A Survey of Optimization in Deep Neural Network

Chi-Jui Ho, Che-Hsien Lin

B04507009, B04901040

Abstract

Nowadays, deep neural network (DNN) plays an important role in the development of many fields. To guarantee the performance of a DNN model, the convergence analysis is important since it indicates how the model reduces the value of loss function in the training process. In this report, we focus on the convergence analysis of classical DNN models. The scope of analysis involves why and how fast the model reduces the loss function. To guarantee the convergence rate, an important requirement is over-parameterization, which means that the network has sufficient number of parameters. Our analysis starts from the basic case to the general case of DNN models. By generalizing the concepts in the analysis of basic cases, we show that the convergence rate of a general DNN model can be guaranteed via over-parameterization.

1 Introduction

In recent years, deep neural network (DNN) based algorithms have been applied to many fields and brought great improvement. Take computer vision as example, DNN-based algorithms have been developed for image segmentation [1] and optical flow estimation [2]. Although DNN-based methods are widely adopted, its working principle gets less attention than its amazing results.

In our previous experience on other courses about deep learning, we often focused more on the results rather than the theory. Thus, we used to put emphasis on how to design the network structure and training schedule. However, there are few discussions about how the training performance of DNN is guaranteed. In this report, we describe how the NN guarantees the training performance from the basic case to the general case.

2 Background

Just like other machine learning methods, the purpose of NN is to estimate the ground truth y_i of a data point x_i. Specifically, DNN performs a linear or non-linear transformation by using a series of matrix operations and activation functions. In the training process, we iteratively optimize the DNN model to make the estimate as accurate as possible. Consider the basic case of NN, which is called "one-layer network" in this report: For some data point $x_i \in \mathbb{R}^d$, we use a matrix $w \in \mathbb{R}^{k\times d}$ to predict the label $y_i \in \mathbb{R}$ by using the following transform:

$$f(x, w) = \sum_{i=1}^{k} \phi(w_i^T x), \tag{1}$$

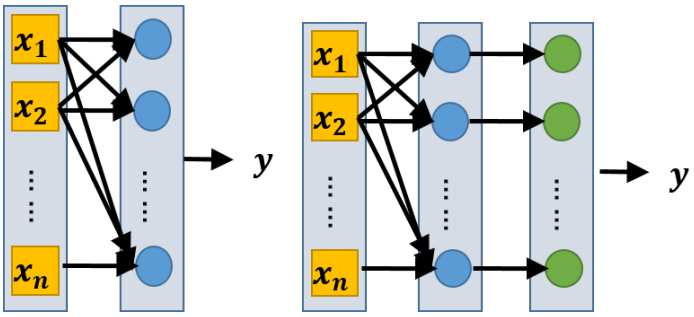

Figure 1: Illustrations of one-layer network (left) and two-layer network (right)

where ϕ denotes the activation function that can be linear or non-linear and $f(x, w)$ denotes the transformation of DNN. In the training process, our purpose is to make $f(x, w)$ and y as close as possible. Then, if we replace the direct sum with weighted sum, $f(x, w)$ is expressed as the following form, which is called "two-layer network" in this report:

$$f(x, w, a) = \sum_{i=1}^{k} a_i \phi(w_i^T x), \tag{2}$$

The illustration of one-layer and two-layer network are provided in Fig. 2. Note that the term "one-layer" and "two-layer" are not unified in our reference. This is why we give clear definitions here.

Then, let us consider the general form of DNN. When the network repeatedly uses matrix operation and activation functions to make the prediction, a multi-layer network is constructed. An expression of a H-layer deep neural network is given as follow:

$$x^{(h)} = \phi(w^{(h)} x^{(h-1)}), 1 \leq h \leq H, \tag{3}$$

$$f(x, w, a) = a^{(T)} x^{(H)}. \tag{4}$$

Again, note that the number of layers H represents the required number of matrix operation before making the prediction. After making the prediction $f(x, w, a)$, we would like to evaluate how good the performance is. Thus, we use a loss function for evaluation. Since the form of the loss function depends on the objective, in this report, we express the loss functions in the assumptions of each section.

Therefore, the model training is actually a process to iteratively search w, a and reduce the loss function until the searching results converge. Our descriptions focus on two major questions on model training: whether and how fast the searching results converge.

All the structures described above have a general name called "fully-connected network," which means that a neuron is connected to all the neurons in neighboring layers. However, this may take a lot of computational cost because when the dimension goes large, the amount of trainable parameters is very large. Therefore, an extension called convolutional neural network (CNN) [3] is developed. CNN uses the concept called "shared weight" to save the amount of trainable parameters. Specifically, the idea of CNN is widely adopted in computer vision topics.

We note that the optimization process of 1 has been discussed in Unit 5 of this lecture. Nevertheless, the scope only covers the convergence analysis under strong assumptions, such as assuming the objective

function is strongly-convex or smooth. In real cases, these assumptions are usually not held. Hence, if we depict the objective function w.r.t. some dimensions (the depiction is also called landscape), we may fail to find the optimal solution in the training process [4]. If the objective function is convex, we can compute the gradient to check whether the current point is at the global minimum. However, if the function is not convex, we may find some points called "local minimum" or "saddle point" that have low gradient but are not global minimum. Thus, it is important to get rid of these points in the training process.

To guarantee the convergence rate, there is an important condition called over-parameterization. This term means that the total number of neurons is more than the number of training samples. Actually, the requirement of over-parameterization is more strict according to the desired convergence rate and the network structure. With over-parameterization, the convergence rate is guaranteed since the model is powerful enough to represent the training data.

Moreover, if no assumptions are given in model training, the optimization process is very difficult. Specifically, even with very simple network structure, it is an NP-complete problem to determine whether the network is able to make the prediction match the ground truth with some weights and thresholds [5]. Therefore, we require some assumptions to analyze the convergence rate of the network. Intuitively, when different assumptions are given, different results are derived.

The following sections describe the convergence analysis from basic case to general case. That is, the description starts from one-layer network to multi-layer network.

3 One-layer Network

In this section, we first formulate the problem of model training in one-layer network. Then, we show that the convergence rate is guaranteed with some proof ideas.

3.1 Problem Formulation

The mathematical expression of a one-layer network is provided in 1. Specifically, we use rectified linear unit (ReLU) as activation function $\phi(\cdot)$ in this section. That is, $\phi(x) = max\{0, x\} = (x)^+$. In this section, we assume that the input data x is drawn from i.i.d. Gaussian distribution $\mathcal{N}(0, 1)$, and the corresponding label y is generated from a teacher network parameterized by w^*, that is,

$$y = f(x, w^*) = \sum_{i=1}^{k} (w_i^{*T} x)^+. \tag{5}$$

In the training process, we focus on the population risk instead of empirical risk. As we can see, the population risk takes the expectation over the square loss:

$$\mathcal{L}(w) = \mathbb{E}_{x \sim \mathcal{N}(0,1)} \left[(f(x, w) - f(x, w^*))^2 \right]. \tag{6}$$

Therefore, we can view the model training as a process that recovers w^* from some w by making the loss as small as possible:

$$\min_{w} \mathcal{L}(w). \tag{7}$$

3.2 Results and Proof

There are several papers solving the one-layer network optimization problem under different optimization algorithms, such as gradient descent (GD) [3] [6], stochastic gradient descent (SGD) [7], and projected gradient descent (pGD) [8]. All of them obtained high probability convergence guarantee. As shown in the results, if we put some constraints on the data pair (x, y), we obtain satisfied results regardless of the non-convex property of the objective function.

Note that the loss function we define in 6 only depends on w and w^*. By simple algebra, we find that the loss function and it's gradient have close-form solutions. Since we get rid of the dependency on data pair (x, y), we guarantee the training performance by some analyses. The overall idea of analyses in these papers are similar. Here we show one proof idea that is based on a CNN model [3]. The model can be expressed as follows:

$$f(x, w) = \sum_{i=1}^{k} (w^T x_i)^+. \tag{8}$$

In this model, the data $x \in \mathbb{R}^{kd}$ is partitioned into k patches. To further derive the loss function, we define the following function $g(\cdot, \cdot)$:

$$g(u, v) = \frac{1}{2\pi}\|u\|\|v\|\Big(\sin\theta_{u,v} + (\pi - \theta_{u,v})\cos\theta_{u,v}\Big). \tag{9}$$

where $\theta_{u,v}$ is the angle between u and v. Then, the loss function expressed in 6 can be expanded as follows:

$$\mathcal{L}(w) = \frac{1}{k^2}\Big[\gamma\|w\|^2 - 2kg(w, w^*) - 2\beta\|w\|\|w^*\|\Big], \tag{10}$$

where $\beta = \frac{k^2-k}{2\pi}$ and $\gamma = \beta + \frac{k}{2}$.

By calculating the partial derivatives, we can obtain the following lemma:

For $k > 1$, $\mathcal{L}(w)$ has three critical points:

(a) A local maximum at $w = 0$

(b) A unique global maximum at $w = w^*$

(c) A degenerate saddle point at $w = -(\frac{k^2-k}{k^2+(\pi-1)k})w^*$

Using GD with proper initialization of w to update the model. We guarantee that the loss function converges to an ϵ accurate solution after $\mathcal{O}(\frac{1}{\epsilon^2})$ iterations w.h.p.

However, the method we use in this section is not easy to be generalized. Thus, we need some different viewpoints to solve the optimization problem. This is why we continue our study to two-layer network.

4 Two-layer Network

In this section, we first show that the simple expansion technique can also be used in two-layer network. Then, to find some general concepts, we give two different viewpoints to analyze the two-layer network problem.

4.1 Simple Expansion

In this subsection, we consider a two-layer CNN as learning model [9]:

$$f(x, w, a) = \sum_{i=1}^{k} a_i (w^T x_i)^+. \tag{11}$$

For training data and loss function, we use the same setting as we described in one-layer network, to wit, we assume that the data point x follows i.i.d.Gaussian distribution $\mathcal{N}(0, 1)$ and label y is generated by a teacher network parameterized by (w^*, a^*). The population risk 6 is used as metric.

With the similar idea used in the previous section, we expand the loss function as follows:

$$\mathcal{L}(w) = \frac{\pi - 1}{2\pi}(\|w^*\|^2\|a^*\|^2 + \|a\|^2) - (\frac{2g(w, w^*)}{\|w\|} - \frac{\|w^*\|}{\pi})a^T a^* + \frac{\|w^*\|^2}{2\pi}(1^T a^*)^2 + \frac{1}{2\pi}(1^T a)^2 - 2\|w^*\|1^T a \cdot 1^T a^*, \tag{12}$$

where $g(\cdot, \cdot)$ is defined in 9. Then, we work on this function to get high probability convergence guarantee.

Note that the proof here also assumes that x follows i.i.d. Gaussian and the label y comes from a teacher network. Nevertheless, this assumption is not practical, we may want to get rid of this setting as well as keep the convergence guarantee. Furthermore, another interesting result is that under the high probability convergence guarantee, the objective function 6 does have a spurious local minimum. This fact raised our interest in studying the landscape of objective function.

In the following two method, we relax the assumptions of the training data pair (x, y), yet we can obtain convergence guarantee via over-parameterization, which means that the number of trainable parameters are larger than training samples.

4.2 Gram Matrix Analysis

Du et al. analyze the convergence rate by considering the induced gram matrix during model training [10]. The model is a typical two-layer network 2 with a ReLU activation function:

$$u(t) = f(x, w(t), a) = \sum_{i=1}^{k} a_i (w_i^T(t) x)^+. \tag{13}$$

In their setup, the second layer is fixed. So the model training is actually applying GD to update the first layer. Furthermore, w is updated with infinitesimal step size. Thus, we can view w as a continuous function that depends on time t. For simplicity, we use $u(t)$ to denote the prediction given by the model at time t .

The training process is actually an empirical risk minimization problem with a quadratic loss:

$$\mathcal{L}(w, a, x, y) = \frac{1}{2}\sum_{i=1}^{n}(f(x_i, w, a) - y_i)^2. \tag{14}$$

We assume WLOG that the data x satisfied $\|x_i\|_2 = 1$ and label y satisfies $\|y\|_2 < C$ for some constant C and for any $i \neq j$, $x_i \neq x_j$. Note that the constraint on x can be further loosened. When the constraint is loosened by some factor C_L, the requirement of over-parameterization is also loosened by the same

factor. Moreover, the assumptions here are more practical than those in the previous section. Under these assumptions, if the over-parameterization condition $k = \Omega(\frac{n^6}{\lambda_0^4 \delta 3})$ is satisfied, we have the following results with probability at least $1 - \delta$:

$$\|u(t) - y\|_2^2 \leq \exp(-\lambda_0 t)\|u(0) - y\|_2^2. \tag{15}$$

This result guarantees the linear convergence rate in model training. Then, we provide the proof sketch.

First, we calculate the loss function dynamics:

$$\frac{d}{dt}\|y - u(t)\|_2^2 = -2(y - u(t))^T H(t)(y - u(t)), \tag{16}$$

where $H(t)$ is the induced gram matrix with the following expression:

$$H_{ij} = \frac{1}{k} x_i^T x_j \sum_{r=1}^{k} \mathbb{I}\{x_i^T w_r(t) \geq 0, x_j^T w_r(t) \geq 0\}. \tag{17}$$

Moreover, it can be shown that for all time step t, if over-parameterization is satisfied, there is a lower bound of the least eigenvalue of $H(t)$ w.h.p.:

$$\lambda_{min}(H) > \frac{\lambda_0}{2}. \tag{18}$$

Combine 16 and 18, we derive the following result:

$$\frac{d}{dt}\|y - u(t)\|_2^2 \leq -\lambda_0 \|y - u(t)\|_2^2. \tag{19}$$

By integration on both sides, the result 15 is derived.

4.3 Landscape Analysis

Another viewpoint to analyze the convergence rate is to consider the landscape of the objective function. The concept is to consider the whole training process to travel in the valley with the purpose of finding some global minimum in the landscape, which implies that the loss function is minimized. Therefore, if the landscape has some desired properties, such as smoothness or no local minimum, it might be easy to converge to global minimum. Soltanolkotabi et al. study two-layer network with this viewpoint [11] and obtain favorable results. Let's start with the model structure. They use the model defined in 2 with quadratic activation function:

$$f(x, w, a) = \sum_{i=1}^{k} a_i (w_i^T x)^2. \tag{20}$$

The loss function they used is empirical quadratic loss 14. For any $w \in \mathbb{R}^{k \times d}$, if over-parameterized condition $kd \geq n$ is satisfied, the following result is derived by matrix analysis and algebra: if

$$\nabla \mathcal{L}(w) = 0 \text{ and } \nabla^2 \mathcal{L}(w) \succeq 0, \tag{21}$$

then w is the global minimum of $\mathcal{L}(w)$.

This result implies two things. One is that all local minimum are actually global minimum. The other is that every saddle point has a direction of negative curvature. Let us consider the latter case in detail. For a saddle point w^s, although it satisfied $\nabla\mathcal{L}(w^s) = 0$, the point is not the global minimum. Therefore, for each saddle point, there must be some direction that causes negative curvature and makes $\nabla^2\mathcal{L}(w^s) \succeq 0$ unsatisfied. That is, in the training process, we do not have to be afraid of being converged to sub-optimal solution in local minimum or saddle points.

Although the quadratic activation function is not commonly used, this paper still inspired us to study the landscape analysis of the objective function. In general cases that use multi-layer network, we believe that when some assumptions such as over-parameterization conditions are satisfied, we can still guarantee some desired properties of the landscape and thus guarantee the convergence of multi-layer network.

5 Multi-layer Network

In this section, we generalize the two concepts in the convergence analysis of two-layer network to multi-layer network. One is to consider the induced gram matrix and the other one is to consider the properties on the landscape. The expression of the network is given in 3 and 4. For simplicity, we assume the number of neurons at each layer of the network are the same. That is, every w_h is a $k \times k$ matrix except for the first layer.

5.1 Gram Matrix Analysis

Follow the idea of gram matrix analysis in the previous section. We take the same assumptions on training data (x, y). Moreover, we gave a looser constraint on the activation function $\phi(\cdot)$. We only require the function to be Lipschitz, smooth, analytical, and not a polynomial function. The loss function here is quadratic loss, just as defined previously 14.

Similar to the derivation in previous section, we derive the induced gram matrix $G_{ij}(t)$ during model training:

$$G_{ij}(t) = \sum_{h=1}^{H} \left\langle \frac{\partial f(x_i,t)}{\partial W^{(h)}(t)}, \frac{\partial f(x_j,t)}{\partial W^{(h)}(t)} \right\rangle + \left\langle \frac{\partial f(x_i,t)}{\partial a(t)}, \frac{\partial f(x_j,t)}{\partial a(t)} \right\rangle \tag{22}$$

However, this gram matrix depends on many parameters, including neural network parameters, which are time-variant. Thus, it is not easy to derive the desired properties of $G_{ij}(t)$ as we did in two-layer network cases. To address this issue, a fixed matrix $K^{(H)} \in \mathbb{R}^{n\times n}$ is used to give an approximation of $G_{ij}(t)$ for all t. That is, during the training process, the induced gram matrix $G_{ij}(t)$ will not be too far away from the fixed gram matrix $K^{(H)}$ under over-parameterization. The expression of $K^{(H)}$ is a little bit complicated since it is recursively defined, so we omit it in this report. The complete form can be found in the reference [12].

Here we provide the key ideas of the proof. The least eigenvalue of $K^{(H)}$ is denoted as λ_0. Then, with over-parameterization, the following properties are guaranteed:

$$\lambda_{min}(G^{(H)}(0)) \geq \frac{3}{4}\lambda_0 \tag{23}$$

$$\|G^{(H)}(t) - G^{(H)}(0)\| \leq \frac{1}{4}\lambda_0 \tag{24}$$

	Time Complexity	Model Complexity
Gram Matrix	$\Theta(n^2 2^{O(H)} \log \frac{1}{\epsilon})$	$\Omega(n^4 2^{O(H)})$
Landscape	$\Theta(\frac{n^6 H^2}{\delta^2} \log \frac{1}{\epsilon})$	$\Omega(\frac{n^{24} H^{12} d \log^5 k}{\delta^8})$

Table 1: The comparison between two concepts for convergence analysis

By the similar derivation in two-layer network cases, the linear convergence rate is guaranteed. However, one undesired thing is that the requirement of over-parameterization. This is because of that the perturbation in the first layer is exponentially propagating to the H-th layer.

5.2 Landscape Analysis

Li et al. consider the training process as a journey [13]. In this section, most of the assumptions are as same as those given in gram matrix analysis. An additional assumption is that each data point x_i has to be sufficiently far away from any other data point x_j. That is, $\|x_i - x_j\| \geq \delta$.

In this viewpoint, we first guarantee two basic properties of the whole landscape. One is that the magnitude of gradient is lower and upper bounded. The other is that the loss function is nearly smooth. In other words, the value of the function is close enough to the first-order approximation. From these two basic properties, linear convergence rate of the network is guaranteed.

The basic properties are guaranteed based on two things. First, by random initialization of network w, the forward propagation of the network is neither exploded nor vanished. That is, the output of each layer is close to 1. Second, if the global optimal is close to the initialization, the perturbation amount of each layer in the training process is sufficiently small. These two results are helpful in giving the bound of basic properties.

Note that in the derivation, the requirement of over-parameterization and number of iterations are polynomial degree. This is because in the proof based on basic properties, the convergence rate is given by assigning neuron numbers k some polynomial dependency on n and H. However, we note that the dependency is actually large in polynomial degree. The dependency can be loosened if better properties of the landscape is given, such as smoothness. The comparisons of the results of these two viewpoints are provided the next subsection.

5.3 Summary

We summarize the results of the analysis from these two viewpoints in Table 1. It seems that the landscape analysis gives a tighter bound of over-parameterization since it is at polynomial degree. Nonetheless, as we can see in Table 1, the degree of n in these two concepts differs a lot. Moreover, in most cases, the magnitude of n is much larger than that of H. Therefore, if the network is shallow, which means that H can be viewed as a constant, the bound of time and model complexity given by gram matrix analysis may be tighter than that given by landscape analysis. It is not necessary that which concept gives proper bound for model training, the selection of concepts highly depend on the network structure and training data.

6 Conclusion

In our survey, we describe the convergence analysis from the basic case to the general case. When surveying the basics case, We found two concepts that are applicable to the general case. These two concepts both guaranteed the linear convergence rate with over-parameterization. Moreover, these two concepts are based on different assumptions and thus have different requirement of over-parameterization. From the analysis in basic and general form of DNN model, we gained a deeper insight into how the training performance of DNN is guaranteed. We believe that these concept will be helpful for us to conduct the related research in the future.

7 Possible Extensions

In this report, the descriptions are under specific loss functions and activation functions. However, in practice, we often use different activation functions and loss functions according to the objectives. Therefore, one important extension is to survey the convergence analysis under more general activation and loss functions.

Furthermore, instead of only using a fully-connected network, many latest NN techniques use more complicated network structures according to the purpose. For example, a NN-based model for stereo matching may involve convolutional, deconvolutional, and residual neural network. Since the survey of these networks is already in presence, we believe that a more general form of convergence analysis for NN can be developed by integrating the previous conclusions.

8 Contribution of Each Group Member

Chi-Jui Ho: Paper survey, presentation, writing report
Che-Hsien Lin: Paper survey, slides preparation, writing report

Reference

[1] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," *CoRR*, vol. abs/1411.4038, 2014. [Online]. Available: http://arxiv.org/abs/1411.4038

[2] P. Fischer, A. Dosovitskiy, E. Ilg, P. Häusser, C. Hazirbas, V. Golkov, P. van der Smagt, D. Cremers, and T. Brox, "Flownet: Learning optical flow with convolutional networks," *CoRR*, vol. abs/1504.06852, 2015. [Online]. Available: http://arxiv.org/abs/1504.06852

[3] A. Brutzkus and A. Globerson, "Globally optimal gradient descent for a convnet with gaussian inputs," *CoRR*, vol. abs/1702.07966, 2017. [Online]. Available: http://arxiv.org/abs/1702.07966

[4] H. Li, Z. Xu, G. Taylor, and T. Goldstein, "Visualizing the loss landscape of neural nets," *CoRR*, vol. abs/1712.09913, 2017. [Online]. Available: http://arxiv.org/abs/1712.09913

[5] A. Blum and R. L. Rivest, "Training a 3-node neural network is np-complete," in *Proceedings of the First Annual Workshop on Computational Learning Theory*, ser. COLT '88. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988, pp. 9–18. [Online]. Available: http://dl.acm.org/citation.cfm?id=93025.93033

[6] Y. Tian, "An analytical formula of population gradient for two-layered relu network and its applications in convergence and critical point analysis," *CoRR*, vol. abs/1703.00560, 2017. [Online]. Available: http://arxiv.org/abs/1703.00560

[7] Y. Li and Y. Yuan, "Convergence analysis of two-layer neural networks with relu activation," *CoRR*, vol. abs/1705.09886, 2017. [Online]. Available: http://arxiv.org/abs/1705.09886

[8] M. Soltanolkotabi, "Learning relus via gradient descent," *CoRR*, vol. abs/1705.04591, 2017. [Online]. Available: http://arxiv.org/abs/1705.04591

[9] S. S. Du, J. D. Lee, Y. Tian, B. Póczos, and A. Singh, "Gradient descent learns one-hidden-layer CNN: don't be afraid of spurious local minima," *CoRR*, vol. abs/1712.00779, 2017. [Online]. Available: http://arxiv.org/abs/1712.00779

[10] S. S. Du, X. Zhai, B. Póczos, and A. Singh, "Gradient descent provably optimizes over-parameterized neural networks," *CoRR*, vol. abs/1810.02054, 2018. [Online]. Available: http://arxiv.org/abs/1810.02054

[11] M. Soltanolkotabi, A. Javanmard, and J. D. Lee, "Theoretical insights into the optimization landscape of over-parameterized shallow neural networks," *CoRR*, vol. abs/1707.04926, 2017. [Online]. Available: http://arxiv.org/abs/1707.04926

[12] S. S. Du, J. D. Lee, H. Li, L. Wang, and X. Zhai, "Gradient descent finds global minima of deep neural networks," *CoRR*, vol. abs/1811.03804, 2018. [Online]. Available: http://arxiv.org/abs/1811.03804

[13] Z. Allen-Zhu, Y. Li, and Z. Song, "A convergence theory for deep learning via over-parameterization," *CoRR*, vol. abs/1811.03962, 2018. [Online]. Available: http://arxiv.org/abs/1811.03962