ON THE CONVERGENCE OF SGD, ADAM, & AMSGRAD

Hassan Dbouk
Department of Electrical and Computer Engineering
University of Illinois at Urbana-Champaign
Urbana, IL 61801, USA
hdbouk2@illinois.edu

1 INTRODUCTION

The tremendous success of Deep Learning over the pass decade or so has often been credited to the work of Krizhevsky et al. (2012). In their work, they demonstrated how a deep network can be trained efficiently using two GPUs. However, the main reason why training deep networks is relatively easy to parralelize in hardware is due to the simplicity of the Stochastic Gradient Descent (SGD) algorithm (Bottou (2010)).
Despite being easy to implement, the SGD algorithm requires a lot of tuning for hyper-parameters (such as step size schedule) in order for training to converge for large networks where the loss function is high dimensional and highly non-convex. To facilitate the training process, several works have tried to implement optimizers with adaptive learning rates. That is, the learning rate is adjusted dynamically during training.
ADAM, which is derived from adaptive moment estimate, is a recent and popular adaptive optimizer proposed by Kingma & Ba (2015). Empirically it has been shown that ADAM accelerates training of deep networks, compared to SGD, with little overhead in terms of computation. Despite its success, the convergence analysis used for analyzing the ADAM algorithm is actually wrong, as discovered by various sources, including a very recent paper by Reddi et al. (2019).

Reddi et al. (2019) further suggest slight modifications to the ADAM algorithm, which they call AMSGRAD. They also analyze the convergence of AMSGRAD and show that it converges.

2 ERM SETUP

Assume the following notation:

- $\mathsf{X} \subseteq \mathbb{R}^d$ a subset of $d-$dimensional euclidean space.
- Z is a set of tuples z of the form $z = (u, v)$.
- P_z be a probability distribution on $z \in \mathsf{Z}$.
- $f : \mathsf{X} \times \mathsf{Z} \to \mathbb{R}$, is a cost function taking in the parameter vector x and a sample z.

The main idea for using gradient based algorithms is to solve the optimization problem of the form:

$$x^* = \arg\min_{x \in \mathsf{X}} \mathbb{E}_z \left[f(x, z) \right]$$

Where $f(x, z)$ is a cost function for using the parameter $x \in \mathsf{X}$ on a sample $z \in \mathsf{Z}$, following some distribution P_z. In practice, the distribution of z is unknown, but we are usually given a set of N training samples $\{z\}_{n=1}^N$, where z_n are i.i.d. following P_z. Thus the optimization problem becomes an empirical risk minimization (ERM) problem:

$$x^* = \arg\min_{x \in \mathsf{X}} \frac{1}{N} \sum_{n=1}^{N} f(x, z_n)$$

The variable z can be thought of as a tuple (u, v) where u is the feature and v is the label or output that needs to be predicted from u, via a predictor paremeterized by x, and the "loss" of using the

predictor on a sample $z = (u, v)$ is $f(x, z)$.
Assuming that the loss function is convex and differentiable with respect to the parameter x, the gradient of $f(x)$ with respect to x: $\nabla_x f(x)$ is unique and SGD can be used trivially.

3 Online Function Optimization as ERM

The online function minimization framework presented in Zinkevich (2003) has been very popular for studying the convergence behaviour of gradient based optimizers for solving ERM problems. In the online optimization framework, we usually assume the following:

- X has a bounded diameter: $\|x_1 - x_2\|_2 \leq D$ & $\|x_1 - x_2\|_\infty \leq D_\infty, \forall x_1, x_2 \in \mathsf{X}$.
- $f_t : \mathsf{X} \to \mathbb{R}$, $\forall t \in [T]$, is a sequence of convex functions with bounded gradients: $\|\nabla_x f_t(x)\|_2 \leq G$ & $\|\nabla_x f_t(x)\|_\infty \leq G_\infty$.

In the online optimization framework, the goal is to find the sequence $x_t \in \mathsf{X}$ to minimize the total cost incurred for T iterations:

$$J(T) = \sum_{t=1}^{T} f_t(x_t)$$

The catch is that, at each iteration t, the algorithm has access to only the $1, ..., t-1$ functions, so $x_t = A(f_1, f_2, ..., f_{t-1}, x_1, x_2, ..., x_{t-1})$, but the loss for choosing x_t is evaluated at $f_t(x_t)$. Because of this setup, it is often useful to assess the performance of an algorithm by looking at the total regret compared to the best fixed strategy. That is, we compare the cost of running an algorithm A sequentially to that of an offline algorithm that has access to all the T functions, and can choose a fixed solution. The regret can be written as:

$$R(T) = J(T) - \inf_{x^* \in \mathsf{X}} \sum_{t=1}^{T} f_t(x^*)$$

To relate this framework to the ERM framework, we first observe that the sequence of functions $f_t(x)$ can be thought of as a function f evaluated on a sequence of z_t and a parameter vector x: $f(x, z_t) = f_{z_t}(x) = f_t(x)$. Moreover, Cesa-Bianchi et al. (2004) have shown that an online learning algorithm with an average regret $R(T)/T \to 0$ as $T \to \infty$ yields a stochastic optimization algorithm for the ERM problem, which is also seen in class under the generalization of online learning algorithms section. This result essentially allows us to use the online learning framework to prove the convergence of an optimizer in the ERM setup, which is an attractive mathematical tool.

4 SGD

4.1 Algorithm

The SGD algorithm is described in Algorithm (1). The projection operation is used to force the requirement that $x_t \in \mathsf{X}$. Implementation of SGD is very straightforward, and is heavily deployed.

x_1: initial parameter vector;
for *t = 1, 2, 3, ..., T-1* **do**
  $g_t \leftarrow \nabla_x f_t(x_t)$;
  $x_{t+1} \leftarrow \prod(x_t - \alpha_t g_t)$;
end

Algorithm 1: SGD Algorithm

4.2 Regret Bound

As seen in class, and based on the result in Zinkevich (2003), the regret for running SGD after T iterations with a step size $\alpha_t = \frac{\alpha}{\sqrt{t}}$ is upper bounded as follows:

$$R(T) \leq \frac{D^2\sqrt{T}}{2\alpha} + (\sqrt{T} - \frac{1}{2})G^2\alpha$$

Choosing $\alpha = \frac{D}{G\sqrt{2}}$ yields:

$$R(T) \leq DG\sqrt{2T}$$

Thus we have:

$$\lim_{T\to\infty} \frac{R(T)}{T} = 0$$

Therefore $R(T) = o(T)$.
The proof of the bound is explained in the notes, and therefore for brevity I won't rewrite it here.

5 ADAM

5.1 ALGORITHM

The ADAM algorithm is described in Algorithm (2). Note that vector operations are element wise

x_1: initial parameter vector;
$m_0 = v_0 = 0$: initial moments;
$\beta_1, \beta_2 \in [0, 1)$;
for *t = 1, 2, 3, ..., T-1* **do**
  $g_t \leftarrow \nabla_x f_t(x_t)$;
  $m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1)g_t$;
  $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2)g_t^2$;
  $\hat{m}_t \leftarrow m_t/(1 - \beta_1^t)$;
  $\hat{v}_t \leftarrow v_t/(1 - \beta_2^t)$;
  $x_{t+1} \leftarrow \prod(x_t - \alpha_t \hat{m}_t/\sqrt{\hat{v}_t})$;
end

Algorithm 2: ADAM Algorithm

operations. The normalization term for both first and second moment estimates are there to ensure that the estimates are unbiased. To see that, let us unroll the recursion for m_t. Using the fact that $m_0 = 0$, we can write:

$$m_t = (1 - \beta_1) \sum_{k=1}^{t} \beta_1^{t-k} g_k$$

Taking the expectation of both sides (with respect to g_t), we get:

$$\mathbb{E}[m_t] = (1 - \beta_1) \sum_{k=1}^{t} \beta_1^{t-k} \mathbb{E}[g_k]$$

Assuming that $\mathbb{E}[g_k] = \mathbb{E}[g]$, which means that g_t is stationary, we get:

$$\mathbb{E}[m_t] = \mathbb{E}[g](1 - \beta_1) \sum_{k=1}^{t} \beta_1^{t-k} = \mathbb{E}[g](1 - \beta_1)\frac{1 - \beta_1^t}{1 - \beta_1} = \mathbb{E}[g](1 - \beta_1^t)$$

Therefore, to construct an unbiased estimate of the mean, we divide m_t by $1 - \beta_1^2$ as indicated in Algorithm (2). Note that following the same exact line of reasoning, the second moment estimate v_t should be divided by $1 - \beta_2^2$ in order for it to be unbiased.
Intuitively, what ADAM does is that for every parameter dimension $i \in [d]$ an effective learning rate of $\alpha_t/\sqrt{\hat{v}_{t,i}}$ is used to scale the gradient estimate $\hat{m}_{t,i}$. If $g_{t,i}^2$ is large, that means the gradient estimate is not "reliable" and hence a small learning rate is used because of the lack of confidence in the gradient estimate. In practice, ADAM has shown to be a very simple optimizer to use, with little to no tuning required. The recommended setting of the hyperparameters is $\beta_1 = 0.9$ and $\beta_2 = 0.99$, with a fixed step size α typically around 10^{-2}.

5.2 REGRET BOUND

In their paper, Kingma & Ba (2015) claim that under the following assumptions:

- $\beta_1, \beta_2 \in [0, 1)$ and $\beta_1^2/\sqrt{\beta_2} < 1$
- $\alpha_t = \alpha/\sqrt{t}$
- $\beta_{1,t} = \beta_1 \lambda^t$, for some $\lambda \in (0, 1)$
- $\gamma = \frac{\beta_1^2}{\sqrt{\beta_2}}$

the regret for running ADAM for T iterations is upper bounded by:

$$R(T) \leq \frac{D^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} \sqrt{T\hat{v}_{T,i}} + \frac{\alpha(1+\beta_1)G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2} \sum_{i=1}^{d} \|g_{1:T,i}\|_2 + \sum_{i=1}^{d} \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\alpha(1-\beta_1)(1-\lambda)^2}$$

Where $g_{1:T} = [g_1, g_2, ..., g_T]$ and $g_{1:T,i}$ is the vector of the i^th coordinate: $g_{1:T,i} = [g_{1,i}, g_{2,i}, ..., g_{T,i}]$. Using the fact that:

$$\|g_{1:T,i}\|_2 = \sqrt{\sum_{t=1}^{T} g_{t,i}^2} \leq \sqrt{\sum_{t=1}^{T} G_\infty^2} = G_\infty \sqrt{T}$$

Thus Kingma & Ba (2015) claim that $\frac{R(T)}{T} = O(\frac{1}{\sqrt{T}}) \to 0$ as $T \to \infty$. The claim is mathematically incorrect, for a number of reasons. The proof used by Kingma & Ba (2015) has some mistakes, which we will point out in section (7).

5.3 COUNTER EXAMPLE

In a recent paper by Reddi et al. (2019), an online optimization counter example is shown where ADAM fails to converge to the right solution. In fact, the paper proves that, for any constant $\beta_1, \beta_2 \in [0, 1)$ such that $\beta_1^2 < \sqrt{\beta_2}$, there is an online convex optimization problem where ADAM has non-zero average regret i.e., $R(T)/T \nrightarrow 0$ as $T \to \infty$.
The proof assumes that f_t are linear functions over $\mathsf{X} = [-1, 1]$, specifically we have:

$$f_t(x) = \begin{cases} Cx & \text{if } t \mod C = 1 \\ -x & \text{otherwise} \end{cases}$$

for some $C \in \mathbb{N}, C > 2$. Note that the value $x^* = -1$ is the optimal fixed strategy one can use after observing T functions. Assume that $T = nC$, we have:

$$\sum_{t=1}^{T} f_t(x) = \sum_{t=1}^{nC} f_t(x) = n(f_1(x) + f_2(x) + ... + f_C(x)) = n(Cx - (C-1)x) = nx$$

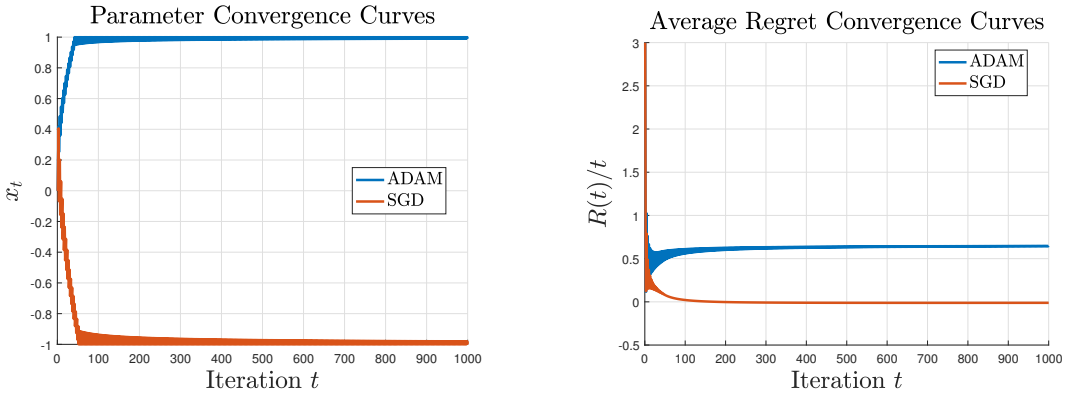
Which is minimized when $x^* = -1$.
The main idea behind the proof of the counter example is that the gradient sequence $g_t = g_t(x) = \nabla_x f_t(x)$ becomes:

$$\nabla_x f_t(x) = \begin{cases} C & \text{if } t \mod C = 1 \\ -1 & \text{otherwise} \end{cases}$$

And then using mathematical induction, they show that using the ADAM update equation, the following inequality holds:

$$x_{t+C} \geq \min\{1, x_t + \frac{\lambda}{\sqrt{t}}\}$$

for some $\lambda > 0$ independent of t. Intuitively, this means there will exist t such that $t \mod C = 0$ with $x_t = 1$, due to the divergence of the sum $\sum \frac{1}{\sqrt{t}}$. Moreover, this implies that ADAM will suffer a regret of at least 2 every C steps, therefore $R(T)/T \nrightarrow 0$ as $T \to \infty$.
In order to verify that ADAM will actually not converge properly in this setup, a simulation is done. For simplicity, we chose $C = 3, \beta_1 = 0, \beta_2 = 0.1$, and $\alpha_t = 0.3/\sqrt{t}$. Figures 1a & 1b show the



(a) Parameter Convergence x_t

(b) Regret Convergence $R(t)/t$

Figure 1: Convergence of SGD vs ADAM

simulation result. For comparison, we also run the SGD algorithm with similar stepsize α_t. Figure 1a shows that the solution x_t will converge to the optimal solution $x^* = -1$ in the case of SGD. However, for ADAM it will converge to the highly un-optimal solution of $x' = 1$. Consequently, Figure 1b shows that the average regret of SGD converges to 0, as opposed to that of ADAM, which converges to some value larger than 0.

6 AMSGRAD

6.1 ALGORITHM

Reddi et al. (2019) propose their own version of ADAM, with some modifications. They call their algorithm AMSGRAD, which is described in Algorithm (3).

x_1: initial parameter vector;
$m_0 = v_0 = \hat{v}_0 = 0$: initial moments;
$\beta_1, \beta_2 \in [0, 1)$;
for *t = 1, 2, 3, ..., T-1* **do**
  $g_t \leftarrow \nabla_x f_t(x_t)$;
  $m_t \leftarrow \beta_{1,t} m_{t-1} + (1 - \beta_{1,t}) g_t$;
  $v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$;
  $\hat{v}_t = \max(v_t, \hat{v}_{t-1})$;
  $x_{t+1} \leftarrow \prod(x_t - \alpha_t m_t / \sqrt{\hat{v}_t})$;
end

Algorithm 3: AMSGRAD Algorithm

Again, the vector operations are element wise operations. The major difference between AMSGRAD and ADAM is the auxiliary second moment estimate $\hat{v}_t = \max(v_t, \hat{v}_{t-1})$. This insures that for every dimension $i \in [d]$:

$$\frac{\sqrt{\hat{v}_{t+1}}}{\alpha_{t+1}} - \frac{\sqrt{\hat{v}_t}}{\alpha_t} \geq 0$$

Which insures a non increasing learning rate for AMSGRAD. Reddi et al. (2019) claim that the above inequality is assumed to be true for ADAM by Kingma & Ba (2015). Although the update rules of ADAM do not guarantee such inequality, the proof presented in Kingma & Ba (2015) and explained in section (7) does not assume the inequality to be true.

6.2 REGRET BOUND

Reddi et al. (2019) claim that under the following assumptions:

- $\beta_1, \beta_2 \in [0, 1)$ and $\beta_1^2 / \sqrt{\beta_2} < 1$

- $\alpha_t = \alpha/\sqrt{t}$
- $\beta_{1,t} = \beta_1 \lambda^t$, for some $\lambda \in (0, 1)$
- $\gamma = \frac{\beta_1}{\sqrt{\beta_2}}$

the regret for running AMSGRAD for T iterations is upper bounded by:

$$R(T) \leq \frac{D_\infty^2}{2\alpha(1-\beta_1)} \sum_{i=1}^{d} \sqrt{T\hat{v}_{T,i}} + \frac{\beta_1 D_\infty^2 G_\infty}{2(1-\beta_1)(1-\lambda)^2} + \frac{\alpha\sqrt{1+\log T}}{(1-\beta_1)^2(1-\gamma)\sqrt{1-\beta_2}} \sum_{i=1}^{d} \|g_{1:T,i}\|_2$$

Where $g_{1:T} = [g_1, g_2, ..., g_T]$ and $g_{1:T,i}$ is the vector of the i^th coordinate: $g_{1:T,i} = [g_{1,i}, g_{2,i}, ..., g_{T,i}]$. Again, we have:

$$\|g_{1:T,i}\|_2 = \sqrt{\sum_{t=1}^{T} g_{t,i}^2} \leq \sqrt{\sum_{t=1}^{T} G_\infty^2} = G_\infty \sqrt{T}$$

which implies that $R(T) = o(T)$, and hence AMSGRAD converges.

7 ADAM REGRET BOUND PROOF

In this section, we present the proof for the regret bound of ADAM claimed by Kingma & Ba (2015). As previously mentioned, the proof contains a few mistakes, which we will highlight.
Recall that the regret after T iterations is:

$$R(T) = \sum_{t=1}^{T} f_t(x_t) - \inf_{x^* \in \mathsf{X}} \sum_{t=1}^{T} f_t(x^*)$$

The main idea of the proof is to use the convexity of $f_t(x)$:

$$f_t(x_t) - f_t(x^*) \leq \nabla f_t(x_t)(x_t - x^*) = \sum_{i=1}^{d} g_{t,i}(x_{t,i} - x_{,i}^*)$$

If $g_{t,i}(x_{t,i} - x_{,i}^*)$ can be upper bounded, then summing over $t \in [T]$ can upper bound the regret:

$$R(T) \leq \sum_{t=1}^{T} \sum_{i=1}^{d} g_{t,i}(x_{t,i} - x_{,i}^*)$$

In order to bound $g_{t,i}(x_{t,i} - x_{,i}^*)$, we start by using the update rule of ADAM:

$$\begin{aligned} x_{t+1} &= x_t - \alpha_t \frac{\hat{m}_t}{\sqrt{\hat{v}_t}} \\ &= x_t - \frac{\alpha_t}{1-\beta_1^t} \left(\frac{\beta_{1,t} m_{t-1} + (1-\beta_{1,t}) g_t}{\sqrt{\hat{v}_t}}\right) \end{aligned}$$

Focusing on the i^{th} dimension, subtracting $x_{,i}^*$ from both sides and taking the square:

$$(x_{t+1,i} - x_{,i}^*)^2 = (x_{t,i} - x_{,i}^*)^2 - \frac{2\alpha_t}{1-\beta_1^t} \left(\frac{\beta_{1,t}}{\sqrt{\hat{v}_{t,i}}} m_{t-1,i} + \frac{1-\beta_{1,t}}{\sqrt{\hat{v}_{t,i}}} g_{t,i}\right)(x_{t,i} - x_{,i}^*) + \alpha_t^2 \left(\frac{\hat{m}_{t,i}}{\sqrt{\hat{v}_{t,i}}}\right)^2$$

Re arranging the terms we get:

$$\begin{aligned} \frac{2\alpha_t(1-\beta_{1,t})}{(1-\beta_1^t)\sqrt{\hat{v}_{t,i}}} g_{t,i}(x_{t,i} - x_{,i}^*) &= (x_{t,i} - x_{,i}^*)^2 - (x_{t+1,i} - x_{,i}^*)^2 \\ &\quad - \frac{2\alpha_t \beta_{1,t}}{(1-\beta_1^t)\sqrt{\hat{v}_{t,i}}} m_{t-1,i}(x_{t,i} - x_{,i}^*) \\ &\quad + \alpha_t^2 \left(\frac{\hat{m}_{t,i}}{\sqrt{\hat{v}_{t,i}}}\right)^2 \end{aligned}$$

Diving both sides by $\frac{2\alpha_t(1-\beta_{1,t})}{(1-\beta_1^t)\sqrt{\hat{v}_{t,i}}}$:

$$\begin{aligned} g_{t,i}(x_{t,i}-x^*_{,i}) &= \frac{(1-\beta_1^t)\sqrt{\hat{v}_{t,i}}}{2\alpha_t(1-\beta_{1,t})}\big((x_{t,i}-x^*_{,i})^2-(x_{t+1,i}-x^*_{,i})^2\big) \\ &- \frac{\beta_{1,t}}{(1-\beta_{1,t})}m_{t-1,i}(x_{t,i}-x^*_{,i}) \\ &+ \frac{\alpha_t(1-\beta_1^t)\sqrt{\hat{v}_{t,i}}}{2(1-\beta_{1,t})}(\frac{\hat{m}_{t,i}}{\sqrt{\hat{v}_{t,i}}})^2 \end{aligned}$$

The term $(x_{t,i}-x^*_{,i})$ in the RHS is annoying to deal with, especially when we end up summing over the dimension d. Using the identity $ab \leq a^2/2 + b^2/2$, with:

$$a = \frac{(\hat{v}_{t-1,i})^{\frac{1}{4}}}{\sqrt{\alpha_{t-1}}}(x^*_{,i}-x_{t,i}) \quad \& \quad b = \frac{\sqrt{\alpha_{t-1}}}{(\hat{v}_{t-1,i})^{\frac{1}{4}}}m_{t-1,i}$$

We have:

$$\frac{\beta_{1,t}}{(1-\beta_{1,t})}m_{t-1,i}(x^*_{,i}-x_{t,i}) \leq \frac{\beta_{1,t}}{(1-\beta_{1,t})}\Big(\frac{\sqrt{\hat{v}_{t-1,i}}}{2\alpha_{t-1}}(x^*_{,i}-x_{t,i})^2 + \frac{\alpha_{t-1}}{2\sqrt{\hat{v}_{t-1,i}}}m^2_{t-1,i}\Big)$$

Plugging back in the previous inequality, we get:

$$\begin{aligned} g_{t,i}(x_{t,i}-x^*_{,i}) &\leq \frac{(1-\beta_1^t)\sqrt{\hat{v}_{t,i}}}{2\alpha_t(1-\beta_{1,t})}\big((x_{t,i}-x^*_{,i})^2-(x_{t+1,i}-x^*_{,i})^2\big) \\ &+ \frac{\beta_{1,t}}{(1-\beta_{1,t})}\Big(\frac{\sqrt{\hat{v}_{t-1,i}}}{2\alpha_{t-1}}(x^*_{,i}-x_{t,i})^2 + \frac{\alpha_{t-1}}{2\sqrt{\hat{v}_{t-1,i}}}m^2_{t-1,i}\Big) \\ &+ \frac{\alpha_t(1-\beta_1^t)\sqrt{\hat{v}_{t,i}}}{2(1-\beta_{1,t})}(\frac{\hat{m}_{t,i}}{\sqrt{\hat{v}_{t,i}}})^2 \end{aligned}$$

Plugging $\alpha_t = \alpha/\sqrt{t}$, summing over $i \in [d]$ and $t \in [T]$, using Lemma (2) and the fact that $\beta_1^t \leq \beta_1 < 1$ and $\frac{1}{1-\beta_{1,t}} \leq \frac{1}{1-\beta_1}$ we have:

$$\begin{aligned} R(T) &\leq \sum_{i=1}^{d}\frac{1}{2\alpha(1-\beta_1)}(x_{1,i}-x^*_{,i})^2\sqrt{\hat{v}_{1,i}} + \sum_{i=1}^{d}\sum_{t=2}^{T}\frac{1}{2(1-\beta_1)}(x_{t,i}-x^*_{,i})^2(\frac{\sqrt{\hat{v}_{t,i}}}{\alpha_t} - \frac{\sqrt{\hat{v}_{t-1,i}}}{\alpha_{t-1}}) \\ &+ \frac{\beta_1\alpha G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2}\sum_{i=1}^{d}\|g_{1:T,i}\|_2 + \frac{\alpha G_\infty}{(1-\beta_1)\sqrt{1-\beta_2}(1-\gamma)^2}\sum_{i=1}^{d}\|g_{1:T,i}\|_2 \\ &+ \sum_{i=1}^{d}\sum_{t=1}^{T}\frac{\beta_{1,t}}{2\alpha_t(1-\beta_{1,t})}(x^*_{,i}-x_{t,i})^2\sqrt{\hat{v}_{t,i}} \end{aligned}$$

Note that, as explained later, the proof of Lemma (2) is not correct, and therefore the above bound is not correct. What is interesting is that Reddi et al. (2019) claim that the proof for ADAM assumes the following inequality:

$$\frac{\sqrt{\hat{v}_{t+1}}}{\alpha_{t+1}} - \frac{\sqrt{\hat{v}_t}}{\alpha_t} \geq 0$$

However, using the bounded diameter assumption, we have $(x_{t,i}-x^*_{,i})^2 \leq D^2_\infty$, therefore the term $\frac{\sqrt{\hat{v}_{t+1}}}{\alpha_{t+1}} - \frac{\sqrt{\hat{v}_t}}{\alpha_t}$ will telescope, without any assumption of it being positive.
On the other hand, the proof assumes that X has a bounded diameter, but the original ADAM algorithm presented by Kingma & Ba (2015) does not involve a projection operation, which seems weird.

7.1 LEMMA 1

Kingma & Ba (2015) claim that:

$$\sum_{t=1}^{T} \sqrt{\frac{g_{t,i}^2}{t}} \leq 2G_\infty \|g_{1:T,i}\|_2$$

They attempt to prove it using mathematical induction on T. However the proof is actually flawed, as we shall see.
First consider the base case $T = 1$, we have:

$$\sqrt{g_{1,i}^2} = \|g_{1,i}\|_2 = |g_{1,i}| \leq G_\infty$$

However, the paper claims that the base case holds:

$$\sqrt{g_{1,i}^2} = \|g_{1,i}\|_2 = |g_{1,i}| \leq 2G_\infty \|g_{1,i}\|_2$$

which holds only if $G_\infty > 0.5$, and that is not stated by Kingma & Ba (2015). Furthermore, examining the inductive step also uncovers another mistake for proving this lemma. To perform the inductive step, assume that the inequality holds for $T = K - 1$, then we have:

$$\sum_{t=1}^{K-1} \sqrt{\frac{g_{t,i}^2}{t}} \leq 2G_\infty \|g_{1:K-1,i}\|_2$$

Need to prove that it holds for $T = K$:

$$\begin{aligned}
\sum_{t=1}^{K} \sqrt{\frac{g_{t,i}^2}{t}} &= \sum_{t=1}^{K-1} \sqrt{\frac{g_{t,i}^2}{t}} + \sqrt{\frac{g_{K,i}^2}{K}} \\
&\leq 2G_\infty \|g_{1:K-1,i}\|_2 + \sqrt{\frac{g_{K,i}^2}{K}} \\
&= 2G_\infty \sqrt{\sum_{t=1}^{K-1} g_{t,i}^2 + g_{K,i}^2 - g_{K,i}^2} + \sqrt{\frac{g_{K,i}^2}{K}} \\
&= 2G_\infty \sqrt{\|g_{1:K,i}\|_2^2 - g_{K,i}^2} + \sqrt{\frac{g_{K,i}^2}{K}}
\end{aligned}$$

Using the inequality:

$$\|g_{1:K,i}\|_2^2 - g_{K,i}^2 \leq \|g_{1:K,i}\|_2^2 - g_{K,i}^2 + \frac{g_{K,i}^4}{4\|g_{1:K,i}\|_2^2} = (\|g_{1:K,i}\|_2 - \frac{g_{K,i}^2}{2\|g_{1:K,i}\|_2})^2$$

We get:

$$\begin{aligned}
\sum_{t=1}^{K} \sqrt{\frac{g_{t,i}^2}{t}} &\leq 2G_\infty (\|g_{1:K,i}\|_2 - \frac{g_{K,i}^2}{2\|g_{1:K,i}\|_2}) + \sqrt{\frac{g_{K,i}^2}{K}} \\
&\leq 2G_\infty (\|g_{1:K,i}\|_2 - \frac{g_{K,i}^2}{2\sqrt{KG_\infty^2}}) + \sqrt{\frac{g_{K,i}^2}{K}} \\
&= 2G_\infty \|g_{1:K,i}\|_2 - \frac{g_{K,i}^2}{\sqrt{K}} + \sqrt{\frac{g_{K,i}^2}{K}}
\end{aligned}$$

Following the proof in Kingma & Ba (2015), they claim that:

$$2G_\infty \|g_{1:K,i}\|_2 - \frac{g_{K,i}^2}{\sqrt{K}} + \sqrt{\frac{g_{K,i}^2}{K}} \leq 2G_\infty \|g_{1:K,i}\|_2$$

If that were true, then the inductive step would hold. However, there is no guarantee that the above inequality holds. Equivalently the claim can be written as:

$$g_{K,i}^2 \leq |g_{K,i}|$$

which only holds if $|g_{K,i}| \leq 1$.
Therefore Lemma 1, which is used in the proof for Lemma (2), does not hold.

7.2 LEMMA 2

Kingma & Ba (2015) claim that:

$$\sum_{t=1}^{T} \frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{v_{t,i}}}} \leq \frac{2}{1-\gamma} \frac{1}{\sqrt{1-\beta_2}} \|g_{1:T,i}\|_2$$

where $\gamma \triangleq \frac{\beta_1^2}{\sqrt{\beta_2}}$ such that $\gamma < 1$. Recall that:

$$\hat{m}_t = \frac{(1-\beta_1)}{(1-\beta_1^t)} \sum_{k=1}^{t} \beta_1^{t-k} g_k$$

$$\hat{v}_t = \frac{(1-\beta_2)}{(1-\beta_2^t)} \sum_{k=1}^{t} \beta_2^{t-k} g_k^2$$

To prove the claim, we start by loooking at the summand:

$$\frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{v_{t,i}}}} = \frac{\sqrt{1-\beta_2^t}}{(1-\beta_1^t)^2} \frac{(\sum_{k=1}^{t}(1-\beta_1)\beta_1^{t-k} g_{k,i})^2}{\sqrt{t\sum_{j=1}^{t}(1-\beta_2)\beta_2^{t-j} g_{j,i}^2}}$$

which implies:

$$\begin{aligned}
\frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{v_{t,i}}}} &\leq \frac{\sqrt{1-\beta_2^t}}{(1-\beta_1^t)^2} \sum_{k=1}^{t} \frac{t((1-\beta_1)\beta_1^{t-k} g_{k,i})^2}{\sqrt{t\sum_{j=1}^{t}(1-\beta_2)\beta_2^{t-j} g_{j,i}^2}} \\
&\leq \frac{\sqrt{1-\beta_2^t}}{(1-\beta_1^t)^2} \sum_{k=1}^{t} \frac{t((1-\beta_1)\beta_1^{t-k} g_{k,i})^2}{\sqrt{t(1-\beta_2)\beta_2^{t-k} g_{k,i}^2}} \\
&= \frac{t(1-\beta_1)^2}{\sqrt{t(1-\beta_2)}} \frac{\sqrt{1-\beta_2^t}}{(1-\beta_1^t)^2} \sum_{k=1}^{t} \Big(\frac{\beta_1^2}{\sqrt{\beta_2}}\Big)^{t-k} |g_{k,i}|
\end{aligned}$$

Using the assumptions, we have $\frac{\sqrt{1-\beta_2^t}}{(1-\beta_1^t)^2} \leq \frac{1}{(1-\beta_1)^2}$, therefore we can write:

$$\frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{v_{t,i}}}} \leq \frac{t}{\sqrt{t(1-\beta_2)}} \sum_{k=1}^{t} \gamma^{t-k} |g_{k,i}|$$

Summing over $t \in [T]$ we get:

$$\sum_{t=1}^{T} \frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{v_{t,i}}}} \leq \sum_{t=1}^{T} \frac{t}{\sqrt{t(1-\beta_2)}} \sum_{k=1}^{t} \gamma^{t-k} |g_{k,i}|$$

However, Kingma & Ba (2015) claim that the resultant upper bound should be:

$$\sum_{t=1}^{T} \frac{\hat{m}_{t,i}^2}{\sqrt{t\hat{v_{t,i}}}} \leq \sum_{t=1}^{T} \frac{|g_{t,i}|}{\sqrt{t(1-\beta_2)}} \sum_{j=0}^{T-t} t\gamma^j$$

which is not straightforward to derive from the previous inequality, and most likely a mistake in the proof.
Assuming that the bound is true, using the upper bound on $\sum_t t\gamma^t < \frac{1}{(1-\gamma)^2}$ and using Lemma (1) (which is also false), then the proof of Lemma (2) is complete.

8 CONCLUSION & FUTURE WORK

In conclusion, adaptive learning rate based gradient optimizers are of much interest for training deep neural networks. ADAM, despite being popular amongst deep learning researches, need not

necessarily converge in convex settings, as shown by the counter example presented by Reddi et al. (2019). Furthermore, AMSGRAD is introduced in Reddi et al. (2019) which is a slightly modified version of ADAM that has guarantees on the convergence in convex settings.
Naturally, it would be interesting to see what extra constraints on the optimization problem can be constructed in order to insure that ADAM converges.
Another interesting direction is comparing the performance of SGD with ADAM and AMSGRAD in different convex settings. SGD with a proper stepsize schedule has always been favored by deep learning researchers, however finding this schedule is often very time consuming. If a proper stepsize schedule is given, will SGD outperform ADAM or AMSGRAD?

REFERENCES

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

Nicolo Cesa-Bianchi, Alex Conconi, and Claudio Gentile. On the generalization ability of on-line learning algorithms. *IEEE Transactions on Information Theory*, 50(9):2050–2057, 2004.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *Proceedings of 3rd International Conference on Learning Representations*, 2015.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pp. 1097–1105, 2012.

Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. *arXiv preprint arXiv:1904.09237*, 2019.

Martin Zinkevich. Online convex programming and generalized infinitesimal gradient ascent. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pp. 928–936, 2003.