Logarithmic Gradient Descent Running time

(Given Minimal Over-Parameterization and crazy assumptions)

Jeff Edmonds *

September 18, 2023

Abstract

Allen-Zhu, Li, and Song [**?**] gave an amazing paper proving that gradient descent starting with random weights converges to an optimal solution in time $D^{\mathcal{O}(1)}$ as long as the machine has $D^{O(1)}$ parameters, where D is the amount of training data. But surely with a trillion data items, the number of iteration does not exceed $\sim\log D$. Previous paper[**?**] had the time depend on $2^{\#\text{ levels}}$. But surely with hundreds of levels, this dependence is not more than $\sim L$. The goal of our paper is to make sufficient reasonable assumptions to get these results to mirror those found in practice. The result obtained is that the number of iterations is $\mathcal{O}(L\log(D))$ as long as certain unrelated vectors remain not perpendicular, and some strange values remain constant.

1 Introduction

In practice, it's often observed that with a sufficient amount of data and parameters, systems like ChatGPT and self-driving cars can be developed. However, as noted by Tsotsos [**?**], a lingering concern remains: practical experience has yet to grasp when, why, and how the process might fail. We propose a worldview in which the number of gradient descent steps to get the optimal solution is $\mathcal{O}(L\log(D))$ where D is the amount of training data and L is the number of layers in the neural network. Note that it does not depend on the width M of the machine. The result does require overfitting and a few other reasonable assumptions.

2 Historical Perspective

A longstanding debate within the realm of machine learning centers on the dilemma of whether the number of parameters, denoted as M', should surpass or fall short of the size of the training data set, represented as D. The former is essential to ensure the existence of weight configurations $\vec{w}$ that produce optimal computations on the training data, while the latter, as demonstrated by Avrim Blum and Ronald Rivest [**?**, **?**] renders the task of finding such weights NP-complete. (See Section **??** for more). The former approach carries the risk of over-fitting, whereas the latter safeguards the machine's capacity to generalize to unseen data (a concept stemming from PAC learning). In contrast, a *smooth* machine exhibits enhanced generalization capabilities. The fundamental idea is that when a new input $\vec{x}$ lies between training inputs $\vec{x}_d$ and $\vec{x}_{d'}$, its output is likely to fall within the range defined by their respective outputs. Bubeck and Sellke [**?**] established that a machine attains smoothness only when the parameter count M' satisfies $M' \geq \Omega(ND)$. Additionally, practical observations substantiate the idea that having an increased number of weights enhances the maneuverability of gradient descent within this higher-dimensional search space, enabling it to navigate obstacles more effectively.

*York University, jeff@cse.yorku.ca, supported by The Natural Sciences and Engineering Research Council of Canada (NSERC)

In 2018, Allen-Zhu, Li, and Song [**?**] gave an amazing paper proving that gradient descent starting with random weights $\vec{w}_0$ converges to an optimal solution in $D^{\mathcal{O}(1)}$ time as long as the machine has $D^{O(1)}$ parameters. Their technique proves that as long as the weights remain within a small ball around $\vec{w}_0$, each step of gradient descent makes a multiplicative decrease in the error until an optimal solution has been found.

A paper by Hui Jiang [**?**] proves that when gradient descent finds critical weights, i.e., a point $\vec{w}$ on the error surface with zero slope, the resulting neural network $NN_{\vec{w}}(x)$ provides accurate responses for each training data point. Edmonds [**?**] decreases the requirement from needing $(1/\epsilon)^N$ parameters to needing the machine to possess at least one hidden layer with a node count M equal to or greater than the number of training data points, denoted as D. Furthermore, they establish a more quantifiable link between the criticality of $\vec{w}$ and the accuracy of the machine's approximation to the supervisor's responses. The number of weights needed to achieve this result is nearly optimal because reducing the number of weights further would render the weight optimization problem NP-complete.

Having more parameters to learn clearly takes more computation time for each step of gradient descent. However, [**?**] proves optimizing the weight is NP-complete with lots of local minimum when the neural network width is less than the amount of training data, while our second result proves that the number of iterations depends on the number of layers L but not on the width M of the machine and only $\mathcal{O}(\log D)$ on the amount D of training data.

3 Statement of Result

Theorem 1 *(Time) When the assumptions are met for the entire duration of the gradient descent algorithm, the number of iterations is* $t = \theta(1+c_{aaah}L)\log(D/\epsilon)$*, where* D *is the amount of training data,* L *is the number of layers in the neural network, and* c_{aaah} *is a strange value we will assume is* $\theta(1)$*. The required assumptions are as follows:*

1. **Error Function:** The error $Error(\vec{w})$ is assumed to change from $\Theta(D)$ to ϵ.
   Here $Error(\vec{w}) = \sum_{d\in[D]} \left(NN_{\vec{w}}(\vec{x}_d) - y^*_d\right)^2 = \sum_{d\in[D]} \Theta(1) = \Theta(D)$.
   We are assuming that the supervisor's answer y^*_d has been normalized to $\mathcal{O}(1)$ and the networks output $NN_{\vec{w}}(\vec{x}_d)$ is kept to $\mathcal{O}(1)$.
2. **Activation Function:** Though the proof uses sigmoid, the only property of it that we use is that $\frac{\delta sigmoid(z)^2}{\delta z^2} / \frac{\delta sigmoid(z)}{\delta z} = 1-2y \in [-1,1]$. If the activation function where linear, then $\frac{\delta activation(z)^2}{\delta z^2} = 0$ and the $c_{aaah}L$ part of the time would disappear. One might conclude that the same holds for a piece-wise linear function like ReLU. However, it does not because the Taylor approximations breaks down as soon as one of the activation functions crosses such a transition.
3. **$\mathcal{O}(1)$ Step Size:** In order to avoid second-order curvature terms, the step size is set as follows. In order to avoid higher-order curvature terms, the step size γ_t cannot exceed one.

$$\gamma_t \quad = \quad Min\left(\left[\theta(1+c_{aaah}L)\cdot\sum_d\sum_m\left(\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta\vec{w}_m}\right)^2\right]^{-1}, 1\right)$$

   Note that this value changes with each time step. We generally simplify all the notation by dropping the t subscripts.

The problem with decreasing γ_t to one is that this decreases the running time by the same factor. To avoid this, we add the assumption that the original defined amount is $\mathcal{O}(1)$. Lemma 2 helps to justify this by proving it is $\theta\left(\frac{1}{c_x c_z D L^3}\right)$ where we justify c_x and c_z are $\theta(1)$ as follows.

- Here $c_x = Avg_{d,\ell}\ |\vec{x}_{\langle d,\ell\rangle}|^2 = Avg_{d,\ell}\ \sum_k \left(x_{\langle d,\ell,k\rangle}\right)^2$.
  In practice, machine learners do not want the values across a layer to blow up or decay layer by layer. We expect this to be $\Theta(1)$.
- Here $c_z = Avg_{d,\ell}\ \sum_j \left(\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}}\right)^2$.
  This is similar to the sensitivity requirement in Theorem **??**. We expect this to be $\Theta(1)$, because if the pre-activation values $z_{\langle d,\ell,j\rangle}$ of all M_ℓ nodes at level ℓ change by ϵ, we likely want the output $NN_{\vec{w}}(\vec{x}_d)$ to change by about $\Theta(\epsilon)$.

4. **Non-Perpendicular:** Define a D-dimensional vector space with a dimension for each training input $\vec{x}_d$. Let $\vec{Loss}$ denote the vector $\langle NN_{\vec{w}}(\vec{x}_d) - y^*_d \mid d \in [D]\rangle$ and for each weight $\vec{w}_m$, let $\vec{Dir}_m$ denote the vector $\left\langle \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_m} \middle|\ d \in [D] \right\rangle$. These vectors $\vec{Loss}$ and $\vec{Dir}_m$ are not coordinated in that a little noise added to the supervisor's output $\langle y^*_d\rangle$ weights changes the first, and a little noise added to the weights $\vec{w}$ changes the second. We require that angle α_m between them is not 90^{O} so that $cos(\alpha_m) \geq \Omega(1)$.
   Unlike Theorem **??** proving that there are no local minimum, this result does not "need" the width M to be big. However, having more nodes M, means more vectors $\vec{Dir}_m = \left\langle \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_m} \middle|\ d \in [D] \right\rangle$ so "more" of them are going to be not perpendicular to $\vec{Loss}$. Consider the M edges from the M nodes at level $\ell' - 1$ to a fixed node $z_{\langle d,\ell',j\rangle}$. The matrix $\left\langle \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta w_{\langle \ell',k,j\rangle}} \middle|\ k \in [M], d \in [D] \right\rangle$ is equal to the matrix $\left\langle \frac{\delta x_{\langle d,\ell',k\rangle}}{|} \ k \in [M], d \in [D] \right\rangle$ times the vector $\left\langle \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell',j\rangle}} \middle|\ d \in [D] \right\rangle$. Lemma **??** proves that the second matrix is has rank D so so does the first matrix. However, linear independence does not help that much because if the columns are all perpendicular to each other, then all but one could be perpendicular to $\vec{Loss}$. In this case, the bound on the running time increases by a factor of D.
5. **Negations in Sum:** The proof will want to factor a funny value $\left(1-2y_{\langle d,\ell,j\rangle}\right) w_{\langle \ell,k,j\rangle}$ out of a funny sum. We define $\left(1-2y_{\langle d,\ell,?\rangle}\right) w_{\langle \ell,k,?\rangle}$ to be the value to make it work, namely so that

$$\sum_{j\in[M_\ell]} \left(1-2y_{\langle d,\ell,j\rangle}\right) \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}} w_{\langle \ell,k,j\rangle}{}^2 \leq \left(1-2y_{\langle d,\ell,?\rangle}\right) w_{\langle \ell,k,?\rangle} \cdot \sum_{j\in[M_\ell]} \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}} w_{\langle \ell,k,j\rangle}.$$

   If all of these numbers were positive, we would simply factor out the maximum of these values. Having different signs, however, makes it awkward. The sum we want to be small could not cancel at all, while the one we want to be large could cancel to zero. In order to handle this, we assume that we are able to factor out $\mathcal{O}(1)$ of their average value. The justification is that if in both sums the sign of these M_ℓ terms were chosen randomly, then in both sums, all but $\theta(\sqrt{M_\ell})$ of the terms are expected to cancel.
6. **Mid-Sigmoid:** We define c_{aaah} so that $\forall\, \langle t,d,\ell,k\rangle$, (or at least on "average") $\left|w_{\langle \ell,k,?\rangle}\right| \leq c_{aaah} \left|\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta x_{\langle d,\ell,k\rangle}}\right|$. We will assume that this $c_{aaah} \leq \mathcal{O}(1)$. (In the event that $c_{aaah} < o(1)$, the theorem is stated as $\theta(1+c_{aaah}L)$.) The justification is as follows.

Notation: Recall $w_{\langle \ell,k,j\rangle}$ represents the weight from layer ℓ's k^{th} input node with post-activation value $x_{\langle d,\ell,k\rangle}$ to its j^{th} output node with pre-activation value $z_{\langle d,\ell,j\rangle}$.

Proportional: The equation is $\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta x_{\langle d,\ell,k\rangle}} = \sum_j \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}} w_{\langle \ell,k,j\rangle}$, which is proportional to scaling all the $w_{\langle \ell,k,j\rangle}$ by the same factor.

Simple Case: For better understanding assume that node $x_{\langle d,\ell,k\rangle}$ feeds into a single sigmoid that is the output $NN_{\vec{w}}(\vec{x}_d)$.

$$\begin{aligned}\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta x_{\langle d,\ell,k\rangle}} &= \sum_j \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}} w_{\langle \ell,k,j\rangle} = \frac{\delta sig(z_{\langle d,L\rangle})}{\delta z_{\langle d,L\rangle}} w_{\langle L,k,?\rangle} = sig(z_{\langle d,L\rangle})(1-sig(z_{\langle d,L\rangle}))w_{\langle Y,k,?\rangle} \\ c_{aaah} &= \left| w_{\langle \ell,k,?\rangle} / \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta x_{\langle d,\ell,k\rangle}} \right| = \frac{1}{sig(z_{\langle d,L\rangle})(1-sig(z_{\langle d,L\rangle}))}\end{aligned}$$

It is not surprising that the running time increases inverse proportionally to how small the sigmoid outputs get. Theorem **??** assumes that these are all $\theta(1)$.

Non-Blowup: In practice, machine learners do not want the values across a layer to blow up or decay layer by layer.

$\boldsymbol{w_{\langle \ell,k,j\rangle}}$**:** The relation is $z_{\langle d,\ell,j\rangle} = \sum_{k\in[M_{\ell-1}]} w_{\langle \ell,k,j\rangle} \times x_{\langle d,\ell,k\rangle}$. Rearranged "gives" $|w_{\langle \ell,k,j\rangle}|$ needs to be $\frac{|z_{\langle d,\ell,j\rangle}|}{\left|\sum_{k\in[M_{\ell-1}]} ??\right| \times |x_{\langle d,\ell,k\rangle}|}$. Being the input to the sigmoid, we want $\left|z_{\langle d,\ell,j\rangle}\right| \in \theta(1)$. Because $\sum_{k\in[M_{\ell-1}]} \left(x_{\langle d,\ell,k\rangle}\right)^2$ is expected to be $\theta(1)$, $\left|x_{\langle d,\ell,k\rangle}\right|$ is expected to be $\theta\left(1/\sqrt{M_{\ell-1}}\right)$. If the sign of these $M_{\ell-1}$ terms were chosen randomly, then all but $\theta\left(\sqrt{M_{\ell-1}}\right)$ of the terms are expected to cancel. Together this gives us that $|w_{\langle \ell,k,j\rangle}|$ needs to be $\frac{\theta(1)}{\theta\left(\sqrt{M_{\ell-1}}\right)\times\theta\left(1/\sqrt{M_{\ell-1}}\right)} = \theta(1)$.

$\boldsymbol{\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta x_{\langle d,\ell,k\rangle}}}$**:** The relation is $\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta ??} = \sum_{k\in[M_{\ell-1}]} \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta x_{\langle d,\ell,k\rangle}} \times \frac{\delta x_{\langle d,\ell,k\rangle}}{\delta ??}$. Rearranged "gives" $\left|\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta x_{\langle d,\ell,k\rangle}}\right|$ needs to be $\frac{\left|\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta ??}\right|}{\left|\sum_{k\in[M_{\ell-1}]} ??\right| \times \left|\frac{\delta x_{\langle d,\ell,k\rangle}}{\delta ??}\right|}$. Because $|NN_{\vec{w}}(\vec{x}_d)|$ is expected to be $\theta(1)$, we expect ϵ events to change it by $\theta(\epsilon \times 1)$. This gives that $\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta ??}$ needs to be $\theta(1)$. Because $\left|x_{\langle d,\ell,k\rangle}\right|$ is expected to be $\theta\left(1/\sqrt{M_{\ell-1}}\right)$, we expect our ϵ event to change it by $\theta\left(\epsilon \times 1/\sqrt{M_{\ell-1}}\right)$. This gives that $\left|\frac{\delta x_{\langle d,\ell,k\rangle}}{\delta ??}\right|$ needs to be $\theta\left(1/\sqrt{M_{\ell-1}}\right)$. Again if the sign of the $M_{\ell-1}$ terms in a sum were chosen randomly, then all but $\theta\left(\sqrt{M_{\ell-1}}\right)$ of the terms are expected to cancel. Together this gives $\left|\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta x_{\langle d,\ell,k\rangle}}\right|$ needs to be $\frac{\theta(1)}{\theta\left(\sqrt{M_{\ell-1}}\right)\times\theta\left(1/\sqrt{M_{\ell-1}}\right)} = \theta(1)$.

$\boldsymbol{c_{aaah}}$**:** This gives $c_{aaah} = \left|w_{\langle \ell,k,?\rangle} / \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta x_{\langle d,\ell,k\rangle}}\right| = \theta(1/1)$.

3.1 Proof Idea

The Error function is quadratic when it is defined as $Error(\vec{w}) = \sum_d (NN_{\vec{w}}(\vec{x}_d) - y^*_d)^2$ and the neural network has only linear activation functions. On this, gradient descent solves the problem in logarithmic time. Given current values w_t, error $Error(w_t) = (w_t - w_*)^2$, derivative $Error'(w) = 2(w_t - w_*)$, update $w_{t+1} = w_t - \gamma Error'(w)$, each iteration decreases the error amount $|w_t - w_*|$ and the error measure $(w_t - w_*)^2$ by a constant factor, which leads to logarithmic time. What remains

is to take a step size small enough that the non-linearity of the neural network has an insignificant effect.

The effect of the non-linearity is characterized by its second derivatives. This paper has a few lemmas that approximate these second derivatives by converting them into the product of first derivatives. Lemma 5 for example wants to prove $\frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta w_{\langle \ell,k,j\rangle}\,\delta w_{\langle \ell',k',j'\rangle}} \leq \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta w_{\langle \ell,k,j\rangle}} \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta w_{\langle \ell',k',j'\rangle}}$. This is proved by induction on the number of sigmoids L–ℓ between the node/edge layer ℓ being considered and the output layer L. This is easy when differentiating with respect to different things, i.e. $w_{\langle \ell,k,j\rangle}$ vs $w_{\langle \ell',k',j'\rangle}$ or when what is being passed through is linear. Lemma 8 handles the case when neither is true, namely moving through the non-linear sigmoid activation function on the same node twice. The relation being "passed through" is $y_{\langle d,\ell,j\rangle} = sigmoid(z_{\langle d,\ell,j\rangle})$. The lemma proves $\left[\frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta y_{\langle d,\ell,j\rangle}{}^2} \leq c\left(\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta y_{\langle d,\ell,j\rangle}}\right)^2\right] \Rightarrow \left[\frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta z_{\langle d,\ell,j\rangle}{}^2} \leq c\left(\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}}\right)^2 + \left(1-2y_{\langle d,\ell,j\rangle}\right)\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}}\right]$. The second term is the ugly one leading to c_{aaah}. The induction collects one such c_{aaah} for each sigmoid it passes through. The c on the first term is how many times it has been collected so far. Not having the c on the second ugly term means that it is collected additively $c_{aaah}+c_{aaah}+\ldots+c_{aaah} = c_{aaah}\times L$ instead of multiplicatively $c_{aaah}\times c_{aaah}\times\ldots\times c_{aaah} = c_{aaah}^L$.

3.2 Linear Approximation

Proof: We begin the proof by considering what occurs in each step of gradient descent.

$$\begin{aligned}
\Delta\vec{w} &= \gamma\Delta Error(\vec{w}) \;=\; \gamma\left\langle \tfrac{\delta Error}{\delta\vec{w}_1},\ldots,\tfrac{\delta E}{\delta\vec{w}_{M'}}\right\rangle \\
Error(\vec{w}+\Delta\vec{w}) & \quad \text{(by Taylor approximation)} \\
&= Error(\vec{w}) + \textstyle\sum_m\left[\frac{\delta Error}{\delta\vec{w}_m}\cdot\Delta\vec{w}_m\right] + \text{curvature } \Theta(\Delta\vec{w}^2) \text{ terms} \\
&= Error(\vec{w}) + \textstyle\sum_m\left[\frac{\delta Error}{\delta\vec{w}_m}\cdot\left(-\gamma\frac{\delta Error}{\delta\vec{w}_m}\right)\right] + \Theta(\gamma^2) \\
&= Error(\vec{w}) - \gamma\textstyle\sum_m\left(\frac{\delta Error}{\delta\vec{w}_m}\right)^2 + \Theta(\gamma^2) \\
&= Error(\vec{w}) - \Omega((1+c_{aaah}L)^{-1})\; Error(\vec{w}) \quad \text{(by Lemmas 1 and 3)} \\
&= \left(1-\Omega((1+c_{aaah}L)^{-1})\right) Error(\vec{w}) \;=\; e^{-\Omega((1+c_{aaah}L)^{-1}t)} Error(\vec{w}_0)
\end{aligned}$$

Hence, the Error reaches ϵ after $t \leq \theta(1+c_{aaah}L)\log(D/\epsilon)$. ■

Lemma 1 $\sum_m\left(\frac{\delta Error}{\delta\vec{w}_m}\right)^2 \geq \Omega\left(\frac{Error(\vec{w})}{(1+c_{aaah}L)\gamma}\right)$.

Proof: Recall the vectors $\vec{Loss} = \langle NN_{\vec{w}}(\vec{x}_d) - y_d^* \mid d\in[D]\rangle$ and $\forall m\in[M']$, $\vec{Dir}_m = \left\langle \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta\vec{w}_m} \mid d\in[D]\right\rangle$.

By assumption 4 of Theorem 1, the angle α_m between them is not 90^{O} so that $cos(\alpha_m) \geq \Omega(1)$.

$$\begin{aligned}
Error(\vec{w}) &= \textstyle\sum_d \left(NN_{\vec{w}}(\vec{x}_d) - y^*_d\right)^2 \\
\tfrac{\delta Error(\vec{w})}{\delta \vec{w}_m} &= 2\textstyle\sum_d \left(NN_{\vec{w}}(\vec{x}_d) - y^*_d\right) \cdot \tfrac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_m} \;=\; 2\ \vec{Loss} \cdot \vec{Dir}_m \;=\; 2\ cos(\alpha_m)\ |\vec{Loss}|\ |\vec{Dir}_m| \\
&= \Omega(1) \cdot |\vec{Loss}| \cdot |\vec{Dir}_m| \\
\textstyle\sum_m \left(\tfrac{\delta Error(\vec{w})}{\delta \vec{w}_m}\right)^2 &= \textstyle\sum_m \Omega(1) \cdot |\vec{Loss}|^2 \cdot |\vec{Dir}_m|^2 \\
&= \Omega(1) \cdot \left[\textstyle\sum_d \left(NN_{\vec{w}}(\vec{x}_d) - y^*_d\right)^2\right] \cdot \left[\textstyle\sum_m \sum_d \left(\tfrac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_m}\right)^2\right] \\
&= \Omega(1) \cdot [Error(\vec{w})] \cdot [\theta(1{+}c_{aaah}L) \cdot \gamma]^{-1} \text{ (By the definitions of } Error \text{ and } \gamma.)
\end{aligned}$$

■

This is not why the step-size γ is set to $\left[\theta(1{+}c_{aaah}L) \cdot \sum_d \sum_m \left(\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_m}\right)^2\right]^{-1}$, but it sure is convenient.

Lemma 2 $[\theta(1{+}c_{aaah}L) \cdot \gamma]^{-1} = \sum_m |\vec{Dir}_m|^2 \;=\; \sum_m \sum_d \left(\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_m}\right)^2 = c_x c_z DL.$

Proof: We begin by considering the influence of weight $w_{\langle \ell,k,j\rangle} \in \mathbb{R}$.

$$\begin{aligned}
z_{\langle d,\ell,j\rangle} &= \textstyle\sum_k w_{\langle \ell,k,j\rangle} \times x_{\langle d,\ell,k\rangle} \\
\tfrac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta w_{\langle \ell,k,j\rangle}} &= \tfrac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}} \cdot \tfrac{\delta z_{\langle d,\ell,j\rangle}}{\delta w_{\langle \ell,k,j\rangle}} \;=\; \tfrac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}} \cdot x_{\langle d,\ell,k\rangle} \\
\textstyle\sum_{d,\ell,k,j} \left(\tfrac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta w_{\langle \ell,k,j\rangle}}\right)^2 &= \textstyle\sum_{d,\ell} \sum_j \sum_k \left(\tfrac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}}\right)^2 \cdot \left(x_{\langle d,\ell,k\rangle}\right)^2 \;=\; \textstyle\sum_{d,\ell} \left[\sum_j \left(\tfrac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}}\right)^2\right] \cdot \left[\sum_k \left(x_{\langle d,\ell,k\rangle}\right)^2\right] \\
&= c_z c_x DL
\end{aligned}$$

■

3.3 Step Size γ and Second Derivatives

Lemma 3 *Setting the step size to* $\gamma = \left[\theta(1{+}c_{aaah}L) \cdot \sum_d \sum_m \left(\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_m}\right)^2\right]^{-1}$ *is small enough to avoid second-order curvature terms.*

Proof: The Taylor expansion is $Error(\vec{w}{+}\gamma\Delta Error) = Error(\vec{w}) - E'\gamma + \frac{1}{2}E''\gamma^2 + \frac{1}{3!}E'''\gamma^3 + \ldots$. We avoid the second order term, i.e. $E''\gamma^2 \leq E'\gamma$, by setting $\gamma = \frac{E'}{E''}$. This in turn ensures the higher order terms decrease geometrically, at least in the cases when $Error(\vec{w})$ is w^c and e^{cw}. The problem cases are cw, $sigmoid(0)$, and $sin(0)$ because $E'' = 0$, giving $\gamma = \infty$. We avoided this by

assuming that γ is $\mathcal{O}(1)$ and rounding down to one.

$$\begin{aligned}
E'\gamma &= \sum_m \left(\frac{\delta Error}{\delta \vec{w}_m}\right)^2 \gamma \text{ (Theorem 1)} \\
E''\gamma^2 &= \sum_m \sum_{m'} \left[\frac{\delta Error^2}{\delta \vec{w}_m\ \delta \vec{w}_{m'}} \cdot \Delta\vec{w}_m \cdot \Delta\vec{w}_{m'}\right] \\
&\leq \sum_m \sum_{m'} \left[\left[\theta(1+c_{aaah}L) \cdot \sum_d \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_m} \cdot \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_{m'}}\right] \cdot \left(-\gamma \frac{\delta Error}{\delta \vec{w}_m}\right) \cdot \left(-\gamma \frac{\delta Error}{\delta \vec{w}_{m'}}\right)\right] \\
&\qquad \text{(by Lemma 4)} \qquad\qquad \text{(Gradient Descent Step)} \\
&= \theta(1+c_{aaah}L) \cdot \sum_d \left[\sum_m \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_m} \cdot \frac{\delta Error}{\delta \vec{w}_m}\right]^2 \gamma^2 \text{ (Think of this as the dot product of two vectors.)} \\
&= \theta(1+c_{aaah}L) \cdot \sum_d \left[\left\langle \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_m} | m \right\rangle \cdot \left\langle \frac{\delta Error}{\delta \vec{w}_m} | m \right\rangle\right]^2 \gamma^2 \\
&= \theta(1+c_{aaah}L) \cdot \sum_d \left[cos(\alpha) \cdot \left|\left\langle \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_m} | m \right\rangle\right| \cdot \left|\left\langle \frac{\delta Error}{\delta \vec{w}_m} | m \right\rangle\right|\right]^2 \gamma^2 \\
&\leq c_{aaah}L \cdot \left[\sum_d \sum_m \left(\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_m}\right)^2\right] \cdot \left[\sum_m \left(\frac{\delta Error}{\delta \vec{w}_m}\right)^2\right] \gamma^2 \\
\gamma = \frac{E'}{E''} &= \left[\theta(1+c_{aaah}L) \cdot \sum_d \sum_m \left(\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_m}\right)^2\right]^{-1}
\end{aligned}$$

■

Lemma 4 $\frac{\delta Error(\vec{w})^2}{\delta \vec{w}_m\ \delta \vec{w}_{m'}} \leq \theta(1+c_{aaah}L) \cdot \sum_d \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_m} \cdot \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_{m'}}$.

Proof:

$$\begin{aligned}
\frac{\delta Error(\vec{w})}{\delta \vec{w}_m} &= 2 \sum_d (NN_{\vec{w}}(\vec{x}_d) - y^*_d) \cdot \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_m} \\
\frac{\delta Error(\vec{w})^2}{\delta \vec{w}_m\ \delta \vec{w}_{m'}} &= 2 \sum_d \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_{m'}} \cdot \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_m} \\
&+ 2 \sum_d (NN_{\vec{w}}(\vec{x}_d) - y^*_d) \cdot \frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta \vec{w}_m\ \delta \vec{w}_{m'}}
\end{aligned}$$

$|NN_{\vec{w}}(\vec{x}_d) - y^*_d|$ is assumed to be $\mathcal{O}(1)$. Lemmas 5 proves that the second term is at most $c_{aaah}L$ times the first. ■ ■

Lemma 5 *Whether or not ($x_{\langle d,\ell,k\rangle}$ and $x_{\langle d,\ell' \leq \ell,k'\rangle}$) and ($w_{\langle \ell,k,j\rangle}$ and $w_{\langle \ell' \leq \ell,k',j'\rangle}$) represent different or the same node/edge:*

$$\frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta x_{\langle d,\ell,k\rangle}\ \delta x_{\langle d,\ell',k'\rangle}} \leq c_{aaah}(L-\ell) \cdot \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta x_{\langle d,\ell,k\rangle}} \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta x_{\langle d,\ell',k'\rangle}} \tag{1}$$

$$\frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta w_{\langle \ell,k,j\rangle}\ \delta w_{\langle \ell',k',j'\rangle}} \leq c_{aaah}(L-\ell) \cdot \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta w_{\langle \ell,k,j\rangle}} \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta w_{\langle \ell',k',j'\rangle}} \tag{2}$$

Proof of Lemma 5.1: This is proved by induction on the number of sigmoids $L-\ell$ between the node/edge layer ℓ being considered and the output layer L. When the two nodes/edge are in different layers, i.e., $\ell > \ell'$, the proof moves the second one up increasing ℓ' until they are in the same layer. Being about the same, the proof here focuses on the case in which they are in the same layer, i.e., $\ell = \ell'$. The hardest case is when the two nodes are the same.

The base case occurs at the output $NN_{\vec{w}}(\vec{x}_d)$ which is the output $y_{\langle d,L-1,j\rangle}$ to the $L-1^{st}$ layer. If there were one, this would also be the input $x_{\langle d,L,k\rangle}$ to the L^{th} layer. The number of sigmoids

between this $x_{\langle d,L,k\rangle}$ and the output is $L-L=0$. Here, the statement is true because $0 = \frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta x_{\langle d,L,k\rangle}{}^2} = c_{aaah}(L-L)\left(\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta x_{\langle d,L,k\rangle}}\right)^2 = 0\times 1$.

The induction step proves $\frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta x_{\langle d,\ell,k\rangle}\ \delta x_{\langle d,\ell',k'\rangle}} \leq c_{aaah}(L-\ell)\cdot \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta x_{\langle d,\ell,k\rangle}}\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta x_{\langle d,\ell',k'\rangle}}$ with $L-\ell$ sigmoids after it. By way of induction, assume the same statement is true for each $x_{\langle d,\ell+1,j\rangle}$ with $L-\ell-1$ sigmoids after it. Recall that $x_{\langle d,\ell+1,j\rangle}$ is different notation for $y_{\langle d,\ell,j\rangle}$ shifted one layer up. This gives us the precondition for Lemmas 6, 7, and 8 with $c=L-\ell-1$. We continue the proof as follows.

$$
\begin{array}{rcl}
\forall j, z_{\langle d,\ell,j\rangle} & = & \sum_k w_{\langle \ell,k,j\rangle}\times x_{\langle d,\ell,k\rangle} \\
\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta x_{\langle d,\ell,k\rangle}} & = & \sum_j \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}}\frac{\delta z_{\langle d,\ell,j\rangle}}{\delta x_{\langle d,\ell,k\rangle}} \ = \ \sum_j \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}} w_{\langle \ell,k,j\rangle} \\
\frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta x_{\langle d,\ell,k\rangle}\ \delta x_{\langle d,\ell,k'\rangle}} & = & \sum_j \frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta z_{\langle d,\ell,j\rangle}\ \delta x_{\langle d,\ell,k'\rangle}} w_{\langle \ell,k,j\rangle} \ = \ \sum_j \left[\sum_{j'} \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}\ \delta z_{\langle d,\ell,j'\rangle}} w_{\langle \ell,k',j'\rangle}\right] w_{\langle \ell,k,j\rangle} \\
 & & \text{(There is an extra term when the two nodes } \langle \ell,j\rangle = \langle \ell,j'\rangle \text{ are the same.)} \\
 & \leq & \sum_j \sum_{j'} c_{aaah}(L-\ell-1)\cdot \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}}\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j'\rangle}} w_{\langle \ell,k,j\rangle} w_{\langle \ell,k,j'\rangle} \text{ (by Lemma 7)} \\
 & & + \ \sum_{j\in[M_\ell]} \left(1-2y_{\langle d,\ell,j\rangle}\right)\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}} w_{\langle \ell,k,j\rangle}{}^2 \text{ (by Lemma 8)} \\
1^{st} \text{ term} & = & c_{aaah}(L-\ell-1)\cdot \left(\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta x_{\langle d,\ell,k\rangle}}\right)^2 \\
2^{nd} \text{ term} & & \text{By assumption 5 of Theorem 1, we can pretend that the } \left(1-2y_{\langle d,\ell,j\rangle}\right) w_{\langle \ell,k,j\rangle} \\
 & & \text{can be factored out.} \\
 & = & \left(1-2y_{\langle d,\ell,?\rangle}\right) w_{\langle \ell,k,?\rangle}\cdot \sum_{j\in[M_\ell]} \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}} w_{\langle \ell,k,j\rangle} \ = \ \left(1-2y_{\langle d,\ell,?\rangle}\right) w_{\langle \ell,k,?\rangle}\cdot \left(\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta x_{\langle d,\ell,k\rangle}}\right) \\
 & & \text{Here } y_{\langle d,\ell,?\rangle} \text{ is the output of the sigmoid activation function and hence } 1-2y_{\langle d,\ell,?\rangle}\in[-1..1]. \\
 & & \text{By assumption 6 of Theorem 1, } \left|w_{\langle \ell,k,?\rangle}\right| \leq c_{aaah}\left|\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta x_{\langle d,\ell,k\rangle}}\right| \\
 & \leq & c_{aaah}\left(\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta x_{\langle d,\ell,k\rangle}}\right)^2
\end{array}
$$

The number of $\left(\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta x_{\langle d,\ell,k\rangle}}\right)^2$ introduced by the 1^{st} term is $c_{aaah}(L-\ell-1)$ and the number introduced by the 2^{nd} is c_{aaah} for a total of $c_{aaah}(L-\ell)$. ■

Lemma 6 *Whether or not $w_{\langle \ell,k,j\rangle}$ and $w_{\langle \ell',k',j'\rangle}$ are the same weight:*
$\left[\frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta z_{\langle d,\ell,j\rangle}\ \delta z_{\langle d,\ell',j'\rangle}} = c\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}}\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell',j'\rangle}}\right] \Rightarrow \left[\frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta w_{\langle \ell,k,j\rangle}\ \delta w_{\langle \ell',k',j'\rangle}} = c\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta w_{\langle \ell,k,j\rangle}}\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta w_{\langle \ell',k',j'\rangle}}\right]$

Proof of Lemma 5.2 and 6:

$$
\begin{array}{rcl}
z_{\langle d,\ell,j\rangle} & = & \sum_k w_{\langle \ell,k,j\rangle}\times x_{\langle d,\ell,k\rangle} \\
\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta w_{\langle \ell,k,j\rangle}} & = & \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}}\frac{\delta z_{\langle d,\ell,j\rangle}}{\delta w_{\langle \ell,k,j\rangle}} \ = \ \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}} x_{\langle d,\ell,k\rangle} \\
\frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta w_{\langle \ell,k,j\rangle}\ \delta w_{\langle \ell',k',j'\rangle}} & = & \frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta z_{\langle d,\ell,j\rangle}\ \delta z_{\langle d,\ell',j'\rangle}}\frac{\delta z_{\langle d,\ell',j'\rangle}}{\delta w_{\langle \ell',k',j'\rangle}} x_{\langle d,\ell,k\rangle} \ = \ \frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta z_{\langle d,\ell,j\rangle}\ \delta z_{\langle d,\ell',j'\rangle}} x_{\langle d,\ell,k\rangle} x_{\langle d,\ell',k'\rangle} \\
 & \leq & c\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}}\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell',j'\rangle}} x_{\langle d,\ell,k\rangle} x_{\langle d,\ell',k'\rangle} \text{ (by assumption)} \\
 & = & c\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta w_{\langle \ell,k,j\rangle}}\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta w_{\langle \ell',k',j'\rangle}}
\end{array}
$$

■

Lemma 7 *If we are considering two different nodes, i.e.,* $\langle \ell, j\rangle \neq \langle \ell', j'\rangle$,
then $\left[\frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta y_{\langle d,\ell,j\rangle}\ \delta y_{\langle d,\ell',j'\rangle}} = c\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta y_{\langle d,\ell,j\rangle}}\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta y_{\langle d,\ell',j'\rangle}}\right] \Rightarrow \left[\frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta z_{\langle d,\ell,j\rangle}\ \delta z_{\langle d,\ell',j'\rangle}} = c\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}}\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell',j'\rangle}}\right]$

Proof:

$$
\begin{aligned}
y_{\langle d,\ell,j\rangle} &= sigmoid(z_{\langle d,\ell,j\rangle}) \\
\\
\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}} &= \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta y_{\langle d,\ell,j\rangle}}\frac{\delta y_{\langle d,\ell,j\rangle}}{\delta z_{\langle d,\ell,j\rangle}} = \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta y_{\langle d,\ell,j\rangle}}\frac{\delta sig(z_{\langle d,\ell,j\rangle})}{\delta z_{\langle d,\ell,j\rangle}} \\
\frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta z_{\langle d,\ell,j\rangle}\ \delta z_{\langle d,\ell',j'\rangle}} &= \frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta y_{\langle d,\ell,j\rangle}\ \delta y_{\langle d,\ell',j'\rangle}}\frac{\delta sig(z_{\langle d,\ell,j\rangle})}{\delta z_{\langle d,\ell,j\rangle}}\frac{\delta sig(z_{\langle d,\ell',j'\rangle})}{\delta z_{\langle d,\ell',j'\rangle}} \\
&\leq c\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta y_{\langle d,\ell,j\rangle}}\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta y_{\langle d,\ell',j'\rangle}}\frac{\delta sig(z_{\langle d,\ell,j\rangle})}{\delta z_{\langle d,\ell,j\rangle}}\frac{\delta sig(z_{\langle d,\ell',j'\rangle})}{\delta z_{\langle d,\ell',j'\rangle}} \text{ (by assumption)} \\
&\leq c\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}}\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell',j'\rangle}}
\end{aligned}
$$

■

Lemma 8 *If we are considering the same node twice, then*
$\left[\frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta y_{\langle d,\ell,j\rangle}{}^2} \leq c\left(\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta y_{\langle d,\ell,j\rangle}}\right)^2\right] \Rightarrow \left[\frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta z_{\langle d,\ell,j\rangle}{}^2} \leq c\left(\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}}\right)^2 + \left(1-2y_{\langle d,\ell,j\rangle}\right)\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}}\right]$

Proof:

$$
\begin{aligned}
y_{\langle d,\ell,j\rangle} &= sigmoid(z_{\langle d,\ell,j\rangle}) \\
y &= sig(z) = \tfrac{1}{1+e^{-z}} \text{ and } \tfrac{\delta y}{\delta z} = y(1-y) \\
\frac{\delta y^2}{\delta z^2} &= \frac{\delta y}{\delta z}(1-y) + \frac{\delta(1-y)}{\delta z} = [y(1-y)](1-y) - y[y(1-y)] = [1-2y][y(1-y)] \\
&\in [-1,1]\frac{\delta y}{\delta z} \\
\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}} &= \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta y_{\langle d,\ell,j\rangle}}\frac{\delta y_{\langle d,\ell,j\rangle}}{\delta z_{\langle d,\ell,j\rangle}} = \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta y_{\langle d,\ell,j\rangle}}\frac{\delta sig(z_{\langle d,\ell,j\rangle})}{\delta z_{\langle d,\ell,j\rangle}} \\
\frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta z^2_{\langle d,\ell,j\rangle}} &= \frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta y_{\langle d,\ell,j\rangle}}\frac{\delta sig(z_{\langle d,\ell,j\rangle})}{\delta z_{\langle d,\ell,j\rangle}} + \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta y_{\langle d,\ell,j\rangle}}\frac{\delta sig(z_{\langle d,\ell,j\rangle})^2}{\delta z_{\langle d,\ell,j\rangle}{}^2} \\
1^{st} \text{ term} &= \frac{\delta NN_{\vec{w}}(\vec{x}_d)^2}{\delta y^2_{\langle d,\ell,j\rangle}}\left(\frac{\delta sig(z_{\langle d,\ell,j\rangle})}{\delta z_{\langle d,\ell,j\rangle}}\right)^2 = c\left(\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta y_{\langle d,\ell,j\rangle}}\right)^2\left(\frac{\delta sig(z_{\langle d,\ell,j\rangle})}{\delta z_{\langle d,\ell,j\rangle}}\right)^2 = c\left(\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}}\right)^2 \\
2^{nd} \text{ term} &= \left[\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}}/\frac{\delta sig(z_{\langle d,\ell,j\rangle})}{\delta z_{\langle d,\ell,j\rangle}}\right]\left(\frac{\delta sig(z_{\langle d,\ell,j\rangle})}{\delta z_{\langle d,\ell,j\rangle}}\right)^2 = \left(1-2y_{\langle d,\ell,j\rangle}\right)\frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta z_{\langle d,\ell,j\rangle}}
\end{aligned}
$$

■

4 Mathematical Insight

We initiate our discussion with an intuitive overview of our approach. Our primary objective is to prove that only globally optimal critical points exist. When considering sigmoid activation functions, we make the assumption that they never reach zero or exhibit flat regions. Consequently, we contend that as the weights w_m undergo changes, the output of the neural network consistently varies. In practical terms, this implies that for any fixed input $\vec{x}$, the relationship between $NN_{\vec{w}}(x)$

and $\vec{w}$ does not encompass critical points. The error surface $Error(\vec{w}) = \sum_d (NN_{\vec{w}}(\vec{x}_d) - y_d^*)^2$ has local minimums only because of this imposed quadratic. We prove that all such critical points are optimal in that for each training input $\vec{x}_d$, the neural network produces an output $NN_{\vec{w}}(\vec{x}_d)$ equaling the supervisor's answer y_d^*.

Hui Jiang, in his work [**?**], effectively introduces the following framework: Consider a vector space of dimension D, where each dimension corresponds to a training input $\vec{x}_d$. Given the current set of weights $\vec{w}$, we can measure the discrepancy between the neural network's response, denoted as $NN_{\vec{w}}(\vec{x}_d)$, and the supervisor's provided answer y_d^* using the vector $\vec{Loss}$, defined as $\vec{Loss} = \langle NN_{\vec{w}}(\vec{x}_d) - y_d^* \,|\, d \in [D] \rangle$. For each weight $\vec{w}_m$, we define $\vec{Dir}_m$ as the vector of derivatives $\langle \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_m} \,|\, d \in [D] \rangle$. We can assemble a matrix **Dir** using these vectors, where each row corresponds to $\vec{Dir}_m$. This framework enables us to express the direction of descent, i.e., the vector of derivatives of the Error function with respect to each $\vec{w}_m$, as follows:

$$\begin{aligned} Error(\vec{w}) &= \textstyle\sum_d \left(NN_{\vec{w}}(\vec{x}_d) - y_d^*\right)^2 \\ \tfrac{\delta Error(\vec{w})}{\delta \vec{w}_m} &= 2 \textstyle\sum_d \left(NN_{\vec{w}}(\vec{x}_d) - y_d^*\right) \cdot \tfrac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_m} \;=\; 2\ \vec{Loss} \cdot \vec{Dir}_m \\ \nabla Error(\vec{w}) &= \left\langle \tfrac{\delta Error(\vec{w})}{\delta \vec{w}_1}, \ldots, \tfrac{\delta Error(\vec{w})}{\delta \vec{w}_{M'}} \right\rangle \;=\; 2\mathbf{Dir} \cdot \vec{Loss} \end{aligned}$$

If the neural network has more weights than data, $M' \geq D$, the matrix **Dir** has more rows than columns. Our task is to prove that it has full rank D. This then gives

$$\vec{Loss} \;=\; \tfrac{1}{2}\mathbf{Dir}^{-1} \cdot \nabla Error(\vec{w})$$

Having the weights $\vec{w}$ situated at a critical point implies that each of these Error derivatives is zero, namely $\nabla Error(\vec{w}) = \vec{0}$. This results in $\vec{Loss} = \langle NN_{\vec{w}}(\vec{x}_d) - y_d^* \mid d \in [D] \rangle = \vec{0}$, signifying that the neural network provides the optimal answers for all of the training data points.

Hui Jiang's work [**?**] establishes that his version of the matrix **Dir** indeed possesses a rank of D. Edmonds [**?**] decreases the requirement from needing $(1/\epsilon)^N$ parameters to needing the machine to possess at least one hidden layer with a node count M equal to or greater than the number of training data points, denoted as D.

Theorem 1 proves that the number of iterations is $\theta(1+c_{aaah}L)\log(D/\epsilon)$ as long as during the entire gradient descent, the angle α_m between $\vec{Loss}$ and $\vec{Dir}_m$ is not 90^{o} so that $cos(\alpha_m) \geq \Omega(1)$. This may be why in practice, noise is added to gradient descent. Perhaps naively, the authors argue that this is a reasonable assumption. These vectors are not coordinated in that a little noise added to the supervisor's output $\langle y_d^* \rangle$ changes $\vec{Loss} = \langle NN_{\vec{w}}(\vec{x}_d) - y_d^* \mid d \in [D] \rangle$ and a little noise added to the weights $\vec{w}$ changes $\vec{Dir}_m = \left\langle \frac{\delta NN_{\vec{w}}(\vec{x}_d)}{\delta \vec{w}_m} \middle| \ d \in [D] \right\rangle$.

If the gradient descent computation reached a point where all M' of the vectors $\vec{Dir}_m$ were perpendicular to $\vec{Loss}$, then the computation would stop in a local minimum. If a $\frac{r}{M'}$ fraction ("randomly" chosen) of them are non-perpendicular, the rate of progress would decrease by a factor of $\frac{r}{M'}$. However, such progress would move the computation past such an unfortunate place.

Unlike Theorem **??**, Theorem 1, does not "need" the width M to be big. However, having more nodes M means having more vectors $\vec{Dir}_m$, and hence "more" of them will not be perpendicular to $\vec{Loss}$. Theorem **??** proves that D of the vectors $\vec{Dir}_m$ are linearly independent. This, however, does not help that much because if these $\vec{Dir}_m$ were perpendicular to each other and one of them equals $\vec{Loss}$, then all but this one will be perpendicular to $\vec{Loss}$. On the other hand, if the vectors

$\vec{Dir}_m$ span the D dimensional space, then at least one of them is not in the $D-1$ dimensional subspace of vectors perpendicular to $\vec{Loss}$. This keeps the computation moving. These are the types of problems arising in [**?**] that we are ignoring.