Exploring the effect of depth on optimization

Final Report

Authors: *Charlie Hou, Hrishikesh Khandeparkar, Gene Li*

Abstract

In a recent paper by Arora et al., it was discovered empirically that increasing depth in linear neural networks accelerates training for ℓ_p losses for $p > 2$. For our project we studied the asymptotic convergence rate for a toy problem from the above setting. We show that the asymptotic convergence rate for gradient descent for ℓ_p regression is $\theta(t^{-\frac{1}{p-2}})$. We also show that the asymptotic convergence rate for gradient descent on a deep linear neural net is also the same. Thus, the effect of depth is reducing the constant in ill conditioned problems. We confirm these results empirically.

Furthermore, Arora et al. experimentally found that increasing depth in a neural network similar to LeNet 5 accelerated training significantly for the MNIST dataset. In reproducing their results, we experimentally find that increasing depth for this nonlinear network does not accelerate the "true loss" (cross entropy) but substantially decreases the value of the (L2) regularizer . This suggests that depth is better at finding low-norm solutions with error similar to that of the solutions found by gradient descent.

1 Introduction

Deep learning's recent practical success has led to interest in understanding how it works on a theoretical level. One of the most interesting phenonmenona in deep learning is how depth in a neural network can improve model performance. Recent literature studying the model complexity of deep networks [MLP16] found that for deeper neural networks, the VC-dimension and fat-shattering dimension are much smaller than for shallow networks when the function to be learned is a composition of functions.

Furthermore, other papers have characterized the optimization landscape of deep networks: [HM16] found that for deep enough linear ResNets, (1) there exists global optimum that has low norm (2) any critical point with low norm is a global minimum. In addition, [LK17] found that in deep linear networks, all local minima are global minima.

In light of these developments, which seem to suggest that depth (1) can reduce the sample complexity of a model under conditions (2) doesn't seem to introduce a "difficult" optimization landscape, it is natural to wonder if depth itself could accelerate model training. This was studied in [ACH18], where they found that when training $\ell_p(p > 2)$ loss on linear networks, training is implicitly accelerated, since depth's effect seems to emulate momentum.

They also experimentally find that on the MNIST dataset, an archiecture similar to LeNet 5 trains significantly faster with more depth. In our paper, we want to expand on some questions raised in [ACH18]. What exactly are the convergence properties of adding depth? What are the details of what is happening on their experiment in the MNIST dataset?

In general, it is hard to study the effect of depth on neural networks because adding a layer along with a non-linearity significantly increases the model capacity.

To isolate the effect of depth, a popular model studied in literature is a *Linear Neural Net*.

Definition 1.1 (Linear Neural Net)**.**
A Linear Neural Net with d layers is a function $f : \mathbb{R}^n \to \mathbb{R}^m$ defined by d matrices $M_1, M_2, ..., M_d$ such that

$$f(\mathbf{x}) = M_d M_{d-1} ... M_2 M_1 \mathbf{x}$$

The benefit of studying the effect of depth in Linear Neural Nets is that depth does not affect model capacity. Indeed every Linear Neural Net of depth d can be viewed as a single linear mapping acting on a vector.

However, from the point of view of optimization, adding depth can significantly change the landscape of optimization. Adding depth can change the landscape of optimization to become non-convex. For instance the problem we study - realizable ℓ_p regression - can be formulated as follows

$$\min_{M} \mathop{\mathbb{E}}_{\mathbf{x},\mathbf{y}\sim D} [(M\mathbf{x} - M^*\mathbf{y})^p]$$

This is a convex problem and gradient descent converges to the true minimum i.e $M \to M^*$ as $t \to \infty$. However, even adding depth via a scalar to this problem makes the problem non-convex in s and M

$$\min_{M,s} \mathop{\mathbb{E}}_{\mathbf{x},\mathbf{y}\sim D} [(sM\mathbf{x} - M^*\mathbf{y})^p]$$

Understanding the of optimization of this non-convex objective is a central theme of our paper.

2 Convergence for Gradient Descent

To understand the effect of depth, we study the following toy problem. The input data points are simply $(0, 1) \to y_1$ and $(1, 0) \mapsto y_2$. Thus, we attempt to optimize l_p loss for the following objective (constants are omitted for simplicity)

$$\min_{m_1,m_2} (m_1 - y_1)^p + (m_2 - y_2)^p$$

We prove the following lemma in the context of the above toy problem

Lemma 2.1. Gradient descent with learning rate η converges to error ϵ at the rate

$$\epsilon = \Theta(\frac{1}{(\eta t)^{1/p-2}})$$

We prove this lemma in the appendix. We also give some rough evidence for why the dependence of acceleration with depth on t should be the same order of magnitude.

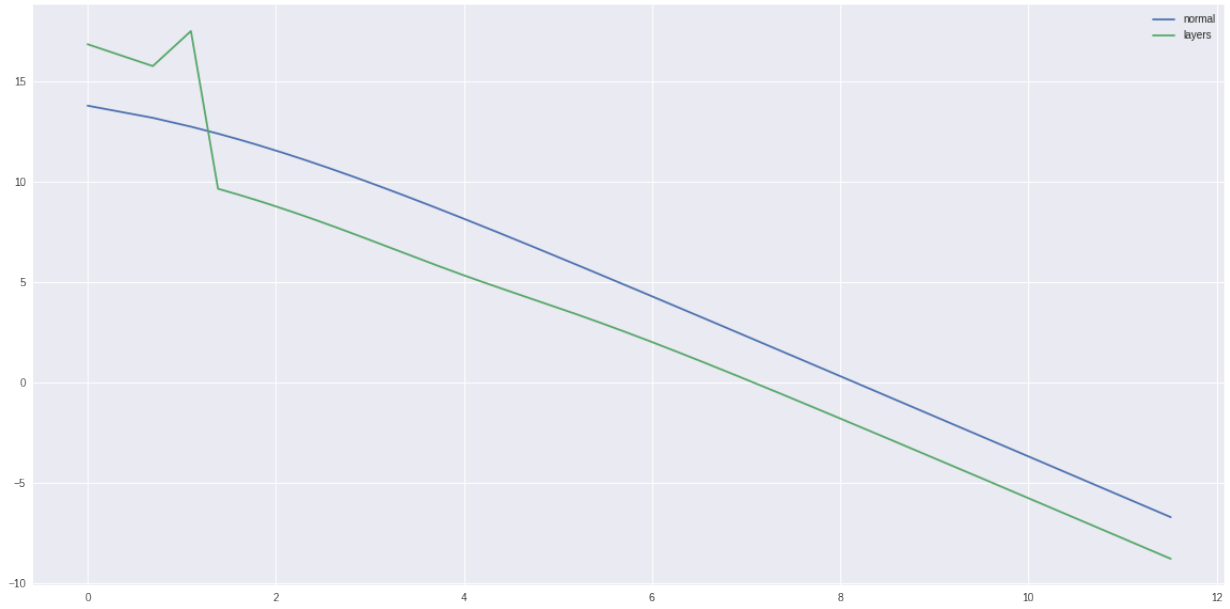


Figure 1: *The log-time vs log-error plot for gradient descent on ℓ_4 loss. Note that the slope of the line is approximately $1/2$. It is also interesting to see that the "acceleration" that occurs still shares the same asymptotic rate with respect to t.*

We empirically confirm this finding by running gradient descent on simulated problems. Note that the log-time vs log-error plots have slope $-\frac{1}{p-2}$ as predicted by the lemma. Further graphs and results are attached in the appendix.

3 Experiments on Non-linear Networks

We first tried to reproduce the results given in the MNIST experiment that [ACH18] had. The results are reproduced in figure 1.

We observed the same results that [ACH18] found: depth on the MNIST example does accelerate training. However, if we subtract the regularizer loss out, we find that the cross entropy loss of the overparameterized network does not outperform the cross entropy loss of the normal network. So in a sense, overparameterization does not seem to accelerate the "accuracy" of the network. To verify this, we also trained the two networks on just cross entropy and also found negligible difference in training loss.

We propose an explanation to this phenomenon: the (L2) regularizer term makes the optimization landscape of the loss function more "L2-like". Recall that [ACH18] showed depth induces momentum in linear networks (which may lead to acceleration) when the loss function is ℓ_p $(p > 2)$. So by adding the L2 penalty, the loss function (roughly speaking) looks more like ℓ_p $(p > 2)$, so (perhaps) the added depth leads to acceleration.

We summarize our finding below after experimenting with various other regularizations.

Conjecture 3.1 (Effect of Depth on Non-linear Networks)**.**

Figure 2: *The MNIST example reproduced: a LeNet 5-like architecture's training loss. "Normal" denotes the original network, and "overparameterized" denotes the original network, except with any matrix replaced as the product of two matrices. "No reg." means that we took out the regularizer term (L2) out of the loss calculation, to get a sense of what the "true training error" was. Note that when we calculate the regularizer term in the overparameterized setting, we take the product of two matrices and then take the norm for fair comparison with the "normal" version.*

(1) Adding depth to non-linear networks finds lower-norm solutions under ℓ_p regularization while not compromising the loss without the regularization term
(2) Depth in non-linear networks accelerates training with regularization terms because the loss with regularization makes the landscape more like ℓ_p loss.

Another thing that is interesting to note is how empirically depth's advantage seems to only be at the beginning of training, but asymptotically the losses decrease the same. In figure 2 one can see that the decrease in the loss for the over-parameterized model becomes fairly parallel to the loss function for the regular model. While the lemma we proved earlier shows that the asymptotic rate with respect to t varies the same, we do not have any rigorous results on the important initial iterates of the optimization, where depth seems to perform very well.]

4 Conclusion

We gave provide some theory on the asymptotics of convergence for depth in ℓ_p $p > 2$ and performed some empirical investigation on the effect of depth in non-linear networks. Some future directions would include investigating the effect of depth on cross-entropy loss.

Finally, and probably the most important thing to investigate is the behavior of the over-parameterized network's optimization in the first few steps. What causes it to work so well at the beginning? And why does this advantage disappear towards the end?

References

[HM16] M. Hardt and T. Ma. "Identity Matters in Deep Learning". In: *ArXiv e-prints* (Nov. 2016). arXiv: 1611.04231 [cs.LG] (1).

[MLP16] Hrushikesh Mhaskar, Qianli Liao, and Tomaso A. Poggio. "Learning Real and Boolean Functions: When Is Deep Better Than Shallow". In: *CoRR* abs/1603.00988 (2016). arXiv: 1603.00988. URL: http://arxiv.org/abs/1603.00988 (1).

[LK17] H. Lu and K. Kawaguchi. "Depth Creates No Bad Local Minima". In: *ArXiv e-prints* (Feb. 2017). arXiv: 1702.08580 [cs.LG] (1).

[ACH18] Sanjeev Arora, Nadav Cohen, and Elad Hazan. "On the Optimization of Deep Networks: Implicit Acceleration by Overparameterization". In: *CoRR* abs/1802.06509 (2018). arXiv: 1802.06509. URL: http://arxiv.org/abs/1802.06509 (1, 3).

A Analysis of toy problem

We prove the lemma from section 2.

Lemma 2.1. First, we note that the gradient update rule for m_i is as follows

$$m_i^{(t+1)} = m_i^{(t+1)} - \eta(m_i^{(t+1)} - y_i)^{p-1}$$

which can be written as

$$m_i^{(t+1)} - y_i = m_i^{(t+1)} - y_i - \eta(m_i^{(t+1)} - y_i)^{p-1}$$

and thus definig $\Delta_i^{(t)} = m_i^{(t)} - y_i$ we get

$$\Delta_i^{(t+1)} = \Delta_i^{(t)}(1 - \eta(\Delta_i^{(t)})^{p-2})$$

We show that this rule is $\Theta(\frac{1}{(\eta t)^{1/p-2}})$ We see this inductively as

$$\begin{aligned}
\Delta_i^{(t+1)} &= \Delta_i^{(t)}(1 - \eta(\Delta_i^{(t)})^{p-2}) \\
&= \Theta(\frac{1}{(\eta t)^{1/p-2}})(1 - \Theta(\frac{1}{t})) \\
&= \Theta(\frac{1}{(\eta)^{1/p-2}}) \left(\Theta(\frac{1}{(t)^{1/p-2}})(1 - \Theta(\frac{1}{t}))\right)
\end{aligned}$$

now we see that

$$\Theta\left(\left(\frac{t}{(t+1)}\right)^{1/p-2}\right) = \Theta\left(\left(1 - \frac{1}{t+1}\right)^{1/p-2}\right) \approx \Theta\left(1 - \frac{1}{2(p-2)(t+1)} + O\left(\frac{1}{t^2}\right)\right) \approx \Theta(1 - \frac{1}{t})$$

which yields that

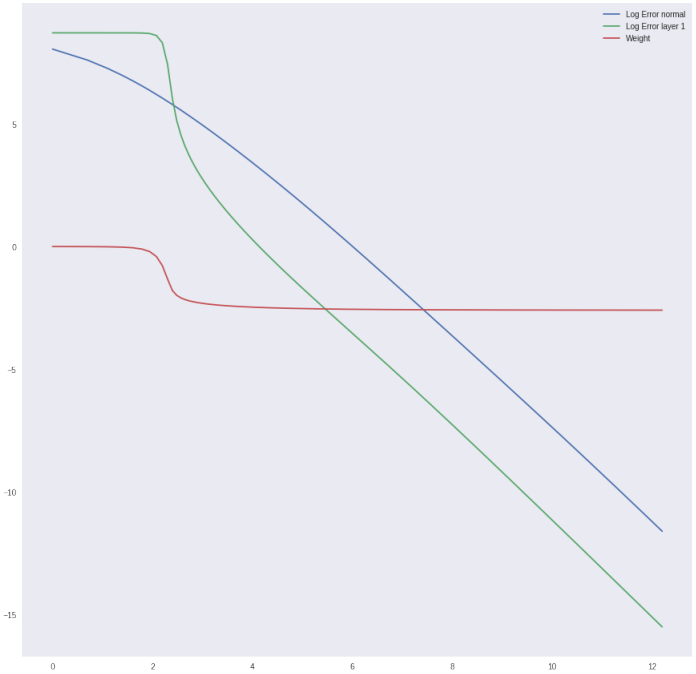
$$\left(\Theta(\frac{1}{(t)^{1/p-2}})(1-\Theta(\frac{1}{t}))\right) \approx \Theta\left(\frac{1}{(t+1)^{1/p-2}}\right)$$

which completes the induction. (we note that the base case is easy, yet messy to prove, because we can assume that for small enough η, the Δ_i converges at best geometrically initially until it Δ_i satisfied the requirement for the base case. Since the geometric convergence is exponentially faster than the convergence rate proven above, the number of iterates initially required to get to the base case are negligible. □

We give some intuition as to why the dependency on t is the same for the depth case too. Note that by adding a scalar the updates now become

$$\begin{aligned}
m_1^{(t+1)} &= m_1^{(t)} - \eta(sm_1^{(t)} - y_1)^{p-1}s \\
m_2^{(t+1)} &= m_2^{(t)} - \eta(sm_2^{(t)} - y_2)^{p-1}s \\
s^{(t+1)} &= s^{(t)} - \eta\left((sm_2^{(t)} - y_2)^{p-1}m_2^{(t)} + (sm_1^{(t)} - y_1)^{p-1}m_1^{(t)}\right)
\end{aligned}$$

Thus, we see that when $(sm_1^{(t)} - y_1)^{p-1}$ and $(sm_2^{(t)} - y_2)^{p-1}$ are small, s doesn't change much, and this just corresponds to a increased learning rate in the equations for m_1, m_2. Thus, it makes sense that the dependence on t is still the same. However, the constant term in the convergence rate changes, and although we cannot theoretically show this, our experiments show this below.



This is a log-error vs log-time plot of gradient descent vs. depth on ℓ_4 regression. The blue line is gradient descent without depth, and the green line is with the addition of a scalar term. The initial slowdown is because of a saddle point at 0. Note that the effect of acceleration is in the high error regime - this is also when the scalar s (red line) undergoes the most change. Once s stabilizes, the convergence rate is exactly the same as regular gradient descent.