Identity Matters in Deep Learning

Gene Li
ELE538B Term Project

April 8, 2018

This is a literature review on the work [1] for the term project of ELE538B: Large Scale Optimization for Data Science, taken in Spring 2018.

1 Introduction

In the field of deep learning, it is common to initialize weights to be "near-zero", for example, $\sim \mathcal{N}(0, \sigma^2)$ for small σ^2. In many neural network architectures, the 0 mapping is represented in when all weights are equal to 0. This sort of neural network design has a flaw that it is difficult to learn the identity transformation, preserving the features that were already good. Moreover, when the depth of such networks is increased, a common phenomenon known as "vanishing gradients" occurs due to the fact that when all of the gradients are near zero, it is hard for the model to learn.

An explicit way to address this is through the use of residual networks, an architecture that represents the identity mapping when all weights are equal to zero. Given an input x to the layer, the output of the layer is represented by $x + h(x)$, where $h(x)$ is some combination of linear and nonlinear transformations. In this way, it is easy to learn the identity mapping.

This work explores the theory of linear residual networks. First, it gives a result for the optimization landscape of deep linear residual networks (i.e. an overparameterization of a linear model without non-linearities in each layer). It next gives a result for the expressivity of non-linear residual networks in terms of the number of parameters required to express any transformation for a dataset of size n and r classes. Lastly, they build a simplified all-convolutional network that performs well on CIFAR10, CIFAR100, and ImageNet.[1]

2 Theoretic Result 1

Consider the following problem setup. We are trying to learn a linear transformation $R : \mathbb{R}^d \to \mathbb{R}^d$ from measurements $y = Rx + \zeta$, where $x, y \in \mathbb{R}^d$ and $\zeta \sim \mathcal{N}(0, I_d)$. Define $\mathcal{D}$ the

[1]In this literature review, I will focus on the main theoretical results, deferring the empirical results to the original work because it is relatively straightforward and not particularly instructive to summarize.

distribution of x, with covariance $\Sigma = \mathbb{E}_{x\sim\mathcal{D}}[xx^T]$.
The classical way of solving this problem is constructing a least squares problem:

$$\min_{A} \mathbb{E}||y - Ax||^2 \tag{1}$$

However, we can consider the deep linear residual network overparameterization of this problem, i.e.

$$\min_{A1,\ldots,A_l} \mathbb{E}||y - (I + A_l)\ldots(I + A_1)x||^2, \tag{2}$$

where we replace the original matrix A with a product of matrices $(I + A_l)\ldots(I + A_1)$, for some $l \in \mathbb{Z}_+$. This new parameterization has the property that the identity mapping is expressed when all $A_i = 0$. Equation 1 is a **convex** optimization problem, however Equation 2 is **nonconvex** optimization problem. Simply by adding depth without changing expressivity of our model induces nonconvexity!

Here I insert a quick word about notation. Let us define $A \in \mathbb{R}^{\text{lxdxd}}$ as the tensor of stacked $A_1, A_2, \ldots, A_l$. In addition, let us call the maximum spectral norm of the A_i:

$$|||A||| := \max_{i\in[l]} ||A_i||. \tag{3}$$

Let us also define the population risk:

$$f(A) := \mathbb{E}||y - (I + A_l)\ldots(I + A_1)x||^2. \tag{4}$$

The main result of this work can be stated as follows:

Theorem 1 *The overparameterized optimization problem for any $\tau < 1$:*

$$\begin{aligned}&\min_{A1,\ldots,A_l} \mathbb{E}||y - (I + A_l)\ldots(I + A_1)x||^2\\ &\text{s.t. } |||A||| \le \tau.\end{aligned}$$

has the property that all critical points of the objective function $f(\cdot)$ are global minimum.

Even though the optimization landscape of Equation 2 is nonconvex, it has this very nice property that all critical points are global minimum. Thus an optimization algorithm only has to find a critical point.

However, there is one caveat. We need to satisfy this condition that the

$$|||A||| \le \tau. \tag{5}$$

Essentially, this means that for a given transformation R, we can find a set of $A_1, \ldots, A_l$ such that $R = (I + A_l)\ldots(I + A_1)$ and all the $I + A_i$ look close to the identity matrix. When R is PSD, this is easy to see: we can simply let each $A_i = R^{1/l} - I$. So first we need a result that says for general R, a solution that satisfies (5) exists:

Theorem 2 *Let l be the number of layers. Let $\gamma := \max\{|\log \sigma_{\max}(R)|, |\log \sigma_{\min}(R)|\}$. Suppose $l \geq 3\gamma$ and $det(R) > 0$. Then there exists $A^* = A_1, ...A_l$ such A^* minimizes population risk $f(\cdot)$ and*

$$|||A^*||| \leq \frac{(4\pi + 3\gamma)}{l}$$

We can think of γ as a constant through a rescaling argument (presented in the original work). Thus we can think of $\tau \sim O(\frac{1}{l})$.

While Theorem 1 shows a strong property about the optimization landscape for this overparameterized problem, it is still not clear that an algorithm like gradient descent will be able to reach the global minimum. It could be the case that we reach a point of small gradient, but our functional value is still far away from the global minimal value of $f(\cdot)$. The work shows that this is not the case: **gradient descent is guaranteed to converge the global minimum, provided that iterates stay inside the region $\mathcal{B}_\tau = \{A : |||A||| \leq \tau\}$.** In particular, the gradient has large norm compared to the error $f(A) - f^{\text{opt}}$:

$$||\nabla f(A)||_F^2 \geq 4l(1-\tau)^{2l-2}\sigma_{\min}(\Sigma)^2(f(A) - f^{\text{opt}}). \tag{6}$$

3 Theoretic Result 2

The previous result is nice because it justifies the use of gradient descent to solve these overparameterized deep linear residual networks. However, we also want to say something about deep **nonlinear** residual networks, which are used in practice. Proving an analogous result of Theorem 1 for nonlinear residual networks is actually very difficult. The paper instead focuses on giving a result for the **expressivity** of nonlinear residual networks: how many parameters do we need to perfectly express any function of a dataset of size n and r classes if we are only allowed to use linear transformation and ReLU activations?

Let's setup the following problem. Suppose we have a dataset of n training examples: $\{(x^{(i)}, y^{(i)})\}_{i\in[n]}$. Our data is $x^{(i)} \in \mathbb{R}^d$, labels $y^{(i)} \in \mathbb{R}^r$ encoded as one-hot basis vectors $e_1, ..., e_r$. The function we are allowed to use the ReLU activated transformation $\mathcal{T}_{U,V,s}(\cdot) : \mathbb{R}^k \to \mathbb{R}^k$ with linear transformations $U, V \in \mathbb{R}^{kxk}$, bias $s \in \mathbb{R}^k$ is defined as follows:

$$\mathcal{T}_{U,V,s}(h) = V\text{ReLU}(Uh + s). \tag{7}$$

We would like to construct a deep residual network with blocks of the form $x + \mathcal{T}_{U,V,s}(x)$. The result the work gives is the follows:

Theorem 3 *Assume that for all $i, j \in [n], i \neq j$, we have $||x^{(i)} - x^{(j)}||^2 \geq \rho$ for some small constant $\rho \geq 0$. Then, there exists a residual network N with $O(n \log n + r^2)$ parameters that expresses the training data: N maps each $x^{(i)}$ to $y^{(i)}$.*

Most neural network architectures have much more than $O(n \log n + r^2)$ parameters, thus this condition is easily met in practice.

A quick note about this result: the work only says that **there exists** a set of weights for a residual network N that are able to express any relabeling. It is not clear **whether the weights for this residual network N can be found by some optimization algorithm** - this is an extremely difficult problem that the work does not attempt to address.

Below we will summarize the construction with $O(n \log n + r^2)$ parameters that the paper uses. The in-depth proofs of why the construction is valid are deferred to the paper.

Construction of residual network N:

1. In the first layer, we map our input x to $h_0 = A_0 x$, where $A_0 \in \mathbb{R}^{kxd}$. This A_0 is taken to be a random matrix.
2. The middle "residual" layers all take the form of:
$$h_j = h_{j-1} + \mathcal{T}_{A_j, B_j, b_j}(h_{j-1}), \forall j \in [l].$$
3. The last layer maps h_l to the predicted y value: i.e.:
$$\hat{y} = \mathcal{T}_{A_{l+1}, B_{l+1}, b_{l+1}}(h_l), \forall j \in [l].$$
Here we have to map $h_l \to \mathbb{R}^r$, so we need $A \in \mathbb{R}^{rxk}$, $b \in \mathbb{R}^r$, $B \in \mathbb{R}^{kxk}$.

We can count up the number of parameters used in this construction. The first layer has kd parameters. Each of the middle residual layers has $2k^2 + k$. The last layer has $kr + r^2 + r$. We make the assumption that the number of labels r and input dimension $d << n$. The choices in the construction are also made:

- k is chosen to be $O(\log n)$.
- l is chosen to be $\lceil n/k \rceil$.

Thus total we have $O(kd + lk^2 + kr + r^2) = O(n \log n + r^2)$.

Now we will give some informal explanations as to why this construction, as well the choices for k and l, allow us to perfectly express the training data.

We can pick random unit vectors $q_1, ..., q_r \in \mathbb{R}^k$, which are preimages of the $e_1, ...e_r \in \mathbb{R}^r$ in the last layer. Then if we want a particular $x^{(i)}$ to map to $y^{(i)} = e_j$, the output of the penultimate layer must be q_j. Lets call this output before the last layer $v^{(i)}$. It's clear that we want these things:

1. We need to construct $A_1, ..., A_l$, $B_1, ..., B_l$ such that for each sample i, $h_0^{(i)}$ passed into the hidden layers gives us an output $v^{(i)} = q_j$.

2. The last layer, parameterized by $A_{l+1}, B_{l+1}, b_{l+1}$, must map each q_j to e_j for all possible labels $j = 1, ..., r$.

Indeed, this is possible! (Refer to Lemma 3.3 in the original work). Essentially, it is possible to show that in a given hidden layer, we can transform a set of k vectors into another set of k vectors, while keeping the rest $n - k$ vectors the same. Thus, in a given layer we can transform k of the $h_0^{(i)}$ into their desired $v^{(i)}$. In order for us to do the transformation for all n vectors $h_0^{(i)}$, we need at least $l = \lceil n/k \rceil$.

As to why we need $k = O(\log n)$, we need this in order to allow the distances between the vectors $x^{(i)}$ to be preserved by the transformation A_0 with high probability (a detail used in the proof - see Appendix B).

4 Conclusions

Let's recap the results of this work. The basic message is that there are some good theoretical reasons for why *identity parameterization* - the basic principle for residual networks - has had success.

In the setting of deep linear residual networks, this paper shows that even though we are changing a convex problem to a nonconvex problem, the resulting nonconvex problem still has a nice landscape: "all critical points are global minima". It is still possible to use a gradient descent algorithm to solve this overparameterized problem with the guarantee we reach the global minima.

In the setting of deep nonlinear residual networks (which is what we are really interested in!), a characterization of the optimization landscape is still an open problem. However, the paper gives a result about the expressiveness of deep nonlinear residual networks: using only hidden layers of the type $x + V\mathrm{ReLU}(Ux)$, we can construct a neural network that has relatively small number parameters yet still perfectly expresses the dataset. This is much simpler than the neural network architectures that are often used in practice, which often include multiple ReLU activations in each layer, along with batch normalization, dropout, and max pooling. Indeed, they show in their empirical section that a simple convolutional residual network performs extremely well on benchmarks CIFAR10, CIFAR100, and ImageNet.

References

[1] Moritz Hardt and Tengyu Ma. Identity matters in deep learning. *CoRR*, abs/1611.04231, 2016.