Journal of Machine Learning Research 1 (2019) 1-48 Submitted 11/15; Published 11/15

Theoretical Analysis of Multi-layer Neural Network: A Survey

Abstract

Though multi-layer neural network has been used in a wide range of areas, the explanation about how it works well is still insufficient. This survey primarily focus on the theoretical analysis about the approximation ability and training process of neural network. For approximation ability, researches show that it can be arbitrarily strong with the increase of network complexity. Qualified relationship between such ability and width/depth has been partially constructed. For training, we investigates the SGD algorithm. To analysis its performance, the trajectory around loss surface critical points are surveyed, and some theoretical results about SGD instances are summarized.

Keywords: multi-layer neural network, theoretical deep learning, optimization theory

1. Introduction

Multi-layer neural network (or neural network/network for short in this survey, if no ambiguity is involved) has been proven a powerful method in a lot of fields, including natural language processing, image recognition, etc. Despite of these successes, the explanation about how it works so well is still largely missing. There are three major problems about the performance of neural network (Poggio et al., 2017):

1. Does DNN have enough expressive power to approximate the real world, or formally, for a specific dataset, is there a set of parameters with which training loss reaches zero or small enough?
2. Stochastic Gradient descent (SGD) and its variants have been applies widely in training DNNs, dose GN always converges to our desired result or how fast it converges?
3. Since modern DNNs is getting deeper and deeper, there are usually more parameters than training samples in a network. Why such overparameterization does not bring about severe over-fitting but instead keep good generalization performance?

A lot of researches have been done to explain and settle these problems, including theoretical analysis, simulations, visualizations. In this survey we focus on the theoretical analysis of the first two problems. We will introduce the development of research on these fields and some representative results.

2. Approximation

2.1 Universal approximation theorem

Because of the complexity and non-linearity, it is natural to conjecture that a neural network, under some mild assumptions, can arbitrarily approximate a wide range of functions.

This means neural network has enough expressive power to approximate lots of relationships in the universe. This topic was widely discussed in late 1980s and early 1990s. At this time, two-layer network is mostly studied. Cybenkot (1989) proved that:

Theorem 1 *Let σ be any continuous function, such that* $\lim_{x\to-\infty}\sigma(x)=0$*,* $\lim_{x\to\infty}\sigma(x)=1$*. Then finite sum of the form*

$$G(x)=\sum_{j=1}^{N}\alpha_j\sigma(y_i^T x+\theta_j) \tag{1}$$

where $\alpha_i,\theta_j\in\mathbb{R},y_i\in\mathbb{R}^n$, are dense in $C(I_n)$. In other words, for any continuous function $f:[0,1]^n\to\mathbb{R}$ and $\epsilon>0$, there is a function G of the above form, such that $\|f-G\|_\infty<\epsilon$.

It is easy to find out that $[0,1]^n$ can be replaced by any compact subset of $\mathbb{R}$.

This is the one of the first version of universal approximation theorem. This version is not perfect enough. For example, the activation function need to be continues and bounded. But some currently widely used activation functions, e.g. ReLU, is not bounded. Besides, the function to be approximated f needs to be continues, which is a strong restriction. There are quite a few other versions and improvements on this theorem, and partially addressed such problems. Hornik et al. (1989) (independent with Cybenkot's work) proved that σ need not be continuous, and instead of using l_∞ metric, any metric induced by a Borel measure is available to represent the approximation. Stinchcombe and White (1989) proved σ do not even need to be bounded. It only need to be measurable with some mild technical restrictions (to avoid being constant or linear function).

2.2 Width for approximation

Even though a network can arbitrarily approximate functions, the storage and computation power is still limited. Therefore the complexity of network need to be considered. The researches in last subsection only provide a exponential lower upper bound for network width. Such width is impractical for real world situations.

Barron (1993) improved above results and shows that a two-layer network with n hidden layer width can approximate a "smooth" function with $O(1/n)$ error. It is intuitive that smoother function is easier to approximate. To quantify the smoothness, Barron uses Fourier representation to introduce a smoothness metric:

Definition 2 *If the Fourier representation of $f:\mathbb{R}^d\to\mathbb{R}$ is*

$$f(x)=\int_{\mathbb{R}^d}e^{i\omega\cdot x}\tilde{f}(\omega)\,\mathrm{d}\omega \tag{2}$$

for some complex-valued function $\tilde{f}(\omega)$ for which $\omega\tilde{f}(\omega)$ is integrable, define

$$C_f=\int_{\mathbb{R}^d}|\omega|\left|\tilde{f}(\omega)\right|\mathrm{d}\omega \tag{3}$$

And the following error bound is proved:

Theorem 3 *Let σ be any continuous function, such that $\lim_{x\to-\infty}\sigma(x)=0$, $\lim_{x\to\infty}\sigma(x)=1$. f is a function with C_f finite. Then there exist $\alpha_i, \theta_j \in \mathbb{R}, y_i \in \mathbb{R}^n$, and*

$$G(x)=\sum_{j=1}^{N}\alpha_j\sigma(y_i^T x+\theta_j)+\alpha_0,^{1} \tag{4}$$

such that

$$\int_{\|x\|_2\le r}(f(x)-G(x))^2\,\mathrm{d}x\le\frac{(2rC_f)^2}{n} \tag{5}$$

$$\alpha_0=f(0),\quad \sum_{k=1}^{n}|\alpha_k|\le 2rC \tag{6}$$

It is noteworthy that this theorem shows that neural network is more efficient than traditional approximation approaches such as polynomial, spline and trigonometric expansions. Such approaches takes a fixed set of basis and uses linear combination of the basis. Barron (1993) also proved that such methods will not reach a error bound smaller than $O(n^{-2/d})$ (d is the dimensionality). Hence neural network performs better when facing the curse of dimensionality [2].

2.3 Depth for approximation

All networks discussed above have only one hidden layer. However, most of modern neural networks have far more than one hidden layer. E.g. ResNet used for image recognition can be as deep as 152 layers, which is proved much more efficient than shallower networks at that time (He et al., 2016). So depth is also an important factor determining the performance of a network.

Though it is shown that two layer network is powerful enough to approximate function, the width requirement might be unacceptable. Can depth help to solve this problem? One strong hint of this question comes from the study of boolean circuit, which is extensively studied (Telgarsky, 2016). As one of the most general results, Ha et al. (2016) proved that for every $d\ge 2$, there is an explicit n-variable Boolean function f, computed by a linear-size depth-d formula, such that any circuit with $d-1$ depth that agree with f on $(1/2+o_n(1))$ fraction of all inputs must have size $\exp(n^{\Omega(1/d)})$.

This result means, when the depth is increased by only 1, the expressive power of boolean circuit grows exponentially. Similarly we can expect such depth separation occurs for neural network, too. Telgarsky (2016) noticed that the superposition of ReLU, e.g. $\sigma(2\sigma(x)-4\sigma(x-1/2))$, produced a function whose figure looks like a triangle, with two oscillation, one up and one down (see Figure 1)

Chaining n such layers will create a function with 2^n oscillations. But intuitively the total oscillations number is linearly bounded with each layer's oscillations. So when depth is fixed, to simulate above depth-n network, width must increase exponentially. Telgarsky formalize this idea and proves:

1. The α_0 term does not change the structure of network: we only need to add a node whose β_i is zero and other parameters chosen properly.
2. Even though C_f might increase as d goes up, Barron points out that in large number of examples, C_f increase only at a polynomial speed with d, instead of exponentially.

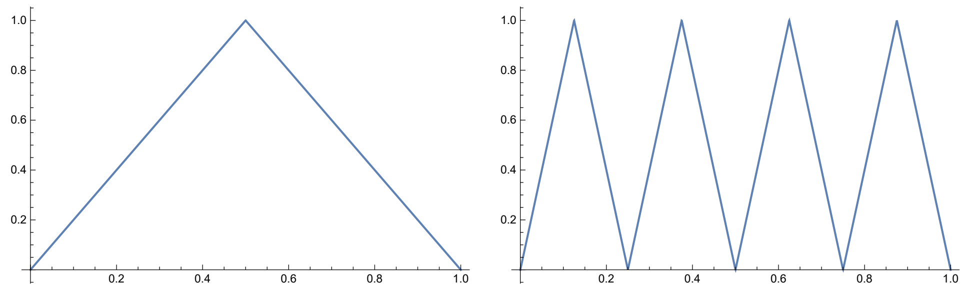

Figure 1: The triangle-like function and three-layer superposition

Theorem 4 *$\forall k, d \in \mathbb{N}_+$, there exists $f : \mathbb{R}^d \to \mathbb{R}$ computed by a neural network with ReLU activation function, with $O(k^3)$ layers and $O(k^3)$ nodes, such that for any function computed by a neural network with $\leq k$ layers and $\leq O(2^k)$ nodes.*

This result, from the perspective of dimension, is rough though. For more accurate proposition, the separation between two-layer network and three-layer network has been mostly studied. (Eldan and Shamir, 2016) proves that three-layer network sometimes needs exponentially smaller width than two-layer networks. However, a universal conclusion about depth separation, like the case of boolean circuit, is still missing.

3. Training

Even though neural networks are expressive enough for most relationships in reality, it is unpractical unless we can find a method to efficiently learn a network from training data. We may wonder whether there is a method to directly calculate the the desired parameters. But Blum and Rivest shows that this is not easy:

Theorem 5 *Consider the network consist of two layers - hidden layer with two node and output layer with one node, n-input, and threshold activation function (see figure 2). For a given set of training data, it is NP-complete to determine if there is proper parameters that can reach zero loss.* [3]

So instead of accurate minimization, we need to approximately minimize loss. If we define a loss function $\mathcal{L}(f_\theta(X), Y)$ to measure the loss given by a function f calculated by a network of parameters set θ training set X and corresponding labels Y, it turns to minimize $\mathcal{L}$. Optimization theory has provided many tools for such problems, among which *stochastic gradient descent* (SGD) has been proven the most practical one. After random initialization of θ, in each iteration, SGD calculates the gradient $\frac{\partial \mathcal{L}_\theta(X,Y)}{\partial \theta}$, then update all parameters proportional to the gradient:

$$\theta \leftarrow \theta + \eta \frac{\partial \mathcal{L}_\theta(X,Y)}{\partial \theta}, \tag{7}$$

where η is a constant called *learning rate*. There are quite a few variants of SGD, all of which depends on the gradient.

3. Since it is NP-complete to determine the existence of parameters, it is also NP-complete to calculate such parameters.

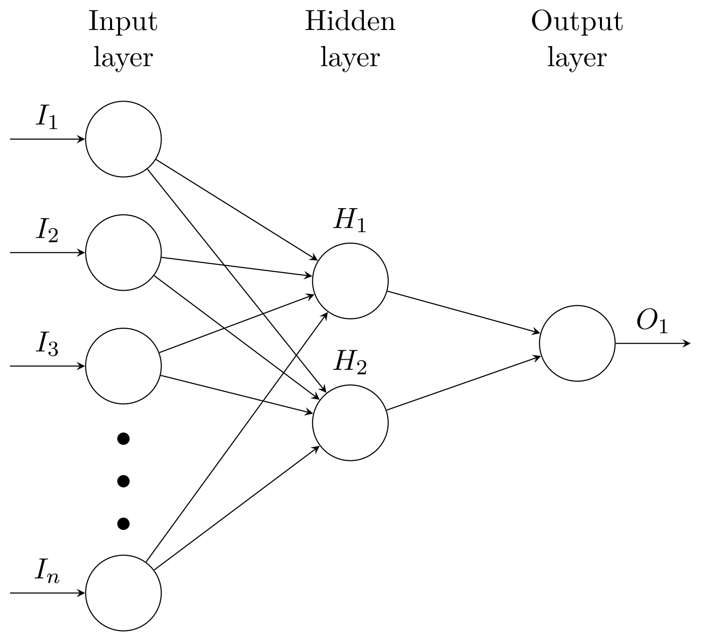

Figure 2: Three node network

3.1 The obstacle of SGD

The loss function, which is usually non-convex and non-smooth, seems hard to minimize via a method as simple as SGD. In practice, when actually training a network with SGD, we may notice sometimes the training loss will not converge stably to an ideal value, but just stalls or oscillate around a bad level. We may think this is the effect of local minima[4]. But in fact, trouble of local minima is not prominent. Ripley (1994) mentioned, "For instance, almost none of the neural net people seem to worry about landing in local minima".

One conjecture is that in neural network, local minima appears rarely in general functions. This conjecture is partially proven true theoretically. Yu (Nov./1992) illustrated that for a two-layer network with sigmoid activation function and square-mean loss function, if the input size is larger than hidden layer width, (under some mild technical condition, again) there is no local minima in its loss surface. Kawaguchi (2016) extent this result for general networks, with some slightly stricter assumptions added.

This seems anti-intuitive, but it is actually not so surprising. As Dauphin et al. (2014) states, "It is often the case that our geometric intuition, derived from experience within a low dimensional physical world, is inadequate for thinking about the geometry of typical error surfaces in high dimensional spaces." For a one dimensional function $\mathbb{R} \to \mathbb{R}$, all critical points[5] are either local or global extremum (except those with zero Hessian matrix). But for high dimensional cases, it is much harder to produce a extremum. Roughly speaking, a critical point is a extremum only if all eigenvalues of its Hessian matrix are of the same

4. Unless particularly specified, global minima is not regarded as a local minima in this article.
5. A point with zero gradient.

sign (positive or negative). For a typical modern dimensionality that is much higher than one thousand, such a matrix is not common.

To find out the situation in practice, Dauphin et al. (2014) investigates the distribution of critical points in high dimensional loss surface and compared their eigenvalues with the corresponding loss. Dauphin et al. uses the fractions of negative eigenvalues among all eigenvalues to measure how a critical point behaves like a local (or global) minima. The result indicates that there is a strong relationship with this fraction and the training loss. In other words, even there is a local minima, its training loss is probably not much worse than global minima. However, even though SGD will not completely stall on a saddle point, it is still slow to jump out from saddle point. Because the gradient around a critical point is relatively small.

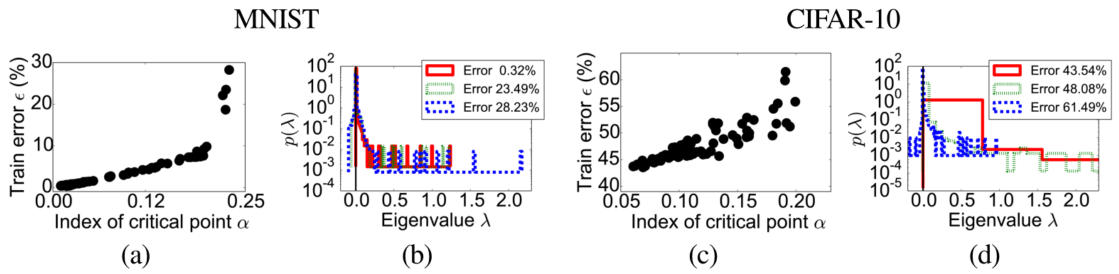

Figure 3: Experiment results of a network trained with MNIST and CIFAR-10 dataset. (a) and (c) demonstrate the relationship between α (the fraction of negative eigenvalues) and corresponding error for all critical points on loss surface. (b) and (d) choose three critical points of different loss, shows the histogram plot of eigenvalues of Hessian matrix.

We can introduce two-order optimization methods, such as Newton method or quasi-Newton method, to speed up the jumping. Dauphin et al. (2014) also propose a new training method called SGN(Saddle-free Newton).4 demonstrate the performance comparing of these methods on saddle points. Though these methods performs well near critical points, they are much slower than SGD, especially for high-dimensional networks that are prevalent nowadays. For example, calculating Hessian matrix spends $O(d^2)$ time compared with $O(d)$ of calculating gradient. Therefore one-order optimization method such as SGD is still the mainstream of neural network.

3.2 Instance analysis

In spite of difficulties above, SGD still works well in practice. And it is non-trivial that SGD often converges to a uniform loss level even with random initialization. In this subsection, we will demonstrate the convergence analysis on some kinds of networks, to somewhat explain the performance of SGD.

Li and Yuan (2017) examine a simple situation: two layer network with ReLU. All training samples are generated from another network of identical structure as the trained

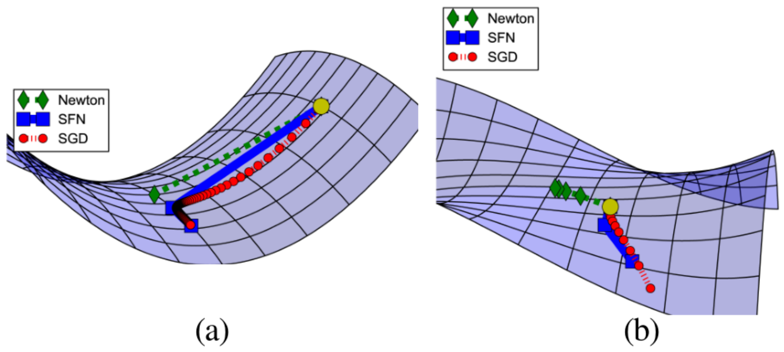

Figure 4: The trajectory around two different kinds of saddle points, trained with different methods.

one, called teacher network. According to universal approximation theorem, this assumption is not unrealistic, as long as the hidden layer is sufficiently wide. Denote the weight matrix of teacher network by W^*. Correspondingly, we called the network to be trained student network, with weight matrix denoted by W.

As Li and Yuan (2017) proves, after random (Gaussian) initialization, the learning process can be divided into two phases. In *phase I*, we choose small enough learning rate η, and apply SGD for $1/16\eta$ steps. In *phase II*, for a sufficiently large T, choose learning rate $\eta = C \log T/T$, where C is a constant determined by the network structure. After T steps, $\mathbb{E}\left\|W - W^*\right\|_F = O(\log T/T)$. [6]

This result has showed the versatility of SGD. Because it is robust against different initialization or training data. This result is still not general enough, because it only applies to two-layer network.

Allen-Zhu et al. (2019) claims, over-parametrization can solve this problem. Over-parametrization means the number of parameters in a network is more than the number of training samples. It gives the network enough flexibility to fit the training data. Over-parametrization is widespread among neural networks nowadays because of their increasing depth.

Allen-Zhu et al. (2019) considers a L-layer network with ReLU activation function. Denote the least relative distance between different training samples by δ. Then for $m \geq \mathsf{poly}(n, L, \delta^{-1})$, if each hidden layer is wider than m, the following result holds.

- If $m \geq \mathsf{poly}(n, L, \delta^{-1})$, starting from random Gaussian initialization and choose $\eta = \Theta(\frac{\delta}{\mathsf{poly}(n,L)m\log^2 m})$, SGD reaches a loss that is at most larger than the global minima by ϵ fraction, using no more than $\mathsf{poly}(n, L, \delta^{-1}) \log \frac{1}{\epsilon}$ iterations with $1 - e^{-\Omega(\log^2 m)}$ possibility.

6. $\|\cdot\|_F$ is Frobenius norm, defined as the square sum of singular values.

- For multi-label classification task, with similar conditions as above, SGD finds a classifier that is totally accurate on training set, after $\mathsf{poly}(n, L, \delta^{-1})$ iterations with the same possibility.

In other words, such networks probability converges in a linear speed, as long as it is over-parametrized enough[7]. It is important to notice that this convergence speed is similar to the case of simple linear neural network. In fact, the paper shows, "For a sufficiently large neighborhood of the random initialization, we prove that the training landscape is almost convex and semi-smooth.". Another key point is the polynomial dependency of L instead of exponential, which somehow explains the good performance of very deep neural network.

4. Conclusion

Neural network is a sophisticated structure with a lot of powerful properties. The complexity and non-linearity makes it a universal approximator to simulate relationships in reality. The Approximation ability is roughly linearly depends on width and exponentially depends on depth.

Neural network can hardly be optimized directly. Then SGD is a widely used method to approximate optimization. Though SGD is largely influenced by critical points on loss surface, we demonstrate that it still provably performs well in many situation.

Yet there is still a long way from the theory and the practice - the network complexity given by universal approximation theory is often unacceptable for calculating, the SGD converge time given by theory might be too long, and the research on basic full-connected feed-forward network cannot be directly applied to miscellaneous structures nowadays. But after all, these researches still matters for their ability to uncover deep learning from alchemic-like masks.

References

Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. A Convergence Theory for Deep Learning via Over-Parameterization. *arXiv:1811.03962 [cs, math, stat]*, June 2019. 120.

A.R. Barron. Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information Theory*, 39(3):930–945, May 1993. ISSN 0018-9448, 1557-9654. doi: 10.1109/18.256500. 2398.

Avrim Blum and Ronald L Rivest. Training a 3-Node Neural Network is NP-Complete. page 8.

G Cybenkot. Approximation by superpositions of a sigmoidal function. page 12, 1989. 11713.

7. It is not strange that both works are all based on ReLU activation function. Because ReLU is piecewise linear, so we can use linear algebra to describe it, simply with some extra data to track the signs. For the latter work, ReLU also plays a role in alleviating gradient exploding or vanishing, thus avoiding the exponential dependency of L.

Yann Dauphin, Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, Surya Ganguli, and Yoshua Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. *arXiv:1406.2572 [cs, math, stat]*, June 2014.

Ronen Eldan and Ohad Shamir. The Power of Depth for Feedforward Neural Networks. page 34, 2016.

Johan Ha, Benjamin Rossman, Rocco A Servedio, and Li-Yang Tan. An average-case depth hierarchy theorem for Boolean circuits. *Journal of the ACM*, 9(4):29, 2016.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, Las Vegas, NV, USA, June 2016. IEEE. ISBN 978-1-4673-8851-1. doi: 10.1109/CVPR.2016.90.

Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural Networks*, 2(5):359–366, January 1989. ISSN 08936080. doi: 10.1016/0893-6080(89)90020-8. 17000.

Kenji Kawaguchi. Deep Learning without Poor Local Minima. page 9, 2016.

Yuanzhi Li and Yang Yuan. Convergence Analysis of Two-layer Neural Networks with ReLU Activation. page 11, 2017. 172.

Tomaso Poggio, Hrushikesh Mhaskar, Lorenzo Rosasco, Brando Miranda, and Qianli Liao. Why and when can deep-but not shallow-networks avoid the curse of dimensionality: A review. *Int. J. Autom. Comput.*, 14(5):503–519, October 2017. ISSN 1751-8520. doi: 10.1007/s11633-017-1054-2. part1.

B. D. Ripley. Neural Networks and Related Methods for Classification. *Journal of the Royal Statistical Society. Series B (Methodological)*, 56(3):409–456, 1994.

Maxwell Stinchcombe and Halbert White. Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions — the Research Networking System. 1989. 321.

Matus Telgarsky. Benefits of depth in neural networks. page 23, 2016.

X.-H. Yu. Can backpropagation error surface not have local minima. *IEEE Trans. Neural Netw.*, 3(6):1019–1021, Nov./1992. ISSN 10459227. doi: 10.1109/72.165604.