Gradient Clipping in Deep Learning: A Dynamical Systems Perspective

Arunselvan Ramaswamy[a]
Dept. of Mathematics and Computer Science, Karlstad University, 651 88 Karlstad, Sweden

Keywords: Deep Learning, Adaptive Gradient Clipping, Dynamical Systems Perspective, Learning Theory, Supervised Learning.

Abstract: Neural networks are ubiquitous components of Machine Learning (ML) algorithms. However, training them is challenging due to problems associated with exploding and vanishing loss-gradients. Gradient clipping is shown to effectively combat both the vanishing gradients and the exploding gradients problems. As the name suggests, gradients are clipped in order to prevent large updates. At the same time, very small neural network weights are updated using larger step-sizes. Although widely used in practice, there is very little theory surrounding clipping. In this paper, we analyze two popular gradient clipping techniques – the classic norm-based gradient clipping method and the adaptive gradient clipping technique. We prove that gradient clipping ensures numerical stability with very high probability. Further, clipping based stochastic gradient descent converges to a set of neural network weights that minimizes the average scaled training loss in a local sense. The averaging is with respect to the distribution that generated the training data. The scaling is a consequence of gradient clipping. We use tools from the theory of dynamical systems for the presented analysis.

1 INTRODUCTION

The proliferation of complex neural network (NN) architectures, coupled with sophisticated Machine Learning (ML) algorithms and cheap increased computational capacity has caused a push towards automation in various aspects of the society. Supervised learning is an important paradigm of ML, where the aim is to learn an unknown map $f : \mathcal{X} \to \mathcal{Y}$ using finitely many examples – $\{(x_i, f(x_i)) \mid x \in \mathcal{X},\ 1 \leq i \leq N\}$ (LeCun et al., 2015). $\mathcal{X}$ is referred to as the input space and $\mathcal{Y}$ as the output space. Typically, $\mathcal{X} \equiv \mathbb{R}^d$, for some $d \geq 1$, and $\mathcal{Y}$ is either discrete or continuous. When $\mathcal{Y}$ is discrete, we are in the setting of classification. When $\mathcal{Y}$ is continuous, it is typically a subset of the real space $\mathbb{R}$, and the setting is called regression. The unknown function f is called the target. The aim is to train a predictor in order to approximate f. The example data used for training is called training data – $\mathcal{D} \equiv \{(x_i, y_i) \mid x \in \mathcal{X},\ y_i \in \mathcal{Y},\ 1 \leq i \leq N\}$. The standard assumption in ML is that the dataset $\mathcal{D}$ is generated by sampling N datapoints from $\mathcal{X} \times \mathcal{Y}$ in an independent manner using the same joint probability distribution, say μ. The reader is referred to (Bishop and Nasrabadi, 2006) for details.

In deep learning, the predictor used to approximate the target is a deep neural network (DNN). It is a NN with two or more hidden layers. The goal is to find a set of DNN weights so that the prediction errors are minimized, and the resulting DNN is a good approximation of the target f. An appropriate *loss function* is defined and the NN weights are iteratively updated to minimize loss using the training data $\mathcal{D}$. Minimizing the loss minimizes the prediction errors. In this paper, we consider NN training using two important loss functions: cross-entropy loss (for classification) and the mean squared error (for regression). Stochastic gradient descent is a simple yet popular choice for training a NN (LeCun et al., 2015). It involves the following update step, at time t, for the NN weight vector $\boldsymbol{\theta}$:

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - a(t)\frac{1}{M} \sum_{j=1}^{M} \nabla_{\theta} \ell\big(\boldsymbol{\theta}(t), x_{i(j)}(t), y_{i(j)}(t)\big), \tag{1}$$

where $a(t)$ is the step-size or learning rate, ℓ is the loss, $\big(x_{i(j)}(t), y_{i(j)}(t)\big) \in \mathcal{D}$, and $M < \infty$ is the sample-batch size. The sample-batch is sampled uniformly from $\mathcal{D}$. For the sake of simplicity in presentation, we let $M = 1$. Also, we use $(x(t), y(t))$ to represent the datapoint sampled at time t for the update. Hence,

[a] https://orcid.org/0000-0001-7547-8111

Ramaswamy, A.
Gradient Clipping in Deep Learning: A Dynamical Systems Perspective.
DOI: 10.5220/0011678000003411
In *Proceedings of the 12th International Conference on Pattern Recognition Applications and Methods (ICPRAM 2023)*, pages 107-114
ISBN: 978-989-758-626-2; ISSN: 2184-4313

(1) becomes

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - a(t)\nabla_{\theta}\ell\big(\boldsymbol{\theta}(t), x(t), y(t)\big). \quad (2)$$

Derivatives in NNs are computed using the back propagation algorithm. In practice, due to the bottleneck in computer capacity, the farther the loss-derivative needs to be backpropagated, the more they *vanish*. In effect, some NN weights are never updated. The complementary problem is that of exploding gradients – numerical instability caused by large update values. These issues are very well documented, see (Rehmer and Kroll, 2020), (Schmidhuber, 2015) and (LeCun et al., 2015). Gradient clipping is a popular solution to the exploding gradient problem. In this paper, we explore two methods, the classic norm-based gradient clipping and the adaptive clipping method. Adaptive clipping additionally addresses the vanishing gradients issue, albeit only partially.

The principle behind gradient clipping is to *clip it* when it grows too large in order to prevent numerically unstable large updates. In the classic version of the norm-based gradient clipping scheme, the gradient is divided (scaled-down) by its norm when the norm exceeds a certain predetermined value. We analyze its stability properties in Section 3.1 and discuss its asymptotic properties in Section 4. In particular, we show that with high probability, the clipped gradient descent is numerically stable and converges to a local minimizer of the loss function, on an average. Although widely used in deep learning solutions and empirically successful there is very little theory surrounding it. In the past researchers have studied the performance under idealized conditions. To the best of our knowledge ours is the first study that analyzes the practical version of the algorithm.

Formally speaking, in the classic method, we divide the gradient update by $\frac{\|\nabla_{\theta}\ell\|}{\lambda}$ for some $\lambda > 0$, when the gradient norm exceeds a predetermined value. There is a variant to this approach that is less conservative called the adaptive gradient clipping method. Here, the update is divided by $\frac{\|\nabla_{\theta}\ell\|}{\lambda\|\boldsymbol{\theta}\|}$, when the gradient norm exceeds a certain multiple of the norm of the NN weights. Here, the updates are larger and convergence faster (Zhang et al., 2019). Although very insightful, again, only an idealized version of this algorithm is analyzed in (Zhang et al., 2019). In this paper, we remedy this. Specifically, we show that numerical stability is not automatic since the updates are in the order of the NN weights. If stable, then we show that the adaptive variant converges to an averaging of the local minimizer of the loss function. The averaging is with respect to the training data distribution.

The organization of this paper is as follows. First, in Section 2, we formally discuss the general gradient descent in the context of deep learning. Next, in Section 3, we introduce the classic and adaptive clipped gradient methods and discuss their numerical stability. Finally, we discuss their asymptotic properties in Section 4.

2 STOCHASTIC GRADIENTS IN DEEP LEARNING

Let us begin with the simple stochastic gradient descent algorithm involved in training any neural network (NN). It is described by the following iteration:

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - a(t)g(t), \quad (3)$$

where $\theta(t)$ represents the vector of NN weights at time t, $a(t)$ is the learning rate or the step-size at time t; $g(t)$ is the loss-gradient. In this paper, we consider two loss functions, one that is popular for classification and the other that is popular for regression. We, in particular, consider the cross entropy and the mean squared losses. It must however be noted that theory presented herein can be generally applied to other loss functions.

In a typical regression application, the output of the NN is a real number. Let us represent the output by $f(x,\theta)$, where x is the input query instance and θ is the NN weight-vector. The mean squared error is then

$$\ell_r = (f(\boldsymbol{\theta}, x) - y)^2, \quad (4)$$

where y is the true label of x. The loss-gradient is then $2\,(f(\boldsymbol{\theta},x) - y)\,\nabla_{\theta} f(\boldsymbol{\theta},x)$. In order to obtain $\nabla_{\theta} f(\boldsymbol{\theta},x)$, the algorithm performs a backward pass (back propagation) through the NN. The forward pass yields $f(x,\theta)$.

In the K-class classification setting, the NN has K neurons (activations) in the output layer. Say, the outputs of these activations are z_i, $1 \le i \le K$. Then, $p_i := e^{z_i} / \sum_{j=1}^{K} e^{z_j}$ is interpreted as the posterior probability that the true label is i. This is called the soft-max classifier, and is trained using the cross-entropy loss:

$$\ell_c = -\sum_{i=1}^{K} \mathbb{1}_{y=i} \log p_i. \quad (5)$$

Here, $\mathbb{1}_{y=i}$ is the indicator function that takes values 1 or 0, depending on whether the true class-label is i or not. In the traditional setting of classification, exactly one label is associated with each query instance x. Suppose the true label is i, then the partial deriva-

tive of ℓ_c with respect to z_k is given by

$$\frac{-\sum_{j=1}^{K} e^{z_j}}{e^{z_i}} \left(\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \mathbb{1}_{k=i} - \frac{e^{z_i} e^{z_k}}{\left(\sum_{j=1}^{K} e^{z_j} \right)^2} \right). \quad (6)$$

We can further simplify the above equation to get $-\mathbb{1}_{k=i} + \frac{e^{z_k}}{\sum_{j=1}^{K} e^{z_j}} = p_k - \mathbb{1}_{k=i}$. This gradient information is back propagated through the NN in order to update the NN weights. We are now ready to make a couple of assumptions in our analysis.

(A1) All activations in the NN are twice continuously differentiable. Popular examples include sigmoid, tanh and the Gaussian error linear unit.

The input to the NN is a vector $x \in \mathbb{R}^d$, for some $d \geq 1$. This input is first *passed through* the input layer, this is represented by $\Sigma(\boldsymbol{\theta}^{ip} x)$, $\boldsymbol{\theta}^{ip}$ is a matrix of dimension $d \times$ the number of activations in the input layer, Σ is an operator that takes a vector z, of some dimension, as input and outputs a vector of the same dimension such that its i^{th} component equals the activation applied to the i^{th} component of z, i.e., $\Sigma(z)_i = \sigma(z_i)$. The vector $\Sigma(\boldsymbol{\theta}^{ip} x)$ is then passed through the first hidden layer to get $\Sigma(\boldsymbol{\theta}^h \Sigma(\boldsymbol{\theta}^{ip} x))$. Here, $\boldsymbol{\theta}^h$ is a matrix of appropriate dimension. It represents the set of NN weights from the first hidden layer. Now, $\Sigma(\boldsymbol{\theta}^h \Sigma(\boldsymbol{\theta}^{ip} x))$ may be passed through more hidden layer, however, for the sake of simplicity we assume there are no more hidden layers. This means that $\Sigma(\boldsymbol{\theta}^h \Sigma(\boldsymbol{\theta}^{ip} x))$ is passed through the output layer, which again looks like $\Sigma(\boldsymbol{\theta}^{op} \Sigma(\boldsymbol{\theta}^h \Sigma(\boldsymbol{\theta}^{ip} x)))$. In the classification setting, the output of the NN, $f(\boldsymbol{\theta}, x)$, is then a class-label distribution given by $\left(e^{z_i} / \sum_{j=1}^{K} e^{z_j} \right)_{1 \leq i \leq K}$, where K is the number of classes and $z_i := \Sigma(\boldsymbol{\theta}^{op} \Sigma(\boldsymbol{\theta}^h \Sigma(\boldsymbol{\theta}^{ip} x)))_i$. Note that $\boldsymbol{\theta}$ is the vector of all NN weights – $\boldsymbol{\theta}^{ip}$, $\boldsymbol{\theta}^h$ and $\boldsymbol{\theta}^{op}$. In regression, $f(\boldsymbol{\theta}, x) = \sum z_i \boldsymbol{\theta}^l{}_i$, the output is a linear combination of the activation outputs from the output layer, and z_i is as defined before. Note that $\boldsymbol{\theta}^l$ is a vector that is also a part of the NN weight vector $\boldsymbol{\theta}$.

Passing the input through the various layers of a NN is called the *forward pass.* It must be noted that the above description of a NN is not complete. In particular, it must be noted that every vector is appended with 1 before passing it through the next layer. In particular $(1, x)$ is the input. Then, $\left(1, \Sigma\left(\boldsymbol{\theta}^{ip} \begin{pmatrix} 1 \\ x \end{pmatrix}\right)\right)$ is the input to the hidden layer and so on. This is done to incorporate bias information. These biases are trainable as a part of the NN weight vector. Again, for the sake of simplicity in presentation we omit the biases.

Claim 1. *Under (A1), $\frac{\partial \ell_r}{\partial \boldsymbol{\theta}_i}$, $\frac{\partial \ell_c}{\partial \boldsymbol{\theta}_i}$, $\frac{\partial^2 \ell_r}{\partial \boldsymbol{\theta}_i \boldsymbol{\theta}_j}$ and $\frac{\partial^2 \ell_c}{\partial \boldsymbol{\theta}_i \boldsymbol{\theta}_j}$ exist and are continuous, where $\boldsymbol{\theta}_i$ and $\boldsymbol{\theta}_j$ are components of $\boldsymbol{\theta}$, ℓ_r and ℓ_c are defined in* (4) *and* (5)*, respectively. Therefore, ℓ_r, ℓ_c $\nabla_\theta \ell_r$ and $\nabla_\theta \ell_c$ are all locally Lipschitz continuous.*

Proof. Let us first consider the case of regression with the loss function being the mean-squared error. Since, (A1) requires that the activation functions are twice continuously differentiable, we directly get that $f(\boldsymbol{\theta}, x)$ is also twice continuously differentiable, as it is a composition of operations that are themselves twice continuously differentiable. Specifically, it is a direct consequence of f being a repeated composition of activations and linear combinations. Hence, $\frac{\partial f(\boldsymbol{\theta}, x)}{\partial \boldsymbol{\theta}_i}$ and $\frac{\partial^2 f(\boldsymbol{\theta}, x)}{\partial \boldsymbol{\theta}_i \boldsymbol{\theta}_j}$ are continuous, directly implying that $\frac{\partial \ell_r}{\partial \boldsymbol{\theta}_i}$ and $\frac{\partial^2 \ell_r}{\partial \boldsymbol{\theta}_i \boldsymbol{\theta}_j}$ are also continuous.

Now, we move onto classification. Recall the loss-gradient with respect to z_k, $1 \leq k \leq K$, when the true label is $y = i$, from (6)

$$p_k - \mathbb{1}_{k=i}.$$

We therefore get:

$$\frac{\partial \ell_c}{\partial \boldsymbol{\theta}_i} = \sum_{k=1}^{K} \frac{\partial z_k}{\partial \boldsymbol{\theta}_i} \frac{\partial \ell_c}{\partial z_k}. \quad (7)$$

Suppose $\boldsymbol{\theta}_i$ is not used to calculate the output value z_k, then clearly $\frac{\partial z_k}{\partial \boldsymbol{\theta}_i} = 0$. For example, $\boldsymbol{\theta}^{op}{}_{2,3}$ is only used in the calculation of z_2, but not to the other $z_j' s$, e.g., $\frac{\partial z_1}{\partial \boldsymbol{\theta}^{op}{}_{2,3}} = 0$. Recall that $\boldsymbol{\theta}^{op}$ is a matrix, hence $\theta_{2,3}$ is the element on row-2 and column-3. The NN weight vector $\boldsymbol{\theta}$ is obtained by flattening all the matrices and appending them. As before, $\frac{\partial^2 z_k}{\partial \boldsymbol{\theta}_i \boldsymbol{\theta}_j}$ is readily calculated, and is continuous as a consequence of (A1). Since $\frac{\partial \ell_c}{\partial z_k} = p_k - \mathbb{1}_{k=i}$, it is also continuous. Also,

$$\frac{\partial^2 \ell_c}{\partial \boldsymbol{\theta}_j \boldsymbol{\theta}_i} = \sum_{k=1}^{K} \left[\frac{\partial^2 z_k}{\partial \boldsymbol{\theta}_j \boldsymbol{\theta}_i} \frac{\partial \ell_c}{\partial z_k} + \frac{\partial z_k}{\partial \boldsymbol{\theta}_i} \left(\sum_{l=1}^{K} \frac{\partial z_l}{\partial \boldsymbol{\theta}_j} \frac{\partial^2 \ell_c}{\partial z_l z_k} \right) \right]. \quad (8)$$

We therefore get the required continuity of $\frac{\partial^2 \ell_c}{\partial \boldsymbol{\theta}_j \boldsymbol{\theta}_i}$ and $\frac{\partial \ell_c}{\partial \boldsymbol{\theta}_i}$.

A function $f : \mathbb{R}^d \to \mathbb{R}^m$ is Lipschtiz continuous suppose there exists L such that $\|f(x) - f(y)\| \leq L\|x - y\|$, where $d, m \geq 1$. It is locally Lipschitz continuous, *iff* for every $x \in \mathbb{R}^d$ $\exists L_x > 0 and r_x > 0$ such

that $\|f(x)-f(y)\| \leq L_x\|x-y\|$ for every $y \in B_{r_x}(x)$, where $B_{r_x}(x)$ is the ball of radius r_x centered at point x. Suppose f is continuously differentiable, then f is locally Lipschtiz continuous (Rudin et al., 1976). Similarly, $\nabla_x f$ is locally Lipschitz continuous when the Hessian is continuous. The second part of the claim, regarding Lipschitzness, is a direct consequence of these. □

3 GRADIENT CLIPPING AND NUMERICAL STABILITY

In this section, we introduce the concept of gradient clipping. We delve into the details of two important gradient clipping methods. We also formally discuss their numerical stability. We begin with a discussion on the classic norm-based gradient clipping that is popular in Deep Learning.

3.1 Gradient Clipping in Deep Learning

Let us return to the popular stochastic gradient descent algorithm given by (3). Suppose assumption (A1) is satisifed, then we get from Claim 1 that $g(t)$ is locally Lipschitz continuous in the $\boldsymbol{\theta}$ coordinate. Recall that the loss-gradient at time t is calculated using the datapoint $(x(t),y(t))$ from the dataset $\mathcal{D}$. In the regression setting the loss gradient $g(t) \equiv 2\big(f(\boldsymbol{\theta}(t),x(t))-y(t)\big)\nabla_\theta f(\boldsymbol{\theta}(t),x(t))$, and it is similarly calculated in the classification setting. At time t, the datapoint $(x(t),y(t))$ is processed from the given training dataset in order to obtain the loss-gradient $g(t)$. There could be more than one datapoint processed, e.g., in a batch-processing implementation of the learning algorithm. For the sake of simplicity in presentation we assume that a single datapoint is processed at every point in time in order to calculate the loss-gradient.

As the NN is highly non-linear the calculated gradient $g(t)$ may be so large that the weight update process is numerically unstable. This is called the exploding gradients problems. This is a common problem when using the cross-entropy loss function. The non-linearity of a NN is directly proportional to its depth. However, this means that the loss-gradients do not propagate back into the network. The weights in the input and the initial hidden layers are updated using very small numerically insignificant gradient values. This is called the vanishing gradient problem, and is common when using deep network architectures.

The exploding gradient problem is often countered by projecting the iterate after every step onto a predetermined compact set $\mathcal{K}$, see (Boyd and Vandenberghe, 2004) and (Vu and Raich, 2022). The projected scheme is given by

$$\begin{aligned} \boldsymbol{\theta}(t+1) &\leftarrow \boldsymbol{\theta}(t)-a(t)g(t), \\ \boldsymbol{\theta}(t+1) &\leftarrow \Pi_{\mathcal{K}}\boldsymbol{\theta}(t+1), \end{aligned} \tag{9}$$

where $\Pi_{\mathcal{K}}$ is the projection operator. It projects $\theta(t+1)$ onto the nearest point from $\mathcal{K}$. Hence, (9) is guaranteed to be numerically stable. However, it does not always converge to the optimal. Gradient clipping is another popular solution to the exploding gradients problem, our main focus here. Intuitively, the idea is to "clip" large gradients in order to maintain stability. In particular, instead of (3) or (9), the update is given by

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - a(t)\frac{g(t)}{c(t)}, \tag{10}$$

where $c(t)$ is the clipping value at time t. In the classical version of norm-based clipping

$$c(t) := \frac{\|g(t)\|}{\lambda} \vee 1, \tag{11}$$

where $\vee$ is the max operator. When $\|g(t)\| > \lambda$, then the gradient update is scaled down by $\frac{\|g(t)\|}{\lambda}$, $\lambda > 0$ is a predetermined value. The NN weights are updated using gradients that are "clipped at λ" to prevent the updates from exploding. In order to formally show numerical stability under norm-based gradient clipping, we need the following assumption on the learning rate.

(A2) $a(t) > 0$ for all $t \geq 0$, $\sum_{t=0}^{\infty} a(t) = \infty$ and $\sum_{t=0}^{\infty} a(t)^2 < \infty$.

We now claim that (10) does not experience "finite-time blowup" with very high probability. By finite-time blowup we mean that the NN weights, updated according to (10), remain within a sphere of radius M centered around the origin for the entire duration of the experiment. We may think of M as a very large positive number – determined by the largest number that can be processed by a computer and the upper-bound on the duration of the experiment.

Lemma 1. *Assuming (A1) and (A2), the stochastic gradient descent with the classic norm-based clipping,* (10)*, does not experience finite blow-up with very high probability.*

Proof. Let us rewrite (10) as follows

$$\boldsymbol{\theta}(s) = \boldsymbol{\theta}(0) - \sum_{t=0}^{s-1} a(t)\frac{g(t)}{c(t)}. \tag{12}$$

Let $T < \infty$ be the duration of the experiment, and let $M > 0$ be a very large arbitrary constant. We need

to show that $\|\boldsymbol{\theta}(t)\| < M$ for $0 \le t \le T$ with very high probability. Since M is arbitrary, we show an equivalent variant of this, specifically we show that $\|\boldsymbol{\theta}(t) - \boldsymbol{\theta}(0)\| < M$ for $1 \le t \le T$ with very high probability. First, we begin by observing that for $s \ge 1$

$$\|\boldsymbol{\theta}(s)\| \le \|\boldsymbol{\theta}(0)\| + \sum_{t=0}^{s-1} a(t) \frac{\|g(t)\|}{c(t)}. \tag{13}$$

Next, we define the following random variables for $0 \le s \le T$, $M_s := \|\boldsymbol{\theta}(0)\| + \sum_{t=0}^{s-1} a(t) \frac{\|g(t)\|}{c(t)}$. We also define the following sigma-algebras – $\mathcal{F}_0 \equiv \sigma\langle \boldsymbol{\theta}(0) \rangle$ and $\mathcal{F}_s \equiv \sigma\Big\langle \boldsymbol{\theta}(t), ((x(u), y(u))) : 0 \le t \le s,\ 0 \le u \le s-1 \Big\rangle$. Since the datapoints from the training dataset are independently generated, we get that $\mathbb{E}[M_{s+1} \mid \mathcal{F}_s] \ge M_s$ for $0 \le s \le T-1$. Hence, $(M_t, \mathcal{F}_t)_{0 \le t \le T}$ is a super-Martinagle. The reader is referred to (Durrett, 2019) or (Durrett, 2018) for the definitions of a super-Martingale and a sigma-algebra. Since $c(t) = \frac{\|g(t)\|}{\lambda} \vee 1$, we get that $\frac{\|g(t)\|}{c(t)} \le \lambda$, further we get that $|M_{t+1} - M_t| \le a(t)\lambda$. It follows from the Hoeffding-Azuma inequality (Bercu et al., 2015) that

$$P(|M_s - M_0| > M) \le 2\mathrm{e}^{-2M^2 / \sum_{t=0}^{s-1} a(t)^2 \lambda^2} \tag{14}$$

for $1 \le s \le T$. Now, since $\sum_{t=0}^{s-1} a(t)^2 \le \underbrace{\sum_{t=0}^{\infty} a(t)^2}_{:=\gamma} < \infty$,

$$P(|M_s - M_0| > M) \le 2\mathrm{e}^{-2M^2 / \sum_{t=0}^{s-1} a(t)^2 \lambda^2} \le 2\mathrm{e}^{-2M^2/\gamma\lambda^2}. \tag{15}$$

Let us say that $a(t) = 1/t$ is the learning rate, then $\sum_{t \ge 1} \frac{1}{t^2} = \frac{\pi^2}{6}$, i.e., $\gamma = \frac{\pi^2}{6}$. Since M is arbitrary, let us fix it as 2^{10}. A typical value for λ is something like 30. Plugging, these values into (15), we get that

$$P(|M_s - M_0| > M) \le \frac{2}{\mathrm{e}^{1416}}. \tag{16}$$

The RHS of (16) is very close to zero. If we choose $s = T$, then we get that $P(|M_T - M_0| \le M) \approx 1$. Since, the super-Martingale sequence is almost surely increasing, we get that $P(|M_s - M_0| \le M \mid 1 \le s \le T) \approx 1$. We get from (12) that $\|\boldsymbol{\theta}(s) - \boldsymbol{\theta}(0)\| = \left\| \sum_{t=0}^{s-1} a(t) \frac{g(t)}{c(t)} \right\|$. The RHS of this equality is further less than $\sum_{t=0}^{s-1} a(t) \frac{\|g(t)\|}{c(t)} = |M_s - M_0|$. Hence, we conclude that $\|\boldsymbol{\theta}(s) - \boldsymbol{\theta}(0)\| \le M$ whenever $|M_s - M_0| \le M$. This gives us the required – $P\big(\|\boldsymbol{\theta}(s) - \boldsymbol{\theta}(0)\| \le M \mid 1 \le s \le T\big) \approx 1$. □

3.2 Adaptive Gradient Clipping in Deep Learning

The classic gradient clipping method may slow down the rate of convergence via conservative updates. Adaptive gradient clipping tries to remedy this by modifying $c(t)$ as follows:

$$c(t) = \frac{\|g(t)\|}{\lambda[\|\boldsymbol{\theta}(t)\| \vee \varepsilon]} \vee 1. \tag{17}$$

Like λ, ε is also predetermined. Usually, ε is chosen to be less than 1. Let us suppose that the NN weights are tiny, in that $\|\boldsymbol{\theta}(t)\| \le \varepsilon$. Also, let $\lambda\varepsilon < \|g(t)\| < \lambda$. Then, the order of update at time t is $\lambda\varepsilon$ in the adaptive clipping scenario, less than the corresponding update when using the classic clipping method. Now, consider the scenario wherein $\boldsymbol{\theta}(t)$ is very large, in particular $\|\boldsymbol{\theta}(t)\| > 1$. When $\|g(t)\| \ge \lambda\|\boldsymbol{\theta}(t)\|$, the update according to the adaptive clip is in the order of $\|\boldsymbol{\theta}(t)\|\lambda$, and the update recommended by the classic method is in the order of λ. In the latter case, the adaptive clipping method updates proportional to the norm of the NN weight vector.

The adaptive clipping given by (17) is simplistic. In practice, the updates for different layers of the NN are clipped differently. In order to update $\boldsymbol{\theta}^{ip}{}_{i,j}$, the adaptive clip method calculates the scaling factor as

$$\frac{\|g(t)\|}{\lambda\left[\|\boldsymbol{\theta}^{ip}(t)\|_F \vee \varepsilon\right]} \vee 1, \tag{18}$$

where $\|\boldsymbol{\theta}^{ip}(t)\|_F$ is the Frobenious norm of the input weight matrix at time t. Suppose we flatten $\boldsymbol{\theta}^{ip}(t)$ into a vector, then one can show that the Frobenius norm equals the Euclidean norm of the flattened version (Halmos, 2017). By clipping updates to different layers differently, the weights are updated in a manner that overcomes the vanishing gradient problem. Although qualitatively different, we will use (17) and not (18) in our analysis, since there is no difference in terms of the steps involved. Further, it greatly reduces kitsch and improves readability and ease of understanding.

Recall that in the classic clipping method (10), the updates are always bounded by λ. However, the update in the adaptive clipping scheme can be in the order of the NN weights. Formally, we claim the following.

Claim 2. *In the adaptive clipping scheme,* (17), $\left\| \frac{g(t)}{c(t)} \right\| \le K(1 + \|\boldsymbol{\theta}(t)\|)$, *where* $K > 0$ *is an iterate independent constant.*

Proof. Suppose $\frac{\|g(t)\|}{\|\boldsymbol{\theta}(t)\| \vee \varepsilon} \le \lambda$, then $c(t) = 1$. Also,

$\|g(t)\| \leq \lambda[\|\boldsymbol{\theta}(t)\| \vee \varepsilon]$. Define $K := \lambda\varepsilon \vee \lambda \vee 1$, then we get that $\left\|\frac{g(t)}{c(t)}\right\| \leq K(1+\|\boldsymbol{\theta}(t)\|)$.

Now, suppose $\frac{\|g(t)\|}{\|\boldsymbol{\theta}(t)\|\vee\varepsilon} > \lambda$, then $c(t) = \frac{\|g(t)\|}{\lambda[\|\boldsymbol{\theta}(t)\|\vee\varepsilon]}$. Further,

$$\left\|\frac{g(t)}{c(t)}\right\| = \frac{\lambda[\|\boldsymbol{\theta}(t)\| \vee \varepsilon]\,\|g(t)\|}{\|g(t)\|}. \tag{19}$$

We can choose the same K as before in order to get the required linear bound even in this case. □

We have thus shown that the updates in the adaptive clipping scheme grows linearly as a function of the NN weight vector. *Stability is therefore not guaranteed.* Additional conditions must be satisfied for the clipped gradient scheme to be numerically stable. There are extensive studies on the stability of systems that grow linearly. The reader is referred to (Ramaswamy and Bhatnagar, 2017), (Ramaswamy and Bhatnagar, 2018) and (Borkar, 2009) for more on this. We assume that sufficient conditions for stability, as discussed in the aforementioned literature, can be verified for the adaptive clipping case. We will, in effect, assume that the adaptive clipping method is stable with very high probability.

(A3) The adaptive clipping gradient descent scheme is numerically stable with very high probability.

4 ASYMPTOTICS OF GRADIENT CLIPPING METHODS

In Section 3, we saw that under mild conditions the clipped gradient descent scheme (10) is numerically stable with very high probability. We also saw that the adaptive clipping scheme is not always numerically stable on account of aggressive gradient updates. However, in this case we dip may dip into the theory of linear systems to ensure stability. Given that the gradient descent algorithm with clipping, classic or adaptive, is numerically stable, the question remains, does it converge to the required set of NN weights that minimizes loss? In this section, we use tools from Dynamical Systems Theory for the convergence analysis of the two gradient clipping methods. Important literature on Dynamical Systems Theory include (Borkar, 2009), (Aubin and Cellina, 2012) and (Benaïm et al., 2005).

For the convergence analysis, we use the theory from (Benaïm et al., 2005). This theory is based on viewing the clipped gradient descent scheme as a stochastic approximation algorithm, and associating with it an ordinary differential equation. The associated o.d.e. has the same asymptotic properties as the stochastic approximation algorithm. Recall the clipped gradient descent scheme.

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - a(t)\frac{g(t)}{c(t)}. \quad \text{(recalled from (10))}$$

Here the clipping constant $c(t)$ is either (11) or (17), depending on if we are in the classic or the adaptive setting. In order to utilize the theory developed in (Benaïm et al., 2005) we rewrite (10) as the stochastic approximation algorithm below

$$\boldsymbol{\theta}(t+1) = \boldsymbol{\theta}(t) - a(t)\left(\mathop{\mathbb{E}}_{(x(t),y(t))\sim\mu}\left[\frac{g(t)}{c(t)}\right] + M(t+1)\right), \tag{20}$$

where $M_{t+1} = \frac{g(t)}{c(t)} - \mathop{\mathbb{E}}_{(x(t),y(t))\sim\mu}\left[\frac{g(t)}{c(t)}\right]$; μ is the joint probability distribution over $\mathcal{X}\times\mathcal{Y}$. Recall that the training dataset $\mathcal{D}$ is generated by sampling from μ.

First, we show that the objective function in (20) is a Marchaud map. A point-to-set map $H : \mathbb{R}^m \to \{\text{ subsets of } \mathbb{R}^m\}$ is called Marchaud, iff it satisfies the following conditions (a) $H(x)$ is convex and compact, (b) $\sup_{z\in H(x)} \|z\| \leq K(1+\|x\|)$ for fixed $K > 0$, and (c) if $x_n \to x$ and $z_n \to z$ in $\mathbb{R}^m$ such that $z_n \in H(x_n)$ for all $n \geq 0$, then $z \in H(x)$ (upper semicontinuity property). Although our objective function $H(\boldsymbol{\theta}) := \mathop{\mathbb{E}}_{(x,y)\sim\mu}\left[\frac{g}{c}\right]$ is a standard point-to-point map, we may view it as a trivial point-to-set map such that $H(\boldsymbol{\theta})$ is the singleton $\left\{\mathop{\mathbb{E}}_{(x,y)\sim\mu}\left[\frac{g}{c}\right]\right\}$.

Lemma 2. *Under assumption (A1), the trivial point-to-set map* $H : \theta \mapsto \left\{\mathop{\mathbb{E}}_{(x,y)\sim\mu}\left[\frac{g}{c}\right]\right\}$*, the objective function* $H(\boldsymbol{\theta})$ *in the clipped gradient descent* (20) *is Marchaud.*

Proof. Since the objective $H(\theta)$ is really a point-to-point map, it is trivially convex and compact. Now, we note that $g \equiv \nabla_\theta \ell_r$ for regression and $g \equiv \ell_c$ for classification. We know from Claim 1 that g is continuous. The clipping function c is also continuous, this follows from the continuity of the "max operator", and from the continuity of $\|g\|$. Hence, $\frac{g}{c}$ is continuous in the $\boldsymbol{\theta}$ variable, keeping x and y fixed. The continuity of $\mathop{\mathbb{E}}_{(x,y)\sim\mu}\left[\frac{g}{c}\right]$ can be shown using the Dominated Convergence Theorem (Rudin et al., 1976) or (Durrett, 2019). The upper semicontinuity property of H boils down to standard continuity as the range of H only consists of singletons.

It is left to show that there exists $K > 0$ such that $\left\| \mathop{\mathbb{E}}_{(x,y)\sim\mu} \left[\frac{g}{c}\right] \right\| \leq K(1+\|\boldsymbol{\theta}\|)$. When the clipping function is given by (10), then we know that $\left\|\frac{g}{c}\right\| \leq \lambda$. For the adaptive clipping function (17), it follows from Claim 2 that there exists $K > 0$ such that $\left\|\frac{g}{c}\right\| \leq K(1+\|\boldsymbol{\theta}\|)$. Therefore, in both the clipping schemes, we get the following:

$$\left\| \mathop{\mathbb{E}}_{(x,y)\sim\mu} \left[\frac{g}{c}\right] \right\| \leq \mathop{\mathbb{E}}_{(x,y)\sim\mu} \left\|\frac{g}{c}\right\|,$$
$$\mathop{\mathbb{E}}_{(x,y)\sim\mu} \left\|\frac{g}{c}\right\| \leq \mathop{\mathbb{E}}_{(x,y)\sim\mu} \left[K(1+\|\boldsymbol{\theta}\|)\right].$$

We have thus shown that the objective in (20) is a Marchaud map. In other words, the *expected clipped loss-gradient* is continuous and grows linearly as a function of the NN weights. □

$M_{t+1} = \frac{g(t)}{c(t)} - \mathop{\mathbb{E}}_{(x(t),y(t))\sim\mu} \left[\frac{g(t)}{c(t)}\right]$, in (20), is interpreted as the sampling error at time t, the error induced when using the sample $(x(t),y(t))$ to calculate the loss-gradient in lieu of the expected clipped loss-gradient. On the sample paths (read "runs of the clipped gradient descent algorithm") where the condition $\sum_{t\geq 0} a(t)M_{t+1} < \infty$ is met, it can be shown that (20) has the desired asymptotic properties. Shortly, we will show that numerical stability of the algorithm implies that the above mentioned condition holds. Hence, the condition holds with very high probability. Before we can state the precise statement, we need to define the following random variables – $\xi(0) = 0$ and $\xi(t+1) = \sum_{s=0}^{t} a(s)M(s+1)$ for $t \geq 0$ – and we need to recall the following filtration – $\mathcal{F}_0 \equiv \sigma\langle\boldsymbol{\theta}(0)\rangle$ and $\mathcal{F}_s \equiv \sigma\Big\langle \boldsymbol{\theta}(t), ((x(u),y(u))) : 0 \leq t \leq s,\ 0 \leq u \leq s-1 \Big\rangle$ for $s \geq 1$.

Lemma 3. *Assuming (A1), (A2) and (A3) we get that $\langle \xi(t), \mathcal{F}_t\rangle_{t\geq 0}$ is a Martingale sequence such that $\lim_{t\to\infty} \xi(t)$ exists with high probability. Hence, $\sum_{t\geq 0} a(t)M(t+1) < \infty$ with high probability.*

Proof. First, observe that $\mathbb{E}[\xi(t+1) \mid \mathcal{F}_t] = \xi_t + \mathbb{E}[a(t)M_{t+1} \mid \mathcal{F}_t]$. In order to conclude that the given sequence is a Martingale we need to show that $\mathbb{E}[a(t)M_{t+1} \mid \mathcal{F}_t] = 0$. Recall that $M_{t+1} = \frac{g(t)}{c(t)} - \mathop{\mathbb{E}}_{(x(t),y(t))\sim\mu} \left[\frac{g(t)}{c(t)}\right]$, hence $\mathbb{E}[a(t)M_{t+1} \mid \mathcal{F}_t] = a(t)\left(\mathbb{E}\left[\frac{g(t)}{c(t)} \mid \mathcal{F}_t\right] - \mathop{\mathbb{E}}_{(x(t),y(t))\sim\mu} \left[\frac{g(t)}{c(t)}\right]\right)$. It however follows from the definition of the filtration that $\frac{g(t)}{c(t)}$ and $\mathcal{F}_t$ are independent, hence $\mathbb{E}\left[\frac{g(t)}{c(t)} \mid \mathcal{F}_t\right] = \mathop{\mathbb{E}}_{(x(t),y(t))\sim\mu} \left[\frac{g(t)}{c(t)}\right]$. This yields the required equality to zero.

Now, we show that the Martingale sequence has a limit with high probability. For this, we need to define the associated quadratic variation process $\langle \xi(t+1)\rangle := (\xi(t+1) - \xi(t)) \bigotimes (\xi(t+1) - \xi(t))$, where $\bigotimes$ is the operator for the element-wise multiplication of vectors and $t \geq 0$. On those sample points where $\sum_{t\geq 1} \langle \xi(t)\rangle < \infty$, the Martingale sequence has a limit (Durrett, 2019). Consider the following:

$$\left\| \sum_{t\geq 1} \langle \xi_t \rangle \right\| \leq \sum_{t\geq 0} a(t)^2 \|M(t+1)\|^2,$$
$$\sum_{t\geq 0} a(t)^2 \|M(t+1)\|^2 \leq \sum_{t\geq 0} a(t)^2 C(1+\|\boldsymbol{\theta}(t)\|^2).$$

The second inequality in the above equation follows from Claim 2 for some $C > 0$. Combining the assumption that $\sum_{t\geq 0} a(t)^2 < \infty$ – (A2) – with Lemma 1 (in case of classic clipping) or (A3) (in case of adaptive clipping), we get that $\sum_{t\geq 0} a(t)^2 C(1+\|\boldsymbol{\theta}(t)\|^2) < \infty$ with high probability. Therefore, $\sum_{t\geq 1} \langle \xi(t)\rangle < \infty$ with high probability. □

In this lemma we have essentially shown that whenever the clipped gradient descent is numerically stable, the errors that arise due to the use of training samples vanish asymptotically. Since we have already shown stability with high probability, the sampling errors also vanish with the same high probability. Now, we need to analyze the *convergence set* of the clipped gradient scheme.

The loss-gradient is a function of the NN weights and the training data. In order to state the main result of this paper, we will make this dependence explicit. Specifically, we let $g(t) \equiv \nabla_\theta \ell(\boldsymbol{\theta}(t), x(t), y(t))$, also note that we have dropped the "r" and "c" subscripts from the loss-gradient to mask the setting being considered – regression or classification. Similarly, we let the clipping function $c(t) \equiv c(\boldsymbol{\theta}(t), x(t), y(t))$. Therefore, in the re-written stochastic approximation equivalent of the clipped gradient descent, we get that the objective function

$$H(\boldsymbol{\theta}) \equiv \mathcal{E}_{(x,y)\sim\mu} \left[\frac{\nabla_\theta \ell(\boldsymbol{\theta}, x, y)}{c(\boldsymbol{\theta}, x, y)}\right]. \tag{21}$$

Before we state the main result of this paper, we recall the clipped gradient descent algorithm below.

$$\boldsymbol{\theta}(t+1) = \theta(t) - a(t)\left(H(\boldsymbol{\theta}(t)) + M(t+1)\right), \tag{(20) recalled}$$

where H is redefined in (21) and $M(t+1)$ is as previously defined following (20).

Theorem 1. *The classic clipped gradient descent converges with high probability to* $\boldsymbol{\theta}(\infty)$ *such that*

$$\mathbb{E}_{(x,y)\sim\mu}\frac{-\nabla_{\theta}\ell(\boldsymbol{\theta}(\infty),x,y)}{c(\boldsymbol{\theta}(\infty),x,y)}=0, \tag{22}$$

provided assumptions (A1) and (A2) are satisfied. If we additionally assume (A3), then the adaptive clipped gradient descent also converges with high probability to $\theta(\infty)$ *satisfying* (22)*.*

Proof. In Lemma 2, we showed that H is a Marchaud map. We also showed that the Martingale noise vanishes asymptotically with high probability, see Lemma 3. We may couple these Lemmas with the assumptions made to apply the theory developed in (Benaïm et al., 2005). It lets us conclude the following. When the clipped gradient descent is numerically stable, which we show happens with high probability, the limit of the clipped gradient descent coincides with the limit of an associated solution to the differential inclusion $\dot{\boldsymbol{\theta}}(t)\in -H(\boldsymbol{\theta}(t))$, the limit is taken as $t\to\infty$. Further, this limit has the property that it is an equilibrium of H. This means, $-H(\boldsymbol{\theta}(\infty))=0$ given that $\boldsymbol{\theta}(\infty)$ is the limit. Since the clipped gradient schemes – classic and adaptive – are stable with high probability, we get that it converges with high probability to $\boldsymbol{\theta}(\infty)$ satisfying (22). □

Colloquially speaking, the clipped gradient descent converges, with high probability, to the set of NN weights with the property that, on an average, the clipped loss-gradient is zero. Since, the clipping factor is always positive, we may conclude that the loss-gradient itself is zero on an average. By "average", we mean that an expectation taken with respect to the data distribution μ. When the loss-gradient is zero, we can conclude that in most circumstances the algorithm has converged to a local minimum of the training loss function.

5 CONCLUSIONS

In this paper, we considered the method of gradient clipping used to control the exploding gradients problem in deep learning. We discussed the classic and adaptive variants of norm-based clipping. For the classic clipping method, we showed that it is numerically stable with high probability. It further converges to a local minimum of the average loss function. In the case of adaptive clipping, we observed that the updates are in the order of the NN weights. Due to the linear growth of the update as a function of the NN weights, one cannot guarantee stability without additional assumptions. However, we observed that we may dip into the theory of linear dynamical systems in order to ensure stability in the adaptive clipping scheme. Once numerical stability is guaranteed, the adaptive clipping method converges, as before, to a local minimizer of the average training loss. It must be noted that the averaging in both variants is with respect to the distribution that generated the training data.

REFERENCES

Aubin, J.-P. and Cellina, A. (2012). *Differential inclusions: set-valued maps and viability theory*, volume 264. Springer Science & Business Media.

Benaïm, M., Hofbauer, J., and Sorin, S. (2005). Stochastic approximations and differential inclusions. *SIAM Journal on Control and Optimization*, 44(1):328–348.

Bercu, B., Delyon, B., and Rio, E. (2015). *Concentration inequalities for sums and martingales*. Springer.

Bishop, C. M. and Nasrabadi, N. M. (2006). *Pattern recognition and machine learning*, volume 4. Springer.

Borkar, V. S. (2009). *Stochastic approximation: a dynamical systems viewpoint*, volume 48. Springer.

Boyd, S. and Vandenberghe, L. (2004). *Convex optimization*. Cambridge university press.

Durrett, R. (2018). *Stochastic calculus: a practical introduction*. CRC press.

Durrett, R. (2019). *Probability: theory and examples*, volume 49. Cambridge university press.

Halmos, P. R. (2017). *Finite-dimensional vector spaces*. Courier Dover Publications.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436–444.

Ramaswamy, A. and Bhatnagar, S. (2017). A generalization of the borkar-meyn theorem for stochastic recursive inclusions. *Mathematics of Operations Research*, 42(3):648–661.

Ramaswamy, A. and Bhatnagar, S. (2018). Stability of stochastic approximations with "controlled markov" noise and temporal difference learning. *IEEE Transactions on Automatic Control*, 64(6):2614–2620.

Rehmer, A. and Kroll, A. (2020). On the vanishing and exploding gradient problem in gated recurrent units. *IFAC-PapersOnLine*, 53(2):1243–1248.

Rudin, W. et al. (1976). *Principles of mathematical analysis*, volume 3. McGraw-hill New York.

Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural networks*, 61:85–117.

Vu, T. and Raich, R. (2022). On asymptotic linear convergence of projected gradient descent for constrained least squares. *IEEE Transactions on Signal Processing*, 70:4061–4076.

Zhang, J., He, T., Sra, S., and Jadbabaie, A. (2019). Why gradient clipping accelerates training: A theoretical justification for adaptivity. In *International Conference on Learning Representations*.