Shallow Neural Networks, Gradient Descent, and Overparameterization

Xingfan Jia
Advisor: Professor Ery Arias-Castro
xij034@ucsd.edu

Abstract

This paper focuses on regression using fully connected one-hidden-layer ReLU activated neural networks. Weights are updated via gradient descent algorithm. We show that when we have the number of hidden nodes exactly as needed, there exist bad local minimums, so the convergence is not guaranteed. However, in the over-parameterized regime, we show that, under certain reasonable conditions, gradient descent will always find the global minimum. And due to the implicit regularization of gradient descent, the solution often has small generalization error surprisingly. We show the above with theoretical and some experimental support, thus accounting for the great success of neural networks nowadays.

Keywords: Regression, Neural Networks, Gradient Descent, Over-parameterization, Implicit Regularization

1. Introduction

Neural network was first introduced in 1960s, as an iformation processing system, and is now a very important factor in statistical learning. For example, in supervised learning, very deep network structures can complete image classification task at the level of human performance. The key of the network are the weights connecting different layers, which transforms inputs to desired outputs. Though the great success should be credited to many techniques applied to the networks, such as deep structures, convolutional layers, skip connections, and etc., but the most fundamental one is gradient descent.

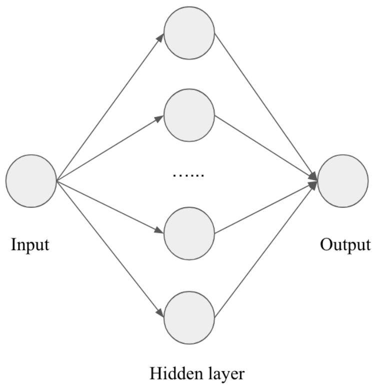


Figure 1: Fully connected one-hidden-layer Neural Network.

In this paper, we focus on analyzing gradient descent of fully connected one-hidden-layer ReLU activated neural network with m hidden nodes. Here is the setting, input $x_i \in \mathbb{R}^d$, output $y \in \mathbb{R}$, and $y = f(x) + e$, where e is some noise. Figure 1 shows the structure of our network. Input is connected to each hidden node by weight $w = [w_1 w_2 ... w_m] \in \mathbb{R}^{d\times m}$, $w_i \in \mathbb{R}^d$, and bias $b = [b_1 b_2 ... b_m]^T \in \mathbb{R}^m$. Output of each node goes through a rectified linear unit(ReLU) with activation function $\sigma(x) = max(0, x)$[1]. Then the output from each ReLU are combined by weight $v = [v_1 v_2 ... v_m] \in \mathbb{R}$ and bias $b' \in \mathbb{R}$. Formally, we consider the network of the following form:

$$f_m(x) = \sum_{i=1}^{m} v_i \sigma(x^T w_i + b_i) + b' \tag{1}$$

Rewrite it in matrix form:

$$f_m(x) = v^T \sigma(x^T w + b) + b'.$$

For simplicity, we expand x and let b be part of the w, then we have

$$f_m(x) = v^T \sigma(x^T w) + b'$$

Given n data pairs (x, y), the goal is to estimate the true underlying function f. We use least mean square error as loss function:

$$L(v, w) = \frac{1}{2n} \sum_{i=1}^{n} (f_m(x_i) - y_i)^2 \tag{2}$$

[1]If there is no non-linear activation function, network output is always a linear function.

.

We use gradient descent to update the weights. Gradient descent is an optimization algorithm that iteratively moves the weights along the direction which decreases the loss function most steeply. This direction is defined by the negative of the gradient. Let $w(t)$ be the weight of time t,

$$w(t+1) = w(t) - \eta \frac{\partial L(v,w)}{\partial w},$$

and

$$\frac{\partial L(v,w)}{\partial w_r} = \sum_{i=1}^{n} (f_m(x_i) - y_i) v_r x_i \mathbb{1}\{x_i^T w_r \geq 0\},$$

where w_r and v_r are weights connect input to rth hidden node and rth hidden node to output respectively, and η is a hyper-parameter learning rate, indicating how far w moves along that direction. Same updating process for v. In practice, people often use Stochastic Gradient Descent (SGD), a variant of gradient descent. Instead of taking derivative of every input data, it takes a batch of input data each time. So for SGD,

$$\frac{\partial L(v,w)}{\partial w_r} = \sum_{i=1}^{l} (f_m(x_i) - y_i) v_r x_i \mathbb{1}\{x_i^T w_r \geq 0\},$$

where l is the batch size, much less than n.

Experiments are conducted in Python with *keras*, a deep learning library backuped by TensorFlow. Noises are i.i.d. random normal.

2. Capacity of Neural Networks

Let's start with an example. Suppose we have a function with domain $[-1, 2]$, as shown in figure 2(b):

$$f(x) = \begin{cases} 0.5x, & \text{if } x \in [-1, 0] \\ 2x, & \text{if } x \in [0, 1] \\ 2, & \text{if } x \in [1, 2], \end{cases}$$

then clearly, one hidden node is not capable of fitting f. As shown in figure 2(a), one hidden node can achieve one of the 4 patterns. However, network with 2 hidden nodes can fit f. One of the possible optimal solution is shown in figure 2(c), where $h_1 = \sigma(-1.5x) = max(-1.5x, 0)$ is the output of first hidden node, $h_2 = \sigma(-2x + 4)$ is the output of second hidden node. $v = [-1, 1]$, $b' = 2$, $y = v^T[h_1, h_2] + b' = -h_1 + h_2 + 2$.

With more nodes, such network can fit more complicate functions. The following theorem states the capacity of single-hidden-layer neural networks.

Theorem 1 (Universal Approximation Theorem)**.** *Let $\sigma(\cdot) : \mathbb{R} \to \mathbb{R}$ be a nonconstant, bounded, and continuous function. Let the space of real-valued continuous functions on any compact set S in $\mathbb{R}^d$ be denoted by $C(S)$. Then, given any $\epsilon > 0$ and any function $f \in C(S)$, there exist an integer m, real constants $v_i, b_i \in \mathbb{R}$, and real vectors $w_i \in \mathbb{R}^d$ for i in {1, 2, ..., m}, such that we may define*

$$f_m(x) = \sum_{i=1}^{m} v_i \sigma(x^T w_i + b_i) + b'$$

which achieves

$$|f_m(x) - f(x)| < \epsilon \textit{ for all } x \in S.$$

This concept was introduced in late 1980s. It was first proved by Hornik, Stinchcombe and White (and also by George Cybenko independently) in 1989 for sigmoid activation functions, and later proved by Stinchcombe and White[1] for other activation functions. Here we use argument from Zhang et al [2] to show proof idea.

Theorem 2 (Zhang et al [2])**.** *There exists a two-layer neural network with ReLU activation and $2n + d$ weights that can represent any function on a sample of size n in d dimensions.*

Proof. For any two interleaving sequences of n real numbers $b_1 < z_1 < b_2 < z_2 < ... < b_n < z_n$, the $n \times n$ matrix $A = [\max(z_i - b_j, 0)]_{ij}$ has full rank, since A is lower triangular and all diagonal entries are nonzero. Its smallest eigenvalue is $\min_i z_i - b_i$.

Let the notations be same as in (1), we want to show that there exist weights w, v, b' so that $y_i = f_m(x_i)$ for all $i \in 1, ..., n$. First, choose w and b such that with $z_i = w^T x_i$ we have $b_1 < z_1 < b_2 < z_2 < ... < b_n < z_n$. Then we can write the n equations of unknown v $y_i = f_m(x_i)$ for $i \in 1, ..., n$ and $y = Av$. Since A has full rank, we can solve the linear system $y = Av$ to find suitable weights. $\square$

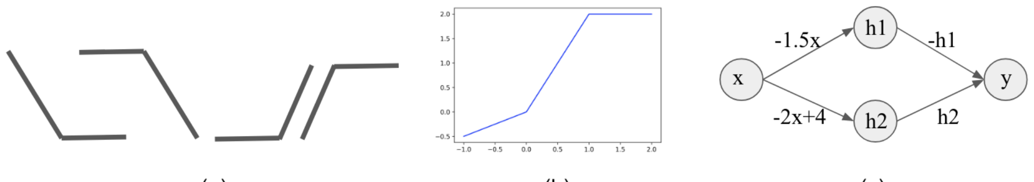


Figure 2: (a) shows 4 patterns of output of a hidden node, and (b) shows function f, and (c) shows how to achieve estimating f with 2 hidden nodes.

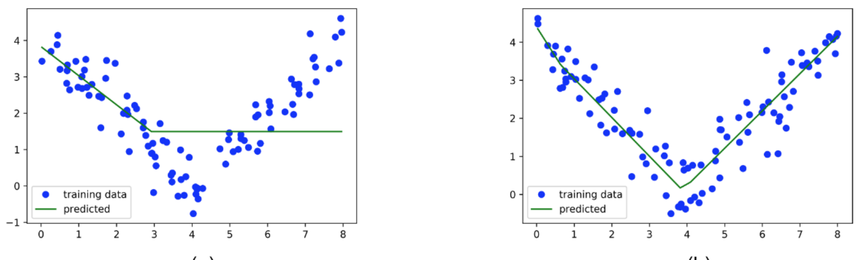


Figure 3: (a) Example of learning process stuck at bad local minimum. Trained with 2 hidden nodes. (b) Trained with 10 hidden nodes.

3. Bad Local Minima

As we can see from (2), the loss function is highly non-convex and non-smooth, with many critical points. In this section, we will talk about the regime where the optimization can be stuck at bad local minima. An example is given in figure 3. The blue dots are training data from some simple function (plus some Gaussian noises) that 2-hidden-node neural network is able to fit. And the green line is the network output estimation f_m, with $h_1 = -0.52x + 1.51$, $h_2 = -0.07x$, and $f_m(x) = 1.54\sigma(h_1) + 0.04\sigma(h_2) + 1.49$. Trying to change any parameter(s) with a fairly small amount would result in an increase in the loss, which is the case of stuck at bad local minima.

Safran and Shamir [3] studies this problem in a slightly different setting. They study the process of fitting w only, i.e. $f_m = \sum_{i=1}^{m} \sigma(w_i x)$. Training data pairs (x, y) are generated using the weight $w^* \in \mathbb{R}^{k\times k}$, $x \in \mathbb{R}^k$, $y = \sum_{i=1}^{k} \sigma(w_i^* x)$. Let L be a thrice-differentiable objective/loss function with a gradient $\nabla L(\cdot)$ and a Hessian $\nabla^2 L(\cdot)$ in a neighborhood of w^m, $w^m = [w_1, w_2, ..., w_m] \in \mathbb{R}^{km}$ is the vector of parameters. Let $R_{w^m,u,t}$ be the remainder from the second-order Taylor expansion of L about a point $w^m \in \mathbb{R}^{km}$, in a direction given by a unit vector $u \in \mathbb{R}^{km}$:

$$L(w^m + tu) = L(w^m) + t\nabla L(w^m)^T u + \frac{1}{2}t^2 u^T t \nabla^2 L(w^m)u + \frac{1}{6}t^3 R_{w^m,u,t}$$

To show the existence of local minima, we want to show that at some point, the loss is significantly greater than 0, gradient norm is very close to 0, and Hessian is strictly positive definite. Now suppose that the point w^m satisfies $\parallel \nabla L(w^m) \parallel \leq \epsilon$, $\parallel \nabla^2 L(w^m) \parallel \succeq \lambda_{min} I$, and $|R_{w^m,u,t}| \leq B_t$, for some positive ϵ, λ_{min}, and B_t (λ_{min} is the smallest eigenvalue of Hessian matrix), uniformly for all u. Fix $\alpha > 0$, and let $B = \sup_{t\in[0,\alpha]} B_t$, by the Taylor expansion above, we have

$$\begin{aligned} L(w^m + tu) &\geq L(w^m) - t \parallel \nabla L(w^m) \parallel \cdot \parallel u \parallel + \frac{t^2}{2}\lambda_{min} \parallel u \parallel^2 - \frac{t^3}{6}B \\ &= L(w^m) - \epsilon t + \frac{t^2}{2}\lambda_{min} - \frac{t^3}{6}B \\ &= L(w^m) + t(\frac{\lambda_{min}}{2}t - \frac{t^2}{6}B - \epsilon) \end{aligned} \tag{3}$$

The second term is strictly positive in the closed interval of $\frac{3\lambda_{min} \pm \sqrt{9\lambda_{min}^2 - 25B\epsilon}}{2B}$. Let $r = \frac{3\lambda_{min} - \sqrt{9\lambda_{min}^2 - 25B\epsilon}}{2B}$, and assuming $r < \alpha$, we have a ball B_r of radius r centered at w^m with boundary S, such that

$$L(w^m) < \min_{w'^m \in S} L(w'^m).$$

Since L is continuous, it is minimized over the ball at w^{*m}, and

$$L(w^{*m}) = \min_{w'^m \in B_r} L(w'^m) \leq L(w_m) < \min_{w'^m \in S} L(w'^m).$$

Thus, w^{*m} must be in the interior of B_r, so it is a local minimum. We refer readers to [3] for exhaustive proof, such as lower bound for λ_{min}, etc.

Safran and Shamir then show the likelihood of converging to bad local minima empirically. Again, k is the number of nodes used to generate training data, and m is the number of nodes used to fit the training data. Result of converging to bad local minima when using exact number of nodes as needed is shown in figure 4 (left half).

There are several ways to deal with this problem. For example, Du et al.[4] show that for such networks with Gaussian inputs, if we initialize network weights as (w, v), among $\{(w, v), (w, -v), (-w, v), (-w, -v)\}$, there is a pair that enables gradient descent to converge to the global minimum. Another direction for dealing with this problem is trying to design networks with loss landscapes that do not have such local minima. Including more nodes (parameters) is one way to achieve that.

In Safran and Shamir's experiment, they also show using one more node than needed could help greatly reduce the probability of converging to bad local minima. Also for the example given in figure 3, figure 3(b) shows the result of using 10 hidden nodes. Fitting the same function using 10 nodes gives much higher convergence rate in the experiment, though the estimated function is kind of zig-zag, where the true function f has only one kink point at $x = 4$.

We can think of two reasons for such behavior:

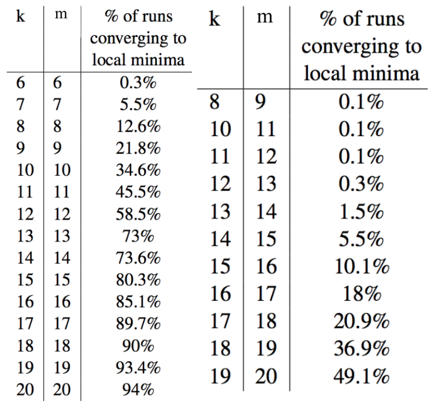
- Since the initialization is random, it is possible that certain nodes are never activated, i.e. rectified linear unit always outputs 0. In this case, we would have insufficient number of nodes.

- Since gradient descent implicitly penalizes the weights to go far from initialization(will be discussed later), the fitting process is very hard if the function requires large weights. However, if there are more nodes, each node with relatively small weights could be added up to achieve that.

4. Over-parameterization Regime

4.1. No Bad Local Minima

Last section encourages the idea of having more parameters than needed. Indeed, in practice, over-parameterized networks actually have great results. For example, AlexNet[5] achieves state-of-art image classification performance with about 650K hidden nodes, 60M parameters. And there is a large amount of work in the literature, saying that there is no bad local minima when the network is over-parameterized under different conditions. Here we use the argument from Soudrya and Carmon [6] to show the following theorem.



k	m	% of runs converging to local minima
6	6	0.3%
7	7	5.5%
8	8	12.6%
9	9	21.8%
10	10	34.6%
11	11	45.5%
12	12	58.5%
13	13	73%
14	14	73.6%
15	15	80.3%
16	16	85.1%
17	17	89.7%
18	18	90%
19	19	93.4%
20	20	94%

k	m	% of runs converging to local minima
8	9	0.1%
10	11	0.1%
11	12	0.1%
12	13	0.3%
13	14	1.5%
14	15	5.5%
15	16	10.1%
16	17	18%
17	18	20.9%
18	19	36.9%
19	20	49.1%

Figure 4: Result of convergence to bad local minima. Cited from [3].

Theorem 3 (Soudrya and Carmon [6]). *Let m be the number of hidden nodes in the network. If $m \geq dn$, then all differentiable local minima(DLM) of the loss function $L(v, w) = \frac{1}{2n}\sum_{i=1}^{n}(f_m(x_i) - y_i)^2$, are global minima with $L(v, w) = 0$.*

Proof. Bias terms are ignored for simplicity. Let $a = [a_1, a_2, ..., a_n]$, where a_i is the slope of the activation function given ith data point x_i. Let $e = [e_1, e_2, ..., e_n]$, where $e_i = f_m(x_i) - y_i$, then we can rewrite

$$\begin{aligned} L(v,w) &= \frac{1}{2n} \parallel e \parallel^2 = \frac{1}{2n}\mathbb{E}e^2 \\ &= \frac{1}{2n}\mathbb{E}(y - v^T diag(a)wx)^2 \\ &= \frac{1}{2n}\mathbb{E}(y - a^T diag(v)wx)^2 \\ &= \frac{1}{2n}\mathbb{E}(y - a^T\tilde{W}x)^2, \end{aligned} \tag{4}$$

where $\tilde{W} = diag(v)w$ for simplicity. Let $\tilde{W}$ be a DLM of loss function, equivalent to (w, v) is a DLM.

To we take derivative of (4), we can switch the order of differentiation and expectation and average over a finite training set. Also, derivative of a with respect to weights is 0. Thus we have the following, which equals 0 because $\tilde{W}$ is a DLM:

$$\nabla_{\tilde{W}} L = \mathbb{E}[eax^T] = 0 \tag{5}$$

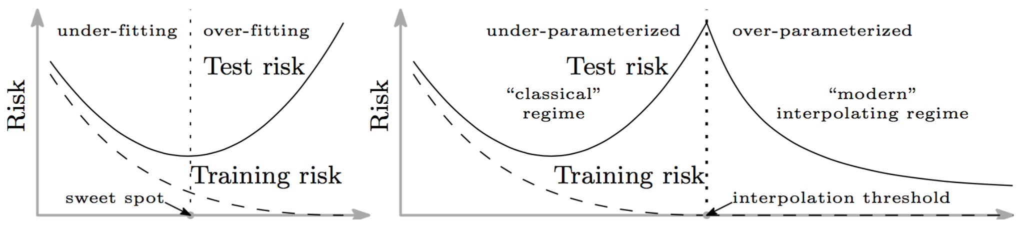


Figure 5: Number of parameters on x-axis. Classic "overfitting" curve on the left, and double descent curve on the right. Cited from [8].

Reshape this using Kronecker product $\otimes$, and define the gradient matrix:

$$G = [A \circ X] = [a_1 \otimes x_1, ..., a_n \otimes x_n] \in \mathbb{R}^{dm \times n}, \tag{6}$$

where $\circ$ is the Khatari-Rao product, and $x_i \in \mathbb{R}^m$. Then (5) becomes

$$Ge = 0 \tag{7}$$

So e lies in the nullspace of G, which is of dimension $n - rank(G)$. If $rank(G) = 0$, then $e = 0$, meaning the DLM is a global minimum.

To show G is generally full rank, we use the fact that if $B \in \mathbb{R}^{b \times n}$ and $C \in \mathbb{R}^{c \times n}$ with $n \leq bc$, we have, (B, C) almost everywhere, $\text{rank}(B \circ C) = n$. But we can not apply the fact directly since A depends on X, instead we apply it for for all (finitely many) possible values of $\text{sign}(wX)$ to get the desired result. $\square$

4.2. Good Generalization Behavior

In statistics, it is very common to encounter the "overfitting" problem when there are too many parameters. As there are many parameters, the model starts to fit the noises of input data, thus having poor generation behavior when predicting unseen data. Suppose we divide available data into two groups, training set and testing set. Training set is used to fit the model, we define the loss of training set as training risk. Then we use the resulting model to predict data in testing set, and define the loss over testing set as testing risk. The classic "overfitting" situation is shown in figure 5 (left plot).

However, Belkin et al.[8] find that test risk will increase and then decrease by keeping adding more parameters to the model. And their experiment with neural networks is shown in figure 6. The intuition behind such behavior is

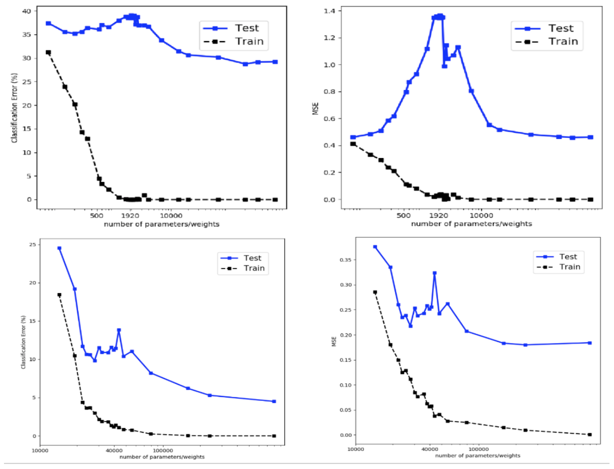


Figure 6: Two sets of experiments (one on each row) on MNIST dataset. Cited from [8].

that, by considering larger function classes, which have higher complexity, we have more ways to interpolate training data. Then we can choose the one with best generalization property among them.

Indeed, for over-parameterized networks, the loss function often has (uncountably infinite) many global minima on a given data set. And such networks, even without any explicit regularization, have good generalization behaviors in practice. It turns out that gradient descent has implicit regularization in terms of L2 norm, i.e. gradient descent always converge to the global minimum with the smallest L2 distance from initialization. There are many works in literature showing such property, for example, N. Azizan, and B. Hassibi[7][9] show that stochastic (a variant of gradient descent) with least-mean-square loss is a minimax optimizer (called H^∞ optimizer) which regularizes the solution in terms of L2 norm, Gunasekar et al.[10] and S. S. Du, et al.[11] use matrix factorization and Gram matrix[2], respectively, to show gradient descent with quadratic loss converges to the solution with minimum nuclear norm of weight matrix from initialization.

We will show work of N. Azizan, and B. Hassibi[7] to illustrate the good

[2]Gram matrix is defined by $H^\infty \in \mathbb{R}^{n\times n}$ with

$$H^\infty_{ij} = \mathbb{E}_{w\sim N(0,\mathbf{I})}[x_i^T x_j \mathbb{1}\{x_i^T w > 0, x_j^T w > 0\}].$$

convergence and implicit regularization of SGD, and refer readers to [7] for detailed proof. Their works is inspired by the H^∞ estimation and control theory that was first introduced in 1980s. The main idea is that, for linear model, SGD has the desired property as mentioned above. For our non-linear model (because of ReLU activation), it has that property locally. And over-parameterization gives infinitely many global minima, ensuring that, with high probability, there is at least one global minimum near random initialization, thus giving non-linear model that desired property.

Let f be a linear function, $y_i = f(x_i, w) + e_i$, and let W be the set of parameters vectors that are consistent with input data, i.e. $L(w) = 0$ for all $w \in W$. Let

$$c_i = y_i - x_i^T w_{i-1}, c_{p,i} = x_i^T w - x_i^T w_{i-1}.$$

Batch size for SGD is 1, and w_i is the weight after ith iterations, i.e. after feeding in ith input data point.

Lemma 1. *As in the setting described above, for any weight w, and for any step size $\eta > 0$, for the SGD iterates $\{w_i\}$ given in*

$$w_i = w_{i-1} - \eta(x_i^T w_{i-1} - y_i)x_i, \tag{8}$$

the following relation holds

$$||w - w_{i-1}||^2 + \eta e_i^2 = ||w - w_i||^2 + \eta(1 - \eta||x_i||^2)c_i^2 + \eta c_{p,i}^2 \tag{9}$$

for all $i \geq 1$.

This can be easily proved with (8) and rewriting $e_i = y_i - x_i^T w$ as $e_i = (y_i - x_i^T w_{i-1}) - (x_i^T w - x_i^T w_{i-1})$. This Lemma views SGD as a process of transferring uncertainty[3].

By summing (9) over $i = 1, ..., T$, we have the following Lemma.

Lemma 2. *Let the setting and condition be the same as in Lemma 1, we have*

$$||w - w_0||^2 + \eta \sum_{i=1}^{T} e_i^2 = ||w - w_T||^2 + \eta \sum_{i=1}^{T} (1 - \eta||x_i||^2)c_i^2 + \eta \sum_{i=1}^{T} c_{p,i}^2. \tag{10}$$

[3]Uncertainty of noise $\{e_i\}$ and uncertainty of weight $w - w_i$.

Now consider the minimax problem,

$$\min_{\{w_i\}} \max_{w,\{e_i\}} \frac{||w - w_T||^2 + \eta \sum_{i=1}^{T} c_{p,i}^2}{||w - w_0||^2 + \eta \sum_{i=1}^{T} e_i^2}. \tag{11}$$

The numerator is the energy of uncertainty of the unknown weight vector plus the energy of uncertainty of the prediction error after using T data points. The denominator is the energy of uncertainty of the unknown weight after initialization plus the energy of uncertainty of the noises. In one word, this minimax estimator minimizes the max energy gain from the uncertainties to prediction errors ($c_{p,i}$), which explains the robustness and conservativeness of the estimator because it safeguards the worst-case scenario. The next theorem shows that SGD is such estimator.

Theorem 4. *For any initialization w_0, any step size $0 < \eta < \min_i \frac{1}{||x_i||^2}$, and any number of steps $T \geq 1$, the stochastic gradient descent iterates $\{w_i\}$ given in (8) are the optimal solution to the minimax problem (11). Furthermore, the optimal minimax value (achieved by SGD) is 1.*

An older paper [9] shows that the algorithm achieves optimal solution 1 as long as w is closed enough to initialization. Having the convergence of SGD, now we show that it regularizes the solution in terms of $l2$ norm without any explicit regularization term. When interpolating, e_i's are zero[4], so $c_i = c_{p,i}$. Then (10) becomes

$$||w - w_0||^2 = ||w - w_T||^2 + \eta \sum_{i=1}^{T} (2 - \eta ||x_i||^2) c_i^2 \tag{12}$$

for all $w \in W$. As $T \to \infty$,

$$\eta \sum_{i=1}^{\infty} (2 - \eta ||x_i||^2) c_i^2 \leq ||w - w_0||^2,$$

so for $0 < \eta < \min_i \frac{1}{||x_i||^2}$, e_i has to go to 0 as $i \to \infty$. When $e_i \to 0$, SGD update vanishes, so $w \to w_\infty$ thus we get convergence. We also have $w_\infty \in W$ because $e_i \to 0$. If we take $T \to \infty$ and minimize both side of (12) with respect to w[5], we have

$$w_\infty = \arg\min_{w \in W} ||w - w_0||^2.$$

[4]Since f is linear in the current setting.

[5]Since (12) does not depend on w, it depends only on x_i, y_i, and w_0.

5. Discussion

This work intends to explain the excellent of over-parameterized neural networks. We provide some of the numerous approaches in the literature to this problem. However, there are still some restrictions and pre-assumptions on the settings in order to make the results valid. We believe that, as the restrictions are being removed, the picture will be much more clear, and people will have better ideas on how to improve their models.

References

[1] M. Stinchcombe, H. White, (1989). “Universal approximation using feedforward networks with non-sigmoid hidden layer activation functions.” In *Proceedings of the International Joint Conference on Neural Networks* (pp. 1:613-618). San Diego: SOS Printing.

[2] C. Zhang, et al. “Understanding deep learning requires rethinking generalization.” *arXiv preprint arXiv:*1611.03530 (2016).

[3] I. Safran, and O. Shamir. “Spurious local minima are common in two-layer relu neural networks.” *arXiv preprint arXiv:1712.08968* (2017).

[4] S. S. Du, et al. “Gradient Descent Learns One-hidden-layer CNN: Don’t be Afraid of Spurious Local Minima.” *arXiv preprint arXiv:*1712.00779 (2017).

[5] A. Krizhevsky, I. Sutskever, and G. E. Hinton. “Imagenet classification with deep convolutional neural networks.” *Advances in Neural Information Processing Systems.* 2012.

[6] D. Soudry, and Y. Carmon. “No bad local minima: Data independent training error guarantees for multilayer neural networks.” *arXiv preprint arXiv:*1605.08361 (2016).

[7] N. Azizan, and B. Hassibi. “Stochastic gradient/mirror descent: Minimax optimality and implicit regularization.” *arXiv preprint arXiv:*1806.00952 (2018).

[8] M. Belkin, et al. “Reconciling modern machine learning and the bias-variance trade-off.” *arXiv preprint arXiv:*1812.11118 (2018).

[9] B. Hassibi, A. H. Sayed, and T. Kailath. “Hoo optimality criteria for LMS and backpropagation.” *Advances in Neural Information Processing Systems.* 1994.

[10] S. Gunasekar, et al. “Implicit regularization in matrix factorization.” *Advances in Neural Information Processing Systems.* 2017.

[11] S. S. Du, et al. “Gradient descent provably optimizes over-parameterized neural networks.” *arXiv preprint arXiv:*1810.02054 (2018).