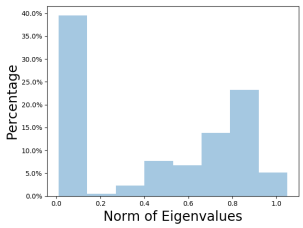
Figure E.3: Convergence success rate vs input norm: MNIST dataset



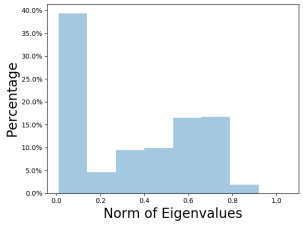
(a) Radius: 1



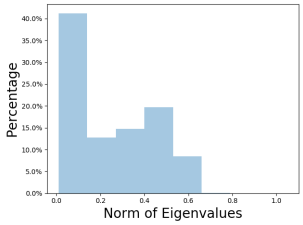
(b) Radius: 5



(c) Radius: 10



(d) Radius: 15



(e) Radius: 20

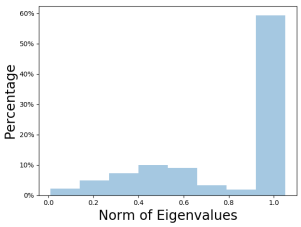
Figure E.4: Specturm Change for Sigmoid

E.3. Basin of Attraction on Mnist

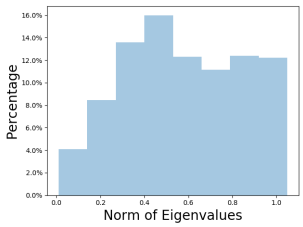
We also test basin of attraction experiments on MNIST dataset to check if we can recover real training examples. The images are prepossessed by subtracting means and rescaled to have different input norms for testing. Similar to the setting before, Figure 4b also shows that larger input norm gives greater basin of attraction for 5 and 20 examples. Notice that because MNIST images have large input dimension, they need larger radius to move out of the linear region.

E.4. Sigmoidal Activations

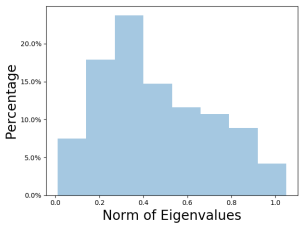
Finally, we show that our results can be extended to different sigmoidal activation functions as well. We chose 2 layer network with hidden size 10000, input dimension 32 and 20 training examples. As before, only settings that can let network converges to training loss below 10^{-7} are included. Figure 5 clearly suggests all the activation functions share similar curves. Notice that both $\tanh$ and erf have large eigenvalue when r is small. This is not a contradiction to our Lemma 3 as their $\dot{=}(0)$ is too large to satisfy the conditions in Lemma 3. The histogram of eigenvalue norm changes for those activation is shown in Figure E.4, Figure E.5, Figure E.6. It is clear that they all follow the same pattern.



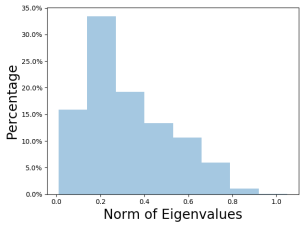
(a) Radius: 1



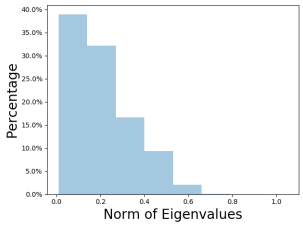
(b) Radius: 5



(c) Radius: 10

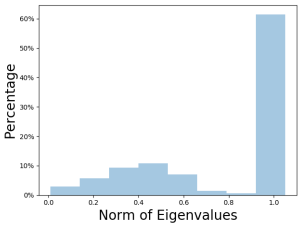


(d) Radius: 15

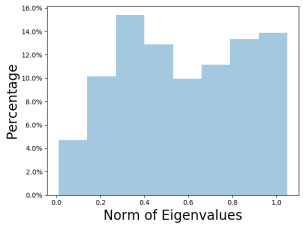


(e) Radius: 20

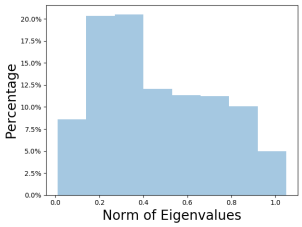
Figure E.5: Specturm Change for Erf



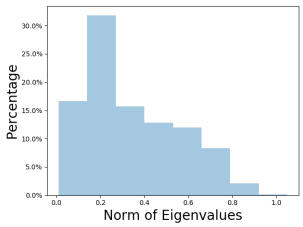
(a) Radius: 1



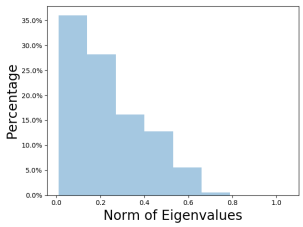
(b) Radius: 5



(c) Radius: 10



(d) Radius: 15



(e) Radius: 20

Figure E.6: Specturm Change for Sigmoid