Gradient Descent Finds Global Minima of Deep Neural Networks

Simon S. Du, Jason D. Lee, Haochuan Li, Liwei Wang,Xiyu Zhai

Jue Guo, Enjamamul Hoq

University at Buffalo

Contents

Introduction

Related Works

Problem Setup

Analysis Techniques

Result
- Deep Fully-Connected Neural Network
- ResNet
- Convolutional ResNet

Conclusion

Abstract

- The paper proves that the gradient descent algorithm can achieve zero training loss in polynomial time for deep over-parameterized neural networks with residual connections (ResNet) and deep residual convolutional neural networks.
- The analysis relies on the particular structure of the Gram matrix induced by the neural network architecture, which allows showing the Gram matrix is stable throughout the training process and this stability implies the global optimality of the gradient descent algorithm.

Introduction I

- The introduction of the paper discusses the mystery of how randomly initialized first-order methods like gradient descent can achieve zero training loss in deep learning, even if the labels are arbitrary.
- The paper explains that over-parameterization is believed to be the main reason for this phenomenon, as only neural networks with sufficiently large capacity can fit all the training data
- The paper also mentions that many neural network architectures are highly over-parameterized in practice, such as Wide Residual Networks.

Introduction II

Main Goal

- Explain the two mysterious phenomena related to the training of deep neural networks:
  - Deep learning is randomly initialized first-order methods like gradient descent achieve zero training loss, even if the labels are arbitrary.
  - The second mysterious phenomenon in training deep neural networks is "deeper networks are harder to train."
- Given, n data points, H layers and width of the network m, The main contributions of the paper are as follows:
  1. In terms of a fully connected feed-forward network, if $m = \Omega\left(\text{poly}(n)2^{O(H)}\right)^1$, then randomly initialized gradient descent converges to zero training loss at a linear rate.
  2. For the ResNet Architecture, zero training loss is achieved at a linear rate if $m = \Omega(\text{poly}(n, H))$.
  3. For p number of patches and $m = (\text{poly}(n, p, H))$, zero training loss at a linear rate is achieved for Convolutional-ResNet.

Related Works I

1. Researchers are studying optimization problems in deep learning, and one approach is to develop a theory for non-convex problems with specific geometric properties.
2. A class of functions with all global minima and negative curvature for every saddle point has been identified, and perturbed gradient descent can find a global minimum for this class

Related Works II

1. One approach to solving the problem is to study the dynamics of a specific algorithm for a specific neural network architecture, as demonstrated in the paper.
2. The paper focuses on minimizing the training loss and proving that randomly initialized gradient descent can achieve zero training loss, rather than recovering the underlying neural network.

Related Works III

The work extends the previous results in several ways

- Deep Networks
- Generalize to ResNet architecture
- Generalize to convolutional networks

Problem Setup I

- Empirical risk minimization problem with the quadratic loss function

$$\min_{\theta} L(\theta) = \frac{1}{2} \sum_{i=1}^{n} \left(f\left(\theta, \mathbf{x}_i\right) - y_i \right)^2$$

  where $\{\mathbf{x}_i\}_{i=1}^n$ are the training inputs, $\{y_i\}_{i=1}^n$ are the labels, θ is the model parameter and f is the prediction function.
- For the prediction function, the neural network is used.
- Following three architectures are used.
  1. Multilayer fully-connected neural networks
  2. Resnet
  3. Convolutional ResNet

Problem Setup II

- Multilayer fully-connected neural networks.

$$\begin{aligned} \mathbf{x}^{(h)} &= \sqrt{\frac{c_\sigma}{m}}\sigma\left(\mathbf{W}^{(h)}\mathbf{x}^{(h-1)}\right), 1 \leq h \leq H \\ f(\mathbf{x}, \theta) &= \mathbf{a}^\top \mathbf{x}^{(H)}. \end{aligned} \tag{1}$$

Here, m = The network width.
H = the number of layers.
Model input = $\mathbf{x} \in \mathbb{R}^d$.
First weight matrix = $\mathbf{W}^{(1)} \in \mathbb{R}^{m \times d}$
$\mathbf{W}^{(h)} \in \mathbb{R}^{m \times m}$ is the weight at the h-th layer for $2 \leq h \leq H$
Output layer = $\mathbf{a} \in \mathbb{R}^m$ is the output layer.
Activation function = $\sigma(\cdot)$ (Lipschitz and Smooth) and is analytic and is not a polynomial function.

Problem Setup III

- Resnet.

$$\mathbf{x}^{(1)} = \sqrt{\frac{c_\sigma}{m}}\sigma\left(\mathbf{W}^{(1)}\mathbf{x}\right)$$
$$\mathbf{x}^{(h)} = \mathbf{x}^{(h-1)} + \frac{c_{res}}{H\sqrt{m}}\sigma\left(\mathbf{W}^{(h)}\mathbf{x}^{(h-1)}\right), \text{ for } 2 \le h \le H$$
$$f_{res}(\mathbf{x}, \theta) = \mathbf{a}^\top \mathbf{x}^{(H)} \tag{2}$$

- Convolutional ResNet

$$\mathbf{x}^{(1)} = \sqrt{\frac{c_\sigma}{m}}\sigma\left(\mathbf{W}^{(1)}\phi_1(\mathbf{x})\right) \in \mathbb{R}^{m\times p},$$
$$\mathbf{x}^{(h)} = \mathbf{x}^{(h-1)} + \frac{c_{res}}{H\sqrt{m}}\sigma\left(\mathbf{W}^{(h)}\phi_h\left(\mathbf{x}^{(h-1)}\right)\right) \in \mathbb{R}^{m\times p} \tag{3}$$
$$\text{for } 2 \le h \le H,$$

Here, operator $\phi_h(\cdot)$ is used to divide $\mathbf{x}^{(h-1)}$ into p patches.

Problem Setup III

- Randomly initialized gradient descent algorithm to find the global minimizer of the empirical loss is considered.
- For every level $h \in [H]$, each entry is sampled from a standard Gaussian distribution, $\mathbf{W}_{ij}^{(h)} \sim N(0,1)$ and each entry of the output layer a is also sampled from $N(0,1)$.
- All layers are trained by gradient descent, for $k = 1, 2, \ldots$, and $h \in [H]$
- Weight Initialization.

$$\begin{aligned} \mathbf{W}^{(h)}(k) &= \mathbf{W}^{(h)}(k-1) - \eta \frac{\partial L(\theta(k-1))}{\partial \mathbf{W}^{(h)}(k-1)}, \\ \mathbf{a}(k) &= \mathbf{a}(k-1) - \eta \frac{\partial L(\theta(k-1))}{\partial \mathbf{a}(k-1)} \end{aligned} \tag{4}$$

here,$\eta > 0$ is the step size.

Analysis Techniques I

- Global convergence of gradient descent proof is based on the study of the dynamics of differences between labels and prediction:

$$u_i(k) = f\left(\theta(k), \mathbf{x}_i\right) \tag{5}$$

- Above equation shows the individual prediction at the k_{th} iteration. $u_i(k)$ represents the output of a neuron in the kth layer of the network.

Analysis Techniques II

- Previous study showed that for a two-layer fully connected neural network, the sequence $\{\mathbf{y} - \mathbf{u}(k)\}_{k=0}^{\infty}$ admits the following dynamics:

$$\mathbf{y} - \mathbf{u}(k+1) = (\mathbf{I} - \eta \mathbf{H}(k))(\mathbf{y} - \mathbf{u}(k))$$

  where $\mathbf{H}(k) \in \mathbb{R}^{n \times n}$ is a Gram matrix with

$$\mathbf{H}_{ij}(k) = \left\langle \frac{\partial u_i(k)}{\partial \mathbf{W}^{(1)}(k)}, \frac{\partial u_j(k)}{\partial \mathbf{W}^{(1)}(k)} \right\rangle.$$

- The key finding is that if m is sufficiently large, $\mathbf{H}(k) \approx \mathbf{H}^{\infty}$ for all k where $\mathbf{H}^{\infty}$ is defined as $\mathbf{H}_{ij}^{\infty} = \mathbb{E}_{\mathbf{w} \sim N(0, \mathbf{I})} \left[\sigma' \left(\mathbf{w}^{\top} \mathbf{x}_i \right) \sigma' \left(\mathbf{w}^{\top} \mathbf{x}_j \right) \mathbf{x}_i^{\top} \mathbf{x}_j \right]$.
- If the m is large, then the dynamics of $\{\mathbf{y} - \mathbf{u}(k)\}_{k=0}^{\infty}$ is approximately linear

$$\mathbf{y} - \mathbf{u}(k+1) \approx \left(\mathbf{I} - \eta \mathbf{H}^{\infty}\right) (\mathbf{y} - \mathbf{u}(k))$$

Analysis Techniques III

- Implementing power method, it can be shown that $\{\mathbf{y} - \mathbf{u}(k)\}_{k=0}^{\infty}$ converges to 0 where the rate is determined by the least eigenvalue of $\mathbf{H}^{\infty}$ and the step size η.
- Adopting the above idea, it can be considered that the sequence $\{\mathbf{y} - \mathbf{u}(k)\}_{k=0}^{\infty}$, which admits the dynamics

$$\mathbf{y} - \mathbf{u}(k+1) = (\mathbf{I} - \eta \mathbf{G}(k))(\mathbf{y} - \mathbf{u}(k)) \tag{6}$$

Analysis Techniques IV

- In the previous equation,

$$\begin{aligned}
&\mathbf{G}_{ij}(k) \\
&= \left\langle \frac{\partial u_i(k)}{\partial \theta(k)}, \frac{\partial u_j(k)}{\partial \theta(k)} \right\rangle \\
&= \sum_{h=1}^{H} \left\langle \frac{\partial u_i(k)}{\partial \mathbf{W}^{(h)}(k)}, \frac{\partial u_j(k)}{\partial \mathbf{W}^{(h)}(k)} \right\rangle + \left\langle \frac{\partial u_i(k)}{\partial \mathbf{a}(k)}, \frac{\partial u_j(k)}{\partial \mathbf{a}(k)} \right\rangle \\
&\triangleq \sum_{h=1}^{H+1} \mathbf{G}_{ij}^{(h)}(k).
\end{aligned}$$

- Here we define, $\mathbf{G}^{(h)} \in \mathbb{R}^{n \times n}$ with $\mathbf{G}_{ij}^{(h)}(k) = \left\langle \frac{\partial u_i(k)}{\partial \mathbf{W}^{(h)}(k)}, \frac{\partial u_j(k)}{\partial \mathbf{W}^{(\mathbf{h})}(k)} \right\rangle$ for $h = 1, \ldots, H$. $\mathbf{G}_{ij}^{(H+1)}(k) = \left\langle \frac{\partial u_i(k)}{\partial \mathbf{a}(k)}, \frac{\partial u_j(k)}{\partial \mathbf{a}(k)} \right\rangle$.

  For all $h \in [H+1]$, each entry of $\mathbf{G}^{(h)}(k)$ is an inner product. Therefore, $\mathbf{G}^{(h)}(k)$ is a positive semi-definite (PSD) matrix for $h \in [H+1]$.

Analysis Techniques V

- Analysis of Random Initialization.
- First, $\mathbf{K}^{(H)}$ is a recursively defined matrix. That makes perturbation (due to randomness of initialization and finite m) from lower layers propagate to the H-th layer.
- Second, this perturbation propagation involves non-linear operations due to the activation function.

Analysis Techniques VI

- Analysis shows that ResNet architecture makes the "perturbation propagation" more stable.
- For a fully connected neural network, consider having perturbation $\left\|\mathbf{G}^{(1)}(0)-\mathbf{K}^{(1)}\right\|_2 \leq \mathcal{E}_1$ in the first layer. This perturbation propagates to the H-th layer and admits the form

$$\left\|\mathbf{G}^{(H)}(0)-\mathbf{K}^{(H)}\right\|_2 \triangleq \mathcal{E}_H \lesssim 2^{O(H)}\mathcal{E}_1.$$

- Which requires $\mathcal{E}_1 \leq \frac{1}{2^{O(H)}}$ and this makes m have exponential dependency on H.
- ResNet shows that the perturbation propagation admits the form

$$\mathcal{E}_H \lesssim \left(1+O\left(\frac{1}{H}\right)\right)^H \epsilon_1 = O\left(\epsilon_1\right)$$

Therefore we do not have the exponential explosion problem for ResNet.

Analysis Techniques VII

- Analysis of Perturbation During Training.
- The Following assumption is made and proved.
  $\mathbf{G}^{(H)}(k)$ is close to $\mathbf{G}^{(H)}(0)$ for $k = 0, 1, \ldots$.
  Here, $\mathbf{G}^{(H)}$ depends on weight matrices from all layers.
- The averaged Frobenious Norm is considered to show that
  $\mathbf{W}^{(h)}(k) - \mathbf{W}^{(h)}(0)$ is small for all $h \in [H]$ and $\mathbf{a}(k) - \mathbf{a}(0)$ is small to establish the fact that $\mathbf{G}^{(H)}(k)$ is close to $\mathbf{G}^{(H)}(0)$

$$\frac{1}{\sqrt{m}} \left\| \mathbf{W}^{(h)}(k) - \mathbf{W}^{(h)}(0) \right\|_F$$

Deep Fully-Connected Neural Network I

- Gradient descent with a constant positive step, size converges to the global minimum at a linear rate.
- The convergence rate depends on the least eigenvalue of the Gram matrix $K_(H)$.
- The Gram matrix $\mathbf{K}^{(H)}$ is recursively defined as follows, for $(i,j) \in [n] \times [n]$, and $h = 1, \ldots, H-1$

$$\mathbf{K}_{ij}^{(0)} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle ,$$

$$\mathbf{A}_{ij}^{(h)} = \begin{pmatrix} \mathbf{K}_{ii}^{(h-1)} & \mathbf{K}_{ij}^{(h-1)} \\ \mathbf{K}_{ji}^{(h-1)} & \mathbf{K}_{jj}^{(h-1)} \end{pmatrix},$$

$$\mathbf{K}_{ij}^{(h)} = c_\sigma \mathbb{E}_{(u,v)^\top \sim N\left(0, \mathbf{A}_{ij}^{(h)}\right)} [\sigma(u)\sigma(v)],$$

$$\mathbf{K}_{ij}^{(H)} = c_\sigma \mathbf{K}_{ij}^{(H-1)} \mathbb{E}_{(u,v)^\top \sim N\left(0, \mathbf{A}_{ij}^{(H-1)}\right)} \left[\sigma'(u)\sigma'(v)\right]$$

- The convergence rate and the amount of over- parameterization depends on the least eigenvalue of this Gram matrix

Deep Fully-Connected Neural Network II

- Convergence Rate of Gradient Descent for Deep Fully-connected Neural Networks. Assume for all $i \in [n], \|\mathbf{x}_i\|_2 = 1, |y_i| = O(1)$ and the number of hidden nodes per layer

$$m = \Omega\left(2^{O(H)} \max\left\{\frac{n^4}{\lambda_{\min}^4\left(\mathbf{K}^{(H)}\right)}, \frac{n}{\delta}, \frac{n^2 \log\left(\frac{Hn}{\delta}\right)}{\lambda_{\min}^2\left(\mathbf{K}^{(H)}\right)}\right\}\right)$$

- The formula implies that the number of hidden nodes per layer needs to increase exponentially with the number of layers to ensure convergence.
- Setting the step size as the following:

$$\eta = O\left(\frac{\lambda_{\min}\left(\mathbf{K}^{(H)}\right)}{n^2 2^{O(H)}}\right)$$

- With probability at least $1 - \delta$ over the random initialization the loss, for $k = 1, 2, \ldots$, the loss at each iteration satisfies

$$L(\theta(k)) \leq \left(1 - \frac{\eta\lambda_{\min}\left(\mathbf{K}^{(H)}\right)}{2}\right)^k L(\theta(0))$$

Deep Fully-Connected Neural Network III

- Above theorem states that if the width m is large enough and setting appropriate step size then gradient descent converges to the global minimum with zero loss at a linear rate.
- The main assumption of the theorem is that we need a large enough width for each layer.
- The width m depends on n, H and $1/\lambda_{\min}\left(\mathbf{K}^{(H)}\right)$.
- The dependency on n is only polynomial, which is the same as previous work on shallow neural networks.

ResNet I

The author further provides the convergence result of GD for ResNet.

- How much over-parameterization is needed to ensure the global convergence of gradient descent?
- The key Gram matrix is defined, whose least eigenvalue determines the convergence rate.

ResNet II: Definition

The Gram Matrix:

$$\mathbf{K}_{ij}^{(0)} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$$

To begin with,

$$\mathbf{K}_{ij}^{(1)} = \mathbb{E}_{(u,v)^\top \sim N} \left(\mathbf{o}, \begin{pmatrix} \mathbf{K}_{ii}^{(0)} & \mathbf{K}_{ij}^{(0)} \\ \mathbf{K}_{ji}^{(0)} & \mathbf{K}_{jj}^{(0)} \end{pmatrix} \right)^{c_\sigma \sigma(u)\sigma(v)}$$

- $\mathbf{K}_{ij}^{(1)}$: $\mathbf{K}_{ij}^{(0)} = \langle \mathbf{x}_i, \mathbf{x}_j \rangle$ is the initial covariance, 0 is the mean.
- c_σ: a constant
- $\sigma(u)$ and $\sigma(V)$: activation functions applied to u and v, respectively.

ResNet III: Definition

$$\mathbf{b}_i^{(1)} = \sqrt{c_\sigma}\mathbb{E}_{u\sim N\left(0,\mathbf{K}_i^{(0)}\right)}[\sigma(u)]$$

- $\mathbf{b}_i^{(1)}$: initial bias term for each neuron i.

$$\mathbf{A}_{ij}^{(h)} = \begin{pmatrix} \mathbf{K}_{ii}^{(h-1)} & \mathbf{K}_{ij}^{(h-1)} \\ \mathbf{K}_{ji}^{(h-1)} & \mathbf{K}_{jj}^{(h-1)} \end{pmatrix}$$

- It defines the matrix $\mathbf{A}_{ij}^{(h)}$ for layer h, it is constructed using the Gram matrix entries from the previous layer $h-1$

ResNet IV: Definition

$$\mathbf{K}_{ij}^{(h)} = \mathbf{K}_{ij}^{(h-1)} + \mathbb{E}_{(u,v)^\top \sim N\left(\mathbf{0}, \mathbf{A}_{ij}^{(h)}\right)} \left[\frac{c_{\text{res}}\, \mathbf{b}_i^{(h-1)} \sigma(u)}{H} + \frac{c_{\text{res}}\, \mathbf{b}_j^{(h-1)} \sigma(v)}{H} + \frac{c_{\text{res}}^2\, \sigma(u)\sigma(v)}{H^2} \right]$$

$$\mathbf{b}_i^{(h)} = \mathbf{b}_i^{(h-1)} + \frac{c_{\text{res}}}{H} \mathbb{E}_{u \sim N\left(0, \mathbf{K}_{ii}^{(h-1)}\right)} [\sigma(u)]$$

$$\mathbf{K}_{ij}^{(H)} = \frac{c_{res}^2}{H^2} \mathbf{K}_{ij}^{(H-1)} \mathbb{E}_{(u,v)^\top \sim N} \left(0, \mathbf{A}_{ij}^{(H-1)}\right) \left[\sigma'(u)\sigma'(v)\right]$$

Theorem 6.1 Overview

- Theorem 6.1 provides a convergence rate for gradient descent training in ResNet
- Assumptions:
  - For all $i \in [n]$, $\|\mathbf{x}_i\|_2 = 1$, and $|y_i| = O(1)$
  - Number of hidden nodes per layer (m) satisfies the over-parameterization condition

Over-parameterization Condition for ResNet

- $m = \Omega\left(\max\left\{\frac{n^4}{\lambda_{\min}^4(\mathbf{K}^{(H)})H^6}, \frac{n^2}{\lambda_{\min}^2(\mathbf{K}^{(H)})H^2}, \frac{n}{\delta}, \frac{n^2\log\left(\frac{Hn}{\delta}\right)}{\lambda_{\min}^2(\mathbf{K}^{(H)})}\right\}\right)$
- The condition ensures sufficient capacity for learning and stability during training and initialization

Step Size and Convergence Rate

- Set the step size $\eta = O\left(\frac{\lambda_{\min}(\mathbf{K}^{(H)})H^2}{n^2}\right)$
- With probability at least $1-\delta$ over random initialization:
  - $L(\theta(k)) \leq \left(1 - \frac{\eta\lambda_{\min}(\mathbf{K}^{(H)})}{2}\right)^k L(\theta(0))$

Convergence Rate Insights

- Convergence rate is polynomial in n and H
- Over-parameterization depends on $\lambda_{\min}(\mathbf{K}^{(H)})$, the smallest eigenvalue of the H-th layer's Gram matrix
- Skip connection block makes the architecture more stable in initialization and training phases

Terms in Over-parameterization Condition

$$m = \Omega\left(\max\left\{\frac{n^4}{\lambda_{\min}^4(\mathbf{K}^{(H)})H^6}, \frac{n^2}{\lambda_{\min}^2(\mathbf{K}^{(H)})H^2}, \frac{n}{\delta}, \frac{n^2\log\left(\frac{Hn}{\delta}\right)}{\lambda_{\min}^2(\mathbf{K}^{(H)})}\right\}\right)$$

- term1 & term2: Ensure Gram matrix stability during training
- term3: Guarantee output in each layer is approximately normalized at initialization
- term4: Bound the size of perturbation of the Gram matrix at initialization

Convolutional ResNet: Overview

- Generalize the convergence result of gradient descent for ResNet to convolutional ResNet
- Determine the necessary over-parameterization to ensure global convergence of gradient descent
- Define the Gram matrix $\mathbf{K}^{(H)}$ for convolutional ResNet

Definition 7.1 - Part 1

- Definition 7.1 focuses on the Gram matrix $\mathbf{K}^{(H)}$ for convolutional ResNet.
- The Gram matrix is recursively defined for $(i, j) \in [n] \times [n], (l, r) \in [p] \times [p]$ and $h = 2, \ldots, H - 1$.
- The first part of the definition describes the base case for $h = 0$:

$$\mathbf{K}_{ij}^{(0)} = \phi_1 (\mathbf{x}_i)^\top \phi_1 (\mathbf{x}_j) \in \mathbb{R}^{p \times p}$$

- $\mathbf{K}_{ij}^{(0)}$ represents the Gram matrix with only the input layer considered.

Definition 7.1 - Part 2

- The second part of the definition calculates $\mathbf{K}_{ij}^{(1)}$:

$$\mathbf{K}_{ij}^{(1)} = \mathbb{E}(\mathbf{u}, \mathbf{v}) \sim N\left(\mathbf{0}, \begin{pmatrix} \mathbf{K}_{ii}^{(0)} & \mathbf{K}_{ij}^{(0)} \\ \mathbf{K}_{ji}^{(0)} & \mathbf{K}_{jj}^{(0)} \end{pmatrix}\right)^{c_\sigma \sigma(\mathbf{u})^\top \sigma(\mathbf{v}),}$$

- This step calculates the Gram matrix for $h = 1$, considering the first layer of hidden nodes.
- $\mathbf{K}_{ij}^{(1)}$ captures the correlations between input data after passing through the first hidden layer.

Definition 7.1 - Part 3

- The definition also calculates $\mathbf{b}_i^{(1)}$ as follows:

$$\mathbf{b}_i^{(1)} = \sqrt{c_\sigma}\mathbb{E}_{\mathbf{u}\sim N\left(\mathbf{0},\mathbf{K}_{ii}^{(0)}\right)}[\sigma(\mathbf{u})]$$

- $\mathbf{b}_i^{(1)}$ represents the bias term for the i^{th} data point at layer $h = 1$.

Definition 7.1 - Part 4

- For $h = 2, \ldots, H-1$, the definition calculates $\mathbf{A}_{ij}^{(h)}$, $\mathbf{H}_{ij}^{(h)}$, $\mathbf{K}_{ij,lr}^{(h)}$, and $\mathbf{b}_{i}^{(h)}$.
- $\mathbf{A}_{ij}^{(h)}$ is calculated as follows:

$$\mathbf{A}_{ij}^{(h)} = \begin{pmatrix} \mathbf{K}_{ii}^{(h-1)} & \mathbf{K}_{ij}^{(h-1)} \\ \mathbf{K}_{ji}^{(h-1)} & \mathbf{K}_{jj}^{(h-1)} \end{pmatrix}$$

- $\mathbf{A}_{ij}^{(h)}$ is used for calculating the expected value of random vectors $\mathbf{u}$ and $\mathbf{v}$.

Definition 7.1 - Part 5

- The definition then calculates $\mathbf{H}_{ij}^{(h)}$:

$$\mathbf{H}_{ij}^{(h)} = \mathbf{K}_{ij}^{(h-1)} + \mathbb{E}_{(\mathbf{u},\mathbf{v})\sim N\left(\mathbf{0},\mathbf{A}_{ij}^{(h-1)}\right)} \left[\frac{c_{res}\mathbf{b}_i^{(h-1)\top}\sigma(\mathbf{u})}{H} + \frac{c_{res}\mathbf{b}_j^{(h-1)\top}\sigma(\mathbf{v})}{H} + \frac{c_{res}^2\sigma(\mathbf{u})^\top\sigma(\mathbf{v})}{H^2} \right]$$

- $\mathbf{H}_{ij}^{(h)}$ captures the interactions between input data after passing through the h^{th} layer of hidden nodes.

Definition 7.1 - Part 6

- The definition then calculates $\mathbf{K}^{(h)}_{ij,lr}$ and $\mathbf{b}^{(h)}_i$:

$$\mathbf{K}^{(h)}_{ij,lr} = \mathrm{tr}\left(\mathbf{H}^{(h)}_{ij,D^{(h)}_l D^{(h)}_r}\right)$$

$$\mathbf{b}^{(h)}_i = \mathbf{b}^{(h-1)}_i + \frac{C_{res}}{H}\mathbb{E}_{\mathbf{u}\sim N\left(\mathbf{0},\mathbf{K}^{(h-1)}_{ii}\right)}[\sigma(\mathbf{u})]$$

- $\mathbf{K}^{(h)}_{ij,lr}$ denotes the (l,r)-th entry of the Gram matrix for the h^{th} layer.
- $\mathbf{b}^{(h)}_i$ represents the bias term for the i^{th} data point at layer h.

Definition 7.1 - Part 7

- Finally, the definition calculates $\mathbf{M}_{ij,lr}^{(H)}$ and $\mathbf{K}_{ij}^{(H)}$ for the last layer:

$$\mathbf{M}_{ij,lr}^{(H)} = \mathbf{K}_{ij,lr}^{(H-1)} \mathbb{E}_{(\mathbf{u},\mathbf{v})\sim N\left(\mathbf{0},\mathbf{A}_{ij}^{(H-1)}\right)} \left[\sigma'\left(u_l\right)\sigma'\left(v_r\right)\right]$$

$$\mathbf{K}_{ij}^{(H)} = \operatorname{tr}\left(\mathbf{M}_{ij}^{(H)}\right)$$

- $\mathbf{K}_{ij}^{(H)}$ represents the Gram matrix for the final layer of the network.
- This definition allows us to analyze the convergence of gradient descent for convolutional ResNet by considering the Gram matrix at different layers.

Theorem 7.1: Convergence Rate of Gradient Descent for Convolutional ResNet

- Assume for all $i \in [n]$, $\|\mathbf{x}_i\|_F = 1$, $|y_i| = O(1)$, and the number of hidden nodes per layer:

$$m = \Omega(\max\{\frac{n^4}{\lambda_0^4 H^6}, \frac{n^4}{\lambda_0^4 H^2}, \frac{n}{\delta}, \frac{n^2 \log\left(\frac{Hn}{\delta}\right)}{\lambda_0^2} \text{poly}(p)\})$$

- If we set the step size $\eta = O\left(\frac{\lambda_0 H^2}{n^2 \text{ poly } (p)}\right)$, then with probability at least $1 - \delta$ over the random initialization we have for $k = 1, 2, \ldots$

$$L(\theta(k)) \leq \left(1 - \frac{\eta \lambda_{\min}\left(\mathbf{K}^{(H)}\right)}{2}\right)^k L(\theta(0))$$

Theorem Insights

- Similar to the ResNet theorem.
- The number of neurons required per layer is polynomial in the depth and the number of data points.
- Step size is polynomially small.
- The only extra term is poly(p) in the requirement of m and η.
- Analysis is similar to ResNet, refer to Section D for details.

Conclusion

- This paper shows that gradient descent on deep over parametrized networks can obtain zero training loss.
- Gram matrix is increasingly stable under over-parametrization.
- Future Research direction:
  1. Addressing the problem of showing gradient descent can also find solutions to low test loss.
  2. Improving the analysis to cover commonly utilized networks is an important open problem.
  3. Extending the analysis to Stochastic Gradient Descent while maintaining linear convergence rate.
  4. Taking minimum eigenvalue values into account to improve the convergence rate.