Implicit Optimization Dynamics in Deep Linear Networks

Henry Steinitz
Courant Institue
New York, NY 10012
steinitz@nyu.edu

Abstract

Arora et al. [1] derive a time evolution equation for deep linear networks trained by gradient descent on a regression task with ℓ_p loss. The equation contains additional terms analogous to the "momentum" terms in gradient descent with momentum. We present a novel proof of this time evolution equation that uses simple algebraic properties of matrix multiplication. We also run an empirical experiment measuring the convergence rate of different linear architectures with respect to the ℓ_2 loss.

1 Introduction

Understanding the functional role of depth in neural networks is a highly active area of research. It's commonly known that a neural network with a single hidden layer is a universal approximator [3]. However, others have showed the existence of learning problems that can be solved with polynomial-sized deep networks, but not polynomial-sized shallow networks [4]. Thus, it is commonly believed that the primary role of depth is the efficient capacity to represent common data distributions, and that this increased capacity makes optimization more difficult.

More recent work suggests that depth may also accelerate optimization [1]. The authors consider deep linear networks since a linear network's capacity is independent of its depth. We present their main result in the next section.

2 Theoretical Results

Consider a deep linear networks with weight matrices $W_1, \ldots W_N$, with output given by

$$y = W_N \ldots W_1 x := W_e x,$$

and define the losses

$$L^1(W) = \frac{1}{p}(\sum_{k=1}^{m}(y_k - Wx_k)^p$$

$$L^N(W_1, \ldots, W_N) = \frac{1}{p}(\sum_{k=1}^{m}(y_k - W_N \ldots W_1 x_k)^p.$$

We begin by presenting a more direct proof of Arora et al.'s gradient descent theorem for deep linear networks.

Lemma 2.1. *Suppose $W_{j+1}^\top W_{j+1} = W_j W_j^T$ for $j = 1, \ldots, N-1$. Then,*

$$W_1^\top \ldots W_N^\top W_N \ldots W_1 = (W_1^\top W_1)^N$$

and

$$W_N \ldots W_1 W_1^\top \ldots W_N^\top = (W_N W_N^\top)^N.$$

Proof. We use induction. Both statements are trivially true when $N = 1$. Now suppose the first holds for $N = k$. Then

$$\begin{aligned}W_1^\top \dots W_{k+1}^\top W_{k+1} \dots W_1 &= W_1^\top (W_2^\top \dots W_{k+1}^\top W_{k+1} \dots W_2) W_1 \\ &= W_1^\top (W_2^\top W_2)^k W_1 \\ &= W_1^\top (W_1 W_1^\top)^k W_1 \\ &= (W_1^\top W_1)^{k+1}.\end{aligned}$$

Similarly, if we suppose the second equality holds for $N = k$, we have

$$\begin{aligned}W_{k+1} \dots W_1 W_1^\top \dots W_{k+1}^\top &= W_{k+1}(W_k \dots W_1 W_1^\top \dots W_k^\top) W_{k+1} \\ &= W_{k+1}(W_k W_k^\top)^k W_{k+1} \\ &= W_{k+1}(W_{k+1}^\top W_{k+1})^k W_{k+1} \\ &= (W_{k+1} W_{k+1}^\top)^{k+1}.\end{aligned}$$

□

We are now in the position to directly compute a step of gradient descent on L^N.

Theorem 2.2. *Assume that $W_1, \dots, W_n$ are updated continuously by gradient descent with respect to L^N. Or equivalently, assume that $\eta << 1$ so that powers of η are negligible. Assume also that their initial values satisfy, for $j = 1, \dots, N-1$:*

$$W_{j+1}^\top(0) W_{j+1}(0) = W_j(0) W_j^\top(0)$$

Then, the end-to-end weight matrix W_e is updated according to the following equation:

$$W_e(t+1) = W_e - \eta \sum_{i=1}^{N} \left((W_e W_e^\top)^{\frac{N-i}{N}} \frac{\partial L^1}{\partial W_e} (W_e^\top W_e)^{\frac{i-1}{N}} \right)$$

We omit the proof that $W_{j+1}^\top(t) W_{j+1}(t) = W_j(t) W_j^\top(t)$ for all t. A simple argument is presented in [1]. Given that condition, we may proceed with a direct calculation.

Proof. For each W_i we have

$$W_i(t+1) = W_i(t) - \eta \frac{\partial d L^N(W_1(t), \dots, W_N(t))}{\partial W_i}$$

From here on, we denote $W_i(t)$ as W_i for convenience. We can then express $W_e(t+1)$ as the product of individual layer steps:

$$\begin{aligned}W_e(t+1) &= W_N(t+1) \dots W_1(t+1) \\ &= (W_N - \eta \frac{\partial L^N}{W_N}) \dots (W_1 - \eta \frac{\partial L^N}{W_1}) \\ &= W_e + \eta (\frac{\partial L^n}{\partial W_N} W_{N-1} \dots W_1 + W_N \frac{\partial L^N}{\partial W_{N-1}} W_{N-2} \dots W_1 + \dots) \\ &\quad + O(\eta^2) + O(\eta^3) + \dots .\end{aligned}$$

Ignoring the higher-order terms and using a condensed notation gives

$$W_e(t+1) = W_e - \eta \sum_{i=1}^{N} \left(\left(\prod_{j=N}^{i+1} W_j \right) \frac{\partial L^N}{\partial W_i} \left(\prod_{j=i-1}^{1} W_j \right) . \right)$$

Note that $\frac{\partial L^N}{\partial W_i}$ may be written in terms of $\frac{\partial L^1}{\partial W_e}$:

$$\begin{aligned}\frac{\partial L^N}{\partial W_i} &= \frac{\partial}{\partial W_i} \frac{1}{p} \sum_{k=1}^{m} (y_k - W_N \dots W_1 x_k)^p \\ &= \prod_{j=i+1}^{N} W_j^\top \frac{\partial L^1}{\partial W_e} \prod_{j=1}^{i-1} W_j^\top .\end{aligned}$$

Plugging this back into our expression for $W_e(t+1)$ and applying our lemma yields

$$W_e(t+1) = W_e - \eta \sum_{i=1}^{N} \left(\left(\prod_{j=N}^{i+1} W_j \right) \left(\prod_{j=i+1}^{N} W_j^\top \right) \frac{\partial L^1}{\partial W_e} \left(\prod_{j=1}^{i-1} W_j^\top \right) \left(\prod_{j=i-1}^{1} W_j \right) \right)$$

$$= W_e - \eta \sum_{i=1}^{N} \left((W_N W_N^\top)^{N-i} \frac{\partial L^1}{\partial W_e} (W_1^\top W_1)^{i-1} \right)$$

Finally, our lemma also implies that $(W_N W_n^T)^N = W_e W_e^T$ and $(W_1^T W_1)^N = W_e^T W_e$. The final result then taking the N^{th} root of these equations:

$$W_e(t+1) = W_e - \eta \sum_{i=1}^{N} \left((W_e W_e^\top)^{\frac{N-i}{N}} \frac{\partial L^1}{\partial W_e} (W_e^\top W_e)^{\frac{i-1}{N}} \right)$$

The fractional powers are defined because $M^\top M$ and $MM^\top$ are positive semidefinite for all matrices M. $\square$

If we then choose to divide by η and take the limit as $\eta \to 0$, our discretized expression becomes the continuous expression presented in [1], sans the regularization constant. Note that our proof doesn't require singular value decompositions or other machinery present in the original proof.

3 Empirical Analysis

We used an automatic differentiation library (PyTorch) to test the performance of various linear network architectures in a simple regression task: UCI Machine Learning Repository's "Gas Sensor Array Drift at Different Concentrations". The code is available at github.com/henrysteinitz/linear-network-analysis.

For each architecture, we search for the approximately optimum learning rate over the values

$$\eta = 1 \times 10^{-12}, 3 \times 10^{-12}, \ldots, 1 \times 10^{-6}, 3 \times 10^{-6}$$

We then use the best learning rate to train each architecture for 30,000 epochs, and compute the record the loss every 50 epochs. We aren't interested in studying overfitting and generalization error, so we validate and compute our loss on the full dataset.

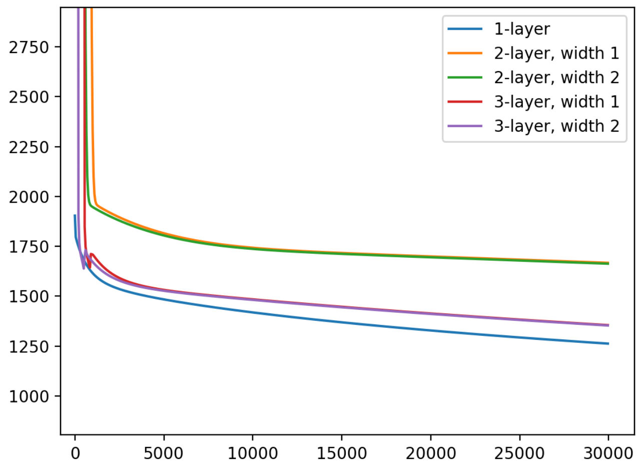

Figure 1: Learning curves of different linear architectures trained by gradient descent on a regression task. The x-axis represents the number of epochs, the y-axis represents the ℓ_2 loss

Our results did show that the optimization dynamics were independent of intermediate layer width, as the time evolution equation predicts. Shallow networks still performed best, followed closely by 3-layer networks. Although [1] did not prove a depth-dependent acceleration for the ℓ_2 loss, it is likely that deeper networks would begin to surpass the shallow networks after a much larger number of epochs.

4 Concluding Remarks

The dynamics of deep gradient descent that [1] derived are only observed because gradient trajectories are not invariant to parameterization. This suggests that our work has connections to the Fisher information matrix and natural gradients [2].

While the acceleration effects of depth on optimization in linear networks are interesting, more work needs to be done to show that similar effects accelerate training in deep nonlinear networks, especially considering other known phenomena such as exploding and vanishing gradients.

Our empirical experiments should be followed up with experiments that run for more epochs, test different architectures, and use ℓ_p losses with $p > 2$. Future work should also test the effects of combining depth with momentum. Our initial tests suggest that deep networks with momentum converge faster than shallow nets with momentum and deep nets without momentum.

References

[1] Arora, S., Cohen, N., Hazan, H.. "On the Optimization of Deep Networks: Implicit Acceleration by Overparameterization" arXiv preprint arXiv:1802.06509v2 (2018).

[2] Martens, James. "New insights and perspectives on the natural gradient method." arXiv preprint arXiv:1412.1193 (2014).

[3] Haykin, Simon *Neural Networks: A Comprehensive Foundation, Volume 2* Prentice Hall. (1998)

[4] Poggio et al. "Why and When Deep - But Not Shallow - Networks Avoid the Curse of Dimensionality: a Review' arXiv preprint arXiv:1611.00740v5' (2017)