博士論文

論文題目　Theory of Macroscopic Dynamics of Learning and Inference in Nonlinear Neural Networks
（非線形ニューラルネットワークにおける学習および推論の巨視的ダイナミクスの理論）

氏　名　吉田　雄紀

Theory of Macroscopic Dynamics of Learning and Inference in Nonlinear Neural Networks

Yuki Yoshida

Supervisor
Masato Okada

The University of Tokyo, 2020
Graduate School of Frontier Sciences
Department of Complexity Science and Engineering

Abstract

With the development of deep learning techniques since the late 2000s, neural networks have become an extremely useful machine learning method in a wide range of applications. Deep learning techniques have been applied to a quite wide range of fields,including image processing (recognition, generation, style transfer etc.), audio processing, natural language processing such as translation and question answering, games such as Go and Poker, robotics, and other prediction tasks that arise in various fields (e.g. protein structure prediction). In many of them, higher performance than conventional methods has been achieved by deep learning techniques.

However, there are many theoretical aspects of neural networks that remain unclear: performance guarantees, interpretability of inferences, optimal structure and hyperparameters, expressivity, generalization performance behavior, etc. In this dissertation, we focus on the fact that it is not clear under what conditions the learning of neural networks succeeds. There are two main reasons for this. For one thing, learning neural networks was considered more difficult in the 1990s than it is now. In particular, the "plateau phenomenon," in which the loss stops decreasing for a long time in the middle of learning,was a problem at the time. The plateau phenomenon was the subject of a great deal of theoretical research at that time, and it was pointed out that it was due to the symmetry inherent in the structure of neural nets. Although this plateau phenomenon is theoretically thought to occur inevitably in neural network learning, it has rarely been seen in recent real-world applications of neural networks. Another is that, even now, we still need to setup hyperparameters by trial and error in order to be most successful in learning neural networks. With the development of technologies for deep learning, a variety of methods have proposed: wide range of network structures, types of layer and activation, parameter initialization, optimization algorithm and its parameters, data preprocessing, types of loss function, etc. This means that we have to choose methods among them which are appropriate for target tasks. Although it has become more important than ever to choose these hyperparameters appropriately for successful learning, most of them currently rely on trial-and-error-based methods such as random search and grid search. To sum up these points, the mechanisms behind the success of deep learning have not been fully explored.

Motivated by the above situation, we analyze the dynamics of learning and inference in neural networks macroscopically using statistical mechanical methods and mean field methods, and examine the conditions for success or failure of neural network learning from various perspectives.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Machine learning, neural network and deep learning

Machine learning is a technology which enables us to let computers learn to do some tasks by data. A lot of methods for machine learning have been developed, including linear regression, logistic regression, support vector machines, decision tree, random forest, and neural networks. Especially, neural networks have been quite highlighted since Hinton et al. took the first step towards deep learning in 2006[5]. After their works, various techniques for training deep neural networks as well as diverse model architectures have been developed. In 2010s, unprecedented high performance have been achieved by deep neural networks in wide range of practical tasks (especially in the field of computer vision). Moreover, it has been shown that deep neural networks perform quite well in tasks for generating data and reinforcement learning tasks. Even in recent years, deep neural networks have yielded many breakthroughs in a number of domains, including natural language processing, sound processing, physics, chemistry, etc.

1.2 Challenges in deep learning

However, there are many theoretical aspects of neural networks that remain unclear: performance guarantees, interpretability of inferences, optimal structure and hyperparameters, expressivity, generalization performance behavior, etc. Sometimes deep learning related techniques are referred as ‘alchemy’ or ‘black magic’, because they somehow work well but we cannot understand how they do so.

Among these black-box aspects of neural networks and deep learning, in this dissertation, we focus on the problems of trainability; it is not clear under what conditions the learning of neural networks succeeds or fails. There are two main reasons for this. For one thing, learning neural networks was considered more difficult in the 1990s than it is now. In particular, the "plateau phenomenon," in which the loss stops decreasing for a long time in the middle of learning, was a problem at the time (see Figure 1.2 for examples of plateau phenomena). The plateau phenomenon was the subject of a great deal of theoretical research at that time, and it was pointed out that it was due to the symmetry inherent

in the structure of neural networks[6]. Although this plateau phenomenon is theoretically thought to occur inevitably in neural network learning, it has rarely been seen in recent real-world applications of neural networks. Another is that, even now, we still need to setup hyperparameters by trial and error in order to be most successful in learning neural networks. With the development of technologies for deep learning, a variety of methods have proposed: wide range of network structures, types of layer and activation, parameter initialization, optimization algorithm and its parameters, data preprocessing, types of loss function, etc. This means that we have to choose methods among them which are appropriate for target tasks. Although it has become more important than ever to choose these hyperparameters appropriately for successful learning, most of them currently rely on trial-and-error-based methods such as random search, grid search and evolutionary methods.

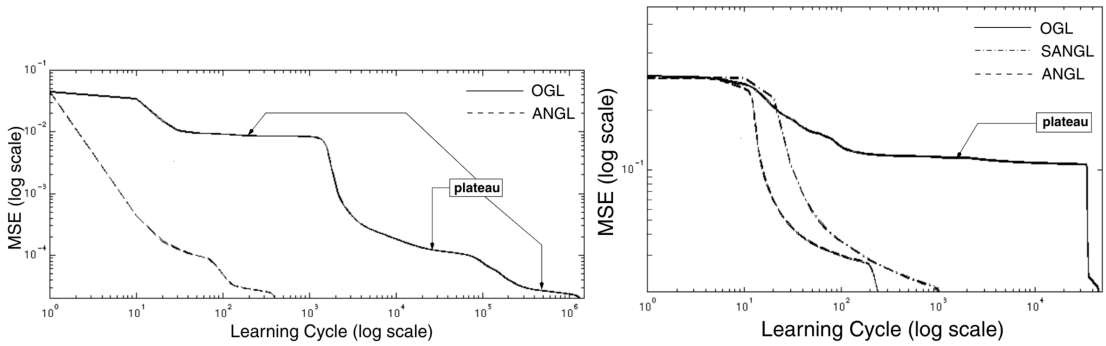

Figure 1.1: Examples of plateau phenomena, observed while learning Mackey-Glass time series regression task and extend XOR classification task. Both figures are reprinted from [1].

1.3 Importance of theoretical understanding of neural networks

In order to overcome this situation, it is important to deepen the theoretical understanding of deep learning. In other words, it would be good for us to be able to understand in what cases learning succeeds or fails without actually performing the learning. With such a theoretical understanding of learning of neural networks, we would be able to present a theory-based guideline for the design of learning, that is the selection of hyperparameters, including network structures, optimization algorithms and data preprocessing. With this motivation, we have been trying to understand the learning of neural networks theoretically.

1.4 Macroscopic methods

A neural network typically has a very large number of parameters. The input signals are propagated through a lot of layers which consist of linear transformations and nonlinear activations. Although they make the network extremely expressive in a efficient

way, they also make analyses of their dynamics of propagating quite tough. In addition, desirable input-output relationship can be obtained by changing their parameters according to the gradient descent method. This learning dynamics is also quite high-dimensional and nonlinear.

For analyzing them, we will use two macroscopic methods: the statistical mechanical method and the mean-field method. Here we briefly introduce them.

1.4.1 Macroscopic method for learning dynamics

While we want to analyze the dynamics of these parameters in training theoretically, it is generally difficult to deal directly with very high-dimensional nonlinear dynamical systems. One way to cope with it is to consider linear neural networks instead of nonlinear neural networks. The learning dynamics of linear neural networks are easy to handle theoretically, as analytical solutions can be obtained under certain assumptions[7]. One drawback, however, is that the rich learning behavior unique to nonlinear neural networks (including the plateau phenomenon described above) is not observed in linear neural networks and cannot be analyzed. Another method is the statistical mechanical method used in this paper. This method was proposed in 1995 by Biehl et al.[8] and Saad et al.[9] to obtain the dynamics of a small number of macroscopic variables by appropriately reducing the dynamics of a large number of microscopic variables. Using this method, the learning dynamics of nonlinear neural networks can be reduced to a dynamics of a few macroscopic variables, or order parameters, under the assumption that the input dimensionality is very large. This method will be introduced in Chapter 2, and used in Chapter 2–4.

1.4.2 Macroscopic method for propagating dynamics

The input-output relationship of deep neural networks themselves are also an important subject of research. Deep neural networks have exponential expressivity with respect to the number of layers[10], which means that a network can represent almost any function depending on the values of the parameters. On the other hand, it is limited to a region in the entire parameter space where the neural network can express the functions required to solve a realistic task. This is important because we cannot explore the entire parameter space in one training with gradient methods; only a narrow region can be reached with the gradient method from the initial value. This means that choosing an initial value located in the region described above is important for successful learning.

More specifically, for a network to express realistic functions, its input-output relationship should not have sensitivity against small noise, and it should not lose information contained in input signals. Thus it is essential to inspect how input signals are propagated through deep neural networks, and how it is affected by network parameters. The mean-field method has been developed for this; it deals with signal propagation through random neural networks with infinite width[11, 2]. We will introduce and use it in Chapter 5.

1.4.3 Advantage of macroscopic methods over other theoretical methods

There are many theoretical studies of deep learning that take approaches other than these macroscopic methods. Importantly, the macroscopic approaches we will use have a common feature; they attempt to examine the average behavior. This is in contrast to, for example, approaches that utilize inequalities to derive bounds and give guarantees of convergence or generalization. Trying to take even the worst case into account sometimes produces only very loose (far from what is actually happening) evaluations. On the other hand, these macroscopic approaches to average behavior seem rather reasonable as models of what is happening in reality, although they usually do not give strict guarantees.

1.5 Outline of this dissertation

This dissertation consists of Chapter 1, which is an introduction, Chapters 2-5, which is the main thesis, and Chapter 6, which is the conclusion. Here we describe the outline of each chapter. See also Figure 1.5 which depicts the overview.

In Chapter 1, we already discussed why the theoretical study of neural networks is important, and briefly described macroscopic analysis methods we will use in the following chapters, the statistical mechanical method and the mean-field method.

In Chapter 2, we analyze the learning dynamics of a nonlinear neural network with Weight Normalization, using a statistical mechanical formulation[12]. The regularization method called Weight Normalization was proposed in 2016[13] and was empirically known to speed up the learning. However, the mechanism of speeding up was unknown. Then we analyze and derive the learning dynamics of a single-layer nonlinear neural network using statistical mechanical formulation. By solving the derived dynamics numerically, we show that Weight Normalization exhibits faster learning regardless of the learning rate, and discuss with what mechanism it is achieved.

In Chapter 3, we focus on the "plateau phenomenon" that prevents successful learning in nonlinear neural networks, and show that this phenomenon can be alleviated depending on the dimensionality of the output of neural networks[14]. The "plateau phenomenon", in which the loss stagnates without decreasing for a long period of time during the training of neural networks, has been known and studied vigorously since the late 1990s. It has been pointed out that the symmetry inherent in the neural network model is the root cause of the plateau phenomenon, which brings a singularity in the metric of the parameter space, which in turn yields a Milnor-like attractor[15, 6]. However, it is empirically quite rare to see a plateau phenomenon in recent deep learning applications, and there is a discrepancy between theory and practice. Although this discrepancy must be caused because the assumptions made in the theoretical analysis are somehow not consistent with the realistic situation, it is unclear what are critical and we should clarify them. In the chapter, we focus on the number of units in the output layer. Existing studies that have analyzed the learning dynamics of nonlinear neural networks using statistical mechanical methods have all assumed a single output unit, and have not considered the case with multiple output units, which is more realistic. Then we analyze and derive the learning

dynamics of a two-layer nonlinear neural network with multiple output units, using the statistical mechanical method, and point out that the number of output units makes an important difference in the plateau phenomenon. Specifically, we show that the plateau phenomenon, which causes a failure of learning, is reduced when there are multiple output units compared to the case with a single output unit.

In Chapter 4, we analyze how the learning dynamics of nonlinear neural networks depend on the statistics of the learning data[16]. Existing studies that have analyzed the learning dynamics of nonlinear neural networks using statistical mechanical formulation, including those described in Chapters 2 and 3, have not considered the statistics of the learning data, especially that of the input signals. Specifically, analyses in existing studies have assumed that the input signal is generated by i.i.d. standard normal distribution. However, the learning behavior can change depending on the statistics of the learning data, and it is not realistic to assume that the input signal is generated by unit Gaussian; according to the manifold hypothesis, the real-world data is expected to concentrate in the vicinity of a low-dimensional manifold embedded in a high-dimensional space[17]. Therefore, in this chapter, we extend the statistical mechanical framework to the more general cases in terms of statistics of the input data, in order to investigate the effect of the statistics of the learning data on the learning dynamics. We then reveal how the learning dynamics and plateau phenomena depend on the statistics of the data. Specifically, we show that the plateau phenomenon becomes less prominent when the covariance matrix of the distribution of input signals has smaller or more dispersed eigenvalues.

In Chapter 5, in contrast to Chapters 2-4 for analyzing learning dynamics with statistical mechanical methods, we work on the dynamics of signal propagation through deep neural networks. For successful learning of deep neural networks, signals and correlations between them need to propagate over many layers without decay or divergence. In recent years, there have been attempts to analyze the dynamics of such signal propagation using the mean-field method. In the mean-field method, we consider a neural network with random weights and infinite width, and approximate the activities of its layers with multivariate Gaussian distributions. These allow us to obtain deterministic equations of evolution of only the macroscopic statistics of the signals, and by further discussing the speed of convergence of their dynamics, we can compute so-called "depth scale" for successful signal propagation[2]. It has been reported that the macroscopic dynamics and depth scales of propagation are analytically determined by mean-field methods for some simple networks and that they are in good agreement with the actual trainability[2, 3, 4]. However, practical neural networks typically have a more complex structure and their depth-scales cannot always be determined analytically. In this chapter, we establish a method for numerically evaluating depth scales for arbitrarily complex networks and show that it is actually possible to predict trainability from depth scales for neural networks with various architectures.

Finally, in Chapter 6, we discuss the potential impact of the series of studies described in this paper on industry and research, as well as the challenges and directions of future research.

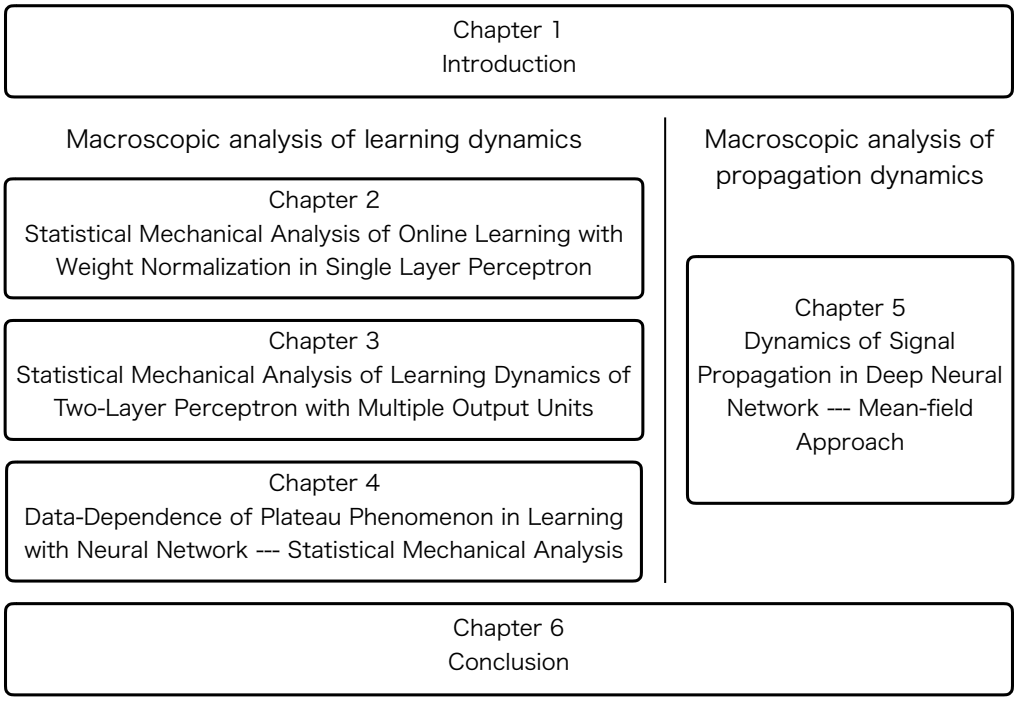

Figure 1.2: Overview of this dissertation.

Chapter 2

Statistical Mechanical Analysis of Online Learning with Weight Normalization in Single Layer Perceptron*

2.1 Introduction

In recent years, large neural networks are commonly used for various tasks in the name of Deep Learning.[18] The success of deep learning is highly indebted to improvements of algorithms for speeding up learning: they include various modifications to gradient descent,[19, 20, 21, 22, 23] and several normalization methods.[24, 13, 25] In particular, the weight normalization (WN) proposed by Salimans & Kingma (2016),[13] which reparametrizes the weight vector as explained below, is spotlighted in terms of easiness of implementation and introduction to various conventional network structures. In most neural networks, each neuron's output can be represented as $y = g(\boldsymbol{W} \cdot \boldsymbol{x} + b)$, where $\boldsymbol{x}$ is an input vector and the activation function g is nonlinear in general. The standard steepest descent method updates its weight vector $\boldsymbol{W}$ as $\Delta \boldsymbol{W} = -\eta \dfrac{\partial \varepsilon}{\partial \boldsymbol{W}}$, where ε is a loss function and $\eta > 0$ is a learning rate. In contrast, in WN, $\boldsymbol{W}$ is decomposed as $\boldsymbol{W} = r\dfrac{\boldsymbol{V}}{|\boldsymbol{V}|}$, and then steepest descent optimization proceeds in accordance with the gradient of r and $\boldsymbol{V}$ instead of $\boldsymbol{W}$, that is, $\Delta r = -\eta \dfrac{\partial \varepsilon}{\partial r}$ and $\Delta \boldsymbol{V} = -\eta \dfrac{\partial \varepsilon}{\partial \boldsymbol{V}}$. This newly proposed optimization method is known to speed up the convergence of the loss function in conventional network structures and machine learning tasks such as image recognition and reinforcement learning. However, it remains unclear why this method works well.

Biehl & Schwarze (1995)[8] and Saad & Solla (1995)[9] established useful techniques on the basis of statistical mechanics, with which we can derive the dynamical equations of order parameters that represent the macroscopic state of the weight vector. Using the

*This work has been published as [12].

techniques, they found analytically and discussed the dynamics of on-line learning in a single layer perceptron and a (two-layered) soft committee machine that learns the input-output relationship of a "teacher network" that has the same structure as a learning one, although their analysis was limited to the conventional steepest descent methods with ordinary parameterization of weight vectors.

In this study, we apply their technique to WN and analyze quantitatively the dynamical evolution of on-line learning in single layer perceptrons. We perform linear stability analysis on a global minimum of the loss function and determine the converging speed of order parameters towards the global minimum. In WN, it shows that an effective learning rate appears that is automatically tuned, and that there exists an optimal initial value of an order parameter for fast converging.

2.2 Model

2.2.1 Student-teacher network formulation

In this study, we focus only on single layer perceptron. That is, we consider a neural network that receives input data $\boldsymbol{x} \in \mathbb{R}^N$, calculates output $s = g(\boldsymbol{J} \cdot \boldsymbol{x})$ (we assume the activation function $g : \mathbb{R} \to \mathbb{R}$ is non-constant and weakly monotonous) , and learns $\boldsymbol{J}$ by teacher data. We treat an ideal situation, in which the teacher data t is determined as $t = g(\boldsymbol{B} \cdot \boldsymbol{x})$; in other words, the learning network (the "student network") learns the input-output relationship of the "teacher network", which has the same structure as the student one and the original fixed weight $\boldsymbol{B}$ (Figure 2.1(a)). We use the squared loss function $\varepsilon = \dfrac{1}{2}(t - s)^2$. (The choice of a loss function is not critical; see Appendix B for general case.)

2.2.2 Statistical mechanical formulation

Stochastic gradient descent

For the statistical mechanical formulation of on-line learning, we introduce further idealization. We assume that the dimension of input data N is very large, and each element of input data $\boldsymbol{x}$ is generated in accordance with i.i.d. normal distribution, $\mathcal{N}(x_i|0, 1/N)$. (Note that $|\boldsymbol{x}| \approx 1$.)* We suppose $|\boldsymbol{B}| = \sqrt{N}$ and define $l(\alpha)$ and $R(\alpha)$ by $|\boldsymbol{J}| = \sqrt{N}\, l(\alpha)$ and $\boldsymbol{B} \cdot \boldsymbol{J} = N\, l(\alpha) R(\alpha)$, where α represents time.[8, 9] $l(\alpha)$ and $R(\alpha)$ are the order parameters; the former one is the measure for the length (norm) of the weight vector of the student network, and the latter one is the direction cosine between the weight vectors of student and teacher (Figure 2.1(b)). The initial values of l and R (l_0 and R_0, respectively) depend on how $\boldsymbol{J}$ is initialized; the value of R_0 converges towards 0 with $N \to \infty$, as long as we choose $\boldsymbol{J}$ from a spherically symmetric distribution.

In the next section, we derive the dynamical equations that govern the order parameters l and R, which capture the macroscopic state of the system, from the dynamics

*This assumption can be relaxed. See the appendix at the end of this dissertation.

of microscopic variables (the weight vector) of the system.

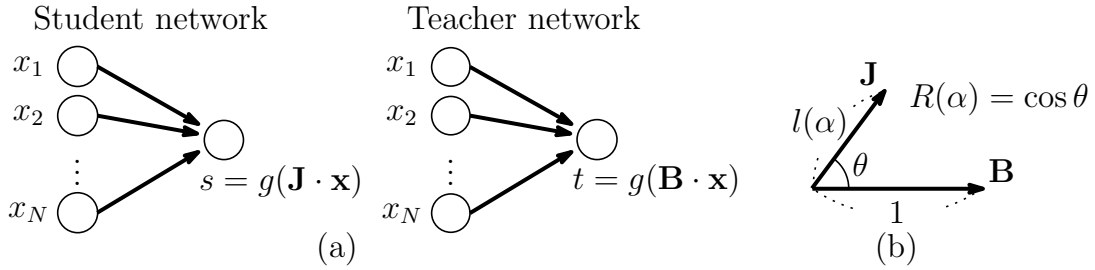

Figure 2.1: (a) Student network and teacher network. (b) Geometrical interpretation of order parameters $l(\alpha)$ and $R(\alpha)$.

Weight normalization

In WN, weight vector $\boldsymbol{J}$ is decomposed into radial length r and direction vector $\boldsymbol{V}$ as $\boldsymbol{J} = r\dfrac{\boldsymbol{V}}{|\boldsymbol{V}|}$, and the gradients of r and $\boldsymbol{V}$ are then used to perform gradient descent optimization. Since we put $|\boldsymbol{J}| = \sqrt{N}\, l(\alpha)$ in the previous section, r equals to $\sqrt{N}\, l(\alpha)$, then we call $l(\alpha)$ "radial length", as well as r. We define $z(\alpha)$ to represent the norm of $\boldsymbol{V}$, as $|\boldsymbol{V}| = \sqrt{N}\, z(\alpha)$. This additional order parameter $z(\alpha)$ shows up because of the redundancy in radial parameterization $\boldsymbol{J} = r\dfrac{\boldsymbol{V}}{|\boldsymbol{V}|}$, where the norm of $\boldsymbol{V}$ is not normalized. In WN, three order parameters $l(\alpha)$, $R(\alpha)$, and $z(\alpha)$ appear in the following macroscopic dynamical equations.

2.3 Theory

In this section, we derive the dynamical equations of the parameters described above.

2.3.1 Dynamical equations of order parameters in SGD

We follow the argument by Biehl & Schwarze (1995)[8] and Saad & Solla (1995)[9] in this subsection. The update rule of on-line learning based on vanilla SGD is written as

$$\Delta \boldsymbol{J} = -\eta \frac{d\varepsilon}{d\boldsymbol{J}} = \eta g'(\boldsymbol{J} \cdot \boldsymbol{x})(t - s)\boldsymbol{x}, \tag{2.1}$$

which gives the update rule of order parameters l and R:

$$\begin{aligned} N\Delta l^2 &= |\boldsymbol{J}(\alpha+1)|^2 - |\boldsymbol{J}(\alpha)|^2 = |\boldsymbol{J}(\alpha) + \eta g'(\boldsymbol{J}(\alpha)\cdot\boldsymbol{x})(t-s)\boldsymbol{x}|^2 - |\boldsymbol{J}(\alpha)|^2 \\ &= 2\eta g'(\boldsymbol{J}(\alpha)\cdot\boldsymbol{x})(t-s)\boldsymbol{J}(\alpha)\cdot\boldsymbol{x} + \eta^2 g'(\boldsymbol{J}(\alpha)\cdot\boldsymbol{x})^2(t-s)^2|\boldsymbol{x}|^2 \\ &= 2\eta g'(lu)(t-s)lu + \eta^2 g'(lu)^2(t-s)^2|\boldsymbol{x}|^2, \\ N\Delta(lR) &= \boldsymbol{B}\cdot\boldsymbol{J}(\alpha+1) - \boldsymbol{B}\cdot\boldsymbol{J}(\alpha) = \boldsymbol{B}\cdot\Delta\boldsymbol{J} \\ &= \eta g'(\boldsymbol{J}(\alpha)\cdot\boldsymbol{x})(t-s)\boldsymbol{B}\cdot\boldsymbol{x} \\ &= \eta g'(lu)(t-s)v, \end{aligned} \tag{2.2}$$

where we define u and v as $\boldsymbol{J}\cdot\boldsymbol{x} = lu$ and $\boldsymbol{B}\cdot\boldsymbol{x} = v$. Since the right hand sides of these equations are $O(N^0)$, the difference terms Δl^2 and $\Delta(lR)$ are $O(N^{-1})$, and therefore we can replace these difference equations with differential ones with $N \to \infty$:

$$\begin{aligned} N\frac{d}{d\alpha}l^2 &= 2\eta A_1 + 2\eta^2 A_2, \\ N\frac{d}{d\alpha}lR &= \eta A_3 \end{aligned} \tag{2.3}$$

$$\begin{aligned} \text{where}\quad A_1(l,R) &= \langle g'(lu)(g(v)-g(lu))lu\rangle, \\ A_2(l,R) &= \frac{1}{2}\langle g'(lu)^2(g(v)-g(lu))^2\rangle, \\ A_3(l,R) &= \langle g'(lu)(g(v)-g(lu))v\rangle. \end{aligned} \tag{2.4}$$

Here the brackets $\langle\cdot\rangle$ represent the expectation when $\boldsymbol{x}$ follows $\mathcal{N}(x_i|0,1/N)$, that is, when (u,v) follows $\mathcal{N}(\boldsymbol{0}, \begin{pmatrix} 1 & R \\ R & 1 \end{pmatrix})$. These differential equations are what we wanted. Note that the generalization error ε_g is represented as

$$\varepsilon_g = \frac{1}{2}\langle (t-s)^2\rangle = \frac{1}{2}\langle (g(v)-g(lu))^2\rangle. \tag{2.5}$$

All expectation terms appearing in (4.4), (3.5) have forms $I_3(y_1,y_2,y_3) := \langle g'(y_1)y_2 g(y_3)\rangle$ or $I_4(y_1,y_2,y_3) := \langle g'(y_1)^2 g(y_2)g(y_3)\rangle$, where y_1, y_2, y_3 is either lu, v, or 0. The I_3 and I_4 can be analytically determined for some activation function g; when (y_1,y_2,y_3) follows a multivariate normal distribution $\mathcal{N}((y_1,y_2,y_3)|0,C)$ where C is covariance matrix, the identity function $g(x) = x$ gives

$$\begin{aligned} I_3 &= \langle y_2 y_3\rangle = C_{23}, \\ I_4 &= \langle y_2 y_3\rangle = C_{23}. \end{aligned} \tag{2.6}$$

The error function $g(x) = \mathrm{erf}(x/\sqrt{2})$ implies

$$\begin{aligned} I_3 &= \frac{2}{\pi}\cdot\frac{1}{\sqrt{(1+C_{11})(1+C_{33})-C_{13}^2}}\frac{C_{23}(1+C_{11})-C_{12}C_{13}}{1+C_{11}}, \\ I_4 &= \frac{4}{\pi^2}\cdot\frac{1}{\sqrt{1+2C_{11}}}\arcsin\frac{(1+2C_{11})C_{23}-2C_{12}C_{13}}{\sqrt{(1+2C_{11})(1+C_{22})-2C_{12}^2}\sqrt{(1+2C_{11})(1+C_{33})-2C_{13}^2}} \end{aligned} \tag{2.7}$$

as shown in Saad & Solla (1995).[9] In the case of $g(x) = \mathrm{ReLU}(x)$ ($:= \max\{0, x\}$), which is commonly used as the activation function in recent neural networks, we found the following formula (see Appendix A for derivation):

$$\begin{aligned} I_3 &= C_{23}\left(\frac{1}{4} + \frac{1}{2\pi}\arcsin\frac{C_{13}}{\sqrt{C_{11}C_{33}}}\right) + \frac{1}{2\pi}\cdot\frac{C_{12}}{C_{11}}\sqrt{C_{11}C_{33} - C_{13}^2}, \\ I_4 &= C_{23}\left[\frac{1}{8} + \frac{1}{4\pi}\left(\arcsin\frac{C_{12}}{\sqrt{C_{11}C_{22}}} + \arcsin\frac{C_{13}}{\sqrt{C_{11}C_{33}}} + \arcsin\frac{C_{23}}{\sqrt{C_{22}C_{33}}}\right)\right] \\ &\quad + \frac{1}{4\pi}\left(\frac{C_{13}}{C_{11}}\sqrt{C_{11}C_{22} - C_{12}^2} + \frac{C_{12}}{C_{11}}\sqrt{C_{11}C_{33} - C_{13}^2} + \sqrt{C_{22}C_{33} - C_{23}^2}\right). \end{aligned} \tag{2.8}$$

2.3.2 Dynamical equations of order parameters in WN

In the case of WN, the update rule of direction vector $\boldsymbol{V}$ and radial parameter r is

$$\begin{aligned} \Delta r &= -\eta\frac{d\varepsilon}{dr} = \eta g'(\boldsymbol{J}\cdot\boldsymbol{x})(t-s)\frac{\boldsymbol{J}\cdot\boldsymbol{x}}{r}, \\ \Delta\boldsymbol{V} &= -\eta\frac{d\varepsilon}{d\boldsymbol{V}} = \eta g'(\boldsymbol{J}\cdot\boldsymbol{x})(t-s)\left(\frac{r}{|\boldsymbol{V}|}\boldsymbol{x} - \frac{\boldsymbol{J}\cdot\boldsymbol{x}}{|\boldsymbol{V}|^2}\boldsymbol{V}\right), \end{aligned} \tag{2.9}$$

which provides the update rule of order parameters l, R, and z:

$$\begin{aligned} N\Delta l &= \sqrt{N}\Delta r = \eta g'(\boldsymbol{J}\cdot\boldsymbol{x})(t-s)\frac{\boldsymbol{J}\cdot\boldsymbol{x}}{l} \\ &= \eta g'(lu)(t-s)u, \\ N\Delta(Rz) &= \Delta(\boldsymbol{B}\cdot\boldsymbol{V}) = \eta g'(\boldsymbol{J}\cdot\boldsymbol{x})(t-s)\left(\frac{r}{|\boldsymbol{V}|}\boldsymbol{x} - \frac{\boldsymbol{J}\cdot\boldsymbol{x}}{|\boldsymbol{V}|^2}\boldsymbol{V}\right)\cdot\boldsymbol{B} \\ &= \eta g'(lu)(t-s)\left(\frac{lv}{z} - \frac{lRu}{z}\right), \\ N\Delta(z^2) &= |\boldsymbol{V}(\alpha+1)|^2 - |\boldsymbol{V}(\alpha)|^2 = 2\boldsymbol{V}(\alpha)\cdot\Delta\boldsymbol{V} + |\Delta\boldsymbol{V}|^2 = |\Delta\boldsymbol{V}|^2 \\ &= \eta^2 g'(\boldsymbol{J}\cdot\boldsymbol{x})^2(t-s)^2\left[\frac{r^2}{|\boldsymbol{V}|^2}|\boldsymbol{x}|^2 - 2\frac{r\boldsymbol{J}\cdot\boldsymbol{x}}{|\boldsymbol{V}|^3}\boldsymbol{V}\cdot\boldsymbol{x} + \frac{(\boldsymbol{J}\cdot\boldsymbol{x})^2}{|\boldsymbol{V}|^2}\right] \\ &= \eta^2 g'(lu)^2(t-s)^2\left(\frac{l^2}{z^2}|\boldsymbol{x}|^2 - \frac{l^2u^2}{Nz^2}\right) \end{aligned} \tag{2.10}$$

(we set $\boldsymbol{J}\cdot\boldsymbol{x} = lu$ and $\boldsymbol{B}\cdot\boldsymbol{x} = v$ again). These difference terms are $O(N^{-1})$; thus, we can replace these equations with differential equations with $N \to \infty$:

$$\begin{aligned} N\frac{d}{d\alpha}l^2 &= 2\eta A_1, \\ N\frac{d}{d\alpha}Rz &= \eta(\frac{l}{z}A_3 - \frac{R}{z}A_1), \\ N\frac{d}{d\alpha}z^2 &= \eta^2\frac{2l^2}{z^2}A_2. \end{aligned} \tag{2.11}$$

The definition of terms A_i are the same as (4.4). Again, all expectation terms have forms like I_3 or I_4, which can be determined for some g. Note that z is monotonously increasing since $A_2 \geq 0$.

2.3.3 Linear stability analysis

In our system of student and teacher single layer perceptrons, the generalization error ε_g equals to 0 if and only if $\boldsymbol{J} = \boldsymbol{B}$; in other words, $\boldsymbol{J} = \boldsymbol{B}$ is the unique global minimum. Values of order parameters at this global minimum are $(l, R) = (1, 1)$. For the system (4.9) for SGD, $(l, R) = (1, 1)$ is a steady state, and for the system (2.11) for WN, $(l, R, z) = (1, 1, z)$ are steady states for all z. We perform stability analysis at these steady points corresponding to the global minimum to evaluate quantitatively the speed of converging towards global optimality.

For the system (4.9) for SGD, the stability matrix at $(l, R) = (1, 1)$, and its eigenvalues and eigenvectors are given as

$$P_{\mathrm{SGD}} = \begin{pmatrix} \eta\dfrac{\partial A_1}{\partial l} & \eta\dfrac{\partial A_1}{\partial R} + \eta^2\dfrac{\partial A_2}{\partial R} \\ 0 & \eta\Big(\dfrac{\partial A_3}{\partial R} - \dfrac{\partial A_1}{\partial R}\Big) - \eta^2\dfrac{\partial A_2}{\partial R} \end{pmatrix},$$

$$\begin{aligned} &\lambda_1 = \eta\frac{\partial A_1}{\partial l}, \quad \boldsymbol{e}_1 = \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \\ &\lambda_2 = \eta\Big(\frac{\partial A_3}{\partial R} - \frac{\partial A_1}{\partial R}\Big) - \eta^2\frac{\partial A_2}{\partial R} \qquad = \frac{\partial A_2}{\partial R}\eta(\eta_c - \eta), \\ &\boldsymbol{e}_2 = \begin{pmatrix} \dfrac{\partial A_1}{\partial R} + \eta\dfrac{\partial A_2}{\partial R} \\ -\dfrac{\partial A_1}{\partial l} + \dfrac{\lambda_2}{\eta} \end{pmatrix} \\ &\text{where} \quad \eta_c := \frac{\partial(A_3 - A_1)}{\partial R} \Big/ \frac{\partial A_2}{\partial R} \end{aligned}$$

where the derivatives of A_i are evaluated at $(l, R) = (1, 1)$. For the system (2.11) for WN, the stability matrix at $(l, R, z) = (1, 1, z_\infty)$, and its eigenvalues and eigenvectors are calculated as

$$P_{\mathrm{WN}} = \begin{pmatrix} \eta\dfrac{\partial A_1}{\partial l} & \eta\dfrac{\partial A_1}{\partial R} & 0 \\ 0 & \dfrac{\eta}{z_\infty^2}\Big(\dfrac{\partial A_3}{\partial R} - \dfrac{\partial A_1}{\partial R}\Big) - \dfrac{\eta^2}{z_\infty^4}\dfrac{\partial A_2}{\partial R} & 0 \\ 0 & \dfrac{\eta^2}{z_\infty^3}\dfrac{\partial A_2}{\partial R} & 0 \end{pmatrix},$$

$$\lambda_1 = \eta\frac{\partial A_1}{\partial l}, \quad \boldsymbol{e}_1 = \begin{pmatrix} 1 \\ 0 \\ 0 \end{pmatrix},$$

$$\lambda_2 = \frac{\eta}{z_\infty^2}\left(\frac{\partial A_3}{\partial R} - \frac{\partial A_1}{\partial R}\right) - \frac{\eta^2}{z_\infty^4}\frac{\partial A_2}{\partial R} \qquad = \frac{\partial A_2}{\partial R}\frac{\eta}{z_\infty^2}\left(\eta_c - \frac{\eta}{z_\infty^2}\right),$$

$$\boldsymbol{e}_2 = \begin{pmatrix} \dfrac{\partial A_1}{\partial R} \\ -\dfrac{\partial A_1}{\partial l} + \dfrac{\lambda_2}{\eta} \\ \dfrac{\eta^2}{z_\infty^3}\dfrac{\partial A_2}{\partial R}\left(-\dfrac{1}{\lambda_2}\dfrac{\partial A_1}{\partial l} + \dfrac{1}{\eta}\right) \end{pmatrix},$$

$$\lambda_3 = 0, \quad \boldsymbol{e}_3 = \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

where the derivatives of A_i are evaluated at $(l, R, z) = (1, 1, z_\infty)$. Since $A_2 = \frac{1}{2}\langle[g'(lu)(g(v) - g(lu))]^2\rangle \geq 0$ and $A_3 - A_1 = \langle g'(lu)(g(v) - g(lu))(v - lu)\rangle \geq 0$ (note that g is monotonous) equal to 0 at global minimum $(1, 1)$, it holds that $\frac{\partial A_2}{\partial R}, \frac{\partial (A_3 - A_1)}{\partial R} \leq 0$. Thus, for SGD system, the necessary and sufficient condition for the stability of the global minimum is $\eta < \eta_c = \frac{\partial (A_3 - A_1)}{\partial R} / \frac{\partial A_2}{\partial R}$. The value of η_c depends on activation function g; for example, $\eta_c = 2$ for $g(x) = x$ and $g(x) = \mathrm{ReLU}(x)$, and $\eta_c = \sqrt{5/3}\pi$ for $g(x) = \mathrm{erf}(x/\sqrt{2})$. In contrast, the stability condition of the global minimum for WN system is $\eta/z_\infty^2 < \eta_c$.

We can evaluate the speeds of order parameters l and R converging towards the global optimum when it is linearly stable, using eigenvalues and eigenvectors of the stability matrix calculated above. For SGD and WN cases, the R-component of the first eigenvector $\boldsymbol{e}_1$ equals to 0, therefore R converges towards the global optimum 1 at the speed of $-\lambda_1$. (We call a variable behaving like $O(e^{\lambda\alpha})$ as "converging at the speed of λ.") In contrast, l converges towards 1 at the speed of $\min\{-\lambda_1, -\lambda_2\}$. The converging speeds of l and R while performing SGD and WN are summarized in Table 1.

Table 2.1: Converging speeds towards global minimum of order parameters l and R. All derivatives of A_i are evaluated at global minimum.

	SGD	WN
Direction cosine R	$-\dfrac{\partial A_2}{\partial R}\eta(\eta_c - \eta)$	$-\dfrac{\partial A_2}{\partial R}\dfrac{\eta}{z_\infty^2}\left(\eta_c - \dfrac{\eta}{z_\infty^2}\right)$
Radial length l	$\min\left\{-\dfrac{\partial A_1}{\partial l}\eta,\ -\dfrac{\partial A_2}{\partial R}\eta(\eta_c - \eta)\right\}$	$\min\left\{-\dfrac{\partial A_1}{\partial l}\eta,\ -\dfrac{\partial A_2}{\partial R}\dfrac{\eta}{z_\infty^2}\left(\eta_c - \dfrac{\eta}{z_\infty^2}\right)\right\}$

Comparing WN and SGD, the converging speed of R, the direction cosine, is different; its effective learning rate is η in SGD and η/z_∞^2 in WN. Since the initial value of z is usually small (Salimans & Kingma (2016)[13] recommend 0.05), the value of η/z^2 is initially large and then decreases during learning because z is monotonously increasing. In SGD, since $-\lambda_2 \propto \eta(\eta_c - \eta)$, a learning rate larger than η_c prevents convergence, while too small a learning rate takes a long time to converge. The optimal learning rate which

maximizes $-\lambda_2$ is $\eta = \frac{\eta_c}{2}$, although the value of η_c is generally unknown. Contrary to this, while performing WN the effective learning rate η/z^2 automatically decreases to near the optimal learning rate $\frac{\eta_c}{2}$. This mechanism seems to realize η-independent fast converging.

Meanwhile, the converging speed of l, radial length, in WN cannot be larger than $-\frac{\partial A_1}{\partial l}\eta$ — that in SGD. Hence, if the initial value of the radial length is near optimal ($l = 1$), WN seems to greatly outperform SGD; otherwise, it is probable that WN yields the same converging speed as SGD does. If the initial value of l is too far from optimal, WN might be slower than SGD because η/z^2 may become too small and turn into a bottleneck in converging.

In the next section, we confirm the correspondence between numerical solutions of differential equations (4.9) and (2.11) about order parameters and simulation results about the original microscopic system, then examine the hypotheses described in this section.

2.4 Experimental results

2.4.1 Consistency between simulation and numerical solution

We performed numerical simulations of original microscopic systems ($N = 10000$) for SGD and WN. We set the weight vector of teacher $\boldsymbol{B}$, the initial weight vector of student $\boldsymbol{J}(0)$, and the initial direction vector $\boldsymbol{V}(0)$ by sampling their elements in accordance with $\mathcal{N}(B_i|0,1)$, $\mathcal{N}(J_i|0,l_0^2)$, and $\mathcal{N}(V_i|0,z_0^2)$, then normalizing them so that they suffice $|\boldsymbol{B}| = \sqrt{N}$, $|\boldsymbol{J}(0)| = \sqrt{N}\,l_0$, and $|\boldsymbol{V}(0)| = \sqrt{N}\,z_0$. We plotted the time course of order parameters l, R, and z and compared them with numerical solutions of differential equations (4.9) and (2.11) about order parameters (Figure 2.2). We confirmed their good agreement.

2.4.2 Dependence of converging speed on learning rate and initial radial length

Next, we computed numerical solutions of differential equations (4.9) and (2.11), investigated how the convergence speed of the generalization error towards 0 depends on η and l_0, and compared the results in the cases of SGD and WN. (Figure 2.3 for $g(x) = x$, and Figure 2.4 for $g(x) = \mathrm{erf}(x/\sqrt{2})$; in the case of $g(x) = \mathrm{ReLU}(x)$, we got almost the same results as Figure 2.3) (See Figure 2.5 for dependence on (η, l_0).)

Dependence on learning rate

When the learning rate η is too large ($\eta > \eta_c$), SGD does not allow the weight vector to converge towards the global minimum, while WN realizes the η-independent converging speed as the same as that in a case of $\eta \approx \eta_c$.

When the weight vector converges to the global optimum, the "effective learning rate" η/z^2 of direction cosine R decreases to near $\eta_c/2$ for a wide range of parameter

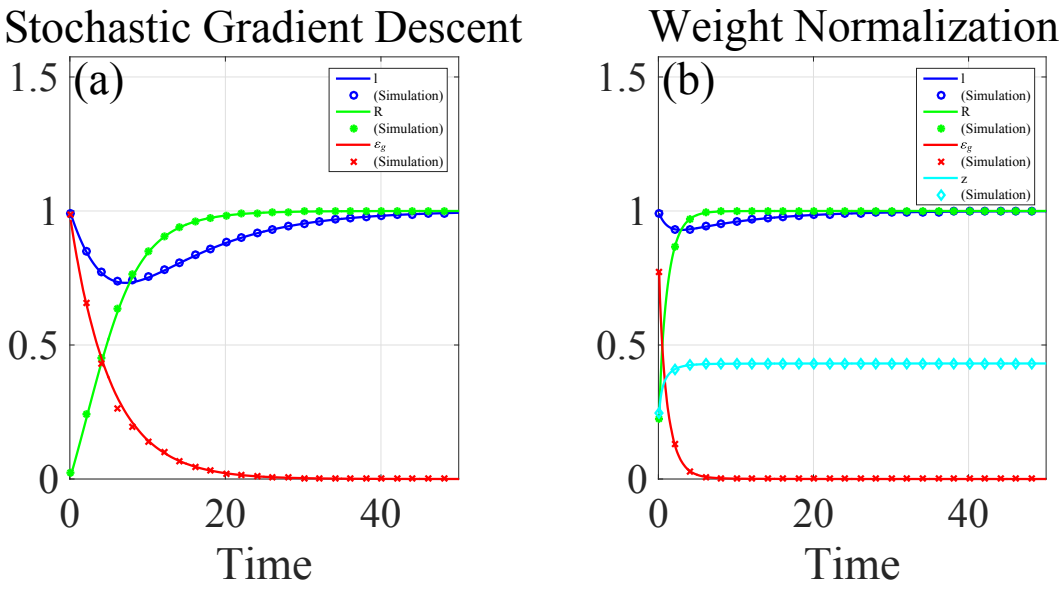

Figure 2.2: Time course of order parameters and generalization error in (a) SGD and (b) WN. blue: l, green: R, red: ε_g, cyan (in (b)): z. Solid lines represent numerical solutions of differential equations (4.9) and (2.11). Markers represent simulation results ($N = 10000$). Initial value of l is $l_0 = 1.0$. In (b), initial value of z is $z_0 = 0.05$. For all cases $\eta = 0.1$ and $g(x) = x$.

values (yellow regions of Figure 2.5(d)). This demonstrates that the "automatic tuning of the learning rate" indeed serves for the η-independent converging speed.

Dependence on initial radial length

Only when the initial value of the radial length is near optimal, a wide range of values of η ($10^{-2} < \eta < 10$ in Figure 2.3(a) and Figure 2.4(a)) yields the same converging speed. Learning with p times smaller η takes $1/p$ times longer in the case of SGD, while in the case of WN such elongation does not occur (within the range of η mentioned above).

In contrast, when the initial value of the radial length is significantly larger than the global minimum, there are cases in which WN takes more time than SGD to converge ($l_0 > 10$ in Figure 2.3(c) and yellow region in Figure 2.5(c)). Figure 2.5(d) indicates that, in such cases, the effective learning rate η/z^2 of the direction cosine parameter R reaches a value below η that regulates l's converging speed, showing that an effective learning rate decreased too much turns into a bottleneck in converging. On the contrary, smaller initial values of the radial length than optimum do not cause such a delay. Therefore, when performing WN we should carefully choose the initial value of the radial length; it seems that one near the optimum is best, and we should at least avoid one much larger than the optimum.

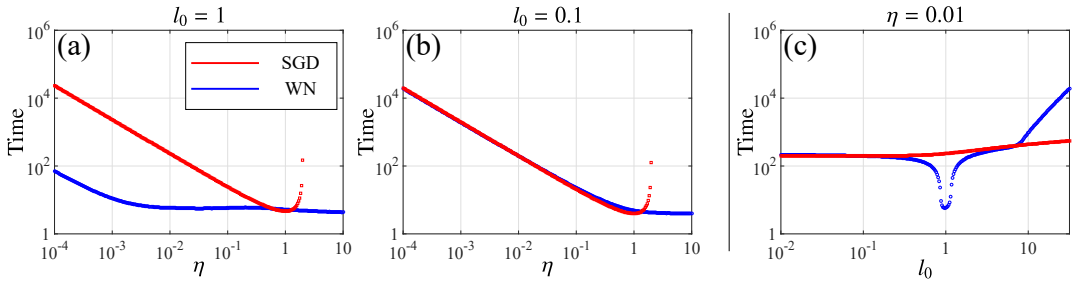

Figure 2.3: (a)(b) Dependence of elapsed time until generalization error ε_g falls below 0.01, on (a)(b) learning rate η and (c) initial radial length l_0. In (a) and (b), l_0 is fixed to 1 (global minimum) and 0.1, respectively. In (c), η is fixed to 0.01. Red symbols: SGD. Blue symbols: WN. Activation function is $g(x) = x$.

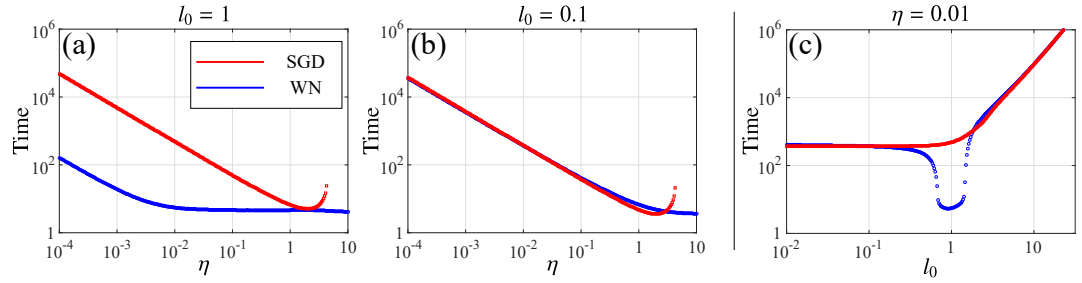

Figure 2.4: (a)(b) Dependence of elapsed time until generalization error ε_g falls below 0.01, on (a)(b) learning rate η and (c) initial radial length l_0. In (a) and (b), l_0 is fixed to 1 (global minimum) and 0.1, respectively. In (c), η is fixed to 0.01. Red symbols: SGD. Blue symbols: WN. Activation function is $g(x) = \mathrm{erf}(x/\sqrt{2})$.

2.5 Conclusion

Using the statistical mechanical method for online learning established by Biehl & Schwarze (1995)[8] and Saad & Solla (1995)[9], we analyzed the weight normalization proposed by Salimans & Kingma (2016)[13] in the simplest situation: that of single layer perceptrons. We revealed quantitatively that the "automatic tuning" of the effective learning rate of the direction cosine plays a critical role in fast convergence. Our theoretical argument does not depend on any specific form of the activation function. Thus, the mechanism of WN we discussed is not dependent on activation and seems to be a proper characteristic of WN.

In this study, we considered single layer perceptrons, although two-or-more-layered networks are often used for practical applications. Unlike a single layer perceptron, a multilayer perceptron has singular points and plateaus in its loss surface, which prevent converging towards its global minimum.[26] The fact that WN exhibits faster convergence even in multilayer networks[13] suggests that WN performs well even under the existence of singular points and plateaus. The statistical mechanical approach was used for a multilayer network by Biehl & Schwarze (1995)[8] and Saad & Solla (1995)[9] who addressed a two-layered soft committee machine and Riegler & Biehl (1995)[26] who analyzed the learning of two-layered perceptrons. Their methods seem to be applicable to the analysis of learning in two-or-more-layered perceptrons with WN.

The advantage of the statistical mechanical method is that we can briefly inspect the behavior of essential quantities like radial length and direction cosine. This statistical mechanical method is also used for theoretical analysis of natural gradients[27] and analysis of learning in networks with ReLU activation[28]. It might be useful for analysis of other recent optimization algorithms[19, 20, 21, 22, 24, 25]. Further success of this approach is expected.

2.A Calculation of expectation terms in the case of ReLU activation

We calculate expectation terms $I_3(y_1, y_2, y_3) := \langle g'(y_1) y_2 g(y_3) \rangle$ and $I_4(y_1, y_2, y_3) := \langle g'(y_1)^2 g(y_2) g(y_3) \rangle$, where (y_1, y_2, y_3) follows multivariate normal distribution $\mathcal{N}((y_1, y_2, y_3)|0, C)$ and $g(x) = \mathrm{ReLU}(x)$ $(:= \max\{0, x\})$.

For I_3,

$$\begin{aligned} I_3 = \langle g'(y_1) y_2 g(y_3) \rangle &= \int_{y_1, y_3 > 0} y_2 y_3 \mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, C) d\boldsymbol{y} \\ &= \frac{1}{(2\pi)^{3/2}\sqrt{|C|}} \int_{y_1, y_3 > 0} y_2 y_3 \exp(-\frac{1}{2}\boldsymbol{y}^T C^{-1} \boldsymbol{y}) d\boldsymbol{y} \\ &= -\frac{1}{(2\pi)^{3/2}\sqrt{|C|}} \frac{\partial}{\partial C_{23}^{-1}} \int_{y_1, y_3 > 0} \exp(-\frac{1}{2}\boldsymbol{y}^T C^{-1} \boldsymbol{y}) d\boldsymbol{y} \\ &= -\frac{1}{\sqrt{|C|}} \frac{\partial}{\partial C_{23}^{-1}} \left(\sqrt{|C|}\, P_3(C) \right). \end{aligned} \tag{2.12}$$

Here the term $P_3(C) := \int_{y_1,y_3>0} \mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, C)d\boldsymbol{y}$ is a "quadrant probability" of multivariate distribution with zero mean and is calculated with the formula:[29]

$$P_3(C) = \frac{1}{4} + \frac{1}{2\pi} \arcsin \frac{C_{13}}{\sqrt{C_{11}C_{33}}}. \tag{2.13}$$

With

$$\begin{aligned} -\frac{1}{\sqrt{|C|}} \frac{\partial}{\partial C_{23}^{-1}} \sqrt{|C|} &= -\frac{1}{\sqrt{|C|}} \frac{\partial}{\partial C_{23}^{-1}} \left(\sqrt{|C|} \int_{\mathbb{R}^3} \mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, C)d\boldsymbol{y} \right) \\ &= \int_{\mathbb{R}^3} y_2 y_3 \mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, C)d\boldsymbol{y} = C_{23}, \end{aligned}$$

equation (2.12) implies

$$I_3 = C_{23}P_3(C) - \frac{\partial P_3(C)}{\partial C_{23}^{-1}},$$

and by substituting (2.13) for $P_3(C)$, we obtain the first row of equation (4.7).

In the same way, for I_4 we get

$$I_4 = C_{23}P_4(C) - \frac{\partial P_4(C)}{\partial C_{23}^{-1}}$$

where

$$P_4(C) := \int_{y_1,y_2,y_3>0} \mathcal{N}(\boldsymbol{y}|\boldsymbol{0}, C)d\boldsymbol{y},$$

and by using the formula[29]

$$P_4(C) = \frac{1}{8} + \frac{1}{4\pi} \left(\arcsin \frac{C_{12}}{\sqrt{C_{11}C_{22}}} + \arcsin \frac{C_{13}}{\sqrt{C_{11}C_{33}}} + \arcsin \frac{C_{23}}{\sqrt{C_{22}C_{33}}} \right),$$

we can derive the second row of equation (4.7).

2.B Case of arbitrary loss functions

In the main text, we assumed the squared loss function $\varepsilon = \frac{1}{2}(t-s)^2$. However, there are many other loss functions which are commonly used, such as the softmax cross entropy loss in classification tasks. In this section, we discuss the case of arbitrary loss function $\varepsilon(s,t)$, where s is the student's output, and t is the teacher's output.

The differential equations (3) and (11), which describe the dynamics of order parameters, still hold if we replace the definition of terms A_i with

$$\begin{aligned} A_1(l,R) &= -\left\langle g'(lu)lu \left.\frac{\partial \varepsilon}{\partial s}\right|_{(s,t)=(g(lu),g(v))} \right\rangle, \\ A_2(l,R) &= \frac{1}{2}\left\langle g'(lu)^2 \left[\left.\frac{\partial \varepsilon}{\partial s}\right|_{(s,t)=(g(lu),g(v))} \right]^2 \right\rangle, \\ A_3(l,R) &= -\left\langle g'(lu)v \left.\frac{\partial \varepsilon}{\partial s}\right|_{(s,t)=(g(lu),g(v))} \right\rangle. \end{aligned}$$

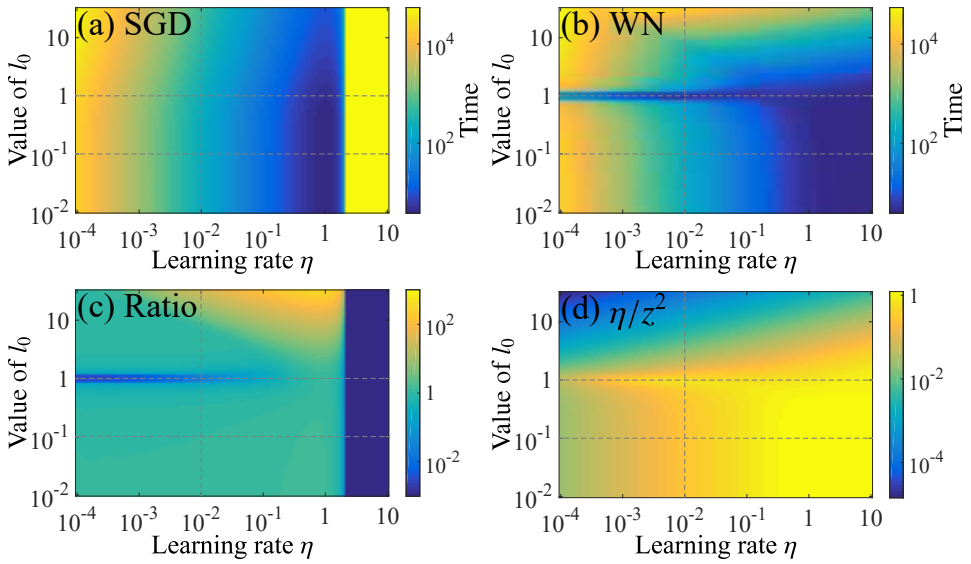

Figure 2.5: (a)(b) Elapsed time until generalization error ε_g falls below 0.01, shown as function of learning rate η and initial radial length l_0. (a) SGD, (b) WN. (c) Ratio of (a) and (b) (WN/SGD). (d) Value of effective learning rate η/z^2 when ε_g reaches < 0.01 in WN. Activation function is $g(x) = x$ in all cases. Three dashed lines correspond to Figure 2.3 (a), (b), and (c).

Whether these new expectation terms can be determined analytically depends on both the form of the activation function $g(x)$ and loss function $\varepsilon(s, t)$.

In the subsection "Linear stability analysis", we required $A_2 \geq 0$, $A_3 - A_1 \geq 0$, and $A_2 = A_3 - A_1 = 0$ at the global minimum $(l, R) = (1, 1)$. These properties are still true, provided that the activation function g is monotonous and the loss function $\varepsilon(s, t)$ with respect to s is single-peaked at the minimum $s = t$, for arbitrary t. In particular, the softmax cross entropy loss

$$\varepsilon(s,t) = H(\mathrm{Ber}(t), \mathrm{Ber}(s)) = -t \log s - (1-t)\log(1-s)$$
$$\text{where} \quad \mathrm{Ber}: \ \text{Bernouli distribution}, \quad H: \ \text{cross entropy}$$

with the sigmoidal activation $g(x) = 1/(1+e^{-x})$ satisfies the conditions above. Hence, our discussion suggests that WN with softmax cross entropy loss also works well.

Therefore, our theoretical argument can be extended to a wide range of loss functions, and WN seems to be advantageous in more general situation.

Chapter 3

Statistical Mechanical Analysis of Learning Dynamics of Two-Layer Perceptron with Multiple Output Units*

3.1 Introduction

Since deep learning was proposed in the late 2000s, neural networks have received great attention. That is because they enabled us to solve real-world tasks in various fields, including image recognition, speech recognition and machine translation tasks, with performance far exceeding conventional methods. However, there are some problems with neural networks left behind. One of them is the "plateau phenomenon," the main topic of this study, which we describe in detail below.

The perceptron is representative one of machine learning methods. Although it was firstly proposed in 1958 [30], there was no efficient learning algorithm at that time, and it was once got obsolete. In 1985, with discovery of the backpropagation method[31], which is a fundamental learning algorithm of neural networks, neural networks began to draw attention once again. However, another problem occurred; that is the "plateau phenomenon," wherein the learning slows down partway through. In the learning process of neural networks, weight parameters of the neural network are updated iteratively so that the loss (gap between the network's output and desired output) decreases. However, typically the loss does not decrease simply, but its decreasing speed slows down significantly partway through learning, and then it speeds up again after a long period of time (see the black line in Figure 3.2(a) and Figure 3.6 in the appendix for example). This is what we call plateau phenomenon. The phenomenon is observed ubiquitously in learning of hierarchical models, including neural networks, radial basis function networks and mixture of expert models[32, 9, 33, 34, 35, 1, 36, 27]. However, in recent years, although many researchers and engineers train hierarchical neural networks, the plateau phenomenon is rarely observed in

*This work has been published as [14].

practical applications of deep learning. Why is that?

With regard to theoretical studies of learning dynamics, that of linear neural networks has been studied analytically[7, 37, 38]. However, the plateau phenomenon is specific to nonlinear neural network, which has nonlinear activation function. Although studying the learning dynamics of nonlinear neural networks is challenging, the underlying mechanism of the plateau phenomenon has been widely studied. It is known empirically that neural networks get trapped into a plateau because they have symmetrical weights such that their input-output relationship is invariant under swapping two units of a hidden layer. The learning dynamics of neural networks have been studied in various settings. Some past studies, using statistical mechanical formalization, derived the learning dynamics of a two-layer soft committee machine[9] and a two-layer perceptron[32]; these researches showed that the weight symmetry results in saddle points that cause plateauing. Furthermore, it has been recognized that such plateau phenomenon stems from singular structure in the parameter space (see [36, 6, 39, 40], or [41] for a review). In the parameter space, a Riemannian metric is naturally induced by Fisher information matrix, which represents how two models identified by slightly different parameters differ as statistical models[42]. This metric is not necessarily regular, but there are regions in the parameter space in which the metric degenerates, called singular regions[6]. More specifically, when we consider a two-layer neural network, if it has two identical weight vectors projected from the input layer to two different hidden units, its input-output relationship can be realized by another model which has one lesser hidden units than the original model. If this downsized model gives local minima of the loss in the downsized parameter space, the original model parameter is in the singular region (see also the section "Singular regions and plateaus" for detailed explanations). The gradient of the loss is zero everywhere within the singular region. Although an isolated saddle point has the same property, the singular region and a saddle point is different to an isolated saddle point in two ways; the singular region has one-or-more dimension, and it is Milnor-like attractor[15], that is, it has a region of attraction which has nonzero measure (see Figure 5 in [6] for a schematic diagram of the singular region). However, all of these researches assume one-dimensional output; in other words, networks that have multiple output units are overlooked, and they may avoid the plateau phenomena, which is a usual situation in modern deep learning and seems to be rational.

For these reasons, we analyze the learning dynamics of a two-layer nonlinear perceptron that has multiple output units, with statistical mechanical formulation. First, we introduce the student-teacher learning setting for ease of analysis [9, 8]. Second, we introduce several order parameters, which can capture macroscopic characteristics of network weights, and derive their evolution equations from that of microscopic variables (i.e. network weights). We solve derived equations numerically to get macroscopic learning dynamics, with which we can discuss plateau phenomenon quantitatively. Under a simple setting, we show that singular-region-driven plateaus diminish or vanish with multidimensional output. We find that the less degenerative (i.e. like one-dimensional output) the model is, the less the model approaches the singular region, and then the more plateau shortens. Further, we show theoretically that singular-region-driven plateaus seldom occur in the learning process if the student and teacher models are initialized orthogonally.

Note that the claim that having multiple output units might alleviate plateau phenomenon is hypothesized by intuitive insight in our previous work [43], but in the current work we examine the hypothesis by experiments and theoretical analyses and show that multiple output units can indeed prevent from approaching the singular region and vanish plateaus.

3.2 Model

We considered a neural network with an input layer (size N), a hidden layer (size K), and an output layer (size O). The network receives input data $\boldsymbol{\xi} \in \mathbb{R}^N$ and calculates output $\boldsymbol{s} = \sum_{i=1}^{K} \boldsymbol{w}_i g(\boldsymbol{J}_i \cdot \boldsymbol{\xi}) \in \mathbb{R}^O$, where g is the activation function. The parameters $\boldsymbol{J}$ and $\boldsymbol{w}$ are optimized during learning depending on the difference between the output $\boldsymbol{s}$ and the teacher data $\boldsymbol{t}$. We considered an ideal situation in which the teacher data $\boldsymbol{t}$ is determined as $\boldsymbol{t} = \sum_{n=1}^{M} \boldsymbol{v}_n g(\boldsymbol{B}_n \cdot \boldsymbol{\xi}) \in \mathbb{R}^O$; in other words, the learning network (the "student network") learns the input-output relationship of the "teacher network", which is also a two-layer network and has N-M-O units and original fixed weights $\boldsymbol{B}$ and $\boldsymbol{v}$ (Figure 4.3(a)). We assumed the squared loss function $\varepsilon = \frac{1}{2}\|\boldsymbol{s} - \boldsymbol{t}\|^2$, which is most commonly used for regression.

For the statistical mechanical formulation of online learning, we introduced further idealization. We assumed that the dimension of input data N is very large and that each element of input data $\boldsymbol{\xi}$ is generated in accordance with i.i.d. normal distribution, $\mathcal{N}(\xi_i|0,1)$.* We put $x_i := \boldsymbol{J}_i \cdot \boldsymbol{\xi}$ and $y_n := \boldsymbol{B}_n \cdot \boldsymbol{\xi}$ and define $Q_{ij} := \boldsymbol{J}_i \cdot \boldsymbol{J}_j = \langle x_i x_j \rangle$, $R_{in} := \boldsymbol{J}_i \cdot \boldsymbol{B}_n = \langle x_i y_n \rangle$, $T_{nm} := \boldsymbol{B}_n \cdot \boldsymbol{B}_m = \langle y_n y_m \rangle$ and $D_{ij} := \boldsymbol{w}_i \cdot \boldsymbol{w}_j$, $E_{in} := \boldsymbol{w}_i \cdot \boldsymbol{v}_n$, $F_{nm} := \boldsymbol{v}_n \cdot \boldsymbol{v}_m$.

The parameters Q_{ij}, R_{in}, T_{nm}, D_{ij}, E_{in}, and F_{nm} defined above capture the state of the system macroscopically; they are therefore called as "order parameters." The first three represent the state of the first layers of the two networks (student and teacher), and the latter three represent their second layers' state (Figure 4.3(b)). Roughly speaking, Q represents the norm of the student's first layer and T represents that of the teacher's first layer. R is related to similarity between the student and teacher's first layer. D, E, F is the second layers' counterpart of Q, R, T. Among these six order parameter matrices, the values of Q_{ij}, R_{in}, D_{ij}, and E_{in} change during learning; their dynamics are what to be determined, from the dynamics of microscopic variables, i.e. connection weights.

3.3 Dynamics of order parameters

In this paper, we adopt the stochastic gradient descent (SGD) learning algorithm, which underlies all conventional algorithms of neural networks used in practice. That is,

*This assumption can be relaxed. See the appendix at the end of this dissertation.

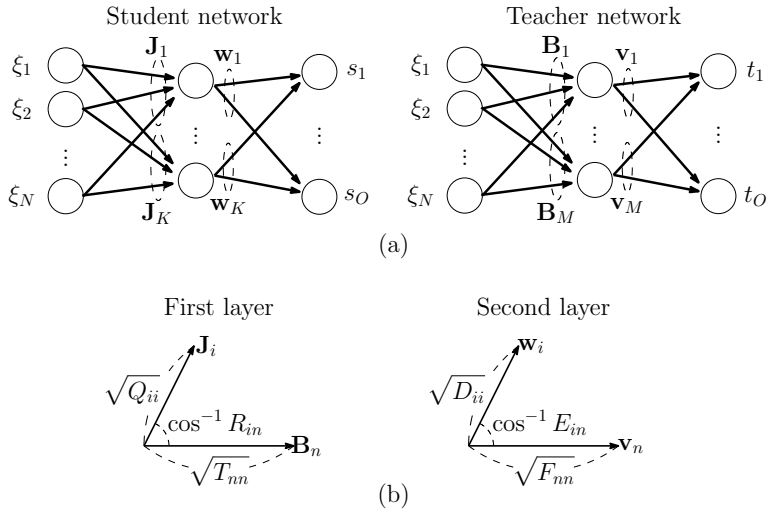

Figure 3.1: (a) Student and teacher networks. (b) Geometrical interpretation of order parameters Q_{ij}, R_{in}, T_{nm}, D_{ij}, E_{in}, and F_{nm}.

every time an input sample is given, the output and sample loss is computed, the differentiation of the sample loss with respect to model parameters is computed, and finally, model parameters are moved against the gradient of the loss. The update rule of connection weights with SGD is written as

$$\begin{aligned}
\Delta \boldsymbol{J}_i &= -\frac{\eta}{N}\frac{d\varepsilon}{d\boldsymbol{J}_i} = \frac{\eta}{N}[(\boldsymbol{t}-\boldsymbol{s})\cdot\boldsymbol{w}_i]g'(x_i)\boldsymbol{\xi} \\
&= \frac{\eta}{N}\left[\left(\sum_{n=1}^{M}\boldsymbol{v}_n g(y_n) - \sum_{j=1}^{K}\boldsymbol{w}_j g(x_j)\right)\cdot\boldsymbol{w}_i\right] g'(x_i)\boldsymbol{\xi}, \\
\Delta \boldsymbol{w}_i &= -\frac{\eta}{N}\frac{d\varepsilon}{d\boldsymbol{w}_i} = \frac{\eta}{N}g(x_i)(\boldsymbol{t}-\boldsymbol{s}) \\
&= \frac{\eta}{N}g(x_i)\left(\sum_{n=1}^{M}\boldsymbol{v}_n g(y_n) - \sum_{j=1}^{K}\boldsymbol{w}_j g(x_j)\right),
\end{aligned} \tag{3.1}$$

in which we set the learning rate as η/N, in order to obtain N-independent macroscopic system. The first equation of (4.1) gives the update rule of order parameters Q_{ij} and R_{in}

in the form of difference equations:

$$\begin{aligned}
\Delta Q_{ij} &= (\boldsymbol{J}_i + \Delta\boldsymbol{J}_i)\cdot(\boldsymbol{J}_j + \Delta\boldsymbol{J}_j) - \boldsymbol{J}_i\cdot\boldsymbol{J}_j \\
&= \boldsymbol{J}_i\cdot\Delta\boldsymbol{J}_j + \boldsymbol{J}_j\cdot\Delta\boldsymbol{J}_i + \Delta\boldsymbol{J}_i\cdot\Delta\boldsymbol{J}_j \\
&= \frac{\eta}{N}\left[\sum_{p=1}^{M} E_{ip}g'(x_i)x_j g(y_p) - \sum_{p=1}^{K} D_{ip}g'(x_i)x_j g(x_p)\right. \\
&\qquad \left. + \sum_{p=1}^{M} E_{jp}g'(x_j)x_i g(y_p) - \sum_{p=1}^{K} D_{jp}g'(x_j)x_i g(x_p)\right] \\
&\quad + \frac{\eta^2}{N^2}\|\boldsymbol{\xi}\|^2\left[\sum_{p,q}^{K,K} D_{ip}D_{jq}g'(x_i)g'(x_j)g(x_p)g(x_q)\right. \\
&\qquad + \sum_{p,q}^{M,M} E_{ip}E_{jq}g'(x_i)g'(x_j)g(y_p)g(y_q) \\
&\qquad \left. - \sum_{p,q}^{K,M} D_{ip}E_{jq}g'(x_i)g'(x_j)g(x_p)g(y_q) - \sum_{p,q}^{M,K} E_{ip}D_{jq}g'(x_i)g'(x_j)g(y_p)g(x_q)\right], \\
\Delta R_{in} &= (\boldsymbol{J}_i + \Delta\boldsymbol{J}_i)\cdot\boldsymbol{B}_n - \boldsymbol{J}_i\cdot\boldsymbol{B}_n \\
&= \Delta\boldsymbol{J}_i\cdot\boldsymbol{B}_n \\
&= \frac{\eta}{N}\left[\sum_{p=1}^{M} E_{ip}g'(x_i)y_n g(y_p) - \sum_{p=1}^{K} D_{ip}g'(x_i)y_n g(x_p)\right].
\end{aligned} \tag{3.2}$$

Since $\|\boldsymbol{\xi}\|^2 \approx N$ and the right hand sizes of these equations are $O(N^{-1})$, we can replace these difference equations with differential ones with $N \to \infty$, by taking the expectation over all input vectors $\boldsymbol{\xi}$:

$$\begin{aligned}
\frac{dQ_{ij}}{d\tilde{\alpha}} &= \eta\left[\sum_{p=1}^{M} E_{ip}I_3(x_i, x_j, y_p) - \sum_{p=1}^{K} D_{ip}I_3(x_i, x_j, x_p)\right. \\
&\qquad \left. + \sum_{p=1}^{M} E_{jp}I_3(x_j, x_i, y_p) - \sum_{p=1}^{K} D_{jp}I_3(x_j, x_i, x_p)\right] \\
&\quad + \eta^2\left[\sum_{p,q}^{K,K} D_{ip}D_{jq}I_4(x_i, x_j, x_p, x_q) + \sum_{p,q}^{M,M} E_{ip}E_{jq}I_4(x_i, x_j, y_p, y_q)\right. \\
&\qquad \left. - \sum_{p,q}^{K,M} D_{ip}E_{jq}I_4(x_i, x_j, x_p, y_q) - \sum_{p,q}^{M,K} E_{ip}D_{jq}I_4(x_i, x_j, y_p, x_q)\right], \\
\frac{dR_{in}}{d\tilde{\alpha}} &= \eta\left[\sum_{p=1}^{M} E_{ip}I_3(x_i, y_n, y_p) - \sum_{p=1}^{K} D_{ip}I_3(x_i, y_n, x_p)\right]
\end{aligned} \tag{3.3}$$

$$\begin{aligned}
&\text{where} \quad I_3(z_1, z_2, z_3) := \langle g'(z_1)z_2 g(z_3)\rangle, \\
&\qquad\qquad I_4(z_1, z_2, z_3, z_4) := \langle g'(z_1)g'(z_2)g(z_3)g(z_4)\rangle.
\end{aligned} \tag{3.4}$$

In these equations, $\tilde{\alpha} := \alpha/N$ represents time (normalized number of steps), and

the brackets $\langle\cdot\rangle$ represent the expectation when the input $\boldsymbol{\xi}$ follows $\mathcal{N}(\xi_i|0,1)$, that is, when $(x_1,\ldots,x_K,y_1,\ldots,y_M)$ follows $\mathcal{N}(\mathbf{0},\begin{pmatrix} Q & R \\ R^T & T \end{pmatrix})$. Likewise, the difference equations of the second layers' order parameters D_{ij} and E_{in} are obtained by the second equation of (4.1) as

$$\begin{aligned}
\Delta D_{ij} &= (\boldsymbol{w}_i + \Delta\boldsymbol{w}_i)\cdot(\boldsymbol{w}_j + \Delta\boldsymbol{w}_j) - \boldsymbol{w}_i\cdot\boldsymbol{w}_j \\
&= \boldsymbol{w}_i\cdot\Delta\boldsymbol{w}_j + \boldsymbol{w}_j\cdot\Delta\boldsymbol{w}_i + \Delta\boldsymbol{w}_i\cdot\Delta\boldsymbol{w}_j \\
&= \frac{\eta}{N}\left[\sum_{p=1}^{M} E_{ip}g(x_j)g(y_p) - \sum_{p=1}^{K} D_{ip}g(x_j)g(x_p)\right. \\
&\qquad\left. + \sum_{p=1}^{M} E_{jp}g(x_i)g(y_p) - \sum_{p=1}^{K} D_{jp}g(x_i)g(x_p)\right] \\
&\quad + \frac{\eta^2}{N^2}\left[\sum_{p,q}^{K,K} D_{pq}g(x_i)g(x_j)g(x_p)g(x_q) + \sum_{p,q}^{M,M} F_{pq}g(x_i)g(x_j)g(y_p)g(y_q)\right. \\
&\qquad\left. -2\sum_{p,q}^{K,M} E_{pq}g(x_i)g(x_j)g(x_p)g(y_q)\right], \\
\Delta E_{in} &= (\boldsymbol{w}_i + \Delta\boldsymbol{w}_i)\cdot\boldsymbol{v}_n - \boldsymbol{w}_i\cdot\boldsymbol{v}_n \\
&= \Delta\boldsymbol{w}_i\cdot\boldsymbol{v}_n \\
&= \frac{\eta}{N}\left[\sum_{p=1}^{M} F_{pn}g(x_i)g(y_p) - \sum_{p=1}^{K} E_{pn}g(x_i)g(x_p)\right].
\end{aligned} \tag{3.5}$$

Again, the right hand sides are $O(N^{-1})$, therefore these difference equations can be rewritten to differential versions with $N\to\infty$, by taking the expectation over all input vectors $\boldsymbol{\xi}$:

$$\begin{aligned}
\frac{dD_{ij}}{d\tilde{\alpha}} &= \eta\left[\sum_{p=1}^{M} E_{ip}I_2(x_j,y_p) - \sum_{p=1}^{K} D_{ip}I_2(x_j,x_p)\right. \\
&\qquad\left. + \sum_{p=1}^{M} E_{jp}I_2(x_i,y_p) - \sum_{p=1}^{K} D_{jp}I_2(x_i,x_p)\right], \qquad (3.6) \\
\frac{dE_{in}}{d\tilde{\alpha}} &= \eta\left[\sum_{p=1}^{M} F_{pn}I_2(x_i,y_p) - \sum_{p=1}^{K} E_{pn}I_2(x_i,x_p)\right] \\
&\qquad \text{where} \quad I_2(z_1,z_2) := \langle g(z_1)g(z_2)\rangle. \qquad (3.7)
\end{aligned}$$

These differential equations govern the macroscopic dynamics. Note that the generalization loss ε_g, the expectation of loss value $\varepsilon(\boldsymbol{\xi}) = \frac{1}{2}\|\boldsymbol{s}-\boldsymbol{t}\|^2$ over all input vectors $\boldsymbol{\xi}$, is represented as

$$\begin{aligned}
\varepsilon_g &= \langle\frac{1}{2}\|\boldsymbol{s}-\boldsymbol{t}\|^2\rangle \\
&= \frac{1}{2}\left[\sum_{p,q}^{M,M} F_{pq}I_2(y_p,y_q) + \sum_{p,q}^{K,K} D_{pq}I_2(x_p,x_q) - 2\sum_{p,q}^{K,M} E_{pq}I_2(x_p,y_q)\right].
\end{aligned} \tag{3.8}$$

Expectation terms I_2, I_3 and I_4 can be analytically determined for some activation functions g, including sigmoid-like $g(x) = \mathrm{erf}(x/\sqrt{2})$[9], and $g(x) = \mathrm{ReLU}(x) =: \max\{0, x\}$ which is commonly used in deep learning[18, 12].

3.4 Singular regions and plateaus

In this section we review the concept of singular regions as the cause of the plateau phenomenon[6].

In general, the input-output relationship (i.e. mapping) of a neural network is determined by the parameter values (i.e. connection weights) of the network. However, this correspondence is not one-to-one; in other words, there could be multiple settings of parameter values that result in one specific model (i.e. input-output mapping). For example, consider two-layer networks that have two hidden units (i.e. $K = 2$) and one output unit (i.e. $O = 1$) as

$$s = w_1 g(\boldsymbol{J}_1 \cdot \boldsymbol{\xi}) + w_2 g(\boldsymbol{J}_2 \cdot \boldsymbol{\xi}) \tag{3.9}$$

where the parameter is $(\boldsymbol{J}_1, \boldsymbol{J}_2, w_1, w_2)$. Then, all of the parameter settings in the parameter regions

$$\begin{aligned} \mathcal{R}_1(\boldsymbol{J}^*, w^*) &:= \{\boldsymbol{J}_1 = \boldsymbol{J}_2 = \boldsymbol{J}^* \text{ and } w_1 + w_2 = w^* \mid (\boldsymbol{J}_1, \boldsymbol{J}_2, w_1, w_2)\}, \\ \mathcal{R}_2(\boldsymbol{J}^*, w^*) &:= \{(\boldsymbol{J}_1 = \boldsymbol{J}^* \text{ and } w_1 = w^* \text{ and } w_2 = 0) \\ &\qquad \text{or } (\boldsymbol{J}_2 = \boldsymbol{J}^* \text{ and } w_1 = 0 \text{ and } w_2 = w^*) \mid (\boldsymbol{J}_1, \boldsymbol{J}_2, w_1, w_2)\} \end{aligned} \tag{3.10}$$

correspond to the same model (i.e. input-output mapping):

$$s = w^* g(\boldsymbol{J}^* \cdot \boldsymbol{\xi}). \tag{3.11}$$

These parameter regions $\mathcal{R}_i$ are called the singular region.

The model (3.11) can be regarded as a $K = 1$ model, parameterized by $\boldsymbol{J}^*$ and w^*. Now suppose that

$$\frac{\partial \varepsilon_g}{\partial w^*} = 0 \quad \text{and} \quad \frac{\partial \varepsilon_g}{\partial \boldsymbol{J}^*} = \boldsymbol{0}; \tag{3.12}$$

note that this occurs when the $K = 1$ model (3.11) gives a local minima of the generalization loss ε_g in the $K = 1$ parameter space. Then, one can show that the gradient of the generalization loss is also zero throughout the singular regions $\mathcal{R}_1$ and $\mathcal{R}_2$ in the original $K = 2$ parameter space; that is, $\partial \varepsilon_g / \partial w_i = 0$ and $\partial \varepsilon_g / \partial \boldsymbol{J}_i = \boldsymbol{0}$ if $(\boldsymbol{J}_1, \boldsymbol{J}_2, w_1, w_2) \in \mathcal{R}_1 \cup \mathcal{R}_2$, provided that (3.12) holds. Moreover, the singular region $\mathcal{R}_1$ has the following properties[44]:

- The region $\mathcal{R}_1$ is partially stable. When the parameter value is in the stable part of $\mathcal{R}_1$, it undergoes a long period of random walk, by fluctuations due to the random sampling of $\boldsymbol{\xi}$. Once the parameter value reached the unstable part of the region, it starts moving away from the region.
- The region $\mathcal{R}_1$ is a Milnor-like attractor[15] in the sense that it has a positive measure of basin of attraction. This means that randomly chosen initial parameter value will get trapped in the region with nonzero probability.

From these points, the singular region is completely different from a saddle point. When trapped in the singular region, the learning process inevitably slows down. That is why the singular region is considered to be the cause of plateaus.

However, the concept of problematic plateaus described above may not be true when a network has multiple outputs ($O > 1$); the loss gradient does not necessarily vanish at the singular region, and the network might not be attracted to the singular region[43]. Thus, we analyzed the learning dynamics of networks that have multiple output units and examined whether or not the networks were trapped in the singular region and plateaus during learning.

3.5 Numerical results

We discuss the dynamics of learning in a two-layer perceptron by numerically solving the differential equations of the order parameters (4.9) and (4.5). For simplicity, we focus on the case with $K = M = 2$ units in the hidden layers and $O = 2$ units in the output layers. In the numerical experiments described below, we initialize the weights of the first layers of the student and teacher networks with $(\boldsymbol{J}_i)_j, (\boldsymbol{B}_i)_j \sim \mathcal{N}(0, 1/N)$ (i.i.d.). This initialization makes the initial values of the first layers' order parameters

$$Q = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad R = \begin{pmatrix} 0 & 0 \\ 0 & 0 \end{pmatrix}, \quad T = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix} \tag{3.13}$$

on average. When N is finite, each component of the matrix Q, R and T has $O(N^{-1})$ noise; it vanishes as $N \to \infty$. It is critical how we initialize the weights of the second layers of the student and teacher networks, $\boldsymbol{v}_i$ and $\boldsymbol{w}_i$. For example, consider the case

$$\boldsymbol{w}_1 = w_1\boldsymbol{c}, \quad \boldsymbol{w}_2 = w_2\boldsymbol{c}; \quad \boldsymbol{v}_1 = v_1\boldsymbol{c}, \quad \boldsymbol{v}_2 = v_2\boldsymbol{c} \tag{3.14}$$

where $\boldsymbol{c}$ is an arbitrary two-dimensional constant vector whose norm is 1. In this setting, the outputs of the teacher and student networks, $\boldsymbol{s}, \boldsymbol{t} \in \mathbb{R}^2$, are confined in the one-dimensional subspace along $\boldsymbol{c}$. This makes the learning process equivalent to one with one output unit. In fact, when (3.14) holds,

$$\boldsymbol{t} = \left[\sum_{n=1}^{M} v_n g(\boldsymbol{B}_n \cdot \boldsymbol{\xi})\right] \boldsymbol{c} = t\boldsymbol{c}, \qquad \boldsymbol{s} = \left[\sum_{i=1}^{K} w_i g(\boldsymbol{J}_i \cdot \boldsymbol{\xi})\right] \boldsymbol{c} = s\boldsymbol{c} \tag{3.15}$$

where we defined scalar t and s as $\displaystyle\sum_{n=1}^{M} v_n g(\boldsymbol{B}_n \cdot \boldsymbol{\xi})$ and $\displaystyle\sum_{i=1}^{K} w_i g(\boldsymbol{J}_i \cdot \boldsymbol{\xi})$ respectively. And from the update rule (4.1),

$$\begin{aligned} \Delta\boldsymbol{J}_i &= \frac{\eta}{N}[(\boldsymbol{t} - \boldsymbol{s}) \cdot \boldsymbol{w}_i]g'(x_i)\boldsymbol{\xi} = \frac{\eta}{N} w_i(t - s)g'(x_i)\boldsymbol{\xi}, \\ \Delta\boldsymbol{w}_i &= \frac{\eta}{N} g(x_i)(\boldsymbol{t} - \boldsymbol{s}) = \frac{\eta}{N} g(x_i)(t - s)\boldsymbol{c} \\ &\text{that is,} \quad \Delta w_i = \frac{\eta}{N} g(x_i)(t - s) \end{aligned} \tag{3.16}$$

which is simply the update rule when both networks have only one output unit.

Thus, how the student network with two-dimensional output learns the teacher network with two-dimensional output largely depends on the initial condition of the weight matrices of their second layers. To see this, we consider an initial condition parameterized by θ:

$$\boldsymbol{w}_1 = \boldsymbol{c}, \quad \boldsymbol{w}_2 = \boldsymbol{c}_\theta; \quad \boldsymbol{v}_1 = \boldsymbol{c}, \quad \boldsymbol{v}_2 = \boldsymbol{c}_\theta \tag{3.17}$$

where $\boldsymbol{c}$ is again an arbitrary two-dimensional constant vector whose norm is 1, and $\boldsymbol{c}_\theta$ is a constant vector which is obtained by rotating $\boldsymbol{c}$ by θ. We refer to the parameter θ as non-degeneracy because it regulates the degeneracy of the weight matrices of the second layers of both networks. We can test various situations by changing θ continuously; $\theta = 0$ makes both matrices degenerate, and $\theta = \pi/2$ makes both matrices orthogonal. The former situation, $\theta = 0$, is equivalent to the one-dimensional output situation, as previously described. The initial condition of the second layers' order parameters D_{ij}, E_{in}, F_{nm}, corresponding to (3.17), is given by

$$D = E = F = \begin{pmatrix} 1 & \cos\theta \\ \cos\theta & 1 \end{pmatrix}. \tag{3.18}$$

Putting these initial conditions together, we examined the learning dynamics in two ways: by simulating the original microscopic system with finite N, i.e. conducting stochastic gradient descent in accordance with the update rule (4.1) , and by solving the differential equations (4.9) and (4.5) of the order parameters numerically under initial conditions that match the initial weights used when simulating the original microscopic system. The black lines in Figure 3.2 show the time courses of the generalization loss ε_g in several typical situations: (a) $\theta = 0$, (b) $\theta = \pi/8$, (c) $\theta = \pi/4$, and (d) $\theta = \pi/2$. To evaluate quantitatively how near the student network is to the singular region $\mathcal{R}_1$ and $\mathcal{R}_2$, we calculated two measures: the overlap of vectors $\boldsymbol{J}_1$ and $\boldsymbol{J}_2$, i.e. $m_{12}^{(1)} := |\boldsymbol{J}_1 \cdot \boldsymbol{J}_2|/\|\boldsymbol{J}_1\|\|\boldsymbol{J}_2\| = |Q_{12}|/\sqrt{Q_{11}Q_{22}}$, and the minimum norm of vectors $\boldsymbol{w}_1$ and $\boldsymbol{w}_2$, i.e. $l_{\min}^{(2)} := \min\{\|\boldsymbol{w}_1\|, \|\boldsymbol{w}_2\|\} = \min\{\sqrt{D_{11}}, \sqrt{D_{22}}\}$. Note that $m_{12}^{(1)}$ measures proximity to the region $\mathcal{R}_1$, and $l_{\min}^{(2)}$ indicates the distance to the region $\mathcal{R}_2$. Figure 3.2 also shows the time evolutions of these measures with blue and red lines, respectively. Results of microscopic simulations are also shown by dots. In every plot in Figure 3.2, the solid lines and dots agree well, meaning that the macroscopic system of order parameters appropriately captures the microscopic system of connection weights. In Figure 3.2(a), the generalization loss ε_g stops during the first ~ 1800 steps, along with high values of $m_{12}^{(1)}$, meaning that falling into the singular region $\mathcal{R}_1$ derived from network's symmetry. In Figure 3.2(b), the plateau shortens. An increase in $m_{12}^{(1)}$ is still observed, although its peak is lower than Figure 3.2(a). In Figure 3.2(c) and (d), there is no apparent plateau. In particular, the overlap $m_{12}^{(1)}$ is always 0 in Figure 3.2(d), signifying that the student network does not approach the singular region $\mathcal{R}_1$ at all during learning. Note also that no approach to the singular region $\mathcal{R}_2$, indicated by high values of $l_{\min}^{(2)}$, is observed in any plot in Figure 3.2 at any time. Figure 3.3 shows the times the plateau begins and ends, depending on θ, calculated by the numerical solutions of the order parameters. Here we define plateaus as "where the logarithm of generalization loss decreases at a rate slower than a half of the typical rate," where the typical rate is measured as the rate when $\varepsilon_g < 10^{-10}$ is achieved. We found that the plateau is observed if and only if roughly $|\theta| < 0.1\pi$ holds. Note that our quantitative definition of plateaus above contains some arbitrariness, but it

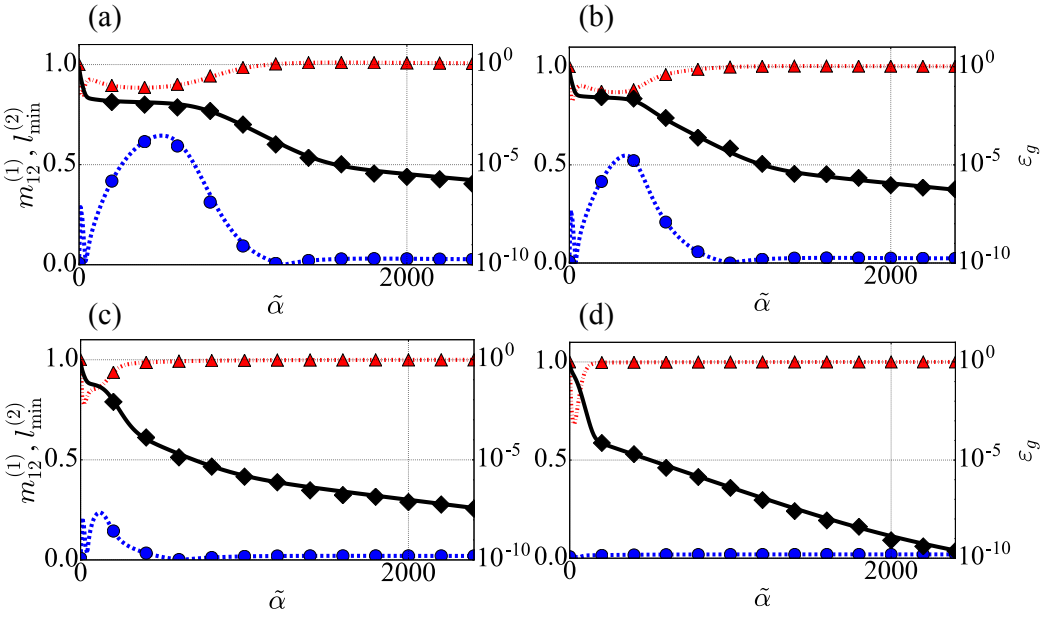

Figure 3.2: Dependence of time course of generalization loss ε_g (black solid line) on non-degeneracy parameter θ. Time course of student's first layer's overlap $m_{12}^{(1)} := |Q_{12}|/\sqrt{Q_{11}Q_{22}}$ (blue dashed line) and minimum norm of student's second norm $l_{\min}^{(2)} := \min\{\sqrt{D_{11}}, \sqrt{D_{22}}\}$ (red dot-dashed line) also shown. These lines indicate how student network is close to singular regions $\mathcal{R}_1$ and $\mathcal{R}_2$, respectively. (a) $\theta = 0$, (b) $\theta = \pi/8$, (c) $\theta = \pi/4$, (d) $\theta = \pi/2$. Simulation results of microscopic systems shown by dots (diamonds, circles, and triangles for ε_g, $m_{12}^{(1)}$ and $l_{\min}^{(2)}$ respectively). Generalization loss of simulation results approximated by averaged sample loss (ε) over 100 contiguous steps. Simulation parameters: $N = 1000$, $\eta = 0.1$ ($\eta/N = 0.0001$). Activation function: $g(x) = \mathrm{erf}(x/\sqrt{2})$.

does not affect the main point; the plateau phenomenon get alleviated as $|\theta|$ increases, as is evident from Figure 3.2.

3.6 Cases for orthogonal second layers

Up to this point, we considered cases in which the second layers of two networks have identical weights, as (3.17). However, this is not practical because it means that the student knows about the teacher in advance. Thus, we consider a slightly different situation:

$$\boldsymbol{w}_1 = \boldsymbol{c}, \quad \boldsymbol{w}_2 = \boldsymbol{c}_{\pi/2}; \quad \boldsymbol{v}_1 = \boldsymbol{c}_{\phi}, \quad \boldsymbol{v}_2 = \boldsymbol{c}_{\phi+\pi/2}, \tag{3.19}$$

wherein the student matrix and teacher matrix are not identical but are both orthogonal. The initial conditions of the order parameters D_{ij}, E_{in}, F_{nm}, corresponding to (3.19), are given by

$$D = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}, \quad E = \begin{pmatrix} \cos\phi & -\sin\phi \\ \sin\phi & \cos\phi \end{pmatrix}, \quad F = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}. \tag{3.20}$$

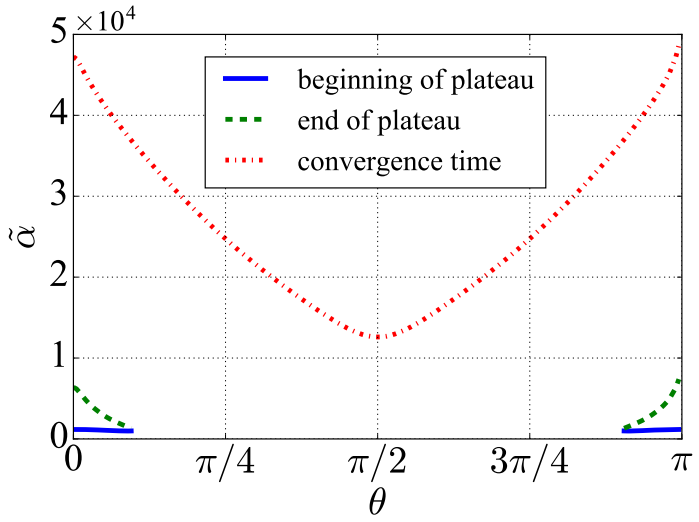

Figure 3.3: Dependence of when plateau starts and ends and when generalization loss converges on non-degeneracy parameter θ. Blue solid line: start of plateau, green dashed line: end of plateau, red dot-dashed line: achieving $\varepsilon_g < 10^{-10}$. See main text for definition of plateau in this figure. Parameters: $\eta = 0.02$. Activation: $g(x) = \mathrm{erf}(x/\sqrt{2})$.

Under these initial conditions, we found that the orthogonality $\boldsymbol{w}_1 \perp \boldsymbol{w}_2$ remains true during learning at any time, as formally stated in the Proposition below. In other words, we prove that the student network stays opposite the singular region $\mathcal{R}_1$.

Proposition. Assume $K = M = 2$ and $O \geq 2$, and consider the situation in which $N \to \infty$. Assume that the activation g is an odd function and that the right sides of the differential equations are Lipschitz continuous in the vicinity of the solution trajectory (Q, R, D, E) during learning. If both the teacher and student network have a pair of orthogonal column vectors in its second layer's weight matrix, that is, $\boldsymbol{v}_1 \perp \boldsymbol{v}_2$ and $\boldsymbol{w}_1 \perp \boldsymbol{w}_2$ hold at the initial state, then the orthogonality $\boldsymbol{w}_1 \perp \boldsymbol{w}_2$ holds at any time during learning. ■

We give the proof of the Proposition in the Appendix.

The numerical solution of the differential equations of the order parameters, under the initial conditions described above, is shown in Figure 3.4. These solutions tell us that there is no approach to the singular regions $\mathcal{R}_1$ and $\mathcal{R}_2$ and that no plateau is seen, except for the case of $\phi = \pi/4$ (Figure 3.4(d)); this plateau is not due to the singular regions but rather to being stuck at a saddle point where the weight vectors of the second layer of the student network are perturbed by the stochastic gradient and can easily escape.

3.7 Simulation results for more practical cases

By using order parameters we analyzed theoretically the case with two hidden units. However, networks with greater number of hidden units are usually used in practice.Also,

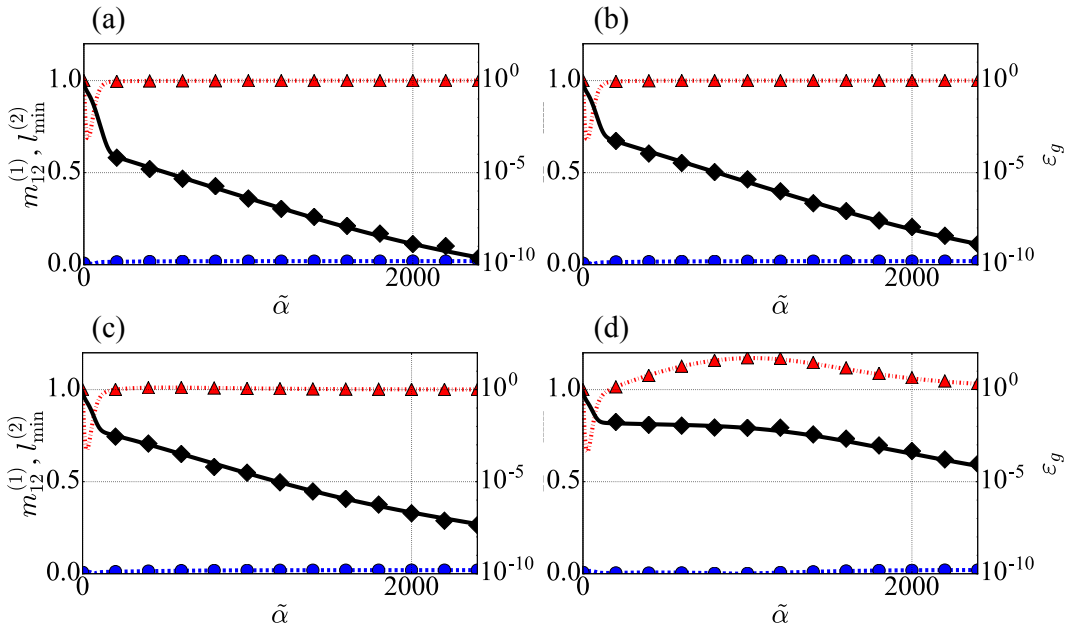

Figure 3.4: Dependence of time course of generalization loss ε_g (black solid line) on rotation parameter ϕ. $m_{12}^{(1)}$ (blue dashed line) and $l_{\min}^{(2)}$ (red dot-dashed line) also shown; see caption of Figure 3.2 for detailed explanation. (a) $\phi = 0$, (b) $\phi = \pi/8$, (c) $\phi = 3\pi/16$, (d) $\phi = \pi/4$. Simulation results of microscopic systems shown by dots (diamonds, circles, and triangles for ε_g, $m_{12}^{(1)}$ and $l_{\min}^{(2)}$ respectively). Generalization loss of simulation results approximated by averaged sample loss (ε) over 100 contiguous steps. Simulation parameters: $N = 1000$, $\eta = 0.1$ ($\eta/N = 0.0001$). Activation: $g(x) = \mathrm{erf}(x/\sqrt{2})$.

the setting above assumes infinite number of samples which is not available in reality. Furthermore, various optimization techniques developed in recent years such as mini-batch, dropout and gradient descent with adaptive learning rate are widely used. In these practical cases the learning dynamics might not be tractable. We examined such cases by numerical simulations of microscopic systems.

In our experiment, a student network and a teacher network both of which have 100 input units and 10 hidden units are used. First we generated 4000 training samples and 1000 test samples by using the teacher network which weights are randomly chosen. We then trained the student network which weights are randomly initialized using training samples only, and computed the training loss and the test loss after every epoch.

In Figure 3.5, the dynamics of the learning in the experiment is shown. (Although this dynamics depends on the initial weights, the qualitative shapes of the plateaus do not depends much; see the appendix.) The output dimension is 1 in Figure 3.5(a) and (c), and 10 in Figure 3.5(b) and (d). These results apparently indicates that the plateau is alleviated by multiple output units.

We also examined the cases with Adam optimizer, bias terms and dropout regularization. All these results are consistent with the idea that multiple outputs mitigate the

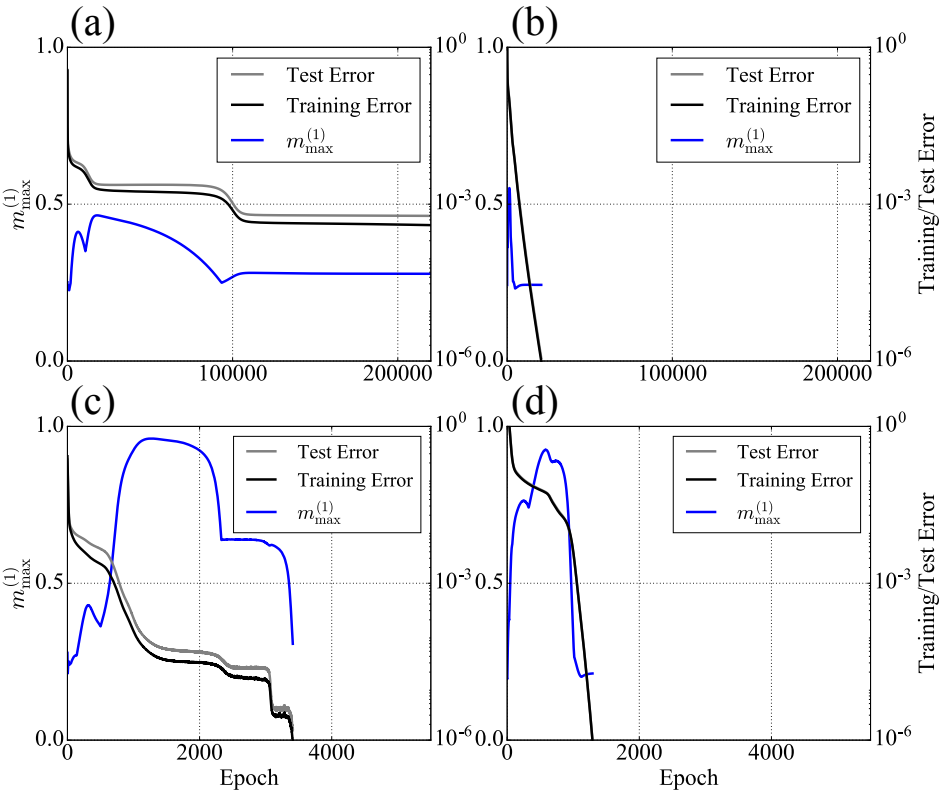

Figure 3.5: Simulation results of time course of training loss (black line) and test loss (gray line), depending on number of output units and choice of optimizers. Time course of student's first layer's maximum overlap $m^{(1)}_{\max} := \max_{i<j} |Q_{ij}|/\sqrt{Q_{ii}Q_{jj}}$ (blue line) also shown. This maximum overlap indicates how student network is close to singular region. (a)(b) stochastic gradient descent (learning rate: 0.01), (c)(d) Adam optimizer. (a)(c) networks with 1 output unit, (b)(d) networks with 10 output units. Mini-batch size: 1000. Activation function: $g(x) = \tanh(x)$.

plateaus due to the singular regions (see the appendix).

3.8 Conclusion

We analyzed the learning dynamics of two-layer networks that have multiple output units by means of statistical mechanical formulation. By defining order parameters, deriving the differential equations they follow, and solving said equations, we clarified experimentally and theoretically that multiple-output networks are less likely to be trapped in plateaus because of singularity than single-output networks are.

Through this paper, we suggest reconsidering the established idea that singular structures cause plateaus and they interrupt learning. However, more general cases, such as cases with deeper neural networks, have yet to be researched.

Acknowledgments This work was supported by JSPS KAKENHI Grant-in-Aid for Scientific Research(A) (No. 18H04106).

3.A Proof of continued orthogonality during learning

This section gives the proof of the Proposition in the main text:

Proposition. Assume $K = M = 2$ and $O \geq 2$, and consider the situation in which $N \to \infty$. Assume that the activation g is an odd function and that the right sides of the differential equations are Lipschitz continuous in the vicinity of the solution trajectory (Q, R, D, E) during learning. If both the teacher and student network have a pair of orthogonal column vectors in its second layer's weight matrix, that is, $\boldsymbol{v}_1 \perp \boldsymbol{v}_2$ and $\boldsymbol{w}_1 \perp \boldsymbol{w}_2$ hold at the initial state, then the orthogonality $\boldsymbol{w}_1 \perp \boldsymbol{w}_2$ holds at any time during learning. ■

Proof. We prove the following claim: suppose $K = M = 2$ and $O \geq 2$, and the activation g is an odd function, then the differential equations (4.9) and (4.5) imply that

$$\begin{aligned} &\text{if} \quad Q \propto I, \quad R + R^T \propto I, \quad D \propto I, \quad E + E^T \propto I, \quad T \propto I, \quad F \propto I \quad (*) \\ &\text{then} \quad \dot{Q} \propto I, \quad \dot{R} + \dot{R}^T \propto I, \quad \dot{D} \propto I, \quad \dot{E} + \dot{E}^T \propto I, \end{aligned} \tag{3.21}$$

where I denotes the 2×2 identity matrix, and $X \propto I$ means that there exists $c \in \mathbb{R}$ such that $X = cI$.

Proving this claim (3.21) is sufficient for the proof of the proposition. Suppose that $\hat{\Theta}(\alpha) = (\hat{Q}(\alpha), \hat{R}(\alpha), \hat{D}(\alpha), \hat{E}(\alpha))$ is one solution of the differential equation. What we have to show is that the solution always lies in the subspace

$$S := \{(Q, R, D, E) \mid Q \propto I,\ R + R^T \propto I,\ D \propto I \text{ and } E + E^T \propto I\}. \tag{3.22}$$

If we denote by $P\hat{\Theta}$ the vector from $\hat{\Theta}$ to the foot of its perpendicular to S (note that this mapping is linear), what we have to show is

$$\boldsymbol{f}(\alpha) := P\hat{\Theta}(\alpha) = \boldsymbol{0} \tag{3.23}$$

for all time α. We have

$$\boldsymbol{f}(0) = \boldsymbol{0} \tag{3.24}$$

by substituting the initial condition (3.13) and (3.20), and we have

$$\begin{aligned} \boldsymbol{f}(\alpha) = \boldsymbol{0} \quad &\Longrightarrow \quad P\frac{d}{d\alpha}\hat{\Theta}(\alpha) = \boldsymbol{0} \quad \text{(from claim (3.21))} \\ &\Longrightarrow \quad \frac{d\boldsymbol{f}}{d\alpha}(\alpha) = P\frac{d}{d\alpha}\hat{\Theta}(\alpha) = \boldsymbol{0} \end{aligned} \tag{3.25}$$

for given time α. These imply that $\boldsymbol{f}(\alpha) \equiv \boldsymbol{0}$ is one solution of the differential equation for $\boldsymbol{f}$. Lipschitz continuity ensures that the uniqueness of the solution of the differential equation, therefore we have $\boldsymbol{f}(\alpha) \equiv \boldsymbol{0}$ for all α.

To prove the claim (3.21), we first show the following lemma.

Lemma 1. If the condition (*) in the claim (3.21) holds, the following relations hold:

$$\begin{aligned} I_2(a_{i_1}, b_{i_2}) &= (-1)^{\delta(i_1=1)\delta(i_2=1)} I_2(a_{j_1}, b_{j_2}), \\ I_3(a_{i_1}, b_{i_2}, c_{i_3}) &= (-1)^{\delta(i_2=1)\delta(i_3=1)} I_3(a_{j_1}, b_{j_2}, c_{j_3}), \\ I_4(a_{i_1}, b_{i_2}, c_{i_3}, d_{i_4}) &= (-1)^{\delta(i_3=1)\delta(i_4=1)} I_4(a_{j_1}, b_{j_2}, c_{j_3}, d_{j_4}), \end{aligned} \tag{3.26}$$

where each of a, b, c and d is either x or y, and $\{i_e, j_e\} = \{1, 2\}$ for $e = 1, 2, 3, 4$.

Proof of Lemma 1. If the condition (*) holds, the covariance matrix of (x_1, x_2, y_1, y_2) is given by

$$\Sigma = \begin{pmatrix} Q_{11} & 0 & R_{11} & R_{12} \\ 0 & Q_{11} & -R_{12} & R_{11} \\ R_{11} & -R_{12} & T_{11} & 0 \\ R_{12} & R_{11} & 0 & T_{11} \end{pmatrix}. \tag{3.27}$$

This matrix Σ has the following property:

$$\mathcal{N}(z_1, z_2, z_3, z_4 \mid 0, \Sigma) = \mathcal{N}(-z_2, z_1, -z_4, z_3 \mid 0, \Sigma). \tag{3.28}$$

Therefore, for arbitrary functions f,

$$\begin{aligned} \langle f(x_1, x_2, y_1, y_2) \rangle &= \int f(x_1, x_2, y_1, y_2) \mathcal{N}(x_1, x_2, y_1, y_2 \mid 0, \Sigma)\ dx_1 dx_2 dy_1 dy_2 \\ &= \int f(-x_2, x_1, -y_2, y_1) \mathcal{N}(-x_2, x_1, -y_2, y_1 \mid 0, \Sigma)\ dx_1 dx_2 dy_1 dy_2 \\ &= \int f(-x_2, x_1, -y_2, y_1) \mathcal{N}(x_1, x_2, y_1, y_2 \mid 0, \Sigma)\ dx_1 dx_2 dy_1 dy_2 \\ &= \langle f(-x_2, x_1, -y_2, y_1) \rangle, \end{aligned} \tag{3.29}$$

and since g is an odd function,

$$\begin{aligned} I_2(a_{i_1}, b_{i_2}) &= \langle g(a_{i_1}) g(b_{i_2}) \rangle \\ &= \langle g((-1)^{\delta(i_1=1)} a_{j_1}) g((-1)^{\delta(i_2=1)} b_{j_2}) \rangle \\ &= (-1)^{\delta(i_1=1)\delta(i_2=1)} \langle g(a_{j_1}) g(b_{j_2}) \rangle \\ &= (-1)^{\delta(i_1=1)\delta(i_2=1)} I_2(a_{j_1}, b_{j_2}), \\ I_3(a_{i_1}, b_{i_2}, c_{i_3}) &= \langle g'(a_{i_1}) b_{i_2} g(c_{i_3}) \rangle \\ &= \langle g'((-1)^{\delta(i_1=1)} a_{i_1}) (-1)^{\delta(i_2=1)} b_{i_2} g((-1)^{\delta(i_3=1)} c_{i_3}) \rangle \\ &= (-1)^{\delta(i_2=1)\delta(i_3=1)} \langle g'(a_{j_1}) g(b_{j_2}) \rangle \\ &= (-1)^{\delta(i_2=1)\delta(i_3=1)} I_3(a_{j_1}, b_{j_2}, c_{j_3}), \\ I_4(a_{i_1}, b_{i_2}, c_{i_3}, d_{i_4}) &= \langle g'(a_{i_1}) g'(b_{i_2}) g(c_{i_3}) g(d_{i_4}) \rangle \\ &= \langle g'((-1)^{\delta(i_1=1)} a_{j_1}) g'((-1)^{\delta(i_2=1)} b_{j_2}) \\ &\qquad \cdot g((-1)^{\delta(i_3=1)} c_{j_3}) g((-1)^{\delta(i_4=1)} d_{j_4}) \rangle \\ &= (-1)^{\delta(i_3=1)\delta(i_4=1)} \langle g'(a_{j_1}) g'(b_{j_2}) g(c_{j_3}) g(d_{j_4}) \rangle \\ &= (-1)^{\delta(i_3=1)\delta(i_4=1)} I_4(a_{j_1}, b_{j_2}, c_{j_3}, d_{j_4}), \end{aligned} \tag{3.30}$$

which is the statement of the lemma. ■

Lemma 2. $I_4(z_1, z_2, z_3, z_4) = I_4(z_2, z_1, z_3, z_4) = I_4(z_1, z_2, z_4, z_3)$.

Proof of Lemma 2. The proof is clear from the definition of I_4. ■

Proof of Proposition. Since the matrices Q, T, D, and F are symmetric, what we have to show is

$$\begin{aligned} \dot{Q}_{11} &= \dot{Q}_{22}, \quad \dot{Q}_{12} = 0, \\ \dot{R}_{11} &= \dot{R}_{22}, \quad \dot{R}_{12} + \dot{R}_{21} = 0, \\ \dot{D}_{11} &= \dot{D}_{22}, \quad \dot{D}_{12} = 0, \\ \dot{E}_{11} &= \dot{E}_{22}, \quad \dot{E}_{12} + \dot{E}_{21} = 0. \end{aligned} \tag{3.31}$$

First, for $\dot{Q}$, we have

$$\begin{aligned} N\dot{Q}_{11} &= 2\eta \left[E_{11}I_3(x_1,x_1,y_1) + E_{12}I_3(x_1,x_1,y_2) - D_{11}I_3(x_1,x_1,x_1)\right] \\ &\quad + \eta^2 \left[D_{11}^2 I_4(x_1,x_1,x_1,x_1) + E_{11}^2 I_4(x_1,x_1,y_1,y_1)\right. \\ &\qquad + E_{11}E_{12}[I_4(x_1,x_1,y_1,y_2) + I_4(x_2,x_2,y_2,y_1)] + E_{12}^2 I_4(x_1,x_1,y_2,y_2) \\ &\qquad - D_{11}E_{11}[I_4(x_1,x_1,x_1,y_1) + I_4(x_1,x_1,y_1,x_1)] \\ &\qquad \left. - D_{11}E_{12}[I_4(x_1,x_1,x_1,y_2) + I_4(x_1,x_1,y_2,x_1)]\right], \\ N\dot{Q}_{22} &= 2\eta \left[E_{11}I_3(x_2,x_2,y_2) - E_{12}I_3(x_2,x_2,y_1) - D_{11}I_3(x_2,x_2,x_2)\right] \\ &\quad + \eta^2 \left[D_{11}^2 I_4(x_2,x_2,x_2,x_2) + E_{11}^2 I_4(x_2,x_2,y_2,y_2)\right. \\ &\qquad - E_{11}E_{12}[I_4(x_2,x_2,y_2,y_1) + I_4(x_1,x_1,y_1,y_2)] + E_{12}^2 I_4(x_2,x_2,y_1,y_1) \\ &\qquad - D_{11}E_{11}[I_4(x_2,x_2,x_2,y_2) + I_4(x_2,x_2,y_2,x_2)] \\ &\qquad \left. + D_{11}E_{12}[I_4(x_2,x_2,x_2,y_1) + I_4(x_2,x_2,y_1,x_2)]\right], \\ N\dot{Q}_{12} &= \eta \left[E_{11}[I_3(x_1,x_2,y_1) + I_3(x_2,x_1,y_2)] + E_{12}[I_3(x_1,x_2,y_2) - I_3(x_2,x_1,y_1)]\right. \\ &\qquad \left. - D_{11}[I_3(x_1,x_2,x_1) + I_3(x_2,x_1,x_2)]\right] \\ &\quad + \eta^2 \left[D_{11}^2 I_4(x_1,x_2,x_1,x_2) + E_{11}^2 I_4(x_1,x_2,y_1,y_2)\right. \\ &\qquad + E_{11}E_{12}[-I_4(x_1,x_2,y_1,y_1) + I_4(x_1,x_2,y_2,y_2)] - E_{12}^2 I_4(x_1,x_2,y_2,y_1) \\ &\qquad - D_{11}E_{11}[I_4(x_1,x_2,x_1,y_2) + I_4(x_1,x_2,y_1,x_2)] \\ &\qquad \left. - D_{11}E_{12}[-I_4(x_1,x_2,x_1,y_1) + I_4(x_1,x_2,y_2,x_2)]\right] \end{aligned} \tag{3.32}$$

from the equation (4.9) and the assumption of the proposition. Applying the lemmas to each I-term, we can confirm $\dot{Q}_{11} = \dot{Q}_{22}$ and $\dot{Q}_{12} = 0$.

With respect to R, we find

$$\begin{aligned} N\dot{R}_{11} &= \eta \left[E_{11}I_3(x_1,y_1,y_1) + E_{12}I_3(x_1,y_1,y_2) - D_{11}I_3(x_1,y_1,x_1)\right], \\ N\dot{R}_{22} &= \eta \left[E_{11}I_3(x_2,y_2,y_2) - E_{12}I_3(x_2,y_2,y_1) - D_{11}I_3(x_2,y_2,x_2)\right], \\ N\dot{R}_{12} &= \eta \left[E_{11}I_3(x_1,y_2,y_1) + E_{12}I_3(x_1,y_2,y_2) - D_{11}I_3(x_1,y_2,x_1)\right], \\ N\dot{R}_{21} &= \eta \left[E_{11}I_3(x_2,y_1,y_2) - E_{12}I_3(x_2,y_1,y_1) - D_{11}I_3(x_2,y_1,x_2)\right], \end{aligned} \tag{3.33}$$

and by applying the lemmas to the I-terms, $\dot{R}_{11} = \dot{R}_{22}$ and $\dot{R}_{12} + \dot{R}_{21} = 0$ are implied.

In the same way, we have

$$\begin{aligned} N\dot{D}_{11} &= 2\eta \left[E_{11}I_2(x_1, y_1) + E_{12}I_2(x_1, y_2) - D_{11}I_2(x_1, x_1)\right], \\ N\dot{D}_{22} &= 2\eta \left[E_{11}I_2(x_2, y_2) - E_{12}I_2(x_2, y_1) - D_{11}I_2(x_2, x_2)\right], \\ N\dot{D}_{12} &= \eta[E_{11}[I_2(x_2, y_1) + I_2(x_1, y_2)] + E_{12}[I_2(x_2, y_2) - I_2(x_1, y_1)] \\ &\quad - D_{11}[I_2(x_2, x_1) + I_2(x_1, x_2)]], \end{aligned} \tag{3.34}$$

and

$$\begin{aligned} N\dot{E}_{11} &= \eta \left[F_{11}I_2(x_1, y_1) - E_{11}I_2(x_1, x_1) + E_{12}I_2(x_1, x_2)\right], \\ N\dot{E}_{22} &= \eta \left[F_{11}I_2(x_2, y_2) - E_{11}I_2(x_2, x_2) - E_{12}I_2(x_2, x_1)\right], \\ N\dot{E}_{12} &= \eta \left[F_{11}I_2(x_1, y_2) - E_{11}I_2(x_1, x_2) - E_{12}I_2(x_1, x_1)\right], \\ N\dot{E}_{21} &= \eta \left[F_{11}I_2(x_2, y_1) - E_{11}I_2(x_2, x_1) + E_{12}I_2(x_2, x_2)\right], \end{aligned} \tag{3.35}$$

and by using the lemmas we can confirm $\dot{D}_{11} = \dot{D}_{22}$, $\dot{D}_{12} = 0$, $\dot{E}_{11} = \dot{E}_{22}$, and $\dot{E}_{12} + \dot{E}_{21} = 0$, the statements of the proposition. ■

3.B Experimental results under more practical settings

We examined learning dynamics under more practical situations than our two-hidden-unit model, including greater number of hidden units, use of bias terms, mini-batch learning, adaptive learning coefficient and regularization techniques, as we showed the typical results in the Figure 3.5 in the main text. In this section, we describe detailed results of our experiment.

3.B.1 Ten hidden units

In the Figure 3.5(a)(b) in the main text, we compared 100-10-1 case (i.e. case with 100 input units, 10 hidden units and 1 output unit) and 100-10-10 case. Here we describe the result in detail. Each of ten subfigures in Figure 3.6 below represents a learning dynamics obtained under the same setting as Figure 3.5(a) in the main text, except for initial weights. Similarly, each of ten subfigures in Figure 3.7 below represents that obtained under Figure 3.5(b) settings.

In Figure 3.6, plateaus and approaching the singular region are observed regardless of the initial weights, and no convergence to the global minimum is achieved during 600000 epochs in all trials (here we define convergence by achieving the loss $< 10^{-6}$). In contrast, in Figure 3.7 there is no plateau observed in large fraction of trials, and all trials succeed in converging to the global minimum within 30000 epochs.

3.B.2 Adam optimizer

We examined whether adaptive learning rate mitigates the plateaus in Figure 3.5(c)(d) in the main text. Figure 3.8 and Figure 3.9 below shows ten results obtained under the Figure 3.5(c) and Figure 3.5(d) situations, respectively. These figures indicate

that adaptive gradient descent methods does not change our conclusion; larger number of output mitigates plateaus and speeds up learning.

3.B.3 Bias terms

We also investigated the effect of the bias terms. That is, we modified the network model to $\boldsymbol{s} = \sum_{i=1}^{K} \boldsymbol{w}_i g(\boldsymbol{J}_i \cdot \boldsymbol{\xi} + b_i) + \boldsymbol{c} \in \mathbb{R}^O$. The results in Figure 3.10 indicates again that increasing output units mitigates plateaus under the presence of bias terms.

3.B.4 Dropout regularization

Furthermore we inspected the effect of the dropout regularization. When using dropout or other regularization techniques such as L2 penalty term, the student network with any weight values cannot achieve zero training error, and the global minima in the training error landscape is not trivial. This makes difficult to compare the converging speed of learning. In Figure 3.12, every trial ends up with close-to-1 maximum overlaps, implying that the global minima are in the singular region unlike no-dropout case, or all cases are trapped in very long plateaus (longer than 600000 epochs). In Figure 3.13, the approach to the singular region are mitigated.

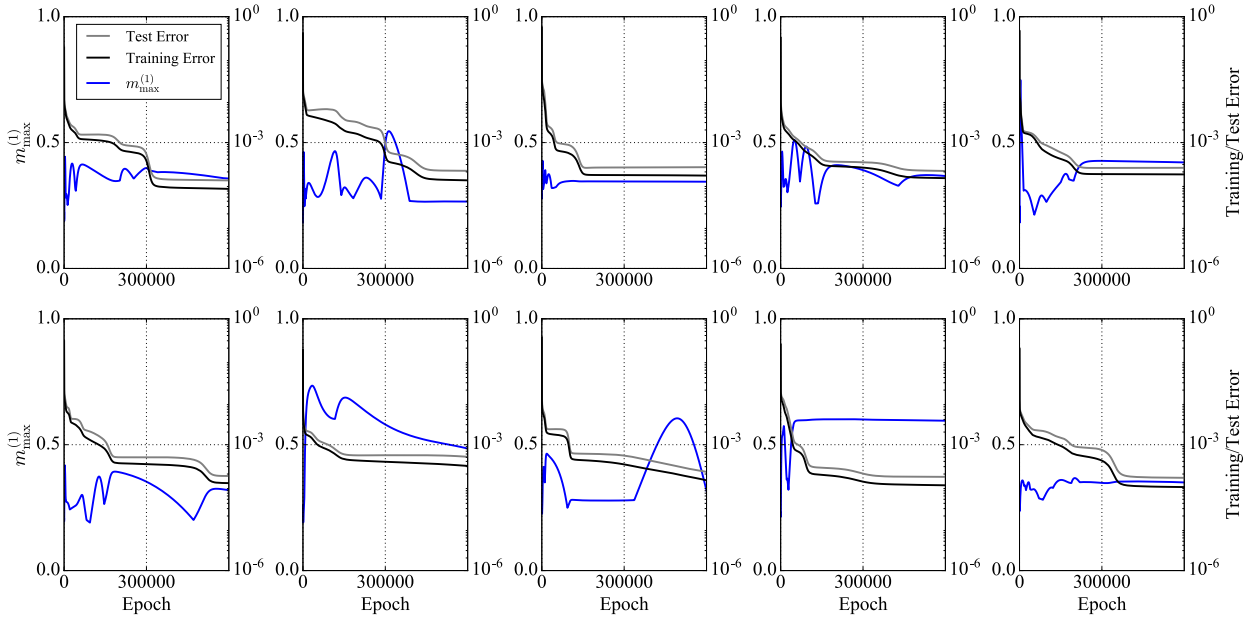

Figure 3.6: Simulation results of time course of training error (black line) and test error (gray line). Time course of student's first layer's maximum overlap $m^{(1)}_{\max} := \max_{i<j} |Q_{ij}|/\sqrt{Q_{ii}Q_{jj}}$ (blue line) also shown. Experimental setting is same as Figure 3.5(a) in the main text; that is, network size is 100 - 10 - 1.

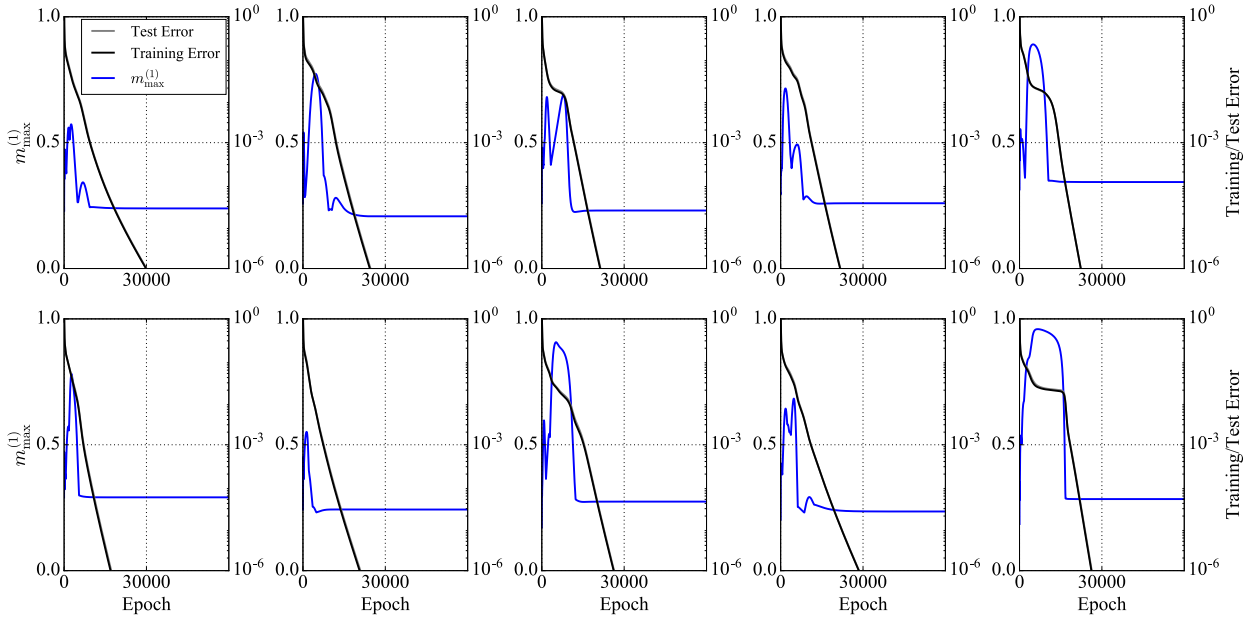

Figure 3.7: Simulation results of time course of training error (black line), test error (gray line), and student's first layer's maximum overlap $m^{(1)}_{\max} := \max_{i<j} |Q_{ij}|/\sqrt{Q_{ii}Q_{jj}}$ (blue line). Experimental setting is same as Figure 3.5(b); that is, network size is 100 - 10 - 10.

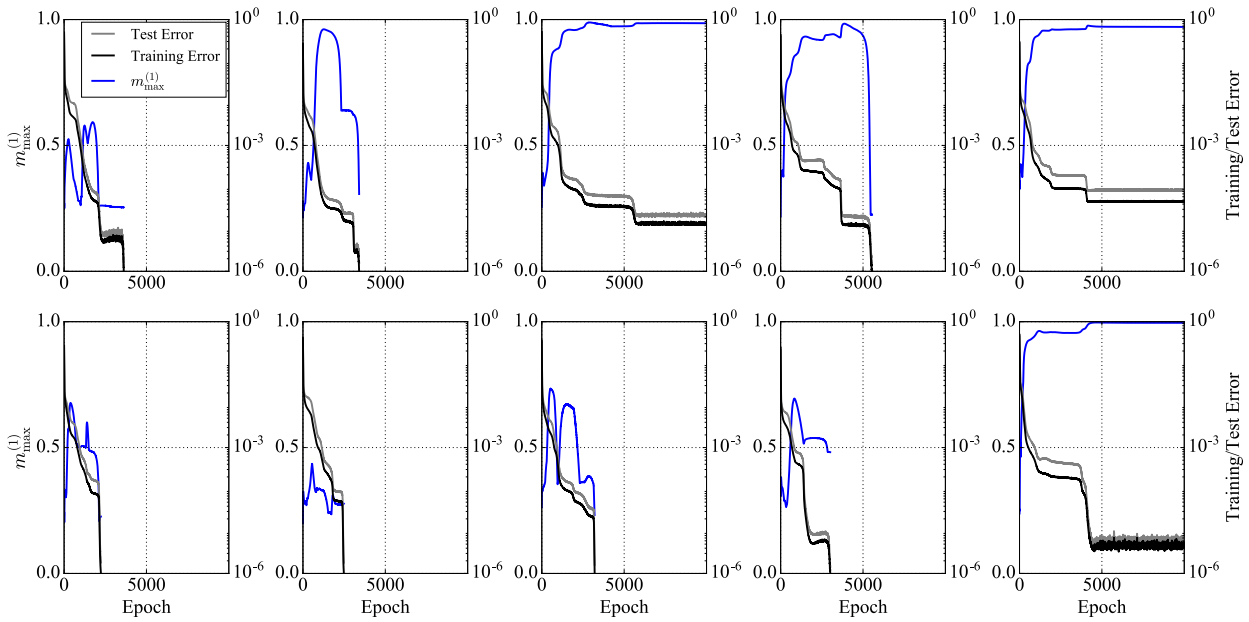

Figure 3.8: Simulation results of time course of training error (black line), test error (gray line), and student's first layer's maximum overlap $m^{(1)}_{\max} := \max_{i<j} |Q_{ij}|/\sqrt{Q_{ii}Q_{jj}}$ (blue line). Experimental setting is same as Figure 3.5(c), i.e. with Adam optimizer and one output unit.

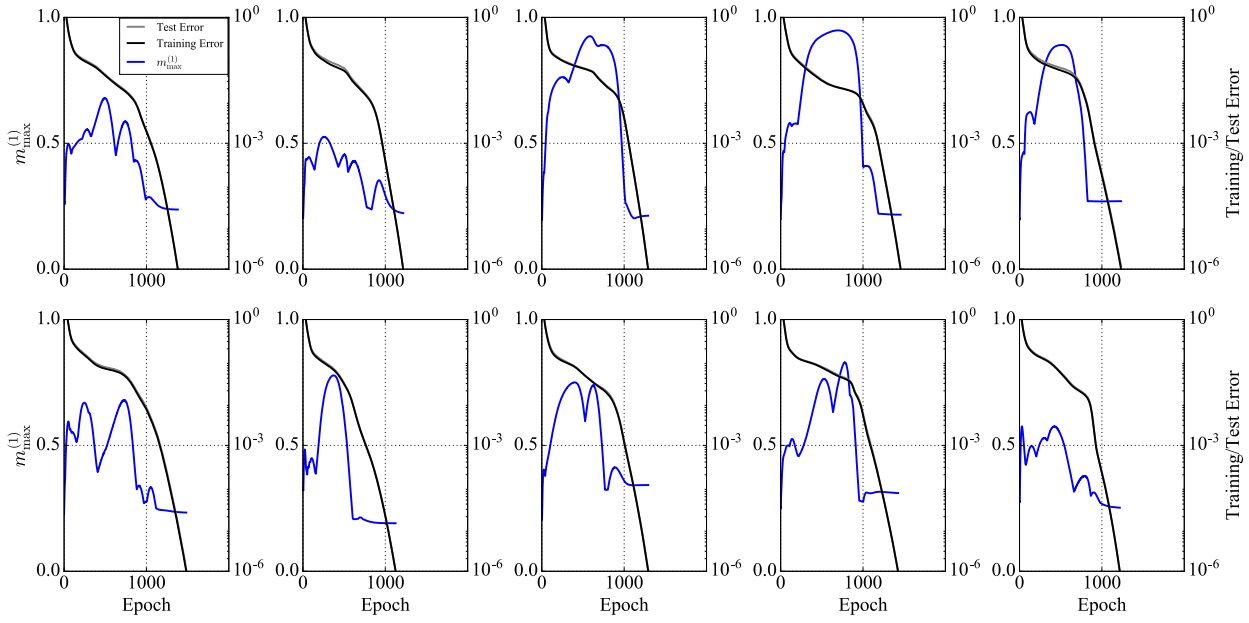

Figure 3.9: Simulation results of time course of training error (black line), test error (gray line), and student's first layer's maximum overlap $m^{(1)}_{\max} := \max_{i<j} |Q_{ij}|/\sqrt{Q_{ii}Q_{jj}}$ (blue line). Experimental setting is same as Figure 3.5(d), i.e. with Adam optimizer and ten output unit.

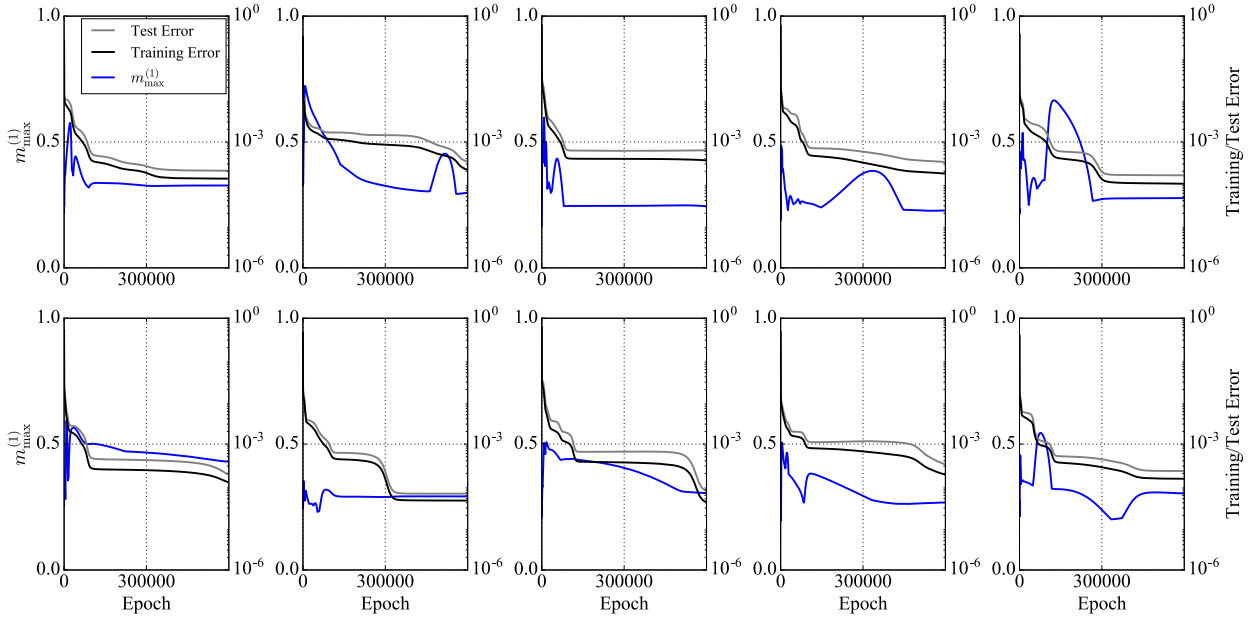

Figure 3.10: Simulation results of time course of training error (black line), test error (gray line), and student's first layer's maximum overlap $m^{(1)}_{\max} := \max_{i<j} |Q_{ij}|/\sqrt{Q_{ii}Q_{jj}}$ (blue line), with use of bias terms. Network size: 100 - 10 - 1.

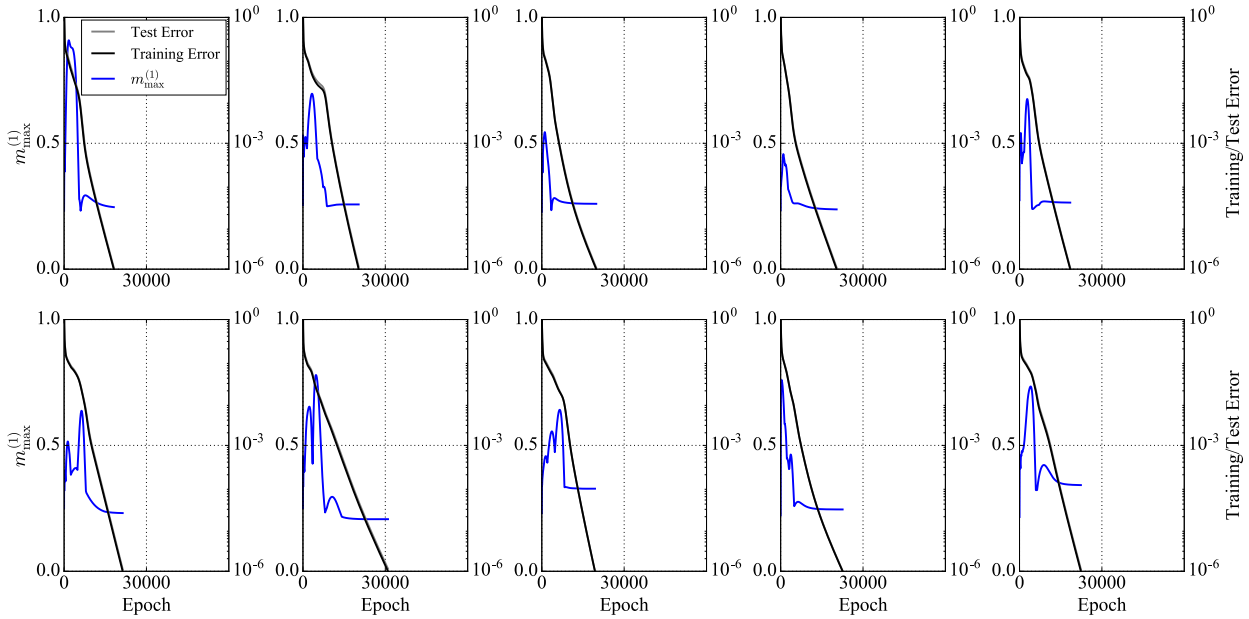

Figure 3.11: Simulation results of time course of training error (black line), test error (gray line), and student's first layer's maximum overlap $m^{(1)}_{\max} := \max_{i<j} |Q_{ij}|/\sqrt{Q_{ii}Q_{jj}}$ (blue line), with use of bias terms. Network size: 100 - 10 - 10.

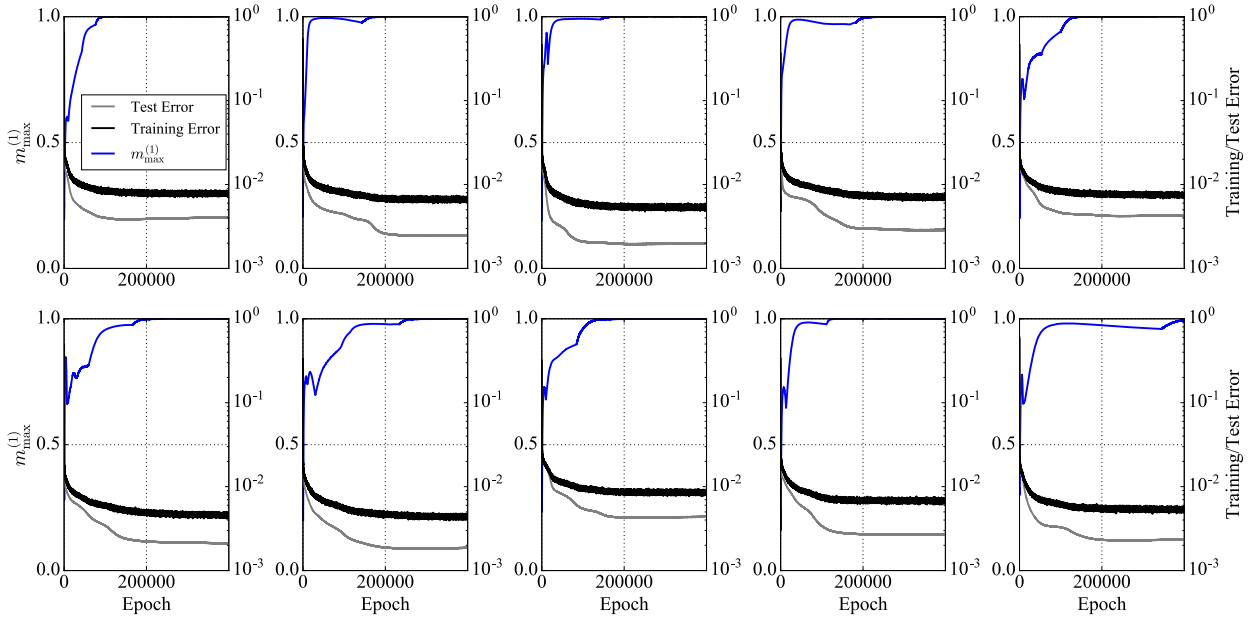

Figure 3.12: Simulation results of time course of training error (black line), test error (gray line), and student's first layer's maximum overlap $m^{(1)}_{\max} := \max_{i<j} |Q_{ij}|/\sqrt{Q_{ii}Q_{jj}}$ (blue line), with use of dropout. Dropout ratio: 1/10. Network size: 100 - 10 - 1.

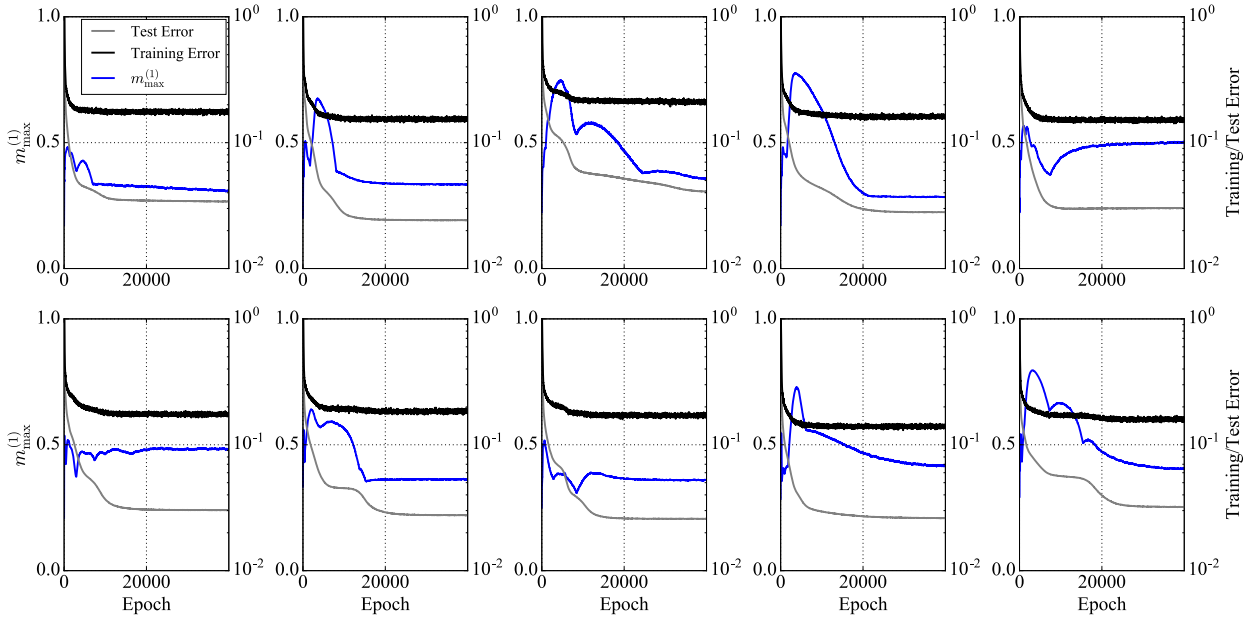

Figure 3.13: Simulation results of time course of training error (black line), test error (gray line), and student's first layer's maximum overlap $m^{(1)}_{\max} := \max_{i<j} |Q_{ij}|/\sqrt{Q_{ii}Q_{jj}}$ (blue line), with use of dropout. Dropout ratio: 1/10. Network size: 100 - 10 - 10.

Chapter 4

Data-Dependence of Plateau Phenomenon in Learning with Neural Network — Statistical Mechanical Analysis *

4.1 Introduction

4.1.1 Plateau Phenomenon

Deep learning, and neural network as its essential component, has come to be applied to various fields. However, these still remain unclear in various points theoretically. The plateau phenomenon is one of them. In the learning process of neural networks, their weight parameters are updated iteratively so that the loss decreases. However, in some settings the loss does not decrease simply, but its decreasing speed slows down significantly partway through learning, and then it speeds up again after a long period of time. This is called as "plateau phenomenon". Since 1990s, this phenomena have been reported to occur in various practical learning situations (see Figure 4.1 (a) and [1, 6]) . As a fundamental cause of this phenomenon, it has been pointed out by a number of researchers that the intrinsic symmetry of neural network models brings singularity to the metric in the parameter space which then gives rise to special attractors whose regions of attraction have nonzero measure, called as Milnor attractor (defined by [15]; see also Figure 5 in [6] for a schematic diagram of the attractor).

4.1.2 Who moved the plateau phenomenon?

However, the plateau phenomenon seldom occurs in recent practical use of neural networks (see Figure 4.1 (b) for example).

*This work has been presented as [16].

In this research, we rethink the plateau phenomenon, and discuss which situations are likely to cause the phenomenon. First we introduce the student-teacher model of two-layered networks as an ideal system. Next, we reduce the learning dynamics of the student-teacher model to a small-dimensional order parameter system by using statistical mechanical formulation, under the assumption that the input dimension is sufficiently large. Through analyzing the order parameter system, we can discuss how the macroscopic learning dynamics depends on the statistics of input data. Our main contribution is the following:

- Under the statistical mechanical formulation of learning in the two-layered perceptron, we showed that macroscopic equations can be derived even when the statistical properties of the input are generalized. In other words, we extended the result of [9] and [32].
- By analyzing the macroscopic system we derived, we showed that the dynamics of learning depends only on the eigenvalue distribution of the covariance matrix of the input data.
- We clarified the relationship between the input data statistics and plateau phenomenon. In particular, it is shown that the data whose covariance matrix has small and dispersed eigenvalues tend to make the phenomenon inconspicuous, by numerically analyzing the macroscopic system.

4.1.3 Related works

The statistical mechanical approach used in this research is firstly developed by [9]. The method reduces high-dimensional learning dynamics of nonlinear neural networks to low-dimensional system of order parameters. They derived the macroscopic behavior of learning dynamics in two-layered soft-committee machine and by analyzing it they point out the existence of plateau phenomenon. Nowadays the statistical mechanical method is applied to analyze recent techniques ([45], [12], [46] and [47]), and generalization performance in over-parameterized setting ([48]) and environment with conceptual drift ([49]). However, it is unknown that how the property of input dataset itself can affect the learning dynamics, including plateaus. Although previous works assume the standard normal distribution for input data, it is not realistic from the viewpoint of the manifold hypothesis, according to which practical data presented in high dimensional ambient spaces are expected to concentrate in the vicinity of a manifold whose dimension is much lower than that of the ambient space[17].

Plateau phenomenon and singularity in loss landscape as its main cause have been studied by [6], [50], [51] and [52]. On the other hand, recent several works suggest that plateau and singularity can be mitigated in some settings. [53] shows that skip connections eliminate the singularity. Another work by [14] points out that output dimensionality affects the plateau phenomenon, in that multiple output units alleviate the plateau phenomenon. However, the number of output elements does not fully determine the presence or absence of plateaus, nor does the use of skip connections. The statistical property of data just can

affect the learning dynamics dramatically; for example, see Figure 4.2 for learning curves with using different datasets and same network architecture. We focus on what kind of statistical property of the data brings plateau phenomenon.

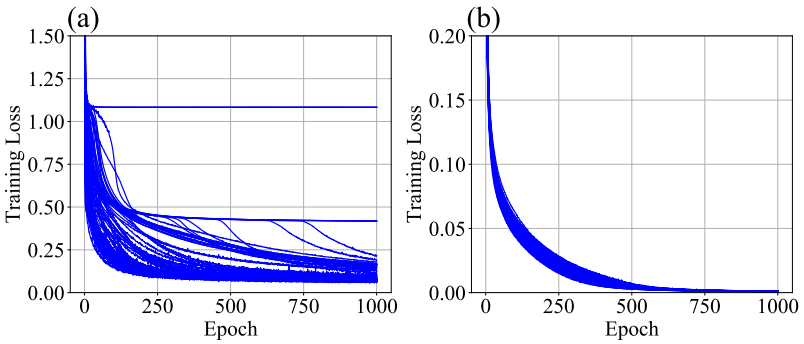

Figure 4.1: (a) Training loss curves when two-layer perceptron with 4-4-3 units and ReLU activation learns IRIS dataset. (b) Training loss curve when two-layer perceptron with 784-20-10 units and ReLU activation learns MNIST dataset. For both (a) and (b), results of 100 trials with random initialization are overlaid. Minibatch size of 10 and vanilla SGD (learning rate: 0.01) are used.

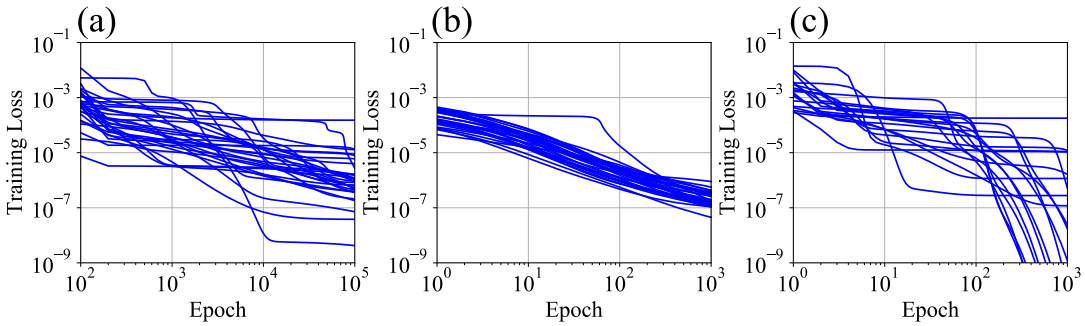

Figure 4.2: Loss curves yielded by student-teacher learning with two-layer perceptron which has 2 hidden units, 1 output unit and sigmoid activation, and with (a) IRIS dataset, (b) MNIST dataset, (c) a dataset in $\mathbb{R}^{60000\times 784}$ drawn from standard normal distribution, as input distribution $p(\boldsymbol{\xi})$. In every subfigure, results for 20 trials with random initialization are overlaid. Vanilla SGD (learning rate: (a)(b) 0.005, (c) 0.001) and minibatch size of 1 are used for all three settings.

4.2 Formulation

4.2.1 Student-Teacher Model

We consider a two-layer perceptron which has N input units, K hidden units and 1 output unit. We denote the input to the network by $\boldsymbol{\xi} \in \mathbb{R}^N$. Then the output can be

written as $s = \sum_{i=1}^{K} w_i g(\boldsymbol{J}_i \cdot \boldsymbol{\xi}) \in \mathbb{R}$, where g is an activation function.

We consider the situation that the network learns data generated by another network, called "teacher network", which has fixed weights. Specifically, we consider two-layer perceptron that outputs $t = \sum_{n=1}^{M} v_n g(\boldsymbol{B}_n \cdot \boldsymbol{\xi}) \in \mathbb{R}$ for input $\boldsymbol{\xi}$ as the teacher network. The generated data $(\boldsymbol{\xi}, t)$ is then fed to the student network stated above and learned by it in the on-line manner (see Figure 4.3). We assume that the input $\boldsymbol{\xi}$ is drawn from some distribution $p(\boldsymbol{\xi})$ every time independently. We adopt vanilla stochastic gradient descent (SGD) algorithm for learning. We assume the squared loss function $\varepsilon = \frac{1}{2}(s-t)^2$, which is most commonly used for regression.

4.2.2 Statistical Mechanical Formulation

In order to capture the learning dynamics of nonlinear neural networks described in the previous subsection macroscopically, we introduce the statistical mechanical formulation in this subsection.

Let $x_i := \boldsymbol{J}_i \cdot \boldsymbol{\xi}$ $(1 \leq i \leq K)$ and $y_n := \boldsymbol{B}_n \cdot \boldsymbol{\xi}$ $(1 \leq n \leq M)$. Then

$$(x_1, \ldots, x_K, y_1, \ldots, y_M) \sim \mathcal{N}\left(0, [\boldsymbol{J}_1, \ldots, \boldsymbol{J}_K, \boldsymbol{B}_1, \ldots, \boldsymbol{B}_M]^T \Sigma [\boldsymbol{J}_1, \ldots, \boldsymbol{J}_K, \boldsymbol{B}_1, \ldots, \boldsymbol{B}_M]\right)$$

holds with $N \to \infty$ by generalized central limit theorem, provided that the input distribution $p(\boldsymbol{\xi})$ has zero mean and finite covariance matrix Σ.*

Next, let us introduce order parameters as following: $Q_{ij} := \boldsymbol{J}_i^T \Sigma \boldsymbol{J}_j = \langle x_i x_j \rangle$, $R_{in} := \boldsymbol{J}_i^T \Sigma \boldsymbol{B}_n = \langle x_i y_n \rangle$, $T_{nm} := \boldsymbol{B}_n^T \Sigma \boldsymbol{B}_m = \langle y_n y_m \rangle$ and $D_{ij} := w_i w_j$, $E_{in} := w_i v_n$, $F_{nm} := v_n v_m$. Then

$$(x_1, \ldots, x_K, y_1, \ldots, y_M) \sim \mathcal{N}(\mathbf{0}, \begin{pmatrix} Q & R \\ R^T & T \end{pmatrix}).$$

The parameters Q_{ij}, R_{in}, T_{nm}, D_{ij}, E_{in}, and F_{nm} introduced above capture the state of the system macroscopically; therefore they are called as "order parameters." The first three represent the state of the first layers of the two networks (student and teacher), and the latter three represent their second layers' state. Q describes the statistics of the student's first layer and T represents that of the teacher's first layer. R is related to similarity between the student and teacher's first layer. D, E, F is the second layers' counterpart of Q, R, T. The values of Q_{ij}, R_{in}, D_{ij}, and E_{in} change during learning; their dynamics are what to be determined, from the dynamics of microscopic variables, i.e. connection weights. In contrast, T_{nm} and F_{nm} are constant during learning.

*Actually we need more conditions for the input distribution. See the appendix at the end of this dissertation.

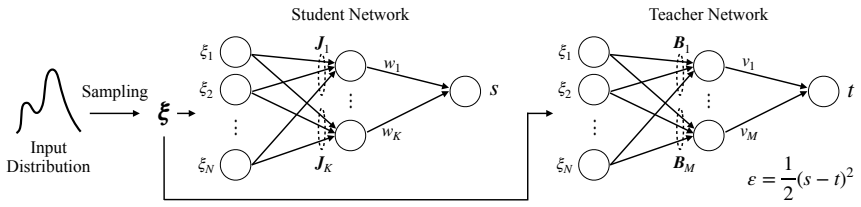

Figure 4.3: Overview of student-teacher model formulation.

Higher-order order parameters

The important difference between our situation and that of [9] is the covariance matrix Σ of the input $\boldsymbol{\xi}$ is not necessarily equal to identity. This makes the matter complicated, since higher-order terms Σ^e $(e = 1, 2, \ldots)$ appear inevitably in the learning dynamics of order parameters. In order to deal with these, here we define some higher-order version of order parameters.

Let us define higher-order order parameters $Q_{ij}^{(e)}$, $R_{in}^{(e)}$ and $T_{nm}^{(e)}$ for $e = 0, 1, 2, \ldots$, as $Q_{ij}^{(e)} := \boldsymbol{J}_i^T \Sigma^e \boldsymbol{J}_j$, $R_{in}^{(e)} := \boldsymbol{J}_i^T \Sigma^e \boldsymbol{B}_n$, and $T_{nm}^{(e)} := \boldsymbol{B}_n^T \Sigma^e \boldsymbol{B}_m$. Note that they are identical to Q_{ij}, R_{in} and T_{nm} in the case of $e = 1$. Also we define higher-order version of x_i and y_n, namely $x_i^{(e)}$ and $y_n^{(e)}$, as $x_i^{(e)} := \boldsymbol{\xi}^T \Sigma^e \boldsymbol{J}_i, y_n^{(e)} := \boldsymbol{\xi}^T \Sigma^e \boldsymbol{B}_n$. Note that $x_i^{(0)} = x_i$ and $y_n^{(0)} = y_n$.

4.3 Derivation of dynamics of order parameters

At each iteration of on-line learning, weights of the student network $\boldsymbol{J}_i$ and w_i are updated with

$$\begin{aligned}
\Delta \boldsymbol{J}_i &= -\frac{\eta}{N} \frac{d\varepsilon}{d\boldsymbol{J}_i} = \frac{\eta}{N} [(\boldsymbol{t} - \boldsymbol{s}) \cdot \boldsymbol{w}_i] g'(x_i) \boldsymbol{\xi} = \frac{\eta}{N} \left[\left(\sum_{n=1}^{M} \boldsymbol{v}_n g(y_n) - \sum_{j=1}^{K} \boldsymbol{w}_j g(x_j) \right) \cdot \boldsymbol{w}_i \right] g'(x_i) \boldsymbol{\xi}, \\
\Delta \boldsymbol{w}_i &= -\frac{\eta}{N} \frac{d\varepsilon}{d\boldsymbol{w}_i} = \frac{\eta}{N} g(x_i)(\boldsymbol{t} - \boldsymbol{s}) = \frac{\eta}{N} g(x_i) \left(\sum_{n=1}^{M} \boldsymbol{v}_n g(y_n) - \sum_{j=1}^{K} \boldsymbol{w}_j g(x_j) \right),
\end{aligned} \tag{4.1}$$

in which we set the learning rate as η/N, so that our macroscopic system is N-independent.

Then, the order parameters $Q_{ij}^{(e)}$ and $R_{in}^{(e)}$ $(e = 0, 1, 2, \ldots)$ are updated with

$$
\begin{aligned}
\Delta Q_{ij}^{(e)} &= (\boldsymbol{J}_i + \Delta\boldsymbol{J}_i)^T \Sigma^e (\boldsymbol{J}_j + \Delta\boldsymbol{J}_j) - \boldsymbol{J}_i^T \Sigma^e \boldsymbol{J}_j = \boldsymbol{J}_i^T \Sigma^e \Delta\boldsymbol{J}_j + \boldsymbol{J}_j^T \Sigma^e \Delta\boldsymbol{J}_i + \Delta\boldsymbol{J}_i^T \Sigma^e \Delta\boldsymbol{J}_j \\
&= \frac{\eta}{N} \left[\sum_{p=1}^{M} E_{ip} g'(x_i) x_j^{(e)} g(y_p) - \sum_{p=1}^{K} D_{ip} g'(x_i) x_j^{(e)} g(x_p) \right. \\
&\qquad \left. + \sum_{p=1}^{M} E_{jp} g'(x_j) x_i^{(e)} g(y_p) - \sum_{p=1}^{K} D_{jp} g'(x_j) x_i^{(e)} g(x_p) \right] \\
&\quad + \frac{\eta^2}{N^2} \boldsymbol{\xi}^T \Sigma^e \boldsymbol{\xi} \left[\sum_{p,q}^{K,K} D_{ip} D_{jq} g'(x_i) g'(x_j) g(x_p) g(x_q) + \sum_{p,q}^{M,M} E_{ip} E_{jq} g'(x_i) g'(x_j) g(y_p) g(y_q) \right. \\
&\qquad \left. - \sum_{p,q}^{K,M} D_{ip} E_{jq} g'(x_i) g'(x_j) g(x_p) g(y_q) - \sum_{p,q}^{M,K} E_{ip} D_{jq} g'(x_i) g'(x_j) g(y_p) g(x_q) \right], \\
\Delta R_{in}^{(e)} &= (\boldsymbol{J}_i + \Delta\boldsymbol{J}_i)^T \Sigma^e \boldsymbol{B}_n - \boldsymbol{J}_i^T \Sigma^e \boldsymbol{B}_n = \Delta\boldsymbol{J}_i^T \Sigma^e \boldsymbol{B}_n \\
&= \frac{\eta}{N} \left[\sum_{p=1}^{M} E_{ip} g'(x_i) y_n^{(e)} g(y_p) - \sum_{p=1}^{K} D_{ip} g'(x_i) y_n^{(e)} g(x_p) \right].
\end{aligned}
\tag{4.2}
$$

Since

$$
\boldsymbol{\xi}^T \Sigma^e \boldsymbol{\xi} \approx N \mu_{e+1} \qquad \text{where} \quad \mu_d := \frac{1}{N} \sum_{i=1}^{N} \lambda_i^d, \qquad \lambda_1, \ldots, \lambda_N : \text{eigenvalues of } \Sigma
$$

and the right hand sides of the difference equations are $O(N^{-1})$, we can replace these difference equations with differential ones with $N \to \infty$, by taking the expectation over all input vectors $\boldsymbol{\xi}$:

$$
\begin{aligned}
\frac{dQ_{ij}^{(e)}}{d\tilde{\alpha}} &= \eta \left[\sum_{p=1}^{M} E_{ip} I_3(x_i, x_j^{(e)}, y_p) - \sum_{p=1}^{K} D_{ip} I_3(x_i, x_j^{(e)}, x_p) \right. \\
&\qquad \left. + \sum_{p=1}^{M} E_{jp} I_3(x_j, x_i^{(e)}, y_p) - \sum_{p=1}^{K} D_{jp} I_3(x_j, x_i^{(e)}, x_p) \right] \\
&\quad + \eta^2 \mu_{e+1} \left[\sum_{p,q}^{K,K} D_{ip} D_{jq} I_4(x_i, x_j, x_p, x_q) + \sum_{p,q}^{M,M} E_{ip} E_{jq} I_4(x_i, x_j, y_p, y_q) \right. \\
&\qquad \left. - \sum_{p,q}^{K,M} D_{ip} E_{jq} I_4(x_i, x_j, x_p, y_q) - \sum_{p,q}^{M,K} E_{ip} D_{jq} I_4(x_i, x_j, y_p, x_q) \right], \\
\frac{dR_{in}^{(e)}}{d\tilde{\alpha}} &= \eta \left[\sum_{p=1}^{M} E_{ip} I_3(x_i, y_n^{(e)}, y_p) - \sum_{p=1}^{K} D_{ip} I_3(x_i, y_n^{(e)}, x_p) \right]
\end{aligned}
\tag{4.3}
$$

$$
\text{where} \quad I_3(z_1, z_2, z_3) := \langle g'(z_1) z_2 g(z_3) \rangle \quad \text{and} \quad I_4(z_1, z_2, z_3, z_4) := \langle g'(z_1) g'(z_2) g(z_3) g(z_4) \rangle. \tag{4.4}
$$

In these equations, $\tilde{\alpha} := \alpha/N$ represents time (normalized number of steps), and the brackets $\langle \cdot \rangle$ represent the expectation when the input $\boldsymbol{\xi}$ follows the input distribution $p(\boldsymbol{\xi})$.

The differential equations for D and E are obtained in a similar way:

$$\frac{dD_{ij}}{d\tilde{\alpha}} = \eta \left[\sum_{p=1}^{M} E_{ip} I_2(x_j, y_p) - \sum_{p=1}^{K} D_{ip} I_2(x_j, x_p) + \sum_{p=1}^{M} E_{jp} I_2(x_i, y_p) - \sum_{p=1}^{K} D_{jp} I_2(x_i, x_p) \right],$$
$$\frac{dE_{in}}{d\tilde{\alpha}} = \eta \left[\sum_{p=1}^{M} F_{pn} I_2(x_i, y_p) - \sum_{p=1}^{K} E_{pn} I_2(x_i, x_p) \right] \tag{4.5}$$

$$\text{where} \quad I_2(z_1, z_2) := \langle g(z_1) g(z_2) \rangle. \tag{4.6}$$

These differential equations (4.3) and (4.5) govern the macroscopic dynamics of learning. In addition, the generalization loss ε_g, the expectation of loss value $\varepsilon(\boldsymbol{\xi}) = \frac{1}{2}\|\boldsymbol{s} - \boldsymbol{t}\|^2$ over all input vectors $\boldsymbol{\xi}$, is represented as

$$\varepsilon_g = \langle \frac{1}{2}\|\boldsymbol{s} - \boldsymbol{t}\|^2 \rangle = \frac{1}{2} \left[\sum_{p,q}^{M,M} F_{pq} I_2(y_p, y_q) + \sum_{p,q}^{K,K} D_{pq} I_2(x_p, x_q) - 2 \sum_{p,q}^{K,M} E_{pq} I_2(x_p, y_q) \right]. \tag{4.7}$$

4.3.1 Expectation terms

Above we have determined the dynamics of order parameters as (4.3), (4.5) and (4.7). However they have expectation terms $I_2(z_1, z_2)$, $I_3(z_1, z_2^{(e)}, z_3)$ and $I_4(z_1, z_2, z_3, z_4)$, where zs are either x_i or y_n. By studying what distribution $\boldsymbol{z}$ follows, we can show that these expectation terms are dependent only on 1-st and $(e+1)$-th order parameters, namely, $Q^{(1)}, R^{(1)}, T^{(1)}$ and $Q^{(e+1)}, R^{(e+1)}, T^{(e+1)}$; for example,

$$I_3(x_i, x_j^{(e)}, y_p) = \int dz_1 dz_2 dz_3 \; g'(z_1) z_2 g(z_3) \; \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \begin{pmatrix} Q_{ii}^{(1)} & Q_{ij}^{(e+1)} & R_{ip}^{(1)} \\ Q_{ij}^{(e+1)} & * & R_{jp}^{(e+1)} \\ R_{ip}^{(1)} & R_{jp}^{(e+1)} & T_{pp}^{(1)} \end{pmatrix})$$

holds, where $*$ does not influence the value of this expression (see the appendix 4.A for more detailed discussion). Thus, we see the 'speed' of e-th order parameters (i.e. (4.3) and (4.5)) only depends on 1-st and $(e+1)$-th order parameters, and the generalization error ε_g (equation (4.7)) only depends on 1-st order parameters. Therefore, with denoting $(Q^{(e)}, R^{(e)}, T^{(e)})$ by $\Omega^{(e)}$ and (D, E, F) by χ, we can write

$$\frac{d}{d\tilde{\alpha}}\Omega^{(e)} = f^{(e)}(\Omega^{(1)}, \Omega^{(e+1)}, \chi), \qquad \frac{d}{d\tilde{\alpha}}\chi = g(\Omega^{(1)}, \chi), \quad \text{and} \quad \varepsilon_g = h(\Omega^{(1)}, \chi)$$

with appropriate functions $f^{(e)}$, g and h. Additionally, a polynomial of Σ

$$P(\Sigma) := \prod_{i=1}^{d} (\Sigma - \lambda_i' I_N) = \sum_{e=0}^{d} c_e \Sigma^e \qquad \text{where} \quad \lambda_1', \ldots, \lambda_d' \quad \text{are distinct eigenvalues of } \Sigma$$

equals to 0, thus we get

$$\Omega^{(d)} = -\sum_{e=0}^{d-1} c_e \Omega^{(e)}. \tag{4.8}$$

Using this relation, we can reduce $\Omega^{(d)}$ to expressions which contain only $\Omega^{(0)}, \dots, \Omega^{(d-1)}$, therefore we can get a closed differential equation system with $\Omega^{(0)}, \dots, \Omega^{(d-1)}$ and χ.

In summary, our macroscopic system is closed with the following order parameters:

Order variables : $Q_{ij}^{(0)}, Q_{ij}^{(1)}, \dots, Q_{ij}^{(d-1)}$, $R_{in}^{(0)}, R_{in}^{(1)}, \dots, R_{in}^{(d-1)}$, D_{ij}, E_{in}

Order constants : $T_{nm}^{(0)}, T_{nm}^{(1)}, \dots, T_{nm}^{(d-1)}$, F_{nm}. (d: number of distinct eigenvalues of Σ)

The order variables are governed by (4.3) and (4.5). For the lengthy full expressions of our macroscopic system for specific cases, see the appendix 4.B.

4.3.2 Dependency on input data covariance Σ

The differential equation system we derived depends on Σ, through two ways; the coefficient μ_{e+1} of $O(\eta^2)$-term, and how (d)-th order parameters are expanded with lower order parameters (as (4.8)). Specifically, the system only depends on the eigenvalue distribution of Σ.

4.3.3 Evaluation of expectation terms for specific activation functions

Expectation terms I_2, I_3 and I_4 can be analytically determined for some activation functions g, including sigmoid-like $g(x) = \mathrm{erf}(x/\sqrt{2})$ (see [9]) and $g(x) = \mathrm{ReLU}(x)$ (see [12]).

4.4 Analysis of numerical solutions of macroscopic differential equations

In this section, we analyze numerically the order parameter system, derived in the previous section[†]. We assume that the second layers' weights of the student and the teacher, namely w_i and v_n, are fixed to 1 (i.e. we consider the learning of soft-committee machine), and that K and M are equal to 2, for simplicity. Here we think of sigmoid-like activation $g(x) = \mathrm{erf}(x/\sqrt{2})$.

4.4.1 Consistency between macroscopic system and microscopic system

First of all, we confirmed the consistency between the macroscopic system we derived and the original microscopic system. That is, we computed the dynamics of the generalization loss ε_g in two ways: (i) by updating weights of the network with SGD (4.1) iteratively, and (ii) by solving numerically the differential equations (4.5) which govern the order parameters, and we confirmed that they accord with each other very well (Figure 4.4). Note that we set the initial values of order parameters in (ii) as values corresponding to initial weights used in (i). For dependence of the learning trajectory on the initial condition, see the appendix 4.C.

[†]Codes are available at `https://github.com/yos1up/data-dependence-of-plateau`. We executed all computations on a standard PC.

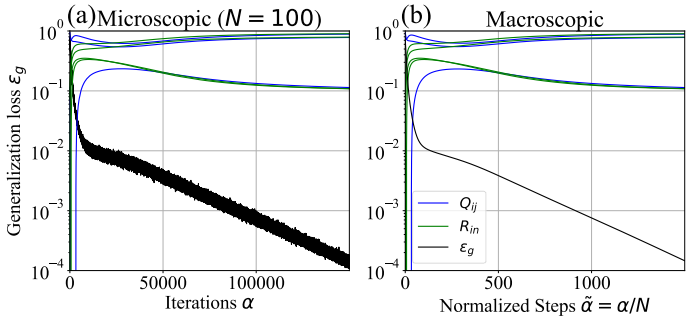

Figure 4.4: Example dynamics of generalization error ε_g computed with (a) microscopic and (b) macroscopic system. Network size: N-2-1. Learning rate: $\eta = 0.1$. Eigenvalues of Σ: $\lambda_1 = 0.4$ with multiplicity $0.5N$, $\lambda_2 = 1.2$ with multiplicity $0.3N$, and $\lambda_3 = 1.6$ with multiplicity $0.2N$. Black lines: dynamics of ε_g. Blue lines: Q_{11}, Q_{12}, Q_{22}. Green lines: $R_{11}, R_{12}, R_{21}, R_{22}$.

4.4.2 Case of scalar input covariance $\Sigma = \sigma I_N$

As the simplest case, here we consider the case that the convariance matrix Σ is proportional to unit matrix. In this case, Σ has only one eigenvalue $\lambda = \mu_1$ of multiplicity N, then our order parameter system contains only parameters whose order is 0 ($e = 0$). For various values of μ_1, we solved numerically the differential equations of order parameters (4.5) and plotted the time evolution of generalization loss ε_g (Figure 4.5(a)). From these plots, we quantified the lengths and heights of the plateaus as following: we regarded the system is plateauing if the decreasing speed of log-loss is smaller than half of its terminal converging speed, and we defined the height of the plateau as the median of loss values during plateauing. Quantified lengths and heights are plotted in Figure 4.5(b)(c). It indicates that the plateau length and height heavily depend on μ_1, the input scale. Specifically, as μ_1 decreases, the plateau rapidly becomes longer and lower. Though smaller input data lead to longer plateaus, it also becomes lower and then inconspicuous. This tendency is consistent with Figure 4.2(a)(b), since IRIS dataset has large μ_1 (≈ 15.9) and MNIST has small μ_1 (≈ 0.112). Considering this, the claim that the plateau phenomenon does not occur in learning of MNIST is controversy; this suggests the possibility that we are observing quite long and low plateaus.

Note that Figure 4.5(b) shows that the speed of growing of plateau length is larger than $O(1/\mu_1)$. This is contrast to the case of linear networks which have no activation; in that case, as μ_1 decreases the speed of learning gets exactly $1/\mu_1$-times larger. In other words, this phenomenon is peculiar to nonlinear networks.

4.4.3 Case of different input covariance Σ with fixed μ_1

In the previous subsection we inspected the dependence of the learning dynamics on the first moment μ_1 of the eigenvalues of the covariance matrix Σ. In this subsection, we explored the dependence of the dynamics on the higher moments of eigenvalues, under fixed first moment μ_1.

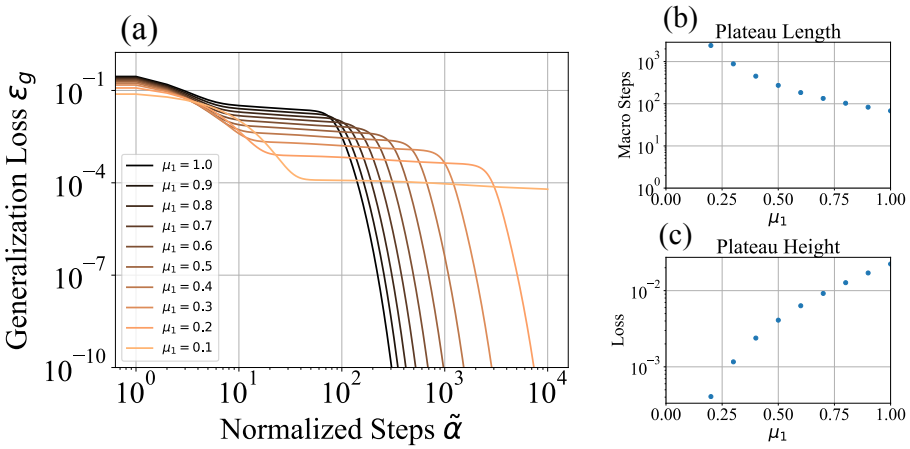

Figure 4.5: (a) Dynamics of generalization error ε_g when input variance Σ has only one eigenvalue $\lambda = \mu_1$ of multiplicity N. Plots with various values of μ_1 are shown. (b) Plateau length and (b) plateau height, quantified from (a).

In this subsection, we consider the case in which the input covariance matrix Σ has two distinct nonzero eigenvalues, $\lambda_1 = \mu_1 - \Delta\lambda/2$ and $\lambda_2 = \mu_1 + \Delta\lambda/2$, of the same multiplicity $N/2$ (Figure 4.6). With changing the control parameter $\Delta\lambda$, we can get eigenvalue distributions with various values of second moment $\mu_2 = \langle \lambda_i^2 \rangle$.

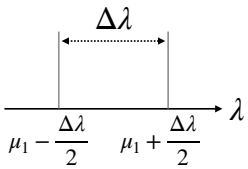

Figure 4.6: Eigenvalue distribution with fixed μ_1 parameterized by $\Delta\lambda$, which yields various μ_2.

Figure 4.7(a) shows learning curves with various μ_2 while fixing μ_1 to 1. From these curves, we quantified the lengths and heights of the plateaus, and plotted them in Figure 4.7(b)(c). These indicate that the length of the plateau shortens as μ_2 becomes large. That is, the more the distribution of nonzero eigenvalues gets broaden, the more the plateau gets alleviated.

4.5 Conclusion

Under the statistical mechanical formulation of learning in the two-layered perceptron, we showed that macroscopic equations can be derived even when the statistical properties of the input are generalized. We showed that the dynamics of learning depends only on the eigenvalue distribution of the covariance matrix of the input data. By nu-

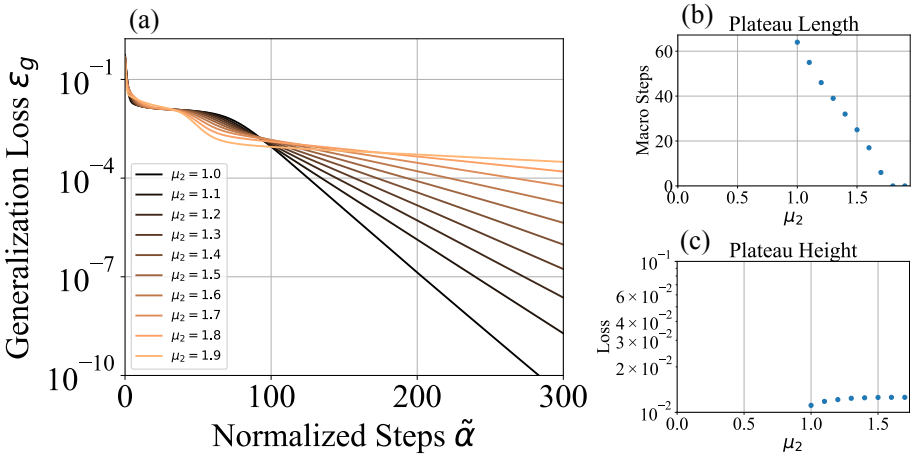

Figure 4.7: (a) Dynamics of generalization error ε_g when input variance Σ has two eigenvalues $\lambda_{1,2} = \mu_1 \pm \Delta\lambda/2$ of multiplicity $N/2$. Plots with various values of μ_2 are shown. (b) Plateau length and (c) plateau height, quantified from (a).

merically analyzing the macroscopic system, it is shown that the statistics of input data dramatically affect the plateau phenomenon.

Through this work, we explored the gap between theory and reality; though the plateau phenomenon is theoretically predicted to occur by the general symmetrical structure of neural networks, it is seldom observed in practice. However, more extensive researches are needed to fully understand the theory underlying the plateau phenomenon in practical cases.

Acknowledgement

This work was supported by JSPS KAKENHI Grant-in-Aid for Scientific Research(A) (No. 18H04106).

4.A Properties of expectation term I_2, I_3 and I_4

The differential equations of learning dynamics (3) and (5) in the main text have expectation terms, $I_2(z_1, z_2), I_3(z_1, z_2, z_3)$ and $I_4(z_1, z_2, z_3, z_4)$. Since their zs are either $x_i^{(e)} = \boldsymbol{\xi}^T \Sigma^e \boldsymbol{J}_i$ or $y_n^{(e)} = \boldsymbol{\xi}^T \Sigma^e \boldsymbol{B}_n$, any tuple $(z_1, z_2, \ldots)$ follows multivatiate normal distribution $\mathcal{N}(\boldsymbol{z}|0, \langle \boldsymbol{z} \cdot \boldsymbol{z}^T \rangle)$ when $N \to \infty$ by generalized central limit theorem, provided that the input $\boldsymbol{\xi}$ has zero mean and finite covariance. Thus the expectation terms only depend on the covariance matrix $\langle \boldsymbol{z} \cdot \boldsymbol{z}^T \rangle$, and their elements can be calculated as $\langle x_i^{(e)} x_j^{(f)} \rangle = Q_{ij}^{(e+f+1)}$,

$\langle x_i^{(e)} y_n^{(f)} \rangle = R_{in}^{(e+f+1)}$ and $\langle y_n^{(e)} y_m^{(f)} \rangle = T_{nm}^{(e+f+1)}$. For example,

$$I_2(x_i, y_p) = \int dz_1 dz_2 \; g(z_1) g(z_2) \; \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \begin{pmatrix} Q_{ii}^{(1)} & R_{ip}^{(1)} \\ & T_{pp}^{(1)} \end{pmatrix}),$$

$$I_3(x_i, x_j^{(e)}, y_p) = \int dz_1 dz_2 dz_3 \; g'(z_1) z_2 g(z_3) \; \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \begin{pmatrix} Q_{ii}^{(1)} & Q_{ij}^{(e+1)} & R_{ip}^{(1)} \\ & Q_{jj}^{(2e+1)} & R_{jp}^{(e+1)} \\ & & T_{pp}^{(1)} \end{pmatrix}),$$

$$I_4(x_i, x_j, y_p, y_q) = \int dz_1 dz_2 dz_3 dz_4 \; g(z_1) g(z_2) g(z_3) g(z_4) \; \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \begin{pmatrix} Q_{ii}^{(1)} & Q_{ij}^{(1)} & R_{ip}^{(1)} & R_{iq}^{(1)} \\ & Q_{jj}^{(1)} & R_{jp}^{(1)} & R_{jq}^{(1)} \\ & & T_{pp}^{(1)} & T_{pq}^{(1)} \\ & & & T_{qq}^{(1)} \end{pmatrix}).$$

Note that all the covariance matrix is symmetric. Their left-bottom sides are not shown for notational simplicity. Substituting these for Is shown in equations (3) and (5) in the main text, we see that the 'speed' of e-th order parameters can be dependent only on 1-st, $(e+1)$-th, and $(2e+1)$-th order parameters.

Here we prove the following proposition, in order to show that the 'speed' of e-th order parameters are not dependent on $(2e+1)$-th order parameters.

Proposition. The expectation term $I_3(z_1, z_2, z_3) := \int dz_1 dz_2 dz_3 \; g'(z_1) z_2 g(z_3) \, \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, C)$ does not depend on C_{22}.

Proof. Since C is positive-semidefinite, we can write $C = VV^T$ for some squared matrix V. Thus, when $\boldsymbol{\xi} \sim \mathcal{N}(0, I_N)$, $A\boldsymbol{\xi} \sim \mathcal{N}(0, C)$ holds. Therefore, we can regard that $z_i (i = 1, 2, 3)$ is generated by $z_i = \boldsymbol{v}_i^T \boldsymbol{\xi}$ where $\boldsymbol{v}_i$ is i-th row vector of V and $\boldsymbol{\xi}$ follows the standard normal distribution.

We can write $\boldsymbol{v}_2 = c_1 \boldsymbol{v}_1 + c_3 \boldsymbol{v}_3 + \boldsymbol{v}^\perp$ for some coefficient $c_1, c_3 \in \mathbb{R}$ and some vector $\boldsymbol{v}^\perp$ perpendicular to $\boldsymbol{v}_1$ and $\boldsymbol{v}_3$. Then I_3 is written as

$$I_3(z_1, z_2, z_3) = \langle g'(z_1) z_2 g(z_3) \rangle = c_1 \langle g'(z_1) z_1 g(z_3) \rangle + c_3 \langle g'(z_1) z_3 g(z_3) \rangle + \langle g'(z_1) \boldsymbol{v}^{\perp T} \boldsymbol{\xi} g(z_3) \rangle.$$

Since $\boldsymbol{\xi} \sim \mathcal{N}(0, I_N)$ and $\boldsymbol{v}^\perp \perp \boldsymbol{v}_1, \boldsymbol{v}_3$ hold, (z_1, z_3) and $\boldsymbol{v}^{\perp T} \boldsymbol{\xi}$ is independent. Therefore the third term in the right hand side of the equation above is

$$\langle g'(z_1) \boldsymbol{v}^{\perp T} \boldsymbol{\xi} g(z_3) \rangle = \langle g'(z_1) g(z_3) \rangle \langle \boldsymbol{v}^{\perp T} \boldsymbol{\xi} \rangle = 0.$$

In addition, we can determine c_1 and c_3 by solving

$$\begin{aligned} C_{12} &= \boldsymbol{v}_2^T \boldsymbol{v}_1 = (c_1 \boldsymbol{v}_1^T + c_3 \boldsymbol{v}_3^T + \boldsymbol{v}^{\perp T}) \boldsymbol{v}_1 = c_1 C_{11} + c_3 C_{13} \quad \text{and} \\ C_{23} &= \boldsymbol{v}_2^T \boldsymbol{v}_3 = (c_1 \boldsymbol{v}_1^T + c_3 \boldsymbol{v}_3^T + \boldsymbol{v}^{\perp T}) \boldsymbol{v}_3 = c_1 C_{13} + c_3 C_{33}. \end{aligned}$$

Together with these, we get

$$I_3(z_1, z_2, z_3) = \frac{(C_{12} C_{33} - C_{13} C_{23}) \; I_3(z_1, z_1, z_3) + (C_{11} C_{23} - C_{12} C_{13}) \; I_3(z_1, z_3, z_3)}{C_{11} C_{33} - C_{13}^2},$$

which shows that I_3 is independent to C_{22}. ■

4.B Full expression of order parameter system

Here we describe the whole system of the order parameters, with specific eigenvalue distribution of Σ.

4.B.1 Case with $\Sigma = \sigma I_N$

In this case, the order parameters are

$$\begin{aligned} &\text{Order variables :} \quad Q_{ij}^{(0)}, \quad R_{in}^{(0)}, \quad D_{ij}, E_{in} \\ &\text{Order constants :} \quad T_{nm}^{(0)}, \quad F_{nm}. \end{aligned}$$

Note that $Q_{ij}^{(1)}$ is identical to $Q_{ij}^{(0)}$. This is same for R and T. The order parameter system is described as following, with omitting $^{(0)}$-s for notational simplicity:

$$\begin{aligned} \frac{dQ_{ij}}{d\tilde{\alpha}} &= \eta \left[\sum_{p=1}^{M} E_{ip} I_3 \begin{pmatrix} Q_{ii} & Q_{ij} & R_{ip} \\ & * & R_{jp} \\ & & T_{pp} \end{pmatrix} - \sum_{p=1}^{K} D_{ip} I_3 \begin{pmatrix} Q_{ii} & Q_{ij} & Q_{ip} \\ & * & Q_{jp} \\ & & Q_{pp} \end{pmatrix} \right. \\ &\quad \left. + \sum_{p=1}^{M} E_{jp} I_3 \begin{pmatrix} Q_{jj} & Q_{ji} & R_{jp} \\ & * & R_{ip} \\ & & T_{pp} \end{pmatrix} - \sum_{p=1}^{K} D_{jp} I_3 \begin{pmatrix} Q_{jj} & Q_{ji} & Q_{jp} \\ & * & Q_{ip} \\ & & Q_{pp} \end{pmatrix} \right] \\ &\quad + \eta^2 \left[\sum_{p,q}^{K,K} D_{ip} D_{jq} I_4 \begin{pmatrix} Q_{ii} & Q_{ij} & Q_{ip} & Q_{iq} \\ & Q_{jj} & Q_{jp} & Q_{jq} \\ & & Q_{pp} & Q_{pq} \\ & & & Q_{qq} \end{pmatrix} + \sum_{p,q}^{M,M} E_{ip} E_{jq} I_4 \begin{pmatrix} Q_{ii} & Q_{ij} & R_{ip} & R_{iq} \\ & Q_{jj} & R_{jp} & R_{jq} \\ & & T_{pp} & T_{pq} \\ & & & T_{qq} \end{pmatrix} \right. \\ &\quad \left. - \sum_{p,q}^{K,M} D_{ip} E_{jq} I_4 \begin{pmatrix} Q_{ii} & Q_{ij} & Q_{ip} & R_{iq} \\ & Q_{jj} & Q_{jp} & R_{jq} \\ & & Q_{pp} & R_{pq} \\ & & & T_{qq} \end{pmatrix} - \sum_{p,q}^{M,K} E_{ip} D_{jq} I_4 \begin{pmatrix} Q_{ii} & Q_{ij} & R_{ip} & Q_{iq} \\ & Q_{jj} & R_{jp} & Q_{jq} \\ & & T_{pp} & R_{pq} \\ & & & Q_{qq} \end{pmatrix} \right], \\ \frac{dR_{in}}{d\tilde{\alpha}} &= \eta \left[\sum_{p=1}^{M} E_{ip} I_3 \begin{pmatrix} Q_{ii} & R_{in} & R_{ip} \\ & * & T_{np} \\ & & T_{pp} \end{pmatrix} - \sum_{p=1}^{K} D_{ip} I_3 \begin{pmatrix} Q_{ii} & R_{in} & Q_{ip} \\ & * & R_{pn} \\ & & Q_{pp} \end{pmatrix} \right] \end{aligned} \tag{4.9}$$

and

$$\begin{aligned} \frac{dD_{ij}}{d\tilde{\alpha}} &= \eta \left[\sum_{p=1}^{M} E_{ip} I_2 \begin{pmatrix} Q_{jj} & R_{jp} \\ & T_{pp} \end{pmatrix} - \sum_{p=1}^{K} D_{ip} I_2 \begin{pmatrix} Q_{jj} & Q_{jp} \\ & Q_{pp} \end{pmatrix} \right. \\ &\quad \left. + \sum_{p=1}^{M} E_{jp} I_2 \begin{pmatrix} Q_{ii} & R_{ip} \\ & T_{pp} \end{pmatrix} - \sum_{p=1}^{K} D_{jp} I_2 \begin{pmatrix} Q_{ii} & Q_{ip} \\ & Q_{pp} \end{pmatrix} \right], \\ \frac{dE_{in}}{d\tilde{\alpha}} &= \eta \left[\sum_{p=1}^{M} F_{pn} I_2 \begin{pmatrix} Q_{ii} & R_{ip} \\ & T_{pp} \end{pmatrix} - \sum_{p=1}^{K} E_{pn} I_2 \begin{pmatrix} Q_{jj} & R_{jp} \\ & T_{pp} \end{pmatrix} \right] \end{aligned} \tag{4.10}$$

,

$$\text{where} \quad I_2(C) = \frac{2}{\pi} \arcsin \frac{C_{12}}{\sqrt{1+C_{11}}\sqrt{1+C_{22}}},$$
$$I_3(C) = \frac{2}{\pi} \cdot \frac{1}{\sqrt{(1+C_{11})(1+C_{33}) - C_{13}^2}} \frac{C_{23}(1+C_{11}) - C_{12}C_{13}}{1+C_{11}},$$
$$I_4(C) = \frac{4}{\pi^2} \cdot \frac{1}{\sqrt{1+2C_{11}}} \arcsin \frac{(1+2C_{11})C_{23} - 2C_{12}C_{13}}{\sqrt{(1+2C_{11})(1+C_{22}) - 2C_{12}^2}\sqrt{(1+2C_{11})(1+C_{33}) - 2C_{13}^2}} \tag{4.11}$$

for $g(x) = \mathrm{erf}(x/\sqrt{2})$ activation, as [9] showed.

4.B.2 Case with Σ which has two distinct eigenvalues, λ_1 of multiplicity $r_1 N$ and λ_2 of multiplicity $r_2 N$

In this case, the order parameters are

Order variables :	$Q_{ij}^{(0)}, Q_{ij}^{(1)},$	$R_{in}^{(0)}, R_{in}^{(1)},$	D_{ij}, E_{in}
Order constants :	$T_{nm}^{(0)}, T_{nm}^{(1)},$	$F_{nm}.$	

Since $\Sigma^2 - (\lambda_1+\lambda_2)\Sigma + \lambda_1\lambda_2 I_N = 0$, the relation $Q_{ij}^{(2)} = (\lambda_1+\lambda_2)Q_{ij}^{(1)} - \lambda_1\lambda_2 Q_{ij}^{(0)}$ holds. This is same for R and T. Then the order parameter system is described as following:

$$
\begin{aligned}
\frac{dQ_{ij}^{(0)}}{d\tilde{\alpha}} &= \eta\left[\sum_{p=1}^{M} E_{ip} I_3 \begin{pmatrix} Q_{ii}^{(1)} & Q_{ij}^{(1)} & R_{ip}^{(1)} \\ & * & R_{jp}^{(1)} \\ & & T_{pp}^{(1)} \end{pmatrix} - \sum_{p=1}^{K} D_{ip} I_3 \begin{pmatrix} Q_{ii}^{(1)} & Q_{ij}^{(1)} & Q_{ip}^{(1)} \\ & * & Q_{jp}^{(1)} \\ & & Q_{pp}^{(1)} \end{pmatrix}\right. \\
&\quad \left. + \sum_{p=1}^{M} E_{jp} I_3 \begin{pmatrix} Q_{jj}^{(1)} & Q_{ji}^{(1)} & R_{jp}^{(1)} \\ & * & R_{ip}^{(1)} \\ & & T_{pp}^{(1)} \end{pmatrix} - \sum_{p=1}^{K} D_{jp} I_3 \begin{pmatrix} Q_{jj}^{(1)} & Q_{ji}^{(1)} & Q_{jp}^{(1)} \\ & * & Q_{ip}^{(1)} \\ & & Q_{pp}^{(1)} \end{pmatrix}\right] \\
&\quad + \eta^2(r_1\lambda_1 + r_2\lambda_2)\left[\sum_{p,q}^{K,K} D_{ip}D_{jq} I_4 \begin{pmatrix} Q_{ii}^{(1)} & Q_{ij}^{(1)} & Q_{ip}^{(1)} & Q_{iq}^{(1)} \\ & Q_{jj}^{(1)} & Q_{jp}^{(1)} & Q_{jq}^{(1)} \\ & & Q_{pp}^{(1)} & Q_{pq}^{(1)} \\ & & & Q_{qq}^{(1)} \end{pmatrix} + \sum_{p,q}^{M,M} E_{ip}E_{jq} I_4 \begin{pmatrix} Q_{ii}^{(1)} & Q_{ij}^{(1)} & R_{ip}^{(1)} & R_{iq}^{(1)} \\ & Q_{jj}^{(1)} & R_{jp}^{(1)} & R_{jq}^{(1)} \\ & & T_{pp}^{(1)} & T_{pq}^{(1)} \\ & & & T_{qq}^{(1)} \end{pmatrix}\right. \\
&\quad \left. - \sum_{p,q}^{K,M} D_{ip}E_{jq} I_4 \begin{pmatrix} Q_{ii}^{(1)} & Q_{ij}^{(1)} & Q_{ip}^{(1)} & R_{iq}^{(1)} \\ & Q_{jj}^{(1)} & Q_{jp}^{(1)} & R_{jq}^{(1)} \\ & & Q_{pp}^{(1)} & R_{pq}^{(1)} \\ & & & T_{qq}^{(1)} \end{pmatrix} - \sum_{p,q}^{M,K} E_{ip}D_{jq} I_4 \begin{pmatrix} Q_{ii}^{(1)} & Q_{ij}^{(1)} & R_{ip}^{(1)} & Q_{iq}^{(1)} \\ & Q_{jj}^{(1)} & R_{jp}^{(1)} & Q_{jq}^{(1)} \\ & & T_{pp}^{(1)} & R_{pq}^{(1)} \\ & & & Q_{qq}^{(1)} \end{pmatrix}\right], \\
\frac{dQ_{ij}^{(1)}}{d\tilde{\alpha}} &= \eta\left[\sum_{p=1}^{M} E_{ip} I_3 \begin{pmatrix} Q_{ii}^{(1)} & (\lambda_1+\lambda_2)Q_{ij}^{(1)} - \lambda_1\lambda_2 Q_{ij}^{(0)} & R_{ip}^{(1)} \\ & * & (\lambda_1+\lambda_2)R_{jp}^{(1)} - \lambda_1\lambda_2 R_{jp}^{(0)} \\ & & T_{pp}^{(1)} \end{pmatrix}\right. \\
&\quad - \sum_{p=1}^{K} D_{ip} I_3 \begin{pmatrix} Q_{ii}^{(1)} & (\lambda_1+\lambda_2)Q_{ij}^{(1)} - \lambda_1\lambda_2 Q_{ij}^{(0)} & Q_{ip}^{(1)} \\ & * & (\lambda_1+\lambda_2)Q_{jp}^{(1)} - \lambda_1\lambda_2 Q_{jp}^{(0)} \\ & & Q_{pp}^{(1)} \end{pmatrix} \\
&\quad + \sum_{p=1}^{M} E_{jp} I_3 \begin{pmatrix} Q_{jj}^{(1)} & (\lambda_1+\lambda_2)Q_{ji}^{(1)} - \lambda_1\lambda_2 Q_{ji}^{(0)} & R_{jp}^{(1)} \\ & * & (\lambda_1+\lambda_2)R_{ip}^{(1)} - \lambda_1\lambda_2 R_{ip}^{(0)} \\ & & T_{pp}^{(1)} \end{pmatrix} \\
&\quad \left. - \sum_{p=1}^{K} D_{jp} I_3 \begin{pmatrix} Q_{jj}^{(1)} & (\lambda_1+\lambda_2)Q_{ji}^{(1)} - \lambda_1\lambda_2 Q_{ji}^{(0)} & Q_{jp}^{(1)} \\ & * & (\lambda_1+\lambda_2)Q_{ip}^{(1)} - \lambda_1\lambda_2 Q_{ip}^{(0)} \\ & & Q_{pp}^{(1)} \end{pmatrix}\right] \\
&\quad + \eta^2(r_1\lambda_1^2 + r_2\lambda_2^2)\left[\sum_{p,q}^{K,K} D_{ip}D_{jq} I_4 \begin{pmatrix} Q_{ii}^{(1)} & Q_{ij}^{(1)} & Q_{ip}^{(1)} & Q_{iq}^{(1)} \\ & Q_{jj}^{(1)} & Q_{jp}^{(1)} & Q_{jq}^{(1)} \\ & & Q_{pp}^{(1)} & Q_{pq}^{(1)} \\ & & & Q_{qq}^{(1)} \end{pmatrix} + \sum_{p,q}^{M,M} E_{ip}E_{jq} I_4 \begin{pmatrix} Q_{ii}^{(1)} & Q_{ij}^{(1)} & R_{ip}^{(1)} & R_{iq}^{(1)} \\ & Q_{jj}^{(1)} & R_{jp}^{(1)} & R_{jq}^{(1)} \\ & & T_{pp}^{(1)} & T_{pq}^{(1)} \\ & & & T_{qq}^{(1)} \end{pmatrix}\right. \\
&\quad \left. - \sum_{p,q}^{K,M} D_{ip}E_{jq} I_4 \begin{pmatrix} Q_{ii}^{(1)} & Q_{ij}^{(1)} & Q_{ip}^{(1)} & R_{iq}^{(1)} \\ & Q_{jj}^{(1)} & Q_{jp}^{(1)} & R_{jq}^{(1)} \\ & & Q_{pp}^{(1)} & R_{pq}^{(1)} \\ & & & T_{qq}^{(1)} \end{pmatrix} - \sum_{p,q}^{M,K} E_{ip}D_{jq} I_4 \begin{pmatrix} Q_{ii}^{(1)} & Q_{ij}^{(1)} & R_{ip}^{(1)} & Q_{iq}^{(1)} \\ & Q_{jj}^{(1)} & R_{jp}^{(1)} & Q_{jq}^{(1)} \\ & & T_{pp}^{(1)} & R_{pq}^{(1)} \\ & & & Q_{qq}^{(1)} \end{pmatrix}\right], \\
\frac{dR_{in}^{(0)}}{d\tilde{\alpha}} &= \eta\left[\sum_{p=1}^{M} E_{ip} I_3 \begin{pmatrix} Q_{ii}^{(1)} & R_{in}^{(1)} & R_{ip}^{(1)} \\ & * & T_{np}^{(1)} \\ & & T_{pp}^{(1)} \end{pmatrix} - \sum_{p=1}^{K} D_{ip} I_3 \begin{pmatrix} Q_{ii}^{(1)} & R_{in}^{(1)} & Q_{ip}^{(1)} \\ & * & R_{pn}^{(1)} \\ & & Q_{pp}^{(1)} \end{pmatrix}\right], \\
\frac{dR_{in}^{(1)}}{d\tilde{\alpha}} &= \eta\left[\sum_{p=1}^{M} E_{ip} I_3 \begin{pmatrix} Q_{ii}^{(1)} & (\lambda_1+\lambda_2)R_{in}^{(1)} - \lambda_1\lambda_2 R_{in}^{(0)} & R_{ip}^{(1)} \\ & * & (\lambda_1+\lambda_2)T_{np}^{(1)} - \lambda_1\lambda_2 T_{np}^{(0)} \\ & & T_{pp}^{(1)} \end{pmatrix}\right. \\
&\quad \left. - \sum_{p=1}^{K} D_{ip} I_3 \begin{pmatrix} Q_{ii}^{(1)} & (\lambda_1+\lambda_2)R_{in}^{(1)} - \lambda_1\lambda_2 R_{in}^{(0)} & Q_{ip}^{(1)} \\ & * \quad 71 & (\lambda_1+\lambda_2)R_{pn}^{(1)} - \lambda_1\lambda_2 R_{pn}^{(0)} \\ & & Q_{pp}^{(1)} \end{pmatrix}\right]
\end{aligned}
\tag{4.12}
$$

, and

$$\frac{dD_{ij}}{d\tilde{\alpha}} = \eta \left[\sum_{p=1}^{M} E_{ip} I_2 \begin{pmatrix} Q_{jj}^{(1)} & R_{jp}^{(1)} \\ & T_{pp}^{(1)} \end{pmatrix} - \sum_{p=1}^{K} D_{ip} I_2 \begin{pmatrix} Q_{jj}^{(1)} & Q_{jp}^{(1)} \\ & Q_{pp}^{(1)} \end{pmatrix} \right.$$
$$\left. + \sum_{p=1}^{M} E_{jp} I_2 \begin{pmatrix} Q_{ii}^{(1)} & R_{ip}^{(1)} \\ & T_{pp}^{(1)} \end{pmatrix} - \sum_{p=1}^{K} D_{jp} I_2 \begin{pmatrix} Q_{ii}^{(1)} & Q_{ip}^{(1)} \\ & Q_{pp}^{(1)} \end{pmatrix} \right], \tag{4.13}$$
$$\frac{dE_{in}}{d\tilde{\alpha}} = \eta \left[\sum_{p=1}^{M} F_{pn} I_2 \begin{pmatrix} Q_{ii}^{(1)} & R_{ip}^{(1)} \\ & T_{pp}^{(1)} \end{pmatrix} - \sum_{p=1}^{K} E_{pn} I_2 \begin{pmatrix} Q_{jj}^{(1)} & R_{jp}^{(1)} \\ & T_{pp}^{(1)} \end{pmatrix} \right]$$

.

4.C Dependence of learning trajectory on initial conditions on macroscopic parameters

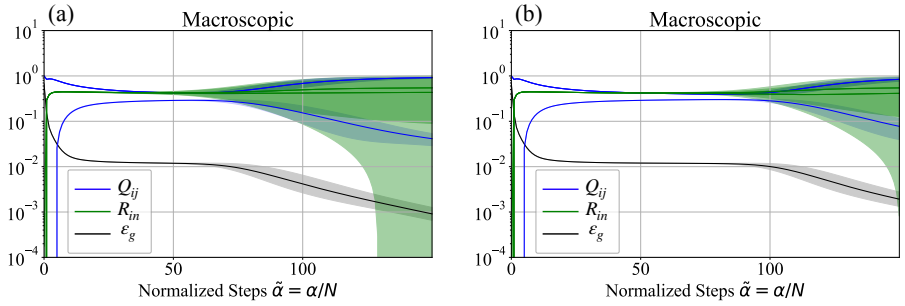

Figure 4.8: Dynamics of generalization error ε_g and order parameters Q_{ij} and R_{in} computed with macroscopic system, and its variability by random weight initialization. Network size: N-2-1. Learning rate: $\eta = 0.1$. Eigenvalues of Σ: $\lambda_1 = 0.3$ with multiplicity $0.5N$, $\lambda_2 = 1.7$ with multiplicity $0.5N$. Black lines: dynamics of ε_g. Blue lines: Q_{11}, Q_{12}, Q_{22}. Green lines: $R_{11}, R_{12}, R_{21}, R_{22}$. (a) $N = 10^5$, (b) $N = 10^7$. In both figures, solid curves and shades represent mean and standard deviation of 100 trials, respectively (note that mean and standard deviation of loss are computed in logarithmic scale).

In the statistical mechanical formulation, by considering N as large, the dynamics of the system is reduced to macroscopic differential equations with small (N-independent) dimensions. The macroscopic system we derived is deterministic in the sense that randomness brought by stochastic gradient descent is vanished. However, note that the trajectory of the macroscopic state can vary in accordance with its initial condition. Figure 4.8 shows this variability with shades.

How does the initial condition affect the learning trajectory? Consider a typical initialization that the microscopic parameters $\boldsymbol{J}_1$, $\boldsymbol{J}_2$, $\boldsymbol{B}_1$ and $\boldsymbol{B}_2$ are initialized as

$(\boldsymbol{J}_i)_k, (\boldsymbol{B}_n)_k \overset{\text{i.i.d.}}{\sim} \mathcal{N}(0, 1/N)$. Then the mean and variance of corresponding initial macroscopic parameters Q, R and T are

$$\begin{aligned}
&\mathbb{E}[Q_{ii}^{(e)}] = \mu_e, \quad \mathbb{V}[Q_{ii}^{(e)}] = \frac{3\mu_{2e}}{N}, \quad \mathbb{E}[Q_{ij}^{(e)}] = 0, \quad \mathbb{V}[Q_{ij}^{(e)}] = \frac{\mu_{2e}}{N}, \\
&\mathbb{E}[R_{in}^{(e)}] = 0, \quad \mathbb{V}[R_{in}^{(e)}] = \frac{\mu_{2e}}{N}, \\
&\mathbb{E}[T_{nn}^{(e)}] = \mu_e, \quad \mathbb{V}[T_{nn}^{(e)}] = \frac{3\mu_{2e}}{N}, \quad \mathbb{E}[T_{nm}^{(e)}] = 0, \quad \mathbb{V}[T_{nm}^{(e)}] = \frac{\mu_{2e}}{N}
\end{aligned}$$

With $N \to \infty$, these probabilistic parameters converge to $(Q^{(e)}, R^{(e)}, T^{(e)}) = (\mu_e I_K, 0, \mu_e I_M)$. However, the solution trajectory starting from just $(\mu_e I_K, 0, \mu_e I_M)$ cannot break the weight symmetry at all. To argue practical learning trajectory, we have to consider the initial value slightly off from that point. How close the initial condition is to that point affects how long it takes to break the weight symmetry, that is, the plateau length. This is why Figure 4.8 (b) with $N = 10^7$ exhibits plateau slightly longer than that of Figure 4.8 (a) with $N = 10^5$.

Chapter 5

Dynamics of Signal Propagation in Deep Neural Network — Mean-field Approach

5.1 Introduction

Since the late 2000s, the various techniques related to training deep neural networks with heavy datasets have been developed successfully. Nowadays, quite wide range of tasks are solved by deep neural networks in end-to-end fashion, thanks to these methods.

However, we lack the knowledge for their mechanisms. Our understanding for why the deep learning is so successful is still not enough. Especially, we have many hyperparameters; size, number and types of layers, weight initialization, methods of gradient descent, and data preprocessing etc. We lack knowledge of how they determine the learning to be successful or unsuccessful. This is why we resort to optimize hyperparameters greedily. In what condition can we train very deep neural networks well?

One of necessary condition for successful training is that information can be propagated through the neural network which typically has many layers, without being degraded.

Recently Schoenholz et al.[2] developed a mean-field approach for analyzing signal propagation in random neural networks. This approach deals with feed-forward neural networks which have random valued weights and infinite layer width, and introduces mean-field approximation. With such setting, the macroscopic evolution of statistics of signals becomes deterministic. They derived the macroscopic dynamics of signal statistics theoretically, in case with fully-connected networks (FCN). Then they discussed their depth scales, which is the time constant of degradation of the signal statistics. Moreover, they found experimentally that the depth scale strongly coincides with trainability; if the number of layers of a network is large/small relative to its depth scale, it is hard/easy to train. Their result is shown in Figure 5.1(a).

Since then, several works have shown that the coincidence described above holds with several neural networks aside from FCN, such as convolutional neural networks (CNN)

and recurrent neural networks (RNN) [3, 4]; they derived the depth scale analytically and confirmed its consistency with actual trainability experimentally. Their results are shown in Figure 5.1(b)(c). These three works demonstrate the validity of mean-field theory of depth scale for predicting trainability without actual training. However, no one knows whether the theory is valid for more realistic situation, in that most of practical neural networks are typically more complex. It is not realistic to derive the depth scale theoretically for every latest neural network architecture. Our main contributions are summarized as below:

- We establish a numerical method for estimating depth scales for forward propagation, which is available regardless of network architecture.
- We examine the applicability of the notion of depth scale of neural networks for predicting their trainability, and find that it works very well in a wide range of neural networks which have finite depth scales.

Aside from above, we also inspect a situation where weights of a neural network are initialized by a long-tailed distribution. Although this situation does not seem to be realistic, this yields an interesting consequence.

5.2 Related works

5.2.1 Edge of chaos and depth scale

Before the deep learning has appeared, there was a study of propagation dynamics in recurrent neural networks [11]. They investigated the dynamics of signal propagation, found the phase transition between ordered and chaotic, and pointed out that it can process complex computations provided that it is at the 'edge of chaos'.

These days, the propagation dynamics is investigated in the context of deep learning, especially in relation to trainability. Theoretical derivation of the depth scale and its consistency with trainability have been reported in case with FCN with/without dropout [2], CNN [3], RNN/LSTM [4, 54], binary neural network [55], quantized neural network [56], residual network [57], graph neural network [58], and batch normalization [59].

5.2.2 Random neural networks

The mathematics of random neural networks has been studied by Amari since the 1970s [60]. His researches at that time were motivated primarily by neuroscience, rather than by neural networks as a method of machine learning.

In the context of deep learning, random neural networks were first focused on by Poole et al [10]. They pointed out that the curvature of a deep neural network can increase exponentially with respect to the number of layers. Subsequently, the discussion of depth scales as described above has developed. In addition, several important quantities for understanding loss landscapes, such as Jacobian, Hessian, and Fisher information matrix, have been analyzed using the mean-field theory [61, 62, 63, 64, 65].

Random networks have also attracted attention for their relation to Gaussian processes. A randomly initialized deep neural network with infinite layer width was found to be equivalent to a Gaussian process [66, 67]. Furthermore, the theory of neural tangent kernel revealed that layer-width infinite neural networks during and after training by a gradient method can also be regarded as Gaussian processes [68, 69].

5.3 Theoretical setup

5.3.1 Depth scale

In this subsection, we review the concept of the depth scale defined in [2].

Here we consider neural networks which consist of repeating L unit architectures. We denote them by $1, 2, \ldots, L$. Each i-th unit architecture has its own parameters θ_i, which determines the actual mapping of i-th unit architecture, namely f_{θ_i}. The mapping of the whole network is described as

$$\mathbb{R}^N \xrightarrow{f_{\theta_1}} \mathbb{R}^N \xrightarrow{f_{\theta_2}} \cdots \xrightarrow{f_{\theta_L}} \mathbb{R}^N.$$

We consider the situation that an input vector $\boldsymbol{x} \in \mathbb{R}^N$ propagates through a unit architecture: $f_{\theta_l}(\boldsymbol{x}^{(l-1)}) = \boldsymbol{x}^{(l)}$. We introduce a macroscopic quantities $q := \|\boldsymbol{x}\|^2/N$. For various types of simple unit architectures (with Gaussian random initialization), it has been shown that the relationship between $q^{(l-1)}$ and $q^{(l)}$ becomes deterministic in the large limit of N; that is, if the squared norm of $\boldsymbol{x}^{(l-1)}$ is $q^{(l-1)}$, the squared norm of $\boldsymbol{x}^{(l)}$ converges to $q^{(l)}$ in distribution when $N \to \infty$. We also consider the propagation of two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N$. Then we introduce overlap, the second macroscopic quantity, namely $c := \boldsymbol{x}^T\boldsymbol{y}/(\|\boldsymbol{x}\|\|\boldsymbol{y}\|N)$. Similar to above, it has been found that the relationship between $(q^{(l-1)}, c^{(l-1)})$ and $(q^{(l)}, c^{(l)})$ becomes deterministic under $N \to \infty$ situation with various simple unit architectures; that is, if the overlap between $\boldsymbol{x}^{(l-1)}$ and $\boldsymbol{y}^{(l-1)}$ is $c^{(l-1)}$, the overlap between $\boldsymbol{x}^{(l)}$ and $\boldsymbol{y}^{(l)}$ converges to $c^{(l)}$ in distribution when $N \to \infty$.

Here we review the case with FCN, by following [2]. In FCN case, each unit architecture is just a fully-connected layer, and its mapping is represented as

$$f_\theta : \boldsymbol{z} \mapsto W\phi(\boldsymbol{z}) + \boldsymbol{b},$$

where $\theta = (W, \boldsymbol{b})$, $W \in \mathbb{R}^{N\times N}, \boldsymbol{b} \in \mathbb{R}^N$ are parameters, and $\phi : \mathbb{R}^N \to \mathbb{R}^N$ is any element-wise activation function. We consider the situation that two signals $\boldsymbol{x}^{(l-1)}$ and $\boldsymbol{y}^{(l-1)}$ are mapped to $\boldsymbol{x}^{(l)}$ and $\boldsymbol{y}^{(l)}$ by f_θ. Assume that W and $\boldsymbol{b}$ are random variables; we assume that each element of W follows $\mathcal{N}(0, \sigma_w^2/N)$, and each element of $\boldsymbol{b}$ follows $\mathcal{N}(0, \sigma_b^2)$. What we are interested in is the joint distribution of $x_k^{(l)}$ and $y_j^{(l)}$ $(1 \le k, j \le N)$, namely

$p(x_1^{(l)}, \ldots, x_N^{(l)}, y_1^{(l)}, \ldots, y_N^{(l)})$. By simple calculations we obtain

$$\begin{aligned}
\mathbb{E}[x_i^{(l)}] &= \mathbb{E}[\sum_{j=1}^{N} W_{ij}\phi(x_j^{(l-1)}) + b_i] = \sum_{j=1}^{N} \mathbb{E}[W_{ij}]\phi(x_j^{(l-1)}) + \mathbb{E}[b_i] = 0, \\
\mathbb{E}[x_i^{(l)} x_j^{(l)}] &= \mathbb{E}[(\sum_{k=1}^{N} W_{ik}\phi(x_k^{(l-1)}) + b_i)(\sum_{h=1}^{N} W_{jh}\phi(x_h^{(l-1)}) + b_j)] \\
&= \sum_{k=1}^{N}\sum_{h=1}^{N} \mathbb{E}[W_{ik}W_{jh}]\phi(x_k^{(l-1)})\phi(x_h^{(l-1)}) + \sum_{k=1}^{N} \mathbb{E}[W_{ik}b_j]\phi(x_k^{(l-1)}) \\
&+ \sum_{h=1}^{N} \mathbb{E}[W_{jh}b_i]\phi(x_h^{(l-1)}) + \mathbb{E}[b_i b_j] \\
&= \sum_{k=1}^{N}\sum_{h=1}^{N} \delta_{ij}\delta_{kh}\sigma_w^2/N\phi(x_k^{(l-1)})\phi(x_h^{(l-1)}) + \delta_{ij}\sigma_b^2 \\
&= \delta_{ij}[\sigma_w^2/N \sum_{k=1}^{N} \phi(x_k^{(l-1)})^2 + \sigma_b^2], \\
\mathbb{E}[x_i^{(l)} y_j^{(l)}] &= \mathbb{E}[(\sum_{k=1}^{N} W_{ik}\phi(x_k^{(l-1)}) + b_i)(\sum_{h=1}^{N} W_{jh}\phi(y_h^{(l-1)}) + b_j)] \\
&= \sum_{k=1}^{N}\sum_{h=1}^{N} \mathbb{E}[W_{ik}W_{jh}]\phi(x_k^{(l-1)})\phi(y_h^{(l-1)}) + \sum_{k=1}^{N} \mathbb{E}[W_{ik}b_j]\phi(x_k^{(l-1)}) \\
&+ \sum_{h=1}^{N} \mathbb{E}[W_{jh}b_i]\phi(y_h^{(l-1)}) + \mathbb{E}[b_i b_j] \\
&= \sum_{k=1}^{N}\sum_{h=1}^{N} \delta_{ij}\delta_{kh}\sigma_w^2/N\phi(x_k^{(l-1)})\phi(y_h^{(l-1)}) + \delta_{ij}\sigma_b^2 \\
&= \delta_{ij}[\sigma_w^2/N \sum_{k=1}^{N} \phi(x_k^{(l-1)})\phi(y_k^{(l-1)}) + \sigma_b^2].
\end{aligned}$$

The main point of the mean-field approximation is replacing the distribution $p(x_1^{(l)}, \ldots, x_N^{(l)}, y_1^{(l)}, \ldots, y_N^{(l)})$ by a Gaussian whose first two moments match that of original one. Using this approximation and letting $N \to \infty$, we obtain

$$\begin{aligned}
\mathbb{E}[x_i^{(l)} x_j^{(l)}] &= \delta_{ij}\left(\sigma_w^2 \int \phi(z)^2 \mathcal{N}(z \mid 0, q^{(l-1)})dz + \sigma_b^2\right), \\
\mathbb{E}[x_i^{(l)} y_j^{(l)}] &= \delta_{ij}\left(\sigma_w^2 \int \phi(z_1)\phi(z_2)\mathcal{N}([z_1, z_2] \mid 0, \begin{pmatrix} q^{(l-1)} & r^{(l-1)} \\ r^{(l-1)} & q^{(l-1)} \end{pmatrix})dz + \sigma_b^2\right),
\end{aligned}$$

if we assume

$$\mathbb{E}[x_i^{(l-1)} x_j^{(l-1)}] = \mathbb{E}[y_i^{(l-1)} y_j^{(l-1)}] = \delta_{ij} q^{(l-1)} \quad \text{and} \quad \mathbb{E}[x_i^{(l-1)} y_j^{(l-1)}] = \delta_{ij} r^{(l-1)}$$
$$(q^{(l-1)}, r^{(l-1)} \in \mathbb{R}\text{: independent of } i \text{ and } j).$$

Note that this assumption includes no correlation between most pairs of variables, which implies their independence since we are regarding the distribution as Gaussian.

We can summarize these results as recurrence relations of q and r:

$$q^{(l)} = \sigma_w^2 \int \phi(z)^2 \mathcal{N}(z \mid 0, q^{(l-1)}) dz + \sigma_b^2,$$
$$r^{(l)} = \sigma_w^2 \int \phi(z_1)\phi(z_2) \mathcal{N}([z_1, z_2] \mid 0, \begin{pmatrix} q^{(l-1)} & r^{(l-1)} \\ r^{(l-1)} & q^{(l-1)} \end{pmatrix}) dz_1 dz_2 + \sigma_b^2.$$

Then we can compute the depth scale (of correlation), defined by convergence speed of $c := r/q$ to its stable point c^*. We write the recurrence relations as

$$q^{(l)} = g_q(q^{(l-1)}, c^{(l-1)}) \quad \text{and} \quad c^{(l)} = g_c(q^{(l-1)}, c^{(l-1)}),$$

and assume that there exist $q^* := \lim_{l\to\infty} q^{(l)}$ and $c^* := \lim_{l\to\infty} c^{(l)}$. Then, at the large l, the dynamics of c can be approximated as

$$c^{(l)} - c^* \propto \exp[-\frac{l}{\delta}] \tag{5.1}$$
$$\text{where} \quad \delta = -\log \left[\frac{\partial}{\partial c} g_c(q^*, c) \Big|_{c=c^*} \right]. \tag{5.2}$$

This δ is the depth scale (of correlation), what we want to compute.

In terms of the ordered and chaotic phases, $c^* = 1$ means ordered phase and $c^* < 1$ means chaotic phase. In the ordered phase, any two different signals get close together infinitesimally during propagation. This means that any information represented as the difference of two signals is lost by propagation. On the contrary, in the chaotic phase, any two signals with arbitrary small difference are pulled apart. The propagation is sensitive to any nonzero noises, in that they are enlarged during propagation. Both behaviours are undesirable for neural networks to learn reasonable functions.

With respect to various types of networks, a number of works (a) formulated equations for propagating (i.e. law of evolution of microscopic parameters which is probabilistic), (b) derived the recurrence equations for macroscopic parameters which are deterministic, then (c) computed the depth scale of the network as negative logarithm of time constant of convergence of macroscopic parameters, and finally (d) confirmed the consistency with actual trainability.

On the contrary, we compute the evolution of macroscopic parameters numerically instead of deriving it theoretically, by sampling the probabilistic evolution of microscopic parameters. Then we estimate the depth scale from evolution of macroscopic parameters we obtained. Our algorithm consists of three steps:

1. Calculate the equilibrium point (q^*, c^*) numerically, by sampling evolution of (q, c). This is accomplished by randomly generating a pair of vectors which has squared norm q and overlap c, propagating them through a unit architecture whose weight is randomly sampled, calculate new q and c from the output (with taking average), and repeating this procedure enough times. Note that q diverges in some cases; we are

able to deal with such cases numerically, by replacing infinity by a very large finite value. We also note that sampled dynamics of c contains some noises. To determine c^* with high precision, we need to lessen this noise by using larger N and increasing number of samples.

2. Sample dynamics of c near (q^*, c^*), and approximate $g_c(q^*, c)$ in (5.2) by curve fitting. Here what we want to know is the shape of the function $c \mapsto g_c(q^*, c)$ at $c \approx c^*$. To do so, we collect a lot of samples $(c, g_c(q^*, c))$, and fit them by some parametric function. Note that the function should be differentiable, since we need to evaluate its derivative at $c = c^*$ in next step.

3. Compute the depth scale δ with the equation 5.2.

In this algorithm, we have to take some large but finite N. We set N as 1000 in the experiment with simple network architectures, and 784 in the experiment with random network architectures. For other hyperparameters and other implementation details of the algorithm, see the appendix 5.A. See the next section for experimental details aside from the estimation algorithm.

5.4 Experiment

In the previous section, we established the method for estimating the depth scale of given neural networks numerically. Now we are to examine the precision of the estimation and apply it to predicting trainability. *

5.4.1 With simple network architectures

First, we confirmed how well our method estimates the depth scale, by using simple networks whose depth scales can be derived theoretically in previous works; fully-connected network (FCN), convolutional neural network (CNN), and recurrent neural network (RNN) [2, 3, 4]. With each of these networks, we compared the depth scale obtained by our numerical method to theoretical values reported by them. Figure 5.1 shows the result of comparison; (d) for FCN, (e) for CNN, and (f) for RNN. All of them demonstrate the consistency with theoretical counterparts (a)–(c).

5.4.2 With various random network architectures

Next, we examined the applicability of the statement: the depth scale can predict the trainability of neural networks. With the simple network described above, this statement has been found to be true. Our interest is that what range of neural networks this statement is valid for.

We generated the unit neural architecture randomly with the following procedure:

*We performed all computations for estimating depth scales with no GPU. We performed all experiments for evaluating trainability with a single GPU (Tesla K80).

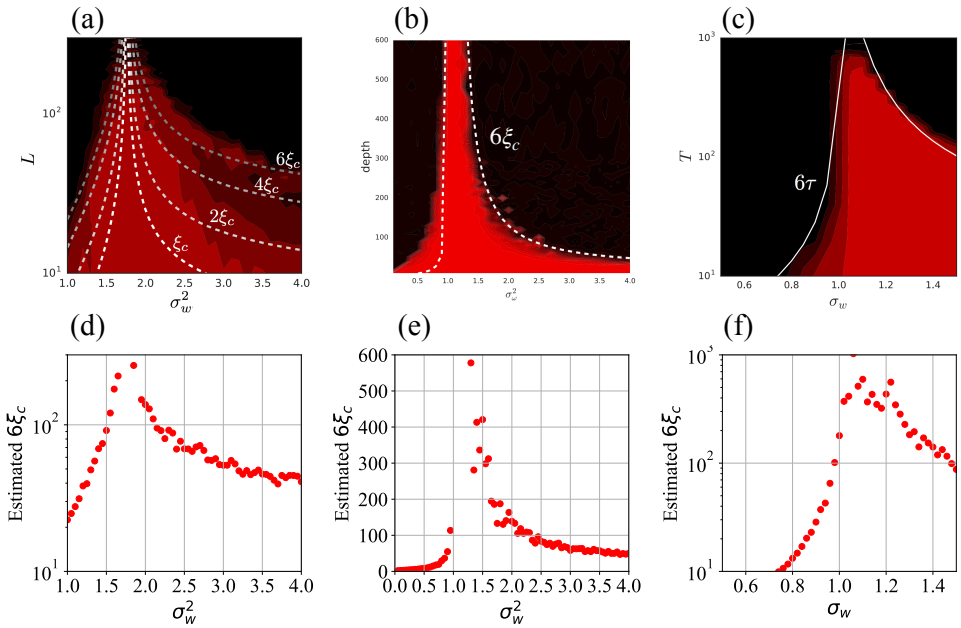

Figure 5.1: (a)–(c) Results reprinted from previous works describing that theoretical depth scale (white lines; multiplied by 6) and actual trainability (red heatmaps) matches well, for (a) FCN, (b) CNN, and (c) RNN [2, 3, 4]. (d)–(f) Estimated depth scale by our numerical methods (multiplied by 6, like (a)–(c)), for (d) FCN, (e) CNN, and (f) RNN. These are consistent with theoretical depth scale described in (a)–(c). In each subfigure, horizontal axis represents hyperparameter σ_w^2 or σ_w, and vertical axis represents number of layers.

- Generate a random directed acyclic graph (DAG) which has nodes $1, 2, \ldots, V$ and E edges. Make sure that there is only one source node 1 (i.e. node with 0 indegree) and sink node V (i.e. node with 0 outdegree). See the appendix 5.A for the algorithm for generating such DAGs. We denote the set of edges of the DAG by $\mathcal{E}$.

- This DAG corresponds to a unit architecture, and each node in the DAG represents a layer. We set randomly the type of the layer function $f^{(v)} : \mathbb{R}^{M^{(v)}} \to \mathbb{R}^N$ among $\{\text{fully-connected}, \text{layer-normalization}, \text{dropout}\}$, the type of element-wise activation $\phi^{(v)} : \mathbb{R}^N \to \mathbb{R}^N$ among $\{\text{ReLU}, \tanh\}$ and the way of aggregation $g^{(v)} : \mathbb{R}^{d^{(v)}N} \to \mathbb{R}^{M^{(v)}}$ of each node v among $\{\text{concatenation}, \text{mean-pooling}, \text{max-pooling}\}$, where $d^{(v)} := \sharp\{u \mid (u \to v) \in \mathcal{E}\}$. Note that $M^{(v)}$ equals to N or $d^{(v)}N$, depending on the type of the layer. The detailed way of randomly choosing them is described in the appendix 5.A.

- With given input $\boldsymbol{x} \in \mathbb{R}^N$, we compute the output of the unit architecture $\boldsymbol{y} \in \mathbb{R}^N$, by computing hidden activations $\boldsymbol{h}^{(v)}$ of all nodes with following recurrence equations:

$$\boldsymbol{y} = \boldsymbol{h}^{(V)},$$
$$\boldsymbol{h}^{(v)} = \begin{cases} \phi^{(v)}\left(f^{(v)}\left(g^{(v)}\left((\boldsymbol{h}^{(u)})_{(u\to v)\in\mathcal{E}}\right) \mid w^{(v)}\right)\right) & (\text{if} \quad v > 1) \\ \phi^{(v)}\left(f^{(v)}\left(\boldsymbol{x} \mid w^{(v)}\right)\right) & (\text{if} \quad v = 1) \end{cases} \in \mathbb{R}^N.$$

  Note that each layer function $f^{(v)}$ has its learnable parameter $w^{(v)}$.

Figure 5.2 shows 15 randomly generated unit architectures, with $V = 5$ and $E = 7$, which we used in the experiment.

5.4.3 Initialization with long-tailed weight distribution

In the mean-field theory of signal propagation in neural networks, we normally assume that the weight values are sampled from distributions which have finite mean and finite variance. This is because such distributions are used at initialization in practice. However, it is suggested that the statistics of weight matrices in neural networks dynamically change during training, in that their spectral densities become long-tailed[70], and therefore the ordinary mean-field theory could be no more valid for trained neural networks. Moreover, Teramae et al. [71] supports that the spontaneous asynchronous activity in recurrent spiking neural networks persist robustly when its weight follows long-tailed distribution, which suggests the usefulness of having long-tailed weights for retaining meaningful activity for long time.

We hypothesize that the long-tailed property of weight distribution can enhance the information propagation in feed-forward neural networks. To quantify the information propagation, we estimate the depth scale numerically. (We also try to derive the depth scale theoretically in the case of long-tailed weights. See the appendix 5.B.)

We also examined whether networks initialized with long-tailed distribution have better actual trainability, by training a real task. We let them learn the MNIST classification task for 1 epoch and evaluated their training accuracy. We used FCNs which have

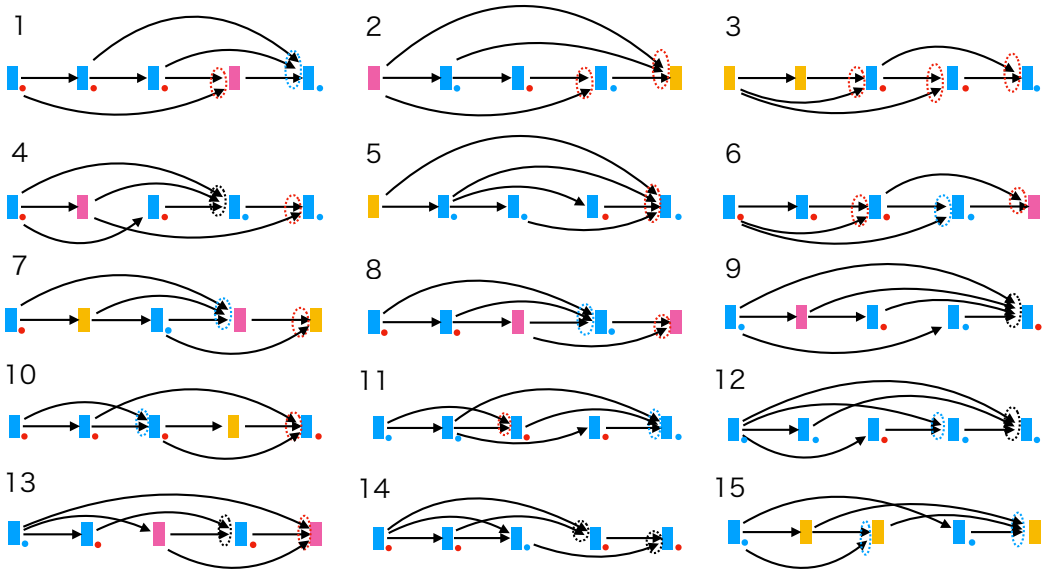

Figure 5.2: Randomly generated unit neural architectures with $V = 5$ and $|\mathcal{E}| = 7$. Blue, yellow and pink nodes represent fully-connected layer, dropout layer and layer-normalization layer, respectively. Blue and pink small filled circles mean ReLU and tanh activation, respectively. Ways of aggregation are described by broken ellipses; blue for mean-pooling, red for max-pooling, and black for concatenating.

hidden layers with size 784 and tanh-like $\mathrm{erf}(x/\sqrt{2})$ activation, and optimized them with vanilla stochastic gradient decent (SGD).

5.5 Result

5.5.1 With various network architectures

With various unit network architectures generated randomly, we examined whether the depth scale appropriately predicts its trainability. Actually, we trained a neural network which consists of L given random network architectures (and one additional fully-connected output layer) and whose weights are initialized by scale σ_w^2, to learn the MNIST classification task, for various values of L and σ_w^2. Since our interest is trainability, we measured the training accuracy, following previous works shown in Figure 5.1(a)–(c). Our hypothesis is that any network which consists of repeated unit architectures and whose weights are initialized with scale σ_w^2 has high trainability, provided that its number of repetition L of the unit architectures is relatively smaller than its depth scale (which is determined by the unit architecture and initialization scale σ_w^2).

Figure 5.3 shows results for 15 distinct unit architectures depicted in Figure 5.2. White lines show estimated depth scales multiplied by 6 (as well as Figure 5.1), and red heatmaps show the trainability measured by training accuracy. They coincide very well in most of 15 cases. We observe that there are some discrepancy between them in the case

11 and 12; when σ_w^2 is larger than $\sim 10^{0.3}$, actual trainability shown by heatmaps degrade with larger σ_w^2, although numerical depth scales depicted by white lines do not get worse.

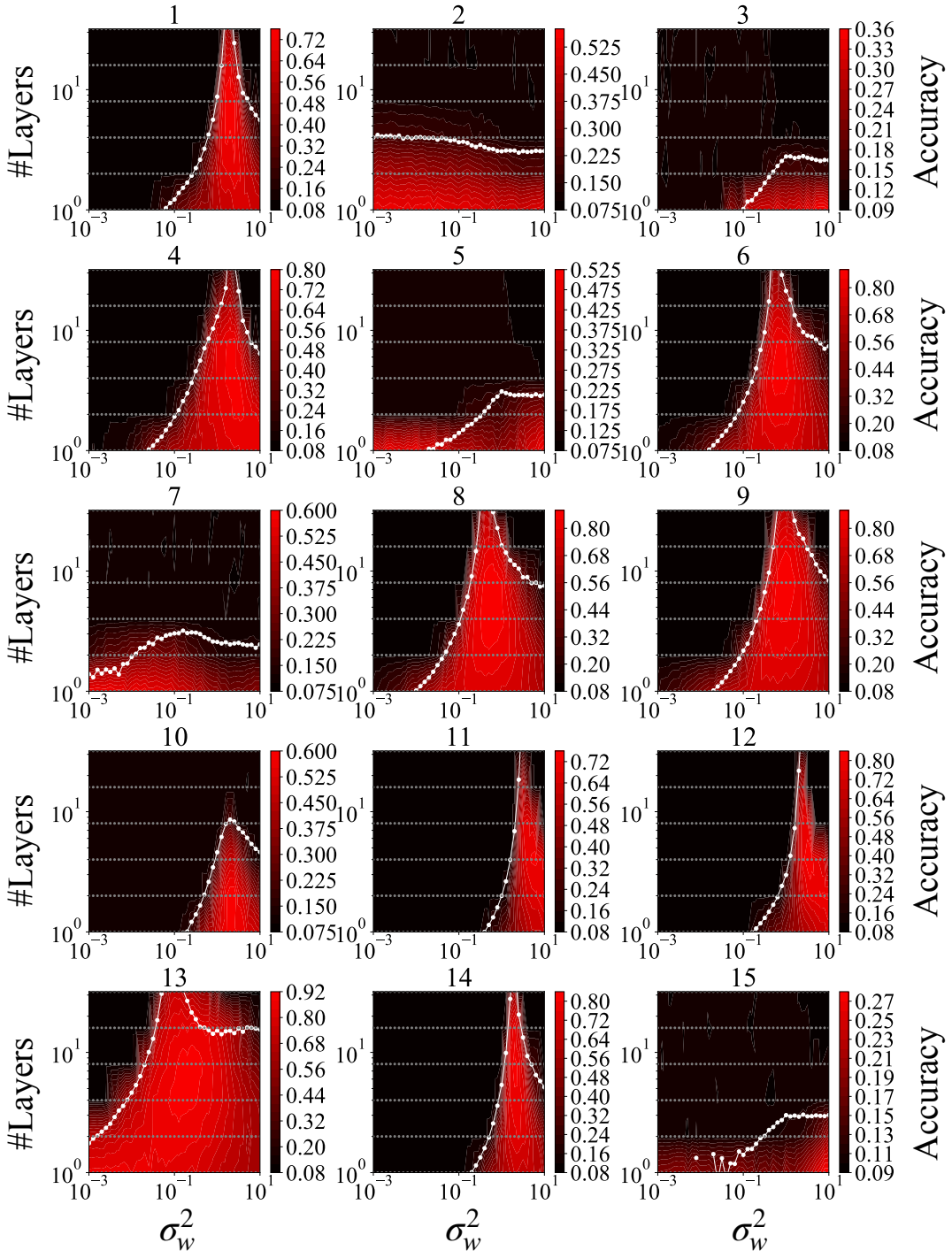

Figure 5.3: Estimated depth scales and actual trainability, in case with 15 distinct unit architectures. Horizontal axes represent weight scale hyperparameter σ_w^2, and vertical axes represent number of unit architectures. σ_b^2 is fixed to 0.05. White lines show estimated depth scale multiplied by 6 (as well as Figure 5.1), and red heatmaps show the trainability measured by training accuracy of MNIST task. We interpolated results at gray points to draw heatmaps. Depth scales and heatmaps coincide very well in most of 15 cases. Number of epochs: 1. Mini-batch size: 100. Optimization: vanilla SGD with learning rate 10^{-3} or 10^{-4} (we adopted better one in each training).

5.5.2 Initialization with long-tailed weight distribution

Then we examined that how long-tailed weight distribution affects the propagation characteristics and trainablity.

Figure 5.4 shows that numerically estimated depth scale of FCN initialized with different weight distributions; Gaussian, Student's t with $\nu = 2$, Cauchy, and Student's t with $\nu = 0.5$, where ν is degrees of freedom. Note that Gaussian distribution and Cauchy distribution are equivalent to Student's t with $\nu = \infty$ and $\nu = 1$ respectively, and that smaller ν yields more long-tailed Student's t-distribution. The figure shows that the more long-tailed the distribution used in initialization is, the wider the range of σ_w with large depth scale. This means that large depth scale can be achieved robustly against the choice of σ_w when we use long-tailed weight distribution.

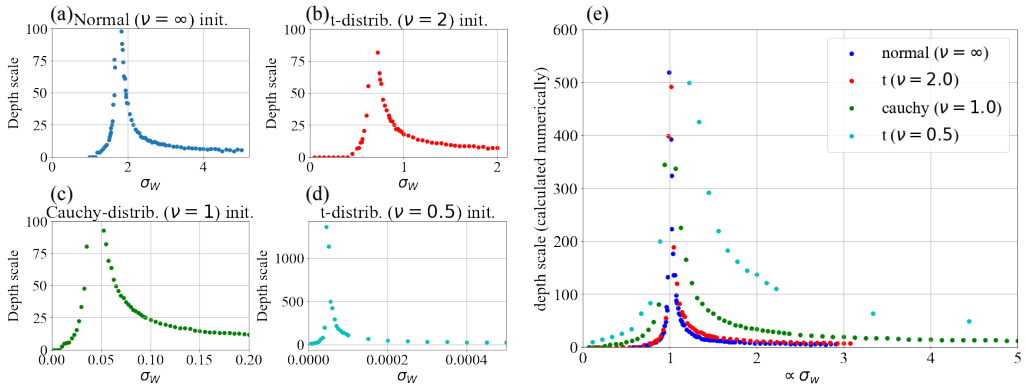

Figure 5.4: Numerically estimated depth scale of FCN initialized with different weight distributions; Gaussian distribution (blue), Student's t-distribution with $\nu = 2$ (red), Cauchy distribution (green) and Student's t-distribution with $\nu = 0.5$ (cyan), where ν represents degrees of freedom. Note that Gaussian distribution and Cauchy distribution are equivalent to Student's t-distribution with $\nu = \infty$ and $\nu = 1$, respectively. (e) Plots of (a)–(d) are overlaid. Horizontal axes are rescaled so that peaks of all plots go to 1.

Next, we examined whether networks initialized with long-tailed distribution have better trainability, by training a real task. Figure 5.5 shows that the accuracy of neural networks. In short, the figure shows that, the more long-tailed distribution we adopt for initialization, the worse the performance.

5.6 Discussion

5.6.1 Depth scale and trainability

We confirmed that depth scale can predict actual trainability well, with respect to a wide range of randomly generated neural network architectures.

However, some cases have to be handled with caution. In the case 11 and 12 in

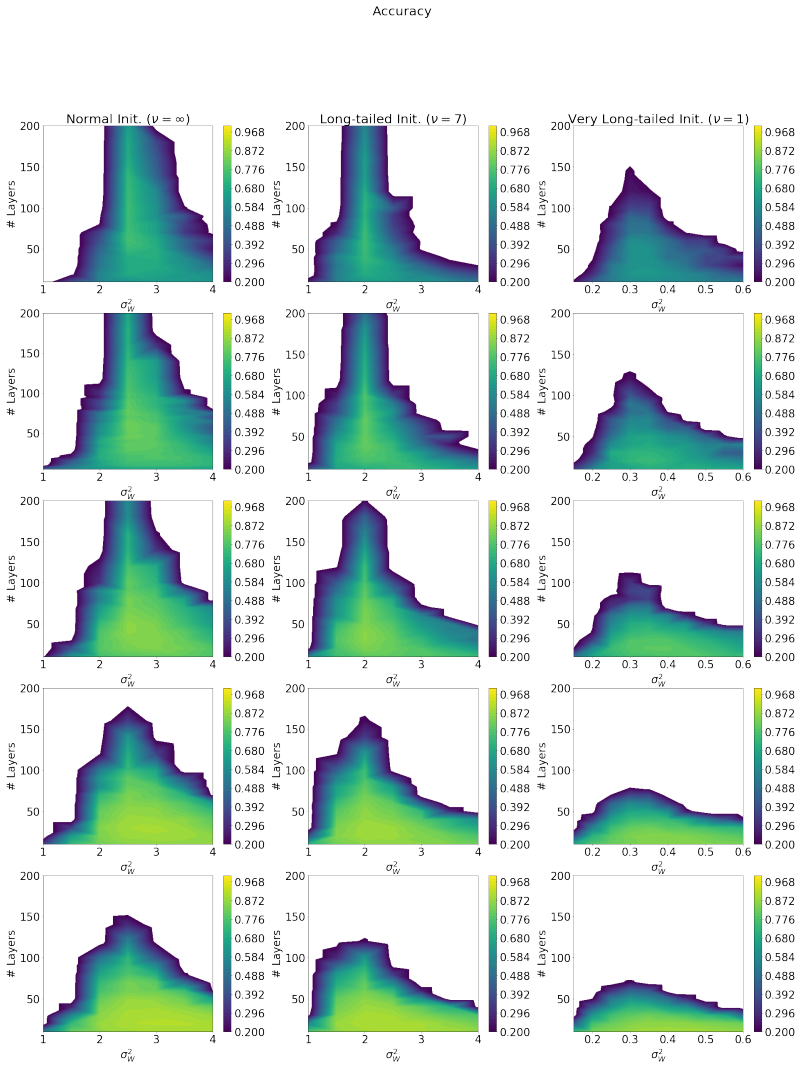

Figure 5.5: Trainability of FCNs, initialized with different weight distributions; (left column) Gaussian distribution, (middle column) Student's t-distribution with $\nu = 7$, (right column) Student's t-distribution with $\nu = 1.0$. Results with different values of learning rate η of SGD are shown; $\eta = 0.0005, 0.001, 0.002, 0.005, 0.01$ from top row to bottom row. In each subfigure, horizontal axis represents σ_w^2 and vertical axis represents number of layers. In each heatmap, region with < 0.2 accuracy is shown by white.

Figure 5.3, we observed some gaps between depth scale curves and actual trainability. A common feature of these two unit structures only is that they contain a simple FCN with ReLU activation as sub-architecture (see architectures 11 and 12 in Figure 5.2). In fact, the discrepancy is also observed in case with just a ReLU FCN. That is, when σ_w^2 becomes large, the depth scale remains quite large but the trainability drops. In this ReLU FCN case the depth scale can be analytically determined, and indeed it also remains quite large in the chaotic (large σ_w^2) region. Then, the question is why the depth scale (numerically and analytically computed) and actual trainability differ.

Other necessary conditions for successful training

The point is that large depth scale is necessary but not sufficient for successful learning. For example, in the situation we described above, we noticed that the norm q diverges to infinity as signals propagate though layers in such large σ_w^2 situation. Apparently this prevents successful learning, so we should consider the condition that the signal norm q converges to a finite value as another necessary condition for successful training. Other than this, it seems to be essential that depth scale of backpropagation (not forward propagation) is also large.

Sufficient conditions for successful training

Although we were able to confirm the usefulness of depth scale for predicting trainability in various neural networks, it is remained untouched that providing theoretical guarantee which ensures the predictability. To ensure the relationship between the depth scale and the trainability more firmly, we have to analyze theoretically not only the propagation dynamics of initialized random networks but also that of trained networks. To do so, it is possible that the recently developed theory with neural tangent kernel (NTK) would be helpful [68, 69]. In this theory, we can treat learning dynamics of neural networks theoretically, by assuming that parameters' change during learning is sufficiently small and we can regard the effect of parameter change to network function as linear. We think that this direction would be important in future works.

5.6.2 Initialization with long-tailed distribution

We found that in the case with long-tailed weight distribution the numerically estimated depth scale becomes more robust but the actual trainability becomes worse. Here we discuss how we can understand these results and what should be inspected further.

The first thing to be considered is the possibility that numerical estimates of depth scales are broken. In the experiment using random unit architectures, the numerically estimated depth scales agreed well with the actual trainability, so it seems that both the accuracy of the depth scale estimation and the predictability of trainability from the depth scales are fine, but in the present case, we cannot say so. The failure of the predictions of trainability could be due to that the layer width N or the number of samples drawn are too small. In this case, increasing them will help convergence to the accurate depth scale.

Other possibility is that even we set them as any large values the numerical algorithm fails to estimate depth scale. This possibility might be eliminated if we can theoretically give a guarantee of the estimation error of the algorithm.

And another thing to be considered is that, as discussed above, even if the numerical estimation of the depth scale is perfect, a large depth scale is not a sufficient condition for successful training. It is important to find other necessary conditions as well as sufficient conditions. Again, depth scale of backpropagation (not forward propagation) seems to be one candidate.

5.7 Conclusion

In this work, we first established the numerical method for estimating depth scale for forward propagation, which is available regardless of network architecture. Then we examined the applicability of the notion of depth scale of neural networks for predicting their trainability, and found that it works very well in a certain range of neural networks with complex architectures. Our work would benefit any deep learning practitioners, in that it suggests how to find optimal initialization strategy of their networks; they can choose their initialization scheme which maximizes depth scales, by numerically estimating them using our framework.

However, we found that it does not work well in the case with initialization with long-tailed weight distributions. Although such initialization is not commonly used so far, it suggests future researches for this framework.

5.A Algorithm details

5.A.1 Algorithm for generating random DAGs

In the experiment with random network architectures, we need to generate a DAG which has the following properties:

- The DAG has V nodes and E edges.
- We call its V nodes by node $1, 2, \ldots, V$. Then, only the node 1 is a source node (whose indegree is 0) and only the node V is a sink node (whose outdegree is 0).

We generated such DAGs by Algorithm 1. In this algorithm, basically we repeat adding an edge in random order, until E edges are added. However, we skip adding an edge if the second property will become unachievable after doing so. We determine this by checking minimum number of further edges needed to suffice demands for all nodes; all intermediate nodes must have nonzero indegree and outdegree. We describe this checking algorithm in `minimum_number_of_edges_needed` function in Algorithm 1. To minimize additional edges, we should append edges which effectively resolve these demands. Let $i < j$ and assume that node i needs any outdegree and node j needs any indegree but neither are achieved yet. In this case, if we append an edge from i to j, we can resolve two demands at the same time. In the algorithm, we count the number of edges needed when we greedily do that way whenever we can.

5.A.2 Algorithm for choosing layer properties

In the experiment with random network architectures, we need to choose (a) types of layers, (b) activation functions, and (c) way of aggregation of inputs, randomly. We did the following. For (a), we chose them among fully-connected, layer-normalization and dropout, with probability 2/3, 1/6 and 1/6, respectively. For (b), we chose them among ReLU and tanh with probability 1/2, for fully-connected layers. We did not add any activation function for layers other than fully-connected. For (c), we selected them among concatenation, mean-pooling and max-pooling with probability 1/3 for fully-connected layers; for not-fully-connected layers, we selected them among mean-pooling and max-pooling with probability 1/2. Additionally, if we got a unit neural architecture which did not contain any fully-connected layers, we discarded it and re-selected randomly, because such unit architecture has few learnable parameters and therefore it is not realistic.

Algorithm 1 Generating random DAG which has one source node and one sink node

Require: V: number of nodes, $1 \leq V$
Require: E: number of edges, $V - 1 \leq E \leq V(V-1)/2$
Ensure: G is a DAG which meets the condition.

$candidate\_edges \leftarrow \{(i, j) \mid 1 \leq i < j \leq V\}$
$cnt \leftarrow 0$
$G \leftarrow$ (graph with V nodes and no edge)
for $(i, j) \in candidate\_edges$ in random order **do**
    if `MINIMUM_NUMBER_OF_EDGES_NEEDED`$(G, (i, j)) > E$ **then**
        continue
    end if
    append an edge from i to j, to graph G
    if (number of edges in G) $= E$ **then**
        break
    end if
end for
return G

function `MINIMUM_NUMBER_OF_EDGES_NEEDED`$(G, (i, j))$
    // If we append an edge $i \to j$ to G,
    // how many additional edges are needed for G to be valid?
    $seq \leftarrow$ (empty list)
    for $k = 1, 2, \ldots, V$ **do**
        if $1 < k \neq j$ and node k of G has 0 indegree **then**
            append -1 to seq
        end if
        if $i \neq k < V$ and node k of G has 0 outdegree **then**
            append $+1$ to seq
        end if
    end for
    $count \leftarrow 0$, $capacity \leftarrow 0$
    for s in seq **do**
        if $s = +1$ **then**
            $count \leftarrow count + 1$
            $capacity \leftarrow capacity + 1$
        else
            if $capacity > 0$ **then**
                $capacity \leftarrow capacity - 1$
            else
                $count \leftarrow count + 1$
            end if
        end if
    end for
    return $count$
end function

5.A.3 Hyperparameters and implementation details of algorithm for estimating depth scale

Algorithm

The algorithm 2 describes how we estimated depth scales numerically.

Hyperparameters

The algorithm 2 has several hyperparameters. In the experiment with random unit architectures, we chose them as following. For computing equilibrium of (q, c):

- Number of samples of weights of unit architecture n: 10.
- Number of iteration h: 300.
- Number of samples of new (q', c') in each iteration s: 1.

And for sampling evolution of c under $q = \hat{q}^*$:

- How to choose $c_1, \ldots, c_u$ near $\hat{c}^*$ (by `get_neighbour`): we chose $u = 100$ points, by set $c_1 = \max\{-1, \hat{c}^* - 0.1\}$, $c_u = \min\{+1, \hat{c}^* + 0.1\}$ and let $c_1, \ldots, c_u$ to be equally spaced.
- Number of samples of weights of unit architecture m: 10.
- Number of samples of new (q', c') in each iteration v: 100.

Additionally, we used cubic polynomial for fitting the recurrence curve $c \mapsto g_c(q^*, c)$ at $c \approx c^*$.

5.B Theoretical discussion

5.B.1 Theoretical discussion for depth scale of networks whose weights have long-tailed distribution

The mean-field theory of signal propagation in neural networks, developed by [2], is based on a simple proposition.

> Let W be a random matrix whose each element follows a distribution with zero mean and σ_W^2/N variance, and let $\boldsymbol{b}$ be a random vector whose each element follows a distribution with zero mean and σ_b^2 variance. We define f by

Algorithm 2 Estimating depth scale

Require: $p(f)$: distribution of unit architecture function f

Ensure: *depth_scale* is an estimated value of the depth scale of the network which consists of sequentially repeated unit architectures (with untied weights)

// Compute equilibrium of (q, c)
Sample $f_1, \ldots, f_n$ from $p(f)$
$(q, c) \leftarrow (1, -0.05)$
for $i = 1, 2, \ldots, h$ **do**
    $q\_list \leftarrow$ (empty list)
    $c\_list \leftarrow$ (empty list)
    for $i = 1, 2, \ldots, n$ **do**
        for $j = 1, 2, \ldots, s$ **do**
            $(q', c') \leftarrow$ `sample_evolution`$(f_i, \hat{q}^*, c)$
            append q' to q_list and c' to c_list
        end for
    end for
    $q \leftarrow$ (mean of q_list), $\quad c \leftarrow$ (mean of c_list)
end for
$(\hat{q}^*, \hat{c}^*) \leftarrow (q, c)$

// Sample evolution of c under $q = \hat{q}^*$
$c_1, \ldots, c_u \leftarrow$ `get_neighbour`$(\hat{c}^*)$
$gc\_list \leftarrow$ (empty list)
Sample $f_1, \ldots, f_m$ from $p(f)$
for $i = 1, 2, \ldots, m$ **do**
    for $j = 1, 2, \ldots, u$ **do**
        for $k = 1, 2, \ldots, v$ **do**
            $(q', c') \leftarrow$ `sample_evolution`$(f_i, \hat{q}^*, c_j)$
            append (c, c') to gc_list
        end for
    end for
end for
$\hat{g} \leftarrow$ `fit_polynomial`(gc_list)
$depth\_scale \leftarrow -1/\log(\hat{g}'(\hat{c}^*))$
return *depth_scale*

function SAMPLE_EVOLUTION(f, q, c)
    randomly generate $\boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathbb{R}^N$ s.t. $\|\boldsymbol{x}_1\|^2/N = \|\boldsymbol{x}_2\|^2/N = q$ and $\boldsymbol{x}_1 \cdot \boldsymbol{x}_2/N = qc$
    $\boldsymbol{y}_1 \leftarrow f(\boldsymbol{x}_1), \quad \boldsymbol{y}_2 \leftarrow f(\boldsymbol{x}_2)$
    $q' \leftarrow \|\boldsymbol{y}_1\|\|\boldsymbol{y}_2\|/N, \quad c' \leftarrow \boldsymbol{y}_1 \cdot \boldsymbol{y}_2/(\|\boldsymbol{y}_1\|\|\boldsymbol{y}_2\|)$
    return (q', c')
end function

$f(\boldsymbol{x}) := \varphi(W\boldsymbol{x} + \boldsymbol{b})$. Then

$$\begin{aligned} f(\boldsymbol{x}) \cdot f(\boldsymbol{y}) &\approx N\mathbb{E}_{W,\boldsymbol{b}}[\varphi(\sum_{j=1}^{N} W_{ij}x_j + b_i)\varphi(\sum_{j=1}^{N} W_{ij}y_j + b_i)] \\ &= NI_2 \begin{pmatrix} \sigma_W^2/N\|\boldsymbol{x}\|^2 + \sigma_b^2 & \sigma_W^2/N\boldsymbol{x}\cdot\boldsymbol{y} + \sigma_b^2 \\ & \sigma_W^2/N\|\boldsymbol{y}\|^2 + \sigma_b^2 \end{pmatrix} \\ \text{where} \quad I_2(\Sigma) &= \int d\boldsymbol{z}\ \varphi(z_1)\varphi(z_2)\mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \Sigma). \end{aligned}$$

For this statement to be valid, the most essential thing is the following:

Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N$ be constant vectors, and let $\boldsymbol{w} \in \mathbb{R}^N$ be a random vector whose each element follows a distribution with zero mean and σ^2/N variance. Then when $N \to \infty$ we get

$$(\boldsymbol{w}^T\boldsymbol{x},\ \boldsymbol{w}^T\boldsymbol{y}) \sim \mathcal{N}\left(\boldsymbol{0},\ \frac{\sigma^2}{N}\begin{pmatrix} \|\boldsymbol{x}\|^2 & \boldsymbol{x}\cdot\boldsymbol{y} \\ \boldsymbol{x}\cdot\boldsymbol{y} & \|\boldsymbol{y}\|^2 \end{pmatrix}\right)$$

by generalized central limit theorem.

What we are interested is the case that the distribution followed by each element of the random vector $\boldsymbol{w}$ is long-tailed so that its variance is undefined. In such case, we obtain the following proposition.

Let $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^N$ be constant vectors, and let $\boldsymbol{w} \in \mathbb{R}^N$ be a random vector whose each element follows Cauchy$(0, \gamma)$, that is, Cauchy distribution with zero mean and γ scale parameter. Then, regardless of N,

$$\boldsymbol{w}^T\boldsymbol{x} \sim \text{Cauchy}(0, \|\boldsymbol{x}\|_1\gamma), \qquad \boldsymbol{w}^T\boldsymbol{y} \sim \text{Cauchy}(0, \|\boldsymbol{y}\|_1\gamma)$$

holds. Although the joint probability distribution of $(u, v) := (\boldsymbol{w}^T\boldsymbol{x},\ \boldsymbol{w}^T\boldsymbol{y})$ is unknown, its characteristic function is given by

$$\Psi(\boldsymbol{t}) := \int dudv\ p(u, v)\exp[i(t_1u + t_2v)] = \exp[-\gamma\|t_1\boldsymbol{x} + t_2\boldsymbol{y}\|_1].$$

Note that this characteristic function is different from that of two-dimensional Cauchy distribution $\exp[-\|\Sigma^{1/2}\boldsymbol{t}\|_2]$.

Chapter 6

Conclusion

In this dissertation, the learning dynamics and the inference (propagation) dynamics of neural networks are analyzed using two macroscopic analysis methods, namely, the statistical mechanical method and the mean-field method. Now we review our works more specifically.

In Chapter 2, we applied the statistical mechanical method to a latest technique for deep learning, weight normalization (WN), and reduced the learning dynamics of a single layer perceptron to a simple three-variable system. Then we compared the systems of order parameters when the standard method and WN are used, and found that the effective learning rate was automatically adjusted in WN. We also showed that while WN is robust to the setting of the learning rate it is sensitive to the weight initialization.

In Chapter 3, we applied the statistical mechanical method to the learning of two-layer perceptron when the number of output elements is general, and derived the dynamics of order parameters. By solving it numerically, we pointed out that the plateau phenomenon does not necessarily occur when there are multiple output elements.

In Chapter 4, we extended the statistical mechanical analysis, in order to investigate the effect of the statistics of training data on the learning dynamics, to a situation where the covariance matrix of the input data distribution is generalized. As a result, the relationship between the eigenvalue distribution of the covariance matrix and the macroscopic dynamics became clear, and it was demonstrated that the plateau phenomenon became less noticeable especially when the eigenvalue distribution had a small scale and a large variance.

In Chapter 5, in contrast to the previous chapters, we conducted research focusing on the signal propagation dynamics in deep neural networks. In previous studies, derivation of the depth scale of signal propagation in random networks using the mean-field method has been performed analytically only for simple networks. We established a numerical method for evaluating the depth scale of arbitrary networks. Furthermore, for networks of various architectures, it was confirmed that the depth scale evaluated by this method and the actual trainability matched well, that is, the method was effective in predicting the trainability.

In this chapter, we review the positions of the above-mentioned studies and discuss

how these studies may influence future theoretical research and industries.

6.1 Position of our research

Neural networks and deep learning, a form of machine learning, have been rapidly used in industry in recent years and have outperformed other machine learning methods. There have been various factors for it. The abundance of available computing resources and data might be one of the reasons. However, it seems most important that a variety of new learning techniques have been developed to enable efficient learning of large networks with many layers. Deep learning is realized not only by just learning a network of many layers literally, but also by being supported by these techniques.

However, in spite of these invention of many learning techniques, deep learning has not become sufficiently clear. Rather, a more confusing situation has arisen than with conventional methods; "How long does it take to learn?", "Is this optimization algorithm more advantageous than that one?", "What is the optimal number of layers and the number of elements in each layer?", "Which method is most suitable for initializing the weights?", etc. In many cases, we can't find out unless we actually do it. In fact, it has been practiced to adjust the enormous number of hyperparameters required for designing and learning neural networks based only on human experience and intuition. In recent years, there have been some methods for automating the search for such hyperparameters (for example, Bayesian optimization, evolutionary computation, and neural architecture search including AutoML). Even with such methods, trial-and-errors remain needed inside them, and the computational cost is still demanded.

In order to improve this situation, it is necessary to understand what is happening in the neural network theoretically. This was the motivation throughout our study. In particular, we have paid attention to macroscopic methods such as the statistical mechanical method and the mean-field methods. This is because we believe that these methods are optimal for modeling what is happening in reality.

6.2 Impacts

6.2.1 Impact on industry

As mentioned in the previous section, the use of neural networks is rapidly expanding in the industrial world. A variety of useful deep learning frameworks (libraries) have been developed, including easy-to-understand tutorials and source codes for learning basic models. Even developers unfamiliar with neural networks and deep learning can easily try something anyway. However, they do not have the know-how to adapt the model to the task appropriately or improve the algorithm or its hyperparameters based on the learning result. When searching on the web, knowledge written by various people based on their experience will be hit, but the situation (tasks and data) faced by them and by the developers might differ, so their information may be useless. In such situations, typically they

resort to search the optimal hyperparameters by force. It is very time-consuming and computationally expensive, so that some might give up on optimization from the beginning. We again note that AutoML-like methods are also demanding too much. Then, how will the situation be resolved if the theoretical understanding of the neural network advances?

It is common that some of the hyperparameter settings make learning very successful, and others do not. We want to know the success or failure of learning without actually learning them. We believe that theories of neural networks will make this possible. In other words, from the theory, we would be able to tell what are recommended or not recommended when designing and learning neural networks. Developers can design a neural network and its training according to their theoretical knowledge without performing the conventional hyperparameter search.

The above is true for all the theoretical methods for neural networks. However, we are particularly interested in macroscopic methods we have used in this dissertation. What we really want is to model the phenomenon which happens in reality. The macroscopic approaches match this demand, because they examine the average behavior, not the worst-case behavior.

6.2.2 Impact on theoretical research

We believe that studies of these directions would be useful in particular and they would be one of the important future direction of theoretical research for neural networks.

Statistical mechanical approach

It is inherently very difficult to grasp learning dynamics of neural networks theoretically, because the input-output relationship of a neural network itself is quite nonlinear, it is parameterized with a huge number of variables, and they change dynamically during learning; the situation is extremely complicated. Therefore, in order to theoretically treat learning of neural networks, it is necessary to consider some simplification. For example, we might consider the case with linear neural networks, or we might build a model which captures the dynamics only in the vicinity of special situations (for example, in the vicinity of singular regions or global solutions). By simplifying the situation, the analysis would become easy, but on the other hand, the properties of the original phenomena that we originally wanted to see are somewhat impaired. In such a situation, the statistical mechanical method would be helpful; it is a prominent method to reduce to a low-dimensional system without impairing the rich behavior of the quite nonlinear phenomenon.

The applicable range of the method is wide. For example, as we analyzed weight normalization in Chapter 2, this method can be applied to other optimization algorithms. For example, by using statistical mechanical methods, learning dynamics with the natural gradient descent method* and the Layer Normalization are analyzed[27, 46]. This method

*The natural gradient descent method[72] is an optimization algorithm that has a theoretical basis in information geometry. Although it speeds up learning greatly in terms of the number of steps, it has a heavy computational cost per step, as it calculates the inverse of the Fisher information matrix (which

provides a framework for comparing and examining innumerably various algorithms, models, and their combinations that have been proposed in recent years.

Finally, we mention a very recent development of our approach. In Chapter 4, we extended existing statistical mechanical methods to analyze the impact of input data statistics on the learning dynamics of nonlinear neural networks. Following this work, a formulation was presented by Goldt et al. that relaxed the assumptions about the statistics of the input data into a more realistic form[73]. They established a formulation based on the manifold hypothesis that realistic data are distributed around a low-dimensional manifold embedded in high-dimensional space, and derived the dynamics of order parameters and generalization loss during learning. This formulation is important for bringing the teacher-student statistical mechanical formulation closer to reality, and further development is expected in the future.

Mean-field approach

The mean-field method discussed in Chapter 5 is a powerful method for analyzing signal propagation in deep neural networks. This method assumes random neural networks and is therefore only applicable a priori to those initialized randomly before the learning starts. However, we believe that the method is likely to be applicable in practice to neural networks during or after learning. The reason for this is twofold. First, We have some observations that it is possible to take into account the correlations between rows of the weight matrices of trained neural networks. Another reason is that the weights of neural network with infinite width vary only slightly from their initial values during training, as shown in the recent theory of neural tangent kernel (NTK)[68, 69]. If the method can be applied to trained neural networks, we believe that it has the potential to solve the necessary and sufficient conditions for successful learning.

6.2.3 Effects on understanding the biological neuronal system

All the theoretical studies of neural networks, including the study of learning and inference dynamics described in this dissertation, are considered to have potential applications in the field of neuroscience. Of course, the majority of research has been conducted without being aware of neuroscience. However, it offers a very good opportunity for looking at neuroscience in a multi-faceted way. Personally, I have thought that it is essential to fully understand artificial neural networks first as a preparation step for understanding neuroscience.

6.3 Conclusion

Anyway, neural networks and deep learning are extremely interesting phenomena from the engineering, mathematical, and even biological perspectives, and I cannot take my eyes off them for a moment, both as a researcher and an engineer.

takes $O(N^3)$ time if done naively, or $O(N^2)$ if done sequentially) in each step.

Appendix A

On input statistics of teacher-student setup

Here we discuss conditions allowed for input statistics in teacher-student formulation used in Chapters 2-4.

A.1 Case with unit covariance

In Chapters 2 and 3, we set up the input distribution $p(\boldsymbol{\xi})$ as $\mathcal{N}(0, 1/N\, I_N)$ and $\mathcal{N}(0, I_N)$, respectively. We then introduced the variables $x_i := \boldsymbol{\xi}^T \boldsymbol{J}_i$ and $y_n := \boldsymbol{\xi}^T \boldsymbol{B}_n$, where $\boldsymbol{J}_i$ is i-th weight vector in the student's first layer and $\boldsymbol{J}_n$ is n-th weight vector in the teacher's first layer. The main point we used there is that $(x_1, \ldots, x_K, y_1, \ldots, y_M)$ follows a multivariate Gaussian distribution when $N \to \infty$. Actually it holds when the input distribution $p(\boldsymbol{\xi})$ is a multivariate Gaussian as we assumed there (even if N is finite). However, it is not only the case we can deal with. As Saad et al. set up [9], we can consider the case in which each ξ_i follows an identical distribution $p(\xi)$ (not necessarily a Gaussian) which has zero mean and finite covariance independently. In that case, $\boldsymbol{\Theta} := (x_1, \ldots, x_K, y_1, \ldots, y_M)$ follows a multivariate Gaussian distribution, if certain conditions are satisfied. What is the conditions?

Considering characteristic function, $\boldsymbol{\Theta}$ follows a multivariate Gaussian if and only if its any projection

$$
\begin{aligned}
&a_1 x_1 + \cdots + a_K x_K + b_1 y_1 + \cdots + b_M y_M \\
&= (a_1 J_{11} + \cdots + a_K J_{K1} + b_1 B_{11} + \cdots + b_M B_{M1})\xi_1 \\
&\quad + \cdots + (a_1 J_{1N} + \cdots + a_K J_{KN} + b_1 B_{1N} + \cdots + b_M B_{MN})\xi_N \\
&= c_1 \xi_1 + \cdots + c_N \xi_N
\end{aligned}
$$

follows a Gaussian. Here we need weighted sum version of the central limit theorem. By Lindeberg's central limit theorem [74], the following condition is sufficient for the sum to converge to Gaussian (in distribution): for any $i = 1, 2, \ldots, N$, $\lim_{N\to\infty} \sum_{j=1}^{N} c_j^2 / c_i^2 = \infty$

holds. Informally, this means that the distribution of J_{ij} and B_{in} is not skewed so that a particular coefficient c_i does not have $O(N^0)$ influence when $N \to \infty$. For example, we let $\boldsymbol{J}_1 = (1, 0, 0, \ldots, 0)$. In this case, $x_1 = \boldsymbol{\xi}^T \boldsymbol{J}_1 = \xi_1$ follows $p(\xi)$ which is not necessarily a Gaussian.

A.2 Case with non-unit covariance

On the contrary, in Chapter 4, we dealt with different setting: we considered the case that $p(\boldsymbol{\xi})$ has zero mean and finite covariance matrix Σ. We defined $x_i^{(e)} := \boldsymbol{\xi}^T \Sigma^{(e)} \boldsymbol{J}_i$ and $y_n^{(e)} := \boldsymbol{\xi}^T \Sigma(e) \boldsymbol{B}_n$. Again, the main point we used there is that all elements in any finite subset of $\{x_i^{(e)} \mid 1 \leq i \leq K, 0 \leq e\} \cup \{y_n^{(e)} \mid 1 \leq n \leq M, 0 \leq e\}$ (as a tuple) follow a multivariate Gaussian distribution when $N \to \infty$. Similar to previous section, the problem is reduced to whether $c_1\xi_1 + \cdots + c_N\xi_N$ follows a Gaussian. In what cases does this hold true? One trivial case is that $\xi_1, \ldots, \xi_N$ are independent random variables. In that case, Lindeberg's central limit theorem can be applied. Additionally, if there is a linear transformation $A\boldsymbol{\zeta} = \boldsymbol{\xi}$ and $\zeta_1, \ldots, \zeta_N$ are independent random variables, Lindeberg's central limit theorem can be also applied. One surprising case is the hidden manifold model proposed by Goldt et al. [73], which models the input as $\boldsymbol{\xi} = f(A\boldsymbol{\zeta})$, where f is an element-wise function, $\boldsymbol{\zeta}$ follows $\mathcal{N}(0, I_D)$, and $A \in \mathbb{R}^{N \times D}$ is a fixed random matrix.

Acknowledgments

I have been indebted to many people in the course of my research. I would like to thank you very much.

My supervisor, Prof. Masato Okada, has been very helpful to me not only in terms of my research, but also in terms of how to present my research, how to write a paper, and how to lead junior colleagues. Thanks to his guidance, I was able to proceed with my research confidently. I would like to express my deepest gratitude.

Dr. Shun-ichi Amari of RIKEN gave me encouragement every time I met him at conferences and seminars. Thank you very much.

I would like to thank everyone in my laboratory, especially my seniors, Mr. Ryo Karakida and Mr. Yoshihiro Nagano, for giving me a lot of guidance on my research. I also had meaningful discussions with my junior, Mr. Shiro Takagi, which led to the brushing up of my research. I was able to have a wide variety of academic discussions with my colleagues on machine learning and related fields, and we were able to motivate each other in our research.

I would like to express my profound gratitude to my family. My lovely wife Rei supported me writing my dissertation at all times, with a great sense of humor. Also, our lovely cat Miita prevented me from being too exhausted due to over-concentration, by meowing every fifteen minutes.

Lastly, I would like to thank my parents for graciously accepting and supporting the drastic change in my career path.

Bibliography

[1] Hyeyoung Park, Shun-ichi Amari, and Kenji Fukumizu. Adaptive natural gradient learning algorithms for various stochastic models. *Neural Networks*, 13(7):755–764, 2000.

[2] Samuel S Schoenholz, Justin Gilmer, Surya Ganguli, and Jascha Sohl-Dickstein. Deep information propagation. *arXiv preprint arXiv:1611.01232*, 2016.

[3] Lechao Xiao, Yasaman Bahri, Jascha Sohl-Dickstein, Samuel S Schoenholz, and Jeffrey Pennington. Dynamical isometry and a mean field theory of cnns: How to train 10,000-layer vanilla convolutional neural networks. *arXiv preprint arXiv:1806.05393*, 2018.

[4] Minmin Chen, Jeffrey Pennington, and Samuel S Schoenholz. Dynamical isometry and a mean field theory of rnns: Gating enables signal propagation in recurrent neural networks. *arXiv preprint arXiv:1806.05394*, 2018.

[5] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[6] Kenji Fukumizu and Shun-ichi Amari. Local minima and plateaus in hierarchical structures of multilayer perceptrons. *Neural Networks*, 13(3):317–327, 2000.

[7] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[8] Michael Biehl and Holm Schwarze. Learning by on-line gradient descent. *Journal of Physics A: Mathematical and general*, 28(3):643, 1995.

[9] David Saad and Sara A Solla. On-line learning in soft committee machines. *Physical Review E*, 52(4):4225, 1995.

[10] Ben Poole, Subhaneil Lahiri, Maithra Raghu, Jascha Sohl-Dickstein, and Surya Ganguli. Exponential expressivity in deep neural networks through transient chaos. In *Advances in neural information processing systems*, pages 3360–3368, 2016.

[11] Nils Bertschinger and Thomas Natschläger. Real-time computation at the edge of chaos in recurrent neural networks. *Neural computation*, 16(7):1413–1436, 2004.

[12] Yuki Yoshida, Ryo Karakida, Masato Okada, and Shun-ichi Amari. Statistical mechanical analysis of online learning with weight normalization in single layer perceptron. *Journal of the Physical Society of Japan*, 86(4):044002, 2017.

[13] Tim Salimans and Diederik Kingma. Weight normalization: A simple reparameterization to accelerate training of deep neural networks. In *Advances in Neural Information Processing Systems*, 2016.

[14] Yuki Yoshida, Ryo Karakida, Masato Okada, and Shun-ichi Amari. Statistical mechanical analysis of learning dynamics of two-layer perceptron with multiple output units. *Journal of Physics A: Mathematical and Theoretical*, 2019.

[15] John Milnor. On the concept of attractor. In *The Theory of Chaotic Attractors*, pages 243–264. Springer, 1985.

[16] Yuki Yoshida and Masato Okada. Data-dependence of plateau phenomenon in learning with neural network—statistical mechanical analysis. In *Advances in Neural Information Processing Systems*, pages 1720–1728, 2019.

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

[18] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[19] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159, 2011.

[20] Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

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

[22] Tijmen Tieleman and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural Networks for Machine Learning*, 4(2), 2012.

[23] Shun-ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[24] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015.

[25] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[26] Peter Riegler and Michael Biehl. On-line backpropagation in two-layered neural networks. *Journal of Physics A: Mathematical and General*, 28(20):L507, 1995.

[27] Hyeyoung Park, Masato Inoue, and Masato Okada. Slow dynamics due to singularities of hierarchical learning machines. *Progress of Theoretical Physics Supplement*, 157:275–279, 2005.

[28] Kazuyuki Hara, Daisuke Saito, and Hayaru Shouno. Analysis of function of rectified linear unit used in deep learning. In *2015 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2015.

[29] M Kendall, A Stuart, and JK Ord. *Vol. 1: Distribution theory*. London [etc.]: Arnold [etc.], 1994.

[30] Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[31] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[32] Peter Riegler and Michael Biehl. On-line backpropagation in two-layered neural networks. *Journal of Physics A: Mathematical and General*, 28(20):L507, 1995.

[33] Michael Biehl, Peter Riegler, and Christian Wöhler. Transient dynamics of on-line learning in two-layered neural networks. *Journal of Physics A: Mathematical and General*, 29(16):4769, 1996.

[34] Jason AS Freeman and David Saad. Online learning in radial basis function networks. *Neural Computation*, 9(7):1601–1622, 1997.

[35] NJ Huh, JH Oh, and K Kang. On-line learning of a mixture-of-experts neural network. *Journal of Physics A: Mathematical and General*, 33(48):8663, 2000.

[36] Masato Inoue, Hyeyoung Park, and Masato Okada. On-line learning theory of soft committee machines with correlated hidden units–steepest gradient descent and natural gradient descent–. *Journal of the Physical Society of Japan*, 72(4):805–810, 2003.

[37] Madhu S Advani and Andrew M Saxe. High-dimensional dynamics of generalization error in neural networks. *arXiv preprint arXiv:1710.03667*, 2017.

[38] Andrew M Saxe, James L McClelland, and Surya Ganguli. A mathematical theory of semantic development in deep neural networks. *arXiv preprint arXiv:1810.10531*, 2018.

[39] Shun-ichi Amari and Tomoko Ozeki. Differential and algebraic geometry of multilayer perceptrons. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, 84(1):31–38, 2001.

[40] Florent Cousseau, Tomoko Ozeki, and Shun-ichi Amari. Dynamics of learning in multilayer perceptrons near singularities. *IEEE Transactions on Neural Networks*, 19(8):1313–1328, 2008.

[41] Shun-ichi Amari, Hyeyoung Park, and Tomoko Ozeki. Singularities affect dynamics of learning in neuromanifolds. *Neural computation*, 18(5):1007–1065, 2006.

[42] Shun-ichi Amari and Hiroshi Nagaoka. Methods of information geometry, volume 191 of translations of mathematical monographs. *American Mathematical Society*, 13, 2000.

[43] Shun-ichi Amari, Tomoko Ozeki, Ryo Karakida, Yuki Yoshida, and Masato Okada. Dynamics of learning in mlp: Natural gradient and singularity revisited. *Neural computation*, 30(1):1–33, 2018.

[44] Haikun Wei, Jun Zhang, Florent Cousseau, Tomoko Ozeki, and Shun-ichi Amari. Dynamics of learning near singularities in layered networks. *Neural computation*, 20(3):813–843, 2008.

[45] Kazuyuki Hara, Daisuke Saitoh, and Hayaru Shouno. Analysis of dropout learning regarded as ensemble learning. In *International Conference on Artificial Neural Networks*, pages 72–79. Springer, 2016.

[46] Shiro Takagi, Yuki Yoshida, and Masato Okada. Impact of layer normalization on single-layer perceptron–statistical mechanical analysis. *Journal of the Physical Society of Japan*, 88(7):074003, 2019.

[47] Michiel Straat and Michael Biehl. On-line learning dynamics of relu neural networks using statistical physics techniques. *arXiv preprint arXiv:1903.07378*, 2019.

[48] Sebastian Goldt, Madhu S Advani, Andrew M Saxe, Florent Krzakala, and Lenka Zdeborová. Dynamics of stochastic gradient descent for two-layer neural networks in the teacher-student setup. *arXiv preprint arXiv:1906.08632*, 2019.

[49] Michiel Straat, Fthi Abadi, Christina Göpfert, Barbara Hammer, and Michael Biehl. Statistical mechanics of on-line learning under concept drift. *Entropy*, 20(10):775, 2018.

[50] Haikun Wei, Jun Zhang, Florent Cousseau, Tomoko Ozeki, and Shun-ichi Amari. Dynamics of learning near singularities in layered networks. *Neural computation*, 20(3):813–843, 2008.

[51] Florent Cousseau, Tomoko Ozeki, and Shun-ichi Amari. Dynamics of learning in multilayer perceptrons near singularities. *IEEE Transactions on Neural Networks*, 19(8):1313–1328, 2008.

[52] Weili Guo, Yuan Yang, Yingjiang Zhou, Yushun Tan, Haikun Wei, Aiguo Song, and Guochen Pang. Influence area of overlap singularity in multilayer perceptrons. *IEEE Access*, 6:60214–60223, 2018.

[53] A Emin Orhan and Xaq Pitkow. Skip connections eliminate singularities. *arXiv preprint arXiv:1701.09175*, 2017.

[54] Dar Gilboa, Bo Chang, Minmin Chen, Greg Yang, Samuel S Schoenholz, Ed H Chi, and Jeffrey Pennington. Dynamical isometry and a mean field theory of lstms and grus. *arXiv preprint arXiv:1901.08987*, 2019.

[55] Zhichao Wang. A mean field theory of binary neural networks. 2018.

[56] Yaniv Blumenfeld, Dar Gilboa, and Daniel Soudry. A mean field theory of quantized deep networks: The quantization-depth trade-off. In *Advances in Neural Information Processing Systems*, pages 7036–7046, 2019.

[57] Ge Yang and Samuel Schoenholz. Mean field residual networks: On the edge of chaos. In *Advances in neural information processing systems*, pages 7103–7114, 2017.

[58] Tatsuro Kawamoto, Masashi Tsubaki, and Tomoyuki Obuchi. Mean-field theory of graph neural networks in graph partitioning. In *Advances in Neural Information Processing Systems*, pages 4361–4371, 2018.

[59] Greg Yang, Jeffrey Pennington, Vinay Rao, Jascha Sohl-Dickstein, and Samuel S Schoenholz. A mean field theory of batch normalization. *arXiv preprint arXiv:1902.08129*, 2019.

[60] S Amari. A mathematical principle of neural networks. *Sangyo Publishing*, 1978.

[61] Jeffrey Pennington and Pratik Worah. Nonlinear random matrix theory for deep learning. In *Advances in Neural Information Processing Systems*, pages 2637–2646, 2017.

[62] Jeffrey Pennington and Yasaman Bahri. Geometry of neural network loss surfaces via random matrix theory. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 2798–2806. JMLR. org, 2017.

[63] Jeffrey Pennington, Samuel Schoenholz, and Surya Ganguli. The emergence of spectral universality in deep networks. In *International Conference on Artificial Intelligence and Statistics*, pages 1924–1932, 2018.

[64] Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. Universal statistics of fisher information in deep neural networks: Mean field approach. *arXiv preprint arXiv:1806.01316*, 2018.

[65] Ryo Karakida, Shotaro Akaho, and Shun-ichi Amari. The normalization method for alleviating pathological sharpness in wide neural networks. In *Advances in Neural Information Processing Systems*, pages 6403–6413, 2019.

[66] Jaehoon Lee, Yasaman Bahri, Roman Novak, Samuel S Schoenholz, Jeffrey Pennington, and Jascha Sohl-Dickstein. Deep neural networks as gaussian processes. *arXiv preprint arXiv:1711.00165*, 2017.

[67] Alexander G de G Matthews, Mark Rowland, Jiri Hron, Richard E Turner, and Zoubin Ghahramani. Gaussian process behaviour in wide deep neural networks. *arXiv preprint arXiv:1804.11271*, 2018.

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

[69] Jaehoon Lee, Lechao Xiao, Samuel Schoenholz, Yasaman Bahri, Roman Novak, Jascha Sohl-Dickstein, and Jeffrey Pennington. Wide neural networks of any depth evolve as linear models under gradient descent. In *Advances in neural information processing systems*, pages 8570–8581, 2019.

[70] Charles H Martin and Michael W Mahoney. Implicit self-regularization in deep neural networks: Evidence from random matrix theory and implications for learning. *arXiv preprint arXiv:1810.01075*, 2018.

[71] Jun-nosuke Teramae, Yasuhiro Tsubo, and Tomoki Fukai. Optimal spike-based communication in excitable networks with strong-sparse and weak-dense links. *Scientific reports*, 2(1):1–6, 2012.

[72] Shun-Ichi Amari. Natural gradient works efficiently in learning. *Neural computation*, 10(2):251–276, 1998.

[73] Sebastian Goldt, Marc Mézard, Florent Krzakala, and Lenka Zdeborová. Modelling the influence of data structure on learning in neural networks: the hidden manifold model. *arXiv preprint arXiv:1909.11500v2*, 2019.

[74] Jarl Waldemar Lindeberg. Eine neue herleitung des exponentialgesetzes in der wahrscheinlichkeitsrechnung. *Mathematische Zeitschrift*, 15(1):211–225, 1922.