The Effect of Batch Normalization in the Symmetric Phase

Shiro Takagi[1], Yuki Yoshida[1], and Masato Okada[1]

The University of Tokyo, Kashiwanoha Chiba 277-0882, Japan

Abstract. Learning neural networks has long been known to be difficult. One of the causes of such difficulties is thought to be the equilibrium points caused by the symmetry between the weights of the neural network. Such an equilibrium point is known to delay neural network training. However, neural networks have been widely used in recent years largely because of the development of methods that make learning easier. One such technique is batch normalization, which is empirically known to speed up learning. Therefore, if the equilibrium point due to symmetry truly affects the neural network learning, and batch normalization speeds up the learning, batch normalization should help escape from such equilibrium points. Therefore, we analyze whether batch normalization helps escape from such equilibrium points by a method called statistical mechanical analysis. By examining the eigenvalue of the Hessian matrix of the generalization error at the equilibrium point, we find that batch normalization delays escape from poor equilibrium points. This contradicts the empirically known finding of speeding up learning, and we discuss why we obtained this result.

Keywords: Neural Network · Symmetric Phase · Batch Normalization.

1 Introduction

A neural network is known as a model that has structural symmetries, i.e., the output can be the same value even if a specific weight value is replaced with another weight value. This symmetry creates a sequence of saddle points and local minima on the error surface in parameter space (hereafter, we call the equilibrium point a symmetric equilibrium point), and these equilibrium points are known to delay the learning of neural networks [33, 11, 40, 3, 2, 38, 9]. Due to this equilibrium point, training of the neural network can be divided into two phases: the symmetric phase, which is an early-stage training phase where training does not proceed because the neural network is trapped around the symmetric point, and the specialization phase, where the neural network escapes from the equilibrium point and begins to learn properly (see Section 3.1 for details). Therefore, the learning behavior of neural networks in the symmetry phase is thought to be a key to determining the difficulty in training neural networks.

Neural networks have been widely used in recent years because of the development of methods that facilitate learning. Batch normalization is one such technique, and it is known that it contributes to faster learning [18, 34, 23, 7, 4, 26]. If batch normalization contributes to faster learning, it is expected to expedite escape from such a symmetric equilibrium point. Therefore, we analyze the eigenvalue of the Hessian matrix of the error at the symmetric equilibrium point to determine whether batch normalization actually helps escape from such a point. The Hessian matrix at a fixed point describes the local curvature of the loss surface at the point, which has a crucial influence on the learning speed of the model around the point; for example, if the absolute values of any negative eigenvalue are small, it will take long time to escape from the fixed point. This requires analytically computing the error and its dynamics. To that end, we derived the learning dynamics of a neural network with batch normalization by statistical mechanical analysis. The statistical mechanical analysis is a method to analytically derive the dynamics of the generalization error in the large limit of the input dimension and is well suitable for the analysis of the symmetric equilibrium point since it enables us to analytically study the learning behavior around the point.

2 Statistical mechanical analysis of two-layered neural network with batch normalization

2.1 Teacher-student learning

For the statistical mechanical analysis, we have to consider the learning under the framework of teacher-student learning [35, 36, 32, 6, 30]. Hence, we analyzed the training of a two-layered neural network by stochastic gradient descent (SGD) in the framework of teacher-student learning, where data used for training once are not used twice. Teacher-student learning refers to supervised learning assuming that both the learner and the true function are neural networks [32, 6]. The true function is called a teacher network, and the learner is called a student network.

Consider the teacher network and student network with N input neurons and 1 output neuron. Each network has one hidden layer, and the number of hidden neurons is M for the teacher and K for the student. We assume that for each update, b new inputs $\boldsymbol{\xi}^u \in \mathbb{R}^N \quad (u = 1, ..., b)$ are used, where each component $\xi_i^u \quad (i = 1, ..., N)$ of input data $\boldsymbol{\xi}^u$ is sampled i.i.d. from the distribution of expected value 0 and variance σ^2. Let the weight matrix of the first layer of the student network be $[\mathbf{J}_1, \ldots, \mathbf{J}_K]^T \in \mathbb{R}^{K \times N}$ and the weight vector of the second layer is $\mathbf{w} \in \mathbb{R}^K$, the weight matrix of the first layer of the teacher network is $[\mathbf{B}_1, \ldots, \mathbf{B}_M]^T \in \mathbb{R}^{M \times N}$ and the weight vector of the second layer is $\mathbf{v} \in \mathbb{R}^M$. The elements of the first-layer weight vector of the student and teacher are $J_{il} \in \mathbb{R} \overset{\text{iid}}{\sim} \mathcal{N}(0, 1/N)$, $B_{nl} \in \mathbb{R} \overset{\text{iid}}{\sim} \mathcal{N}(0, 1/N)$ $(1 \leq i \leq K, 1 \leq n \leq M, 1 \leq l \leq N)$. For simplicity, each element of the second-layer weight vector is assumed to be 1 in both networks, but it is simple to extend to the general case of learning weights. A neural network that fixes each element of the weight vector of the second layer

to a constant is called a soft-committee machine. The activation function of the hidden layer is $\phi : \mathbb{R} \to \mathbb{R}$ and that of the output layer is an identity map. We use the square loss $\varepsilon = \frac{1}{2}(t^u - s^u)^2$ as the loss function.

2.2 Formulation for statistical mechanical analysis

Here, we define the order parameters, which describe the global behavior of the system as $Q_{ij} = \mathbf{J}_i \cdot \mathbf{J}_j$, $R_{in} = \mathbf{J}_i \cdot \mathbf{B}_n$, $T_{nm} = \mathbf{B}_n \cdot \mathbf{B}_m$ [32, 6].

In the limit of large degrees of freedom with the number of input elements $N \to \infty$, the time evolution differential equations of these parameters can be derived, depending on the activation function [32, 6]. The generalized error ε_g, defined as the expected value of the training error with the distribution followed by $\boldsymbol{\xi}$, is a function of the order parameter. Therefore, by deriving the dynamics of the order parameter, the dynamics of the generalization error can also be derived. Since this method is based on statistical mechanics, we call it statistical mechanical analysis.

2.3 Statistical mechanical analysis of neural network with batch normalization

In batch normalization, each input $x_i^u = \mathbf{J}_i \cdot \boldsymbol{\xi}^u$ to each hidden neuron is normalized with the sample mean $\mu_{x_i} = \frac{1}{b}\sum_{u=1}^{b} x_i^u$ and sample standard deviation $\sigma_{x_i} = \sqrt{\frac{1}{b}\sum_{u=1}^{b}(x_i^u - \mu_{x_i})^2}$. Then, the normalized inputs are multiplied by the gain parameter g_i and added by the bias parameter β_i, which are learnable parameters: $g_i \frac{(x_i^u - \mu_{x_i})}{\sigma_{x_i}} + \beta_i$.

Here, we do not subtract the mean μ_{x_i} and do not add β_i, the error of which is negligible when the sample size b is large since x_i always follows the distribution of expected value 0. When b is large, the sample standard deviation for b inputs of each element of the hidden layer is $\sigma_{x_i} \approx \sqrt{\frac{1}{b}\sum_{u=1}^{b}(x_i^u)^2} \approx \sqrt{\langle (x_i^u)^2 \rangle} \approx \sqrt{\mathbf{J}_i^T \langle \boldsymbol{\xi}^u \boldsymbol{\xi}^{u^T} \rangle \mathbf{J}_i} = \sqrt{\sigma^2 ||\mathbf{J}_i||^2} = \sigma\sqrt{Q_{ii}}$. Note that $\langle \cdot \rangle$ is an operation that takes the expected value of the input $\boldsymbol{\xi}$. Hence, inputs to the hidden layer i are reparameterized as $\frac{g_i}{\sigma\sqrt{Q_{ii}}} x_i^u$. Then, the outputs of the student and teacher networks are

$$s^u = \sum_i^K w_i \phi\left(\frac{g_i}{\sigma\sqrt{Q_{ii}}} x_i^u\right) \in \mathbb{R}, \quad t^u = \sum_n^M v_n \phi\left(y_n^u\right) \in \mathbb{R}, \tag{1}$$

where $y_n^u = \mathbf{B}_n \cdot \boldsymbol{\xi}^u$. Note that $\frac{g_i}{\sigma\sqrt{Q_{ii}}} x_i^u$ and y_n^u follow Gaussian distribution of mean 0 since we assume that $N \to \infty$.

The update equations of the weights $\boldsymbol{J}_i$ and the gain parameter g_i by SGD are as follows:

$$\boldsymbol{J}_i \leftarrow \boldsymbol{J}_i - \frac{\eta}{N}\nabla\varepsilon = \boldsymbol{J}_i + \frac{\eta}{Nb}\sum_{u=1}^{b}\left[(t^u - s^u)\cdot w_i\right]\phi'\left(\frac{g_i}{\sigma\sqrt{Q_{ii}}}x_i^u\right)\frac{g_i}{\sigma\sqrt{Q_{ii}}}\boldsymbol{\xi}^u, \quad (2)$$

$$g_i \leftarrow g_i - \frac{\eta}{N}\nabla\varepsilon = g_i + \frac{\eta}{Nb}\sum_{u=1}^{b}\left[(t^u - s^u)\cdot w_i\right]\phi'\left(\frac{g_i}{\sigma\sqrt{Q_{ii}}}x_i^u\right)\frac{1}{\sigma\sqrt{Q_{ii}}}x_i^u, \quad (3)$$

where $\frac{\eta}{N}$ is the learning rate. Note that we assume that $\frac{g_i x_i^u}{\sigma\sqrt{Q_{ii}}Q_{ii}}\boldsymbol{J}_i$ is negligible and omitted it from the update equation of $\boldsymbol{J}_i$ because $\frac{g_i x_i^u}{\sigma\sqrt{Q_{ii}}Q_{ii}}\boldsymbol{J}_i \approx O(\frac{1}{\sqrt{N}})$, while $\frac{g_i}{\sigma\sqrt{Q_{ii}}}\boldsymbol{\xi}^u \approx O(1)$ at initialization. Considering g_i as an order parameter, it is straightforward to derive the update equation of the order parameters. Here, we sum the update equations of the order parameters from time 0 to time $N\mathrm{d}t$, where $N\mathrm{d}t$ is large enough but much smaller than N since $\mathrm{d}t$ is small. As a result, we can derive the dynamics of order parameters as follows:

$$\begin{aligned}\frac{\mathrm{d}Q_{ij}}{\mathrm{d}t} &= \eta\left[\sum_{p=1}^{M} I_3(\hat{x}_i^u, \hat{x}_j^u, y_p^u) - \sum_{p=1}^{K} I_3(\hat{x}_i^u, \hat{x}_j^u, \hat{x}_p^u) + \sum_{p=1}^{M} I_3(\hat{x}_j^u, \hat{x}_i^u, y_p^u) - \sum_{p=1}^{K} I_3(\hat{x}_j^u, \hat{x}_i^u, y_p^u)\right] \\ &\quad + \frac{\eta^2}{b}\left[\sum_{p,q}^{K,K} I_4(\hat{x}_i^u, \hat{x}_j^v, \hat{x}_p^u, \hat{x}_q^v) + \sum_{p,q}^{M,M} I_4(\hat{x}_i^u, \hat{x}_j^v, y_p^u, y_q^v)\right. \\ &\quad \left. - \sum_{p,q}^{K,M} I_4(\hat{x}_i^u, \hat{x}_j^v, \hat{x}_p^u, y_q^v) - \sum_{p,q}^{M,K} I_4(\hat{x}_i^u, \hat{x}_j^v, y_p^u, \hat{x}_q^v)\right], \quad (4)\end{aligned}$$

$$\frac{\mathrm{d}R_{in}}{\mathrm{d}t} = \eta\left[\sum_{p=1}^{M} I_3(\hat{x}_i^u, y_n^u, y_p^u) - \sum_{p=1}^{K} I_3(\hat{x}_i^u, y_n^u, \hat{x}_p^u)\right], \quad (5)$$

$$\frac{\mathrm{d}g_i}{\mathrm{d}t} = \eta\left[\sum_{p=1}^{M} I_3(\hat{x}_i^u, x_i^u, y_p^u) - \sum_{p=1}^{K} I_3(\hat{x}_i^u, x_i^u, \hat{x}_p^u)\right], \quad (6)$$

where $\hat{x}_i = \frac{g_i}{\sigma\sqrt{Q_{ii}}}x_i$ and $I_3(z_1, z_2, z_3) = \langle\phi'(z_1)z_2\phi(z_3)\rangle$ and $I_4(z_1, z_2, z_3, z_4) = \langle\phi'(z_1)\phi'(z_2)\phi(z_3)\phi(z_4)\rangle$ can be calculated analytically when a set of random variables (z_1, z_2, z_3, z_4) follows a multivariate Gaussian distribution of the expected value $\mathbf{0}$ [32, 6, 30]. Note that $\langle\cdot\rangle$ means taking the expected value as of x and y, and the term $\frac{\eta^2}{b}$ is negligible when b is large. The generalization error can also be written as

$$\varepsilon_g = \frac{1}{2}\left[\sum_{p,q}^{M,M} I_2(y_p^u, y_q^u) + \sum_{p,q}^{K,K} I_2(\hat{x}_p^u, \hat{x}_q^u) - 2\sum_{p,q}^{K,M} I_2(\hat{x}_p^u, y_q^u)\right], \quad (7)$$

where $I_2(z_1, z_2) = \langle\phi(z_1)\phi(z_2)\rangle$, which can also be calculated exactly. When the activation function is $\phi(x) = \mathrm{erf}(x/\sqrt{2})$ and b is assumed to be large, the

dynamics of these order parameters and the generalization error are as follows:

$$\frac{\mathrm{d}Q_{ij}}{\mathrm{d}t}=\frac{2\eta}{\pi}\left[\mathcal{Q}_1-\mathcal{Q}_2\right],\quad \frac{\mathrm{d}R_{in}}{\mathrm{d}t}=\frac{2\eta}{\pi}\frac{g_i}{\sigma\sqrt{Q_{ii}}}\left[\mathcal{R}\right],\quad \frac{\mathrm{d}g_i}{\mathrm{d}t}=\frac{2\eta}{\pi g_i}\left[\mathcal{G}\right], \tag{8}$$

where $\mathcal{Q}_1=\mathcal{Q}_1(Q_{ij},R_{in},T_{nm})$, $\mathcal{Q}_2=\mathcal{Q}_2(Q_{ij})$, $\mathcal{R}=\mathcal{R}(Q_{ij},R_{in},T_{nm})$, $\mathcal{G}=\mathcal{G}(Q_{ij},R_{in},T_{nm})$. The exact expressions of these functions are as follows:

$$\mathcal{Q}_1=\sum_{p=1}^{M}\left[\frac{\left(R'_{jp}(1+Q'_{ii})-Q'_{ij}R'_{ip}\right)}{(1+Q'_{ii})\sqrt{(1+Q'_{ii})(1+T'_{pp})-R'^2_{ip}}}+\frac{\left(R'_{ip}(1+Q'_{jj})-Q'_{ji}R'_{jp}\right)}{(1+Q'_{jj})\sqrt{(1+Q'_{jj})(1+T'_{pp})-R'^2_{jp}}}\right], \tag{9}$$

$$\mathcal{Q}_2=\sum_{p=1}^{K}\left[\frac{\left(Q'_{jp}(1+Q'_{ii})-Q'_{ij}Q'_{ip}\right)}{(1+Q'_{ii})\sqrt{(1+Q'_{ii})(1+Q'_{pp})-Q'^2_{ip}}}+\frac{\left(Q'_{ip}(1+Q'_{jj})-Q'_{ji}Q'_{jp}\right)}{(1+Q'_{jj})\sqrt{(1+Q'_{jj})(1+Q'_{pp})-Q'^2_{jp}}}\right], \tag{10}$$

$$\mathcal{R}=\sum_{p=1}^{M}\frac{\left(T'_{np}(1+Q'_{ii})-R'_{in}R'_{ip}\right)}{(1+Q'_{ii})\sqrt{(1+Q'_{ii})(1+T'_{pp})-R'^2_{ip}}}-\sum_{p=1}^{K}\frac{\left(R'_{pn}(1+Q'_{ii})-R'_{in}Q'_{ip}\right)}{(1+Q'_{ii})\sqrt{(1+Q'_{ii})(1+Q'_{pp})-Q'^2_{ip}}}, \tag{11}$$

$$\mathcal{G}=\sum_{p=1}^{M}\frac{\left(R'_{ip}(1+Q'_{ii})-Q'_{ii}R'_{ip}\right)}{(1+Q'_{ii})\sqrt{(1+Q'_{ii})(1+T'_{pp})-R'^2_{ip}}}-\sum_{p=1}^{K}\frac{\left(Q'_{ip}(1+Q'_{ii})-Q'_{ii}Q'_{ip}\right)}{(1+Q'_{ii})\sqrt{(1+Q'_{ii})(1+Q'_{pp})-Q'^2_{ip}}}, \tag{12}$$

$$\varepsilon_g=\frac{1}{\pi}\left[\sum_{p,q}^{M,M}\operatorname{asin}\left(\frac{T'_{pq}}{\sqrt{(1+T'_{pp})(1+T'_{qq})}}\right)\right.$$
$$\left.+\sum_{p,q}^{K,K}\operatorname{asin}\left(\frac{Q'_{pq}}{\sqrt{(1+Q'_{pp})(1+Q'_{qq})}}\right)-2\sum_{p,q}^{K,M}\operatorname{asin}\left(\frac{R'_{pq}}{\sqrt{(1+Q'_{pp})(1+T'_{qq})}}\right)\right]. \tag{13}$$

Note that $Q'_{lk}=\frac{\sigma^2 g_l g_k}{\sigma_{x_l}\sigma_{x_k}}Q_{lk}, R'_{lk}=\frac{\sigma^2 g_l}{\sigma_{x_l}}R_{lk}, T'_{lk}=\sigma^2 T_{lk}$ for $(l,k)=(i,j,n,p,q)$.

3 Error surface near the symmetric equilibrium point

3.1 Symmetric phase

It is known that there are two phases in learning a two-layered neural network [33, 31, 41]. The first phase is called the symmetric phase, where the student network's weights cannot determine which weight of the teacher network they should specialize. This is because there are many parameters that can play the same role due to the symmetry of the neural network. As explained above, such symmetry delays training as the neural network is trapped around a symmetric equilibrium point.

However, as the learning progresses, the neural network gradually escapes from such an equilibrium point due to the small noise, and the weight vectors of the student network correspond to any of the weight vectors of the teacher network. This is the second phase, the specialization phase. In the specialization phase, the student network eventually learns the input/output structure of the teacher network completely. In this paper, we focus on the behavior of neural networks in the symmetric phase, especially around the symmetric equilibrium point.

3.2 Effect of batch normalization at the symmetric equilibrium point

In the following, we consider the case where the number of hidden neurons in the student network and the teacher network is the same: $K = M$. In the symmetric phase, when the covariance matrix of the teacher's weight vectors is isotropic, that is, $T_{nm} = \delta_{nm}$, we can reduce the abovementioned order parameters into a smaller number of order parameters [33]. Note that δ_{nm} is a function that returns 1 when $n = m$ and 0 when $n \neq m$. The reduced order parameters are as follows:

$$Q_{ij} = \begin{cases} Q \ (i = j) \\ C \ (i \neq j) \end{cases}, \quad R_{in} = \begin{cases} R \ (i = n) \\ S \ (i \neq n) \end{cases}, \quad g_i = g.$$

Now, assume that the neural network is at the symmetric equilibrium point. The order parameters R, S, Q, C, and g at this equilibrium point are defined as R^*, S^*, Q^*, C^*, g^*, respectively. Owing to the symmetry between the parameters at the equilibrium point, the equations $Q^* = C^*, R^* = S^*$ hold. Since the relation of $Q^* = C^*$ is conserved up to the first-order terms in the perturbation expansion, the order parameters can be further reduced to four parameters, Q, R, S, and g, and we can derive the dynamics of these four order parameters.

Here, we assume that $Q = \frac{1}{g^2}$ because g is a parameter to scale the magnitude of the normalized input, so it is expected to have the opposite relationship with Q. In fact, we empirically confirmed that this holds through training. Then, the value of the order parameter at the symmetric equilibrium point is $R^* = S^* = \sqrt{\frac{K+(K-1)\tau}{\tau K}}$, $Q^* = \frac{K+(K-1)\tau}{\tau}$, $g^* = \sqrt{\frac{\tau}{K+(K-1)\tau}}$ by solving $\frac{dR}{dt} = \frac{dS}{dt} = \frac{dQ}{dt} = \frac{dg}{dt} = 0$. We represent a vector of order parameters by $\boldsymbol{\Omega} = (R, S, Q, g)$ and the Hessian of the generalization error by $\nabla_{\boldsymbol{\Omega}} \nabla_{\boldsymbol{\Omega}} \varepsilon_g(\boldsymbol{\Omega}) = H$. By estimating the eigenvalue of this Hessian matrix at the symmetric equilibrium point, we can understand the local topography of the error surface. This is because the eigenvalue of the Hessian is a quantity that describes how the perturbation affects the change in error when the parameter is slightly moved from the point where the eigenvalue is evaluated. When some eigenvalues of the Hessian are almost zero, there are directions in which adding a perturbation hardly affects the error since the Hessian is evaluated at the equilibrium point. In other words, the local error surface is flat. Conversely, when a large eigenvalue exists, the terrain is steep where positive eigenvalues imply that there are directions in which the error increases and negative eigenvalues are signs of directions in which the error

decreases. Hence, when the absolute values of any negative eigenvalue are small, it is difficult to escape from this equilibrium point.

Therefore, we calculated the eigenvalues of the Hessian matrix numerically and examined the effect of batch normalization on the local shape of the error surface at the symmetric equilibrium point. Figures 1 and 2 plot the calculated eigenvalues as a function of the hidden layer width K and the input variance τ . Unless otherwise specified, the width of the hidden layer was $K = 2$ and the variance in the input was $\tau = 1$.

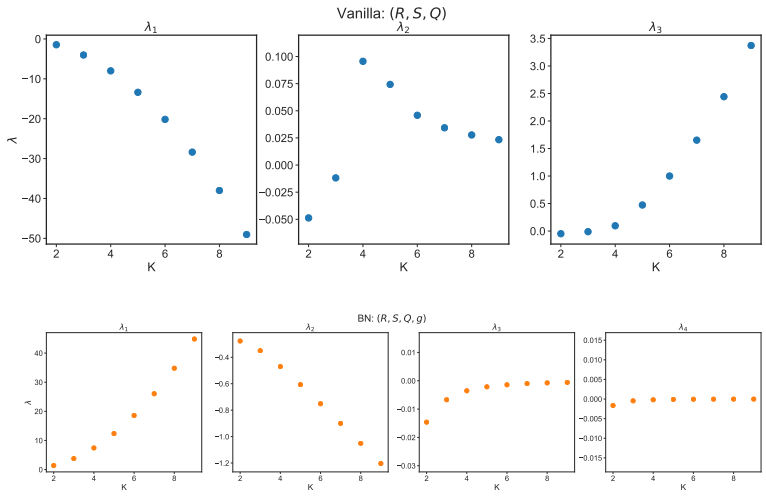

Fig. 1: Eigenvalue of the Hessian matrix for hidden layer K. The vertical axis λ indicates the magnitude of the eigenvalue, and the horizontal axis K indicates the number of hidden neurons. $\lambda_i (i = 1, .., 4)$ is the result for the ith eigenvalue. (a) Normal two-layered neural network without batch normalization. (b) When batch normalization is used. You can see that the absolute value of the negative eigenvalue is relatively smaller when batch normalization is added.

As shown in Figure 1, when batch normalization is used, the direction where the eigenvalue becomes 0 appears for large K. In addition, the absolute value of the negative eigenvalue is smaller when batch normalization is used. Hence, it can be expected that the neural network takes a longer time to escape from the equilibrium point when batch normalization is added.

As shown in Figure 2, a similar tendency was observed for input variance τ. In ordinary neural networks, the absolute value of the negative eigenvalue becomes very large as τ becomes large. In other words, escape from the equilibrium point is expected to be very easy. However, when batch normalization is used, a direction in which the eigenvalue is almost 0 appears as before, and a

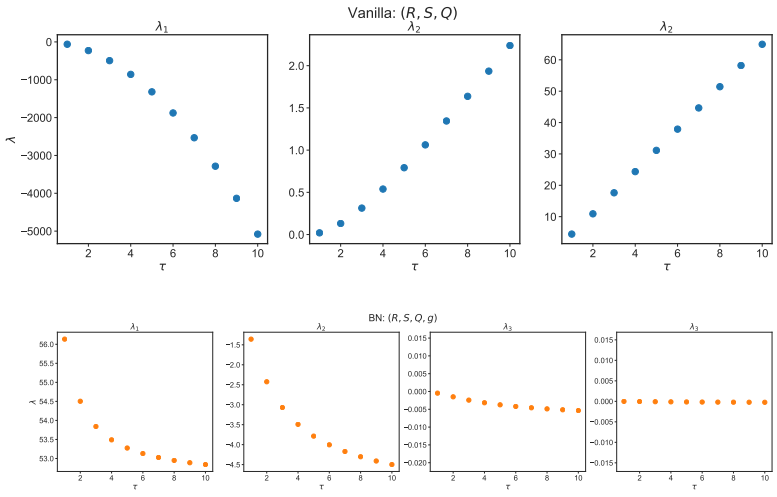

Fig. 2: The same as Fig. 1 except that the x axis is the input variance τ. As shown in Fig. 1, batch normalization decreases the absolute value of the negative eigenvalue.

direction in which the error surface is flat is created. Additionally, the absolute value of the negative eigenvalue is clearly smaller than that of the vanilla neural network. Therefore, we can expect that escape from the equilibrium point will take a longer time when batch normalization is added.

To confirm this, we compare the time evolution of the generalization error of both the vanilla neural network and the batch normalization with large K and τ. The number of iterations is $20,000$, the hidden width is $K = M = 2$ and the input variance is $\tau = 1$ unless otherwise specified. Figure 3 shows the result. The neural network with batch normalization has a longer period when the generalization error hardly changes. This period corresponds to the period when the neural network is near the symmetric equilibrium point. Therefore, as expected, in both cases where the width of the hidden layer K and the input variance τ are large, the neural network with batch normalization takes more time to escape from the symmetric equilibrium point.

4 Related work

The symmetry of neural networks is thought to be a cause of the difficulty of training neural networks. The Hessian of the error surface in the output space of the neural network are known to have a singular point [1, 37, 10]. Previous studies show how this point delays training [33, 11, 40, 3, 2, 38, 9]. When students have extra parameters to express teachers, the global minimum is on the singular

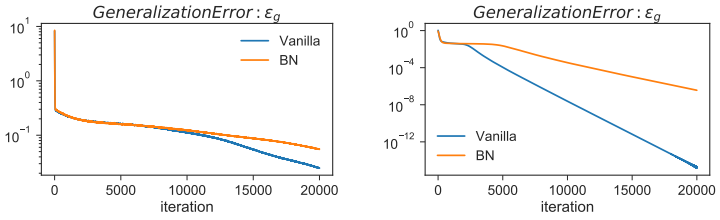

Fig. 3: Generalization error for large K and τ. Vanilla shows the result of a normal two-layered neural network, and BN shows the result when batch normalization is added. The generalization error is calculated not with the order parameter derived by statistical mechanics but with the result of numerical calculation calculated by updating the weights sequentially without using approximation. Note that $\eta = 1$, $N = 100$, $b = 100$, and the total number of iterations is 20,000.

region, and it takes time to reach the solution [9]. Even when local minima and saddles are in singular regions, the neural network is attracted to the point, and training is delayed [40, 39, 33]. Our work is related to the second case.

Statistical mechanical analysis was invented and developed in the 1990s. Saad and Solla derived dynamics of the weight from the input layer to the hidden layer of the two-layered soft-committee machine [32]. Biehl and Schwarze studied the weight from the input layer to the hidden layer for a general two-layered neural network [6]. Yoshida et al. and Goldt et al. derived the learning dynamics of all weights of a general two-layered neural network [12, 42]. We extended the analysis of the dynamics of the soft committee to a case with batch normalization.

Various discussions have focused on why batch normalization speeds up learning. Mitigating internal covariate shift [18], smoothing the error surface [34, 20], decoupling length and direction optimization of weight vectors [23], allowing a larger learning rate [7], and adjusting the effective learning rate [4, 26] might be reasons why batch normalization speeds up neural network training. Although Luo et al. analyzed the dynamics of batch normalization by statistical mechanical analysis [26], this is limited to the discussion of single-layer perceptrons. This is insufficient for our purpose since symmetric equilibrium points do not appear in a single-layer perceptron. Therefore, we derived the dynamics of a two-layered neural network.

5 Discussion

We derived the learning dynamics and generalization error of a two-layered neural network with batch normalization by statistical mechanical analysis. Using them, we analyzed the shape of the error surface in the parameter space around the equilibrium point caused by the symmetry of the network and found that, at

least in the situation we considered, batch normalization delays escape from the equilibrium point. This contradicts the empirical fact that batch normalization speeds up the learning of neural networks. In the following, we consider the cause of this result.

The first possibility is to consider the infinitesimal learning rate. There are many studies that consider the small limit of the learning rate to analyze learning dynamics [27, 19, 8], and statistical mechanical analysis also relies on this assumption. In this method, it is difficult to discuss the finite size effect of learning rates. Since batch normalization is known to smooth the complicated error surface [34, 7], it might be harder to escape with a small learning rate.

The second possibility is that the solution found in practical applications is the symmetric equilibrium point. In other words, it is possible that the network achieved a level of performance without completely breaking the symmetry between the parameters of the networks. In recent years, the learning of neural networks often uses networks with much greater expressive power for solving tasks, and it is well known that there are many parameters that are practically unused in such cases. In fact, a neural network is known to maintain high accuracy even when more than 90 % of its weights are pruned [24, 16, 15, 25]. This is partly because stochastic gradient descent implicitly regularizes neural networks, finding simple functions [29, 28, 14, 5]. In addition, it is known that the solutions found by neural networks have a flat terrain in the vicinity of the error surface, which shows that the learning has been completed while maintaining symmetry. [17, 22, 21]. Furthermore, for example, Yoshida et al. argue that neural networks will not be trapped at the symmetric equilibrium point when neural networks have multiple outputs [42]. Additionally, Yoshida et al. and Goldt et al. claim that the statistical property of the data can mitigate delay learning due to the symmetric equilibrium point [43, 13]. Although we find considerable evidence that neural networks can maintain high symmetry even after training, these are still speculations, and further study on these possibilities is left for future study.

References

1. Amari, S.: Natural gradient works efficiently in learning. Neural Computation **10**(2), 251–276 (1998)
2. Amari, S., Ozeki, T., Karakida, R., Yoshida, Y., Okada, M.: Dynamics of learning in mlp: Natural gradient and singularity revisited. Neural Computation **30**(1), 1–33 (2018)
3. Amari, S., Park, H., Ozeki, T.: Singularities affect dynamics of learning in neuromanifolds. Neural Computation **18**(5), 1007–1065 (2006)
4. Arora, S., Li, Z., Lyu, K.: Theorical analysis of auto rate-tuning by batch normalization. arXiv preprint arXiv:1812.03981 (2018)
5. Arpit, D., Jastrzębski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M.S., Maharaj, T., Fischer, A., annd Yoshua Bengio, A.C., Lacoste-Julien, S.: A closer look at memorization in deep networks. In: Proceedings of the 34th International Conference on Machine Learning (2017)
6. Biehl, M., Schwarze, H.: Learning by on-line gradient descent. Journal of Physics A: Mathematical and General **28**(3), 643 (1995)

7. Bjorck, J., Gomes, G., Selman, B., Weinberger, K.Q.: Understanding batch normalization. In: Advances in Neural Information Processing Systems 31 (2018)
8. Chaudhari, P., Soatto, S.: Stochastic gradient descent performs variational inference, converges to limit cycles for deep networks. In: 6th International Conference on Learning Representations (2018)
9. Cousseau, F., Ozeki, T., Amari, S.: Dynamics of learning in multilayer perceptrons near singularities. IEEE Transactions on Neural Networks **19**(8), 1313–1328 (2008)
10. Fukumizu, K.: A regularity condition of the information matrix of a multilayer perception network. Neural Networks **9**(5), 871–879 (1996)
11. Fukumizu, K., Amari, S.: Local minima and plateaus in hierarchical structures of multilayer perceptrons. Neural Networks **13**, 317–327 (2000)
12. Goldt, S., Advani, M.S., Saxe, A.M., Krzakala, F., Zdeborova, L.: Dynamics of stochastic gradient descent for two-layer neural networks in the teacher-student setup. In: Advances in Neural Information Processing Systems 32 (2019)
13. Goldt, S., Mezard, M., Krzakala, F., Zdeborova, L.: Modelling the influence of data structure on learning in neural networks: the hidden manifold model. arXiv preprint arXiv:1909.11500 (2019)
14. Gunasekar, S., Woodworth, B.E., Bhojanapalli, S., Neyshabur, B., Srebro, N.: Implicit regularization in matrix factorization. In: Advances in Neural Information Processing Systems 30 (2017)
15. Han, S., Pool, J., Tran, J., Dally, W.: Learning both weights and connections for efficient neural network. In: Advances in Neural Information Processing Systems 28 (2015)
16. Hassibi, B., Stork, D.G.: Second order derivatives for network pruning: Optimal brain surgeon. In: Advances in Neural Information Processing Systems 6 (1993)
17. Hochreiter, S., Schmidhuber, J.: Flat minima. Neural Computation **9**(1), 1–42 (1997)
18. Ioffe, S., Szegedy, C.: Batch normalization: Accelerating deep network training by reducing internal covariate shift. In: Proceedings of the 32nd International Conference on Machine Learning (2015)
19. Jastrzebski, S., Kenton, Z., Arpit, D., Ballas, N., Fischer, A., Bengio, Y., Storkey, A.: Three factors influencing minima in sgd. arXiv preprint arXiv:1711.04623 (2017)
20. Karakida, R., Akaho, S., Amari, S.: The normalization method for alleviating pathological sharpness in wide neural networks. In: Advances in Neural Information Processing Systems 32 (2019)
21. Karakida, R., Akaho, S., Amari, S.: Universal statistics of fisher information in deep neural networks: Mean field approach. In: Chaudhuri, K., Sugiyama, M. (eds.) Proceedings of Machine Learning Research. Proceedings of Machine Learning Research, vol. 89, pp. 1032–1041. PMLR (16–18 Apr 2019)
22. Keskar, N.S., Mudigere, D., Nocedal, J., Smelyanskiy, M., Tang, P.T.P.: On large-batch training for deep learning: Generalization gap and sharp minima. In: 5th International Conference on Learning Representations (2017)
23. Kohler, J., Daneshmand, H., Lucchi, A., Zhou, M., Neymeyr, K., Hofmann, T.: Exponential convergence rates for batch normalization: The power of length-direction decoupling in non-convex optimization. arXiv preprint arXiv:1805.10694 (2018)
24. LeCun, Y., Denker, J.S., Solla, S.A.: Optimal brain damage. In: Advances in Neural Information Processing Systems 3 (1990)
25. Li, H., Kadav, A., Durdanovic, I., Samet, H., Graf, H.P.: Pruning filters for efficient convnets. arXiv preprint arXiv:1608.08710 (2016)

26. Luo, P., Wang, X., Shao, W., Peng, Z.: Towards uunderstanding regularization in batch normalization. arXiv preprint arXiv:1809.00846 (2018)
27. Mandt, S., Hoffman, M., Blei, D.: A variational analysis of stochastic gradient algorithms. In: Proceedings of the 33nd International Conference on Machine Learning (2016)
28. Neyshabur, B., Tomioka, R., Salakhutdinov, R., Srebro, N.: Geometry of optimization and implicit regularization in deep learning. arXiv preprint arXiv:1705.03071 (2017)
29. Neyshabur, B., Tomioka, R., Srebro, N.: In search of the real inductive bias: on the role of implicit regularization in deep learning. In: 3rd International Conference on Learning Representations (2015)
30. Riegler, P., Biehl, M.: On-line backpropagation in two-layered neural networks. Journal of Physics A **28**, L507–L513 (1995)
31. Saad, D., Solla, S.A.: Dynamics of on-line gradient descent learning for multilayer neural networks. In: Advances in Neural Information Processing Systems 8 (1995)
32. Saad, D., Solla, S.A.: Exact solution for on-line learning in multilayer neural networks. Physical Review Letters **74**(41), 4337–4340 (1995a)
33. Saad, D., Solla, S.A.: On-line learning in soft committee machines. Physical Review E **52**(4), 4225–4243 (1995b)
34. Santurkar, S., Tsipras, D., Ilyas, A., Mardy, A.: How does batch normalization help optimization? arXiv preprint arXiv:1805.11604 (2018)
35. Schwarze, H.: Learning a rule in a multilayer neural network. Journal of Physics A **26**, 5781–5794 (1993)
36. Seung, H.S., Somopolinsky, H., Tishby, N.: Statistical mechanics of learning from examples. Physical Review A **45**(8), 6056–6091 (1992)
37. Watanabe, S.: Algebraic geometrical methods for hierarchical learning machines. Neural Networks **14**(8), 1049–1060 (2001)
38. Watanabe, S., Amari, S.: Learning coefficients of layered models when the true distribution mismatches the singularities. Neural Computation **15**(5), 1011–1033 (2003)
39. Wei, H., Amari, S.: Dynamics of learning near singularities in radial basis function networks. Neural Networks **21**(7), 989—1005 (2008)
40. Wei, H., Zhang, J., Cousseau, F., Ozeki, T., Amari, S.: Dynamics of learning in multilayer perceptrons near singularities. Neural Computation **20**(3), 813–842 (2008)
41. West, A.H.L., Saad, D., Nabney, I.T.: The learning dynamics of a universal approximator. In: Advances in Neural Information Processing Systems 9 (1996)
42. Yoshida, Y., Karakida, R., Okada, M., Amari, S.: Statistical mechanical analysis of learning dynamics of two-layer perceptron with multiple output units. Journal of Physics A **52**(18), 184002 (2019)
43. Yoshida, Y., Okada, M.: Data-dependence of plateau phenomenon in learning with neural network — statistical mechanical analysis. In: Advances in Neural Information Processing Systems 32 (2019)