Adaptive Optimisation Algorithms for Gradient Descent

Zella Baig

Machine Learning MT22

1 Introduction

Gradient Descent (GD) is an optimisation algorithm commonly used in machine learning in order to find a minimum of a function $F(\vec{x})$, such as in least squares regression. At its most basic, GD may be represented via the iterative scheme

$$\vec{x}_{t+1} = \vec{x}_t - \eta \vec{g}_t, \tag{1}$$

where $\vec{g}_t$ represents the gradient $\nabla F(\vec{x}_t)$, and η represents a hyperparameter known as the *learning rate*, i.e. the "stepsizes"[1] between successive $\vec{x}_t$. Due to the potentially large computational time in calculating the gradient for the entire vector $\vec{x}$ in large dimensions, we often use Stochastic Gradient Descent (SGD) where $\vec{g}_t$ is replaced by a randomly selected vector $\vec{G}_t$ such that $\mathbb{E}\left[\vec{G}_t\right] = \nabla F(\vec{x}_t)$. In practice $\vec{G}$ is chosen to be the gradient of a randomly selected element f_i of $F = \sum_i f_i$ as this is the simplest choice. In this report we use $\vec{g}$ to refer to both the actual and stochastic gradients, specifying the context when appropriate.

As choosing the appropriate value for the learning rate η can be tricky due to overshooting the minima or converging too slowly [15], there have been a host of adaptive algorithms designed in recent years which attempt to overcome this issue. We look at two methods in particular: *AdaGrad-Norm* [19], and *AdaLoss* [23]. We show that these methods are robust to choice of initial hyperparameters under certain assumptions, and further implement these and a third method *AMSGrad* [13] in both synthetic data as well as on the *California Housing Dataset* [17] to demonstrate convergence as a function of initial parameterisation, particularly versus SGD.

2 Overview of Methods

We define the schemes we analyse as

$$\textbf{AdaGrad-Norm:} \quad \vec{x}_{t+1} = \vec{x}_t - \frac{\eta}{b_{t+1}}\vec{g}_t, \qquad b_{t+1}^2 = b_t^2 + \left\|\vec{g}_t\right\|^2, \tag{2}$$

$$\textbf{AdaLoss:} \quad \vec{x}_{t+1} = \vec{x}_t - \frac{\eta}{b_{t+1}}\vec{g}_t, \qquad b_{t+1}^2 = b_t^2 + \alpha|f_i(\vec{x}_t) - c|. \tag{3}$$

where $||\cdot||$ is the ℓ_2 norm, $\eta, b_0 > 0$ are set at runtime, $t = 0, \cdots, N-1$, α is some scaling parameter for the stepsize[2], and c is a constant corresponding to the minimum of f_i.

[1]We use the terms stepsize and learning rate interchangeably to represent the prefactor for $\vec{g}_t$ in gradient descent-based methods.

[2]From now on set the scaling parameter $\alpha = 1$ in order to better compare AdaLoss to AdaGrad-Norm; the original paper introduces it as a parameter but this is *not* done in the paper introducing AdaGrad-Norm

Also note that η, the initial stepsize, is a constant unless we specify otherwise.

2.1 AdaGrad

AdaGrad-Norm and AdaLoss are both *Adaptive Gradient Methods* [23], a class of methods introduced via **AdaGrad** [5] as an extension to the standard (S)GD methods in use at the time. We briefly discuss AdaGrad and use this discussion to motivate our exploration of both AdaGrad-Norm and AdaLoss. There is some ambiguity in how one can define AdaGrad, but the method we refer to is the "coordinate" AdaGrad (named this way as the stepsize is updated on a per-element basis for elements in $\vec{w}$),

$$\vec{x}_{t+1} = \vec{x}_t - \frac{\eta}{\vec{b}_{t+1}} \vec{g}_t, \qquad [\vec{b}_i]^2_{t+1} = [\vec{b}_i]^2_t + [\vec{g}_i]^2. \tag{4}$$

In (4), note that the scaling vector $\vec{b}$ that we build up is updated coordinate by coordinate, as opposed to the constant factor that we see in AdaLoss or AdaGrad-Norm. Also note that we employ an abuse of notation where the factor of $1/\vec{b}_{t+1}$ refers to element-wise multiplication by the reciprocal of $[\vec{b}_i]_{t+1}$.

The intuition behind AdaGrad is similar to other adaptive stepsize methods wherein we desire greater resolution when the function F changes more rapidly; when F has a larger gradient we see that that the factor $[\vec{b}_i]_{t+1}$ increases faster. As each element of $\vec{g}_t$ is scaled by $\eta/[\vec{b}_i]_{t+1}$ we observe a smaller effective stepsize on indices where the gradient is large.

We now tackle the issue of convergence. For non-convex functions, a set of sufficient conditions for the convergence of stepsizes $\{\eta_t := \frac{\eta}{b_{t+1}}\}_{t=1}^T$ is that

$$\sum_{t=1}^{\infty} \eta_t = +\infty \qquad \text{and} \qquad \sum_{t=1}^{\infty} \eta_t^2 < +\infty \tag{5}$$

as shown in [14]. The stepsizes in (4) do not necessarily satisfy this condition as we may have $\vec{g}$ decreasing rapidly enough that the second sum in (5) diverges to $+\infty$ [9], so we work instead on bounding the gradients.

We first introduce several assumptions:

A1 We guarantee that $F(\vec{x})$ is bounded from below by some $F^*(\vec{x}) > -\infty$ for all $\vec{x} \in \mathbb{R}^d$, with d being the dimension we work in.

A2 The gradient is L-Lipschitz continuous: $\forall\ \vec{x}, \vec{y} \in \mathbb{R}^d, \quad \|\nabla F(\vec{x}) - \nabla F(\vec{y})\| \leq L\|\vec{x} - \vec{y}\|$ for some constant L.

when it could have been so we neglect its effect in this analysis.

A3 The gradients are bounded by a value γ: $\|\nabla F(\vec{x})\|_\infty^2 \leq \gamma^2$ for some γ, $\forall \vec{x} \in \mathbb{R}^d$.

Using these assumptions, as done in [4] we can then show that for non-convex yet smooth functions the scheme considered in (4) has convergence (for the norm of the gradient) of $O\left(ln(N)/\sqrt{N}\right)$, though we can tighten these bounds if we modify the scheme in (4) to have terms which incorporate weighted averages of previous stepsizes. We use these results to motivate convergence analysis of AdaGrad-Norm and AdaLoss for smooth, *convex* functions instead, specifically in the case of linear regression.

2.2 AdaGrad-Norm

We now move to discuss the stochastic form of AdaGrad-Norm which is very similar to (4) as discussed earlier. Along with assumption A2, for $k \geq 0$ we also assume that we have random vectors $\vec{\xi}_0, \ldots, \vec{\xi}_{N-1}$ which are linearly independent and that

$$\mathbb{E}\left[\left\|\vec{g}(\vec{x}, \vec{\xi}_k) - \nabla F(\vec{x})\right\|^2\right] \leq \sigma^2. \tag{6}$$

In (6) we have made explicit the dependence of the stochastic gradient on the random vectors $\vec{\xi}_k$; in reality (6) is just stating the idea that the norm of the difference of the stochastic gradient and the gradient itself is bounded by some variance σ^2. We also use a modified A3 to instead refer to the ℓ_2 norm of the gradient being bounded instead of the ∞ norm.

2.2.1 Least Squares Regression with AdaGrad-Norm

In order to build up to our implementation of the algorithm with real data, let us discuss AdaGrad-Norm in the context of Least Squares Regression. Consider the problem of linear regression expressed as:

$$F(\vec{w}) = \frac{1}{2m}\sum_{i=1}^{m}\left(\vec{X}_i^\top \vec{w} - y_i\right)^2 = \frac{1}{2m}\sum_{i=1}^{m} f_i(\vec{w}), \tag{7}$$

where we write F as a function of $\vec{w}$ in order to show that the problem is to find optimal weights $\vec{w}$ such that we minimise F given some inputs $\vec{X}$ and outputs $\vec{y}$. Here we have $\vec{X} \in \mathbb{R}^{m\times d}, \vec{y} \in \mathbb{R}^m$, and $\vec{w} \in \mathbb{R}^d$. Note that $\vec{X}_i$ represents row i in the matrix $\vec{X}$, and $\vec{X}^\top$ the transpose of $\vec{X}$. We define

$$\begin{aligned} \bar{\lambda}_m &> 0, \\ \bar{\lambda}_1 &> 0, \end{aligned}$$

as the smallest and largest singular values of the matrix $\vec{X}^\top\vec{X}$ respectively, scaled[3] by the factor of $1/m$. We pause to link these to the smoothness of $F(\vec{w}) = \frac{1}{2m}\left\|\vec{X}\vec{w} - \vec{y}\right\|^2$ and so

$$\begin{aligned}
\|\nabla F(\vec{w}) - \nabla F(\vec{v})\| &= \frac{1}{m}\left\|\vec{X}^\top\left(\vec{X}\vec{w} - \vec{y}\right) - \vec{X}^\top\left(\vec{X}\vec{v} - \vec{y}\right)\right\| \\
&= \frac{1}{m}\left\|\vec{X}^\top\vec{X}(\vec{w} - \vec{v})\right\| \\
&\le \frac{1}{m}\left\|\vec{X}^\top\vec{X}\right\|\|\vec{w} - \vec{v}\| \\
&\le \bar{\lambda}_1\|\vec{w} - \vec{v}\|. \qquad (8)
\end{aligned}$$

Here we have used the fact that the norm of a matrix is the largest singular value of the matrix. Thus, the smoothness of F (and thus all f_i) is bounded by $\bar{\lambda}_1$. We also introduce $\vec{w}^*$ which leads to the "optimal" (or minimal) value of $F(\vec{w})$; in our problem we thus have $\vec{X}\vec{w}^* = \vec{y}$ leading to $F(\vec{w}^*) = 0$. Note that this immediately implies that $c = 0$ in (3), and hence any further discussion of AdaLoss.

We now introduce the *Restricted Uniform Inequality of Gradients* (RUIG)[24]. We first use the form of F written as a sum over functions f_i, so that we can introduce the RUIG.

Definition 2.1 (RUIG)**.** The *Restricted Uniform Inequality of Gradients* states that if we consider all $\vec{w}$ within balls $D_\varepsilon := \left\{\vec{w} \in \mathbb{R}^d : \|\vec{w} - \vec{w}^*\|^2 > \varepsilon\right\}$ for all $\varepsilon > 0$, then we assume the existence of non-negative constants $\mu, \gamma \in \mathbb{R}$ such that $\mathbb{P}\left(\|\nabla f_i(\vec{w})\|^2 \ge \mu\|\vec{w} - \vec{w}^*\|^2\right) \ge \gamma$ for all $i > 0$.

In essence, the RUIG places a bound on the probability that the norm squared of the gradient of each of the f_i is greater than the squared distance between two vectors $\vec{w}, \vec{w}^*$.

We also introduce the *RUIL* [23].

Definition 2.2 (RUIL)**.** The *Restricted Uniform Inequality of Loss* states that if we consider all $\vec{w}$ within balls $D_\varepsilon := \left\{\vec{w} \in \mathbb{R}^d : \|\vec{w} - \vec{w}^*\|^2 > \varepsilon\right\}$ for all $\varepsilon > 0$, then we assume the existence of some non-negative constants $\mu, \gamma \in \mathbb{R}$ such that $\mathbb{P}\left(\left\langle \vec{X}_{\xi_i} \middle| \vec{w} - \vec{w}^*\right\rangle^2 \ge \mu\|\vec{w} - \vec{w}\|^2\right) \ge \gamma$ for all $i > 0$.

Here $\vec{X}_{\xi_i}$ is some stochastic vector (which in our scenario is a random row within $\vec{X}$), and we also use the "bra-ket" notation for an inner product[4]*.*

[3]This technicality has come about because we aim to replicate the form of Least Squares Regression used in [19].

[4]$\langle\vec{x}|\vec{y}\rangle = \langle\vec{x}, \vec{y}\rangle$. We will continue to use the bra-ket notation for the rest of this report.

This can be thought of as an analogue for the RUIG but for *loss in linear regression* instead of the norm of the gradient; note in particular how the RUIL mimics the RUIG how the stepsize update in AdaLoss mimics the stepsize update in AdaGrad-Norm.

We can now build up the tools necessary to prove stochastic convergence for AdaGrad-Norm in the manner of [23], whilst fleshing out aspects of their proof. We start with several lemmas.

Lemma 1. [5] *Under the assumption of RUIG for AdaGrad-Norm (or RUIL for AdaLoss), there exists $\varepsilon > 0$ such that after*

$$\tau := \left\lceil \frac{\eta^2 \bar{\lambda}_1^2 - b_0^2}{\mu\gamma\varepsilon} + \frac{\delta}{\gamma} \right\rceil + 1 \tag{9}$$

steps we have either

1. $b_\tau > \eta\bar{\lambda}_1$, *or*
2. $\min_t \left(\Delta_t^2 := \|\vec{\boldsymbol{w}}_t - \vec{\boldsymbol{w}}^*\|^2\right) \leq \varepsilon$

with probability $1-\delta_1$ where

$$\delta_1 := \exp\left(-\frac{\delta^2}{2(N\gamma(1-\gamma)+\delta)}\right),$$

and δ is a parameter which we can tune as we wish [24].

Proof. Under RUIG, define the random variables $Z_j, Z = \sum_j Z_j$ defined such that

$$Z_j = \begin{cases} 1 \text{ if } \left\|\boldsymbol{\nabla} f_{\xi_t}(\vec{\boldsymbol{w}}_t)\right\|^2 \geq \mu\Delta_t^2 > \varepsilon\mu \text{ (otherwise the proof is complete)}, \\ 0 \text{ otherwise.} \end{cases} \tag{10}$$

From RUIG we have that $\mathbb{P}\left(Z_j = 1\right) \geq \gamma$, and via Bernstein's Inequality[6] we sum over all Z_j, and apply the inequality to $\mathbb{P}(|Z - N\gamma| > \delta) \leq \delta_1$. We then write:

$$\begin{aligned} \mathbb{P}(|Z - N\gamma| > \delta) &\leq \delta_1, \\ \mathbb{P}(|N\gamma - Z| \leq \delta) &> 1 - \delta_1, \\ \mathbb{P}(Z \geq N\gamma - \delta) &> 1 - \delta_1. \end{aligned}$$

[5] We have corrected the result seen in Lemma A.1 in [23], by picking up a factor η not seen in their result.

[6] Bernstein's Inequality places an upper bound of the form $\mathbb{P}(|X - np| \geq t) \leq \exp\left(-t^2/(np - p^2 + bt)\right)$ up to some constant factors in the RHS [24].

Here we have used the idea that if $\mathbb{P}(X > a) \leq b$, then $\mathbb{P}(X \leq a) > 1 - b$. Using this result for Z, we now sum up

$$b_\tau^2 = b_0^2 + \sum_{t=0}^{\tau-1} \left\|\nabla f_{\xi_t}(\vec{w}_t)\right\|^2 > b_0^2 + \mu\varepsilon(\gamma N - \delta) \geq \eta^2 \bar{\lambda}_1^2, \tag{11}$$

where the result follows from rearranging for N to get at our value for τ after demanding an integer solution. □

We turn to Lemma 2.

Lemma 2. [7] *Considering AdaGrad-Norm, for* $k_0 \in (0, N-1]$ *s.t.* $\exists\ b_{k_0} > \eta\bar{\lambda}_1$ *and* $b_{k_0-1} < \eta\bar{\lambda}_1$, *we have that*

$$\Delta_{k_0-1}^2 \leq \Delta_0^2 + \eta^2\left(\ln\left(\frac{\eta^2\lambda_1^2}{b_0^2}\right) + 1\right). \tag{12}$$

Proof. Begin by noting

$$\Delta_{k_0-1}^2 = \left\|\vec{w}_{k_0-2} - \frac{\eta\vec{g}_{k_0-2}}{b_{k_0-1}} - \vec{w}^*\right\|^2 \tag{13}$$

$$= \left\|\vec{w}_{k_0-2} - \vec{w}^*\right\|^2 + \left\|\frac{\eta\vec{g}_{k_0-2}}{b_{k_0-1}}\right\|^2 - 2\left\langle\frac{\eta\vec{g}_{k_0-2}}{b_{k_0-1}}\middle|\vec{w}_{k_0-2} - \vec{w}^*\right\rangle. \tag{14}$$

Proceeding after noting the last term is non-negative via the definition of an inner product,

$$\Delta_{k_0-1}^2 \leq \Delta_{k_0-2}^2 + \left\|\frac{\eta\vec{g}_{k_0-2}}{b_{k_0-1}}\right\|^2 \tag{15}$$

$$= \Delta_{k_0-2}^2 + \frac{\eta^2\left\|\vec{g}_{k_0-2}\right\|^2}{b_{k_0-1}^2} \tag{16}$$

$$\leq \Delta_0^2 + \eta^2 \sum_{t=0}^{k_0-2} \frac{\left\|\vec{g}_t\right\|^2}{b_{t+1}^2}. \tag{17}$$

Note that the change of index and introduction of the sum comes from telescopically expanding $\Delta_{k_0-2}^2$ from the defintion of our scheme in (4). We now write b_{t+1}^2 in its explicit form to see

$$\Delta_{k_0-1}^2 \leq \Delta_0^2 + \eta^2 \sum_{t=0}^{k_0-2} \frac{\left\|\vec{g}_t\right\|^2}{b_0^2 + \sum_{p=0}^{t}\left\|\vec{g}_p\right\|^2}. \tag{18}$$

[7]We correct the result seen in Lemma A.2 in [23], by picking up a factor of η in the logarithm term.

Then, noting the integral of an expression in a summation is always larger than the summation we write the bound as

$$\Delta_{k_0-1}^2 \leq \Delta_0^2 + \eta^2 \left(\ln \left(\sum_{t=0}^{\tau-2} \|\vec{g}_t\| / b_0^2 \right) + 1 \right) \tag{19}$$

$$\leq \Delta_0^2 + \eta^2 \left(\ln \left(\frac{\eta^2 \bar{\lambda}_1^2}{b_0^2} \right) + 1 \right), \tag{20}$$

where in the last line we recall our definition of k_0. □

Lemma 3. [8] *For AdaGrad-Norm, with our definition of k_0 in Lemma 2 we have that* $(b_M := \max_t b_{k_0+t}) \leq \eta\bar{\lambda}_1 + \frac{\bar{\lambda}_1}{\eta}\Delta_{k_0-1}^2$.

Proof. We write, in a similar manner to the proof of Lemma 2,

$$\Delta_{k_0+t}^2 = \Delta_{k_0+t-1}^2 + \eta^2 \frac{\left\|\vec{g}_{k_0+t-1}\right\|^2}{b_{k_0+1}^2} - 2\eta \frac{1}{b_{k_0+t}} \left\langle \vec{g}_{k_0+t-1} - \nabla f_{k_0+t-1}(\vec{w}^*) \middle| \vec{w}_{k_0+t-1} - \vec{w}^* \right\rangle, \tag{21}$$

where we make use of the fact that $\nabla f_t(\vec{w}^*) = 0$ by definition of the convexity of f_t and RUIG. Now

$$\Delta_{k_0+t}^2 \leq \Delta_{k_0+t-1}^2 + \frac{\eta^2}{b_{k_0+1}^2} \left\|\vec{g}_{k_0+t-1}\right\|^2 - \frac{2\eta}{b_{k_0+t}\bar{\lambda}_1} \left\|\vec{g}_{k_0+t-1} - \nabla f_{k_0+t-1}(\vec{w}^*)\right\| \tag{22}$$

$$\leq \Delta_{k_0+t-1}^2 + \left\|\vec{g}_{k_0+t-1}\right\|^2 \left(\frac{\eta^2}{b_{k_0+t}^2} - \frac{2\eta}{b_{k_0+t}\bar{\lambda}_1} \right) \tag{23}$$

$$\leq \Delta_{k_0+t-1}^2 - \left\|\vec{g}_{k_0+t-1}\right\|^2 \frac{\eta}{b_{k_0+t}\bar{\lambda}_1} \tag{24}$$

$$\leq \Delta_{k_0-1}^2 - \sum_{j=0}^{t} \left\|\vec{g}_{k_0+j-1}\right\|^2 \frac{\eta}{b_{k_0+j}\bar{\lambda}_1}, \tag{25}$$

where (23) follows from our bound on the L-smoothness of f_t via (8), (24) follows from the bound on b_{k_0}, and (25) follows from a telescopic expansion. Then we have

$$\sum_{j=0}^{t} \left\|\vec{g}_{k_0+j-1}\right\|^2 \frac{1}{b_{k_0+j}} \leq \frac{\bar{\lambda}_1}{\eta} \left(\Delta_{k_0-1}^2 - \Delta_{k_0+t}^2 \right). \tag{26}$$

We first express b_{j+1} in a new form before proceeding. We have

$$\left. \begin{aligned} b_j^2 &= b_0^2 + \sum_i^{j-1} \left\|\vec{g}_i\right\|^2 \\ b_{j+1}^2 &= b_0^2 + \sum_i^{j} \left\|\vec{g}_i\right\|^2 \end{aligned} \right\} \Rightarrow b_{j+1}^2 - b_j^2 = \left(b_{j+1} - b_j\right)\left(b_{j+1} + b_j\right) = \left\|\vec{g}_j\right\|^2,$$

[8] Here we arrive at the same result as Lemma A.3 in [23].

and so $b_{j+1} = b_j + \left\|\vec{g}_j\right\|^2 / \left(b_{j+1} + b_j\right)$. Using (26) in this result (with $j+1 \to k_0 + t$) we have

$$b_{k_0+t} \leq b_{k_0-1} + \sum_{j=0}^{t} \left\|\vec{g}_{k_0+j-1}\right\|^2 \frac{1}{b_{k_0+j}} \tag{27}$$

$$\leq \eta\bar{\lambda}_1 + \frac{\bar{\lambda}_1}{\eta}\Delta_{k_0-1}^2. \tag{28}$$

□

We now have all the pieces we need to complete the proof of convergence in the stochastic, least squares case.

2.3 Proof of AdaGrad-Norm Convergence in Least Squares

From Lemma 1 we have shown the existence of a τ such that $b_\tau > \eta\bar{\lambda}_1$. Recalling our k_0, we now add that $k_0 < \tau$; k_0 is the first time index such that $b_t > \eta\bar{\lambda}_1$. We then write, if $k_0 \geq 1$,

$$\Delta_{k_0+t}^2 = \Delta_{k_0+t-1}^2 + \frac{\eta^2}{b_{k_0+t}^2}\left\|\vec{g}_{k_0+t-1}\right\|^2 - \frac{2\eta}{b_{k_0+t}}\left\langle \vec{w}_{k_0+t-1} - \vec{w}^* \middle| \vec{g}_{k_0+t-1}\right\rangle \tag{29}$$

$$\leq \Delta_{k_0+t-1}^2 + \left(\frac{\eta^2\bar{\lambda}_1}{b_{k_0+t}^2} - \frac{2\eta}{b_{k_0+t}}\right)\left\langle \vec{w}_{k_0+t-1} - \vec{w}^* \middle| \vec{g}_{k_0+t-1}\right\rangle \tag{30}$$

$$\leq \Delta_{k_0+t-1}^2 - \frac{\eta}{b_{k_0+t}}\left\langle \vec{w}_{k_0+t-1} - \vec{w}^* \middle| \vec{g}_{k_0+t-1}\right\rangle \tag{31}$$

$$\leq \Delta_{k_0+t-1}^2 - \frac{\eta}{b_M}\left\langle \vec{w}_{k_0+t-1} - \vec{w}^* \middle| \vec{g}_{k_0+t-1}\right\rangle, \tag{32}$$

where (32) follows from Lemma 3. Now recall that $\vec{g}$ is a stochastic gradient chosen such that $\mathbb{E}[\vec{g}] = \nabla F$. Also note that F is $\bar{\lambda}_m$*—strongly convex*[9]. Thus

$$\mathbb{E}\left[\Delta_{k_0+t}^2\right] \leq \Delta_{k_0+t-1}^2 - \frac{\eta}{b_M}\left\langle \vec{w}_{k_0+t-1} - \vec{w}^* \middle| \nabla F\right\rangle \tag{33}$$

$$\leq \left(1 - \frac{\eta\bar{\lambda}_m}{b_M}\right)\Delta_{k_0+t-1}^2 \leq \Pi_{t=0}\left(1 - \frac{\eta\bar{\lambda}_m}{b_M}\right)\Delta_{k_0-1}^2 \tag{34}$$

$$\leq \exp\left\{-\bar{\lambda}_m t / b_M\right\}\Delta_{k_0-1}^2. \tag{35}$$

2.3.1 Markov's Inequality

We introduce *Markov's Inequality* which states that for all $a > 0, \mathbb{P}(X \geq a) \leq \mathbb{E}[X]/a$ [11].

[9] α-strong convexity is the condition that $\left\|\nabla^2 F\right\| \geq \alpha$, where our $\alpha := \bar{\lambda}_m$ as $\nabla^2 F = \vec{x}^\top \vec{x}/m$ [1].

Proof. First introduce

$$Y = \begin{cases} 1 \text{ if } X \geq a, \\ 0 \text{ otherwise.} \end{cases}$$

Now we have two cases. Either $X \geq a$ then $Y = 1$ and so $Y \leq X/a$, or $X < a$ and so $Y = 0 \leq X/a$, achieving equality iff $X = 0$. Thus $\mathbb{E}[Y] = \mathbb{P}(Y = 1) = \mathbb{P}(X \geq a) \leq \mathbb{E}[X]/a$. □

First note that we can re-express Markov's Inequality as

$$\mathbb{P}(X \leq a) \geq 1 - \mathbb{E}[X]/a. \tag{36}$$

We then apply Markov's Inequality in the form (36) to the inequality (35) to get, for some small parameter δ_2,

$$\mathbb{P}\left(\Delta^2_{k_0+t} \leq \frac{1}{\delta_2} \exp\left\{-\bar{\lambda}_m t/b_M\right\} \Delta^2_{k_0-1}\right) \geq 1 - \delta_2. \tag{37}$$

We now need to find some number of steps $\hat{t}$ which would satisfy $\Delta^2_{k_0+\hat{t}} \leq \varepsilon$ when inserted into the inequality in (37). This $\hat{t}$ is simply $\frac{b_M}{\bar{\lambda}_m} \ln\left(\Delta^2_{k_0-1}/(\varepsilon\delta_2)\right)$.

Let us take count of how many timesteps we need in sum. In the case where $b_0 < \eta\bar{\lambda}_1$, we first have at least τ steps from Lemma 1, then another $\hat{t}$ steps; let us define the total number of steps in this case as T_1.

For convenience we first define

$$\Gamma := \Delta_0^2 + \eta^2 \left(2 \ln\left(\frac{\eta\bar{\lambda}_1}{b_0}\right) + 1\right) \tag{38}$$

from the bound we placed in (20) onto the $\Delta^2_{k_0-1}$ term in Lemma 3. Then, to have $\Delta^2_{T_1} \leq \varepsilon$ with probability greater than $1 - \delta_1 - \delta_2$, we require

$$T_1 = \left\lceil \frac{\eta^2\bar{\lambda}_1^2 - b_0^2}{\mu\gamma\varepsilon} + \frac{\delta}{\gamma} + \frac{\bar{\lambda}_1\left(\eta + \frac{1}{\eta}\Gamma\right)}{\bar{\lambda}_m} \ln\left(\frac{\Gamma}{\varepsilon\delta_2}\right) \right\rceil + 1. \tag{39}$$

Alternatively if $k_0 = 0$ then $b_0 > \eta\bar{\lambda}_1$, and so we do not need the first τ steps but we replace $\Delta^2_{k_0-1}$ with Δ_0^2, such that we need

$$T_2 = \left\lceil \frac{b_0 + \frac{\bar{\lambda}_1}{\eta}\Delta_0^2}{\bar{\lambda}_m} \ln\left(\frac{\Delta_0^2}{\varepsilon\delta_2}\right) \right\rceil + 1 \tag{40}$$

steps to attain the same bound $\Delta^2_{T_2} \leq \varepsilon$ with probability greater than $1 - \delta_2$. Thus, we have shown convergence for AdaGrad-Norm in the stochastic setting for least squares regression regardless of the choice of b_0 and η.

2.4 AdaLoss

We perform a very similar analysis on AdaLoss as done in [23]. In order to show stochastic convergence for the least squares problem, we go through the same proof as we did for AdaGrad-Norm. Recall the proof for Lemma 1. We note that replacing the term $\left\|\nabla f_{\xi_t}(\vec{w}_t)\right\|^2$ in (10) with $\left\langle \vec{X}_{\xi_t} \middle| \vec{w} - \vec{w}^*\right\rangle^2$, we have the same result *but under RUIL*, so we have proved this lemma for the AdaLoss case, again with the corrected factor of η compared to Lemma A.1 in [23].

Moving to Lemma 2, note that in the case of AdaLoss the proof proceeds the same way until (18). Now consider that the (stochastic) gradient $\vec{g}_t = \vec{X}_{\xi_t}^{\top}\left(\vec{X}_{\xi_t}\vec{w} - \vec{y}\right)/m$ and thus that $\left\|\vec{g}_t\right\|^2 \leq \bar{\lambda}_1 \left\langle \vec{X}_{\xi_t} \middle| \vec{w}_t - \vec{w}^*\right\rangle^2$ - so we can finish the rest of the proof for the lemma with an additional factor of $\bar{\lambda}_1$ multiplying the η^2 term outside the logarithm compared to the AdaGrad-Norm case, and again picking up the corrected factor of η inside the logarithm as compared to Lemma A.2 in [23]. We state this as

Lemma 4. *Considering AdaLoss, for $k_0 \in (0, N-1]$ s.t. $\exists\ b_{k_0} > \eta\bar{\lambda}_1$ and $b_{k_0-1} < \eta\bar{\lambda}_1$, we have that $\Delta_{k_0-1}^2 \leq \Delta_0^2 + \bar{\lambda}_1\eta^2\left(\ln\left(\frac{\eta^2\bar{\lambda}_1^2}{b_0^2}\right) + 1\right)$.*

We now state an equivalent for Lemma 3 in the case of AdaLoss, which provides an equivalent result to Lemma A.3 in [23].

Lemma 5. *For AdaLoss, in our definition of k_0 in Lemma 4 we have that $(b_M := \max_t b_{k_0+t}) \leq \eta\bar{\lambda}_1 + \frac{1}{\eta}\Delta_{k_0-1}^2$.*

The proof starts by writing

$$\begin{aligned}
\Delta_{k_0+t}^2 &= \Delta_{k_0+t-1}^2 + \eta^2\left\|\vec{X}_{\xi_t}\right\|^2 \frac{\left\langle \vec{X}_{\xi_t} \middle| \vec{w}_{k_0+t-1} - \vec{w}^*\right\rangle^2}{b_{k_0+1}^2} \\
&\quad - \frac{2\eta}{b_{k_0+t}}\left\langle\left\langle \vec{X}_{\xi_t} \middle| \vec{w}_{k_0+t-1} - \vec{w}^*\right\rangle \vec{X}_{\xi_t} \middle| \vec{w}_{k_0+t-1} - \vec{w}^*\right\rangle \qquad (41) \\
&\leq \Delta_{k_0+t-1}^2 + \frac{\eta}{b_{k_0+t}}\left(\frac{\eta\bar{\lambda}_1}{b_{k_0+t}} - 2\right)\left\langle \vec{X}_{\xi_t} \middle| \vec{w}_{k_0+t-1} - \vec{w}^*\right\rangle^2 \qquad (42)
\end{aligned}$$

using the expression for the bound on the gradient. We note that (42) is just the same expression (with a different form for the bound on the gradient squared inserted) as (23), but with an extra factor of $\bar{\lambda}_1$ in the rightmost term due to the step update term in AdaLoss. This directly gets us to the expression in Lemma 5 (which is just the same bound as Lemma 3 but with the rightmost term scaled by $1/\bar{\lambda}_1$). Therefore, for AdaLoss, (in a similar manner to AdaGrad) we first define

$$\tilde{\Gamma} := \Delta_0^2 + \eta^2\bar{\lambda}_1\left(2\ln\left(\frac{\eta\bar{\lambda}_1}{b_0}\right) + 1\right), \qquad (43)$$

so that if $b_0 < \eta\bar{\lambda}_1$ we require

$$\tilde{T}_1 = \left\lceil \frac{\eta^2\bar{\lambda}_1^2 - b_0^2}{\mu\gamma\varepsilon} + \frac{\delta}{\gamma} + \frac{\bar{\lambda}_1\left(\eta + \frac{1}{\eta\bar{\lambda}_1}\tilde{\Gamma}\right)}{\bar{\lambda}_m} \ln\left(\frac{\tilde{\Gamma}}{\varepsilon\delta_2}\right) \right\rceil + 1 \tag{44}$$

steps, and if $b_0 > \eta\bar{\lambda}_1$ we require

$$\tilde{T}_2 = \left\lceil \frac{b_0 + \frac{1}{\eta}\Delta_0^2}{\bar{\lambda}_m} \ln\left(\frac{\Delta_0^2}{\varepsilon\delta_2}\right) \right\rceil + 1 \tag{45}$$

steps, to achieve the same bounds $\Delta_{\tilde{T}_1}^2, \Delta_{\tilde{T}_2}^2 \leq \varepsilon$ with the same respective probabilities as in the case of AdaGrad-Norm.

3 Theoretical Comparison of AdaLoss and AdaGrad-Norm

We start by the assumption that $\bar{\lambda}_1 \geq 1$. This condition is quite nice in the sense that $\bar{\lambda}_1$ is the *largest* singular value of $\vec{X}^\top\vec{X}$, to a scalar factor of $1/m$, so $\bar{\lambda}_1$ may be reasonably thought to hold such a lower bound.

Now consider our expressions for T_1 and $\tilde{T}_1$ in (39) and (44) respectively, in particular the non-shared terms which depend on Γ and $\tilde{\Gamma}$ (defined in (38) and (43) respectively) which we write as functions of $\bar{\lambda}_1$ and b_0. In T_1 these terms take the form

$$\left(\Delta_0^2\bar{\lambda}_1 + A\bar{\lambda}_1 + \bar{\lambda}_1\ln\left(\bar{\lambda}_1/b_0\right)\right)\ln\left(\Delta_0^2 + B + \ln\left(\bar{\lambda}_1/b_0\right)\right), \tag{46}$$

while in $\tilde{T}_1$ they take the form

$$\left(\Delta_0^2 + C\bar{\lambda}_1 + \bar{\lambda}_1\ln\left(\bar{\lambda}_1/b_0\right)\right)\ln\left(\Delta_0^2 + D + \bar{\lambda}_1\ln\left(\bar{\lambda}_1/b_0\right)\right), \tag{47}$$

for constants A, B, C, D. We see that if we let Δ_0^2 dominate (which we expect as this is the norm squared of the difference of the initial weights and final weights) the bound on T_1 is greater than the bound on $\tilde{T}_1$, assuming the other terms in $T_1, \tilde{T}_1$ which we have neglected are either small enough to ignore or of similar size such that we can ignore them. In particular, we let $\eta = 1$ since having shown that the *ratio* of b_0 to η is what determines the maximum number of stepsizes, in practical implementations we expect to leave $\eta = 1$ and vary b_0.

Moving to consider T_2 and $\tilde{T}_2$ (in (40) and (45) respectively), note that $T_2 \geq \tilde{T}_2$ under our assumption that $\bar{\lambda}_1 \geq 1$. Of course, these results are the worst-case scenarios, and as we have also considered the stochastic case the element of randomness will always play a role, as these are simply guarantees for a worst-case bound on the time for convergence. In order to try and glean some practical insight into these methods, let us proceed to a computational implementation.

4 Practical Implementation

We implement all our code using Python, via series of functions which given input data output a series of weights and the losses incorporated iteratively on the data they were trained on. All code is available in Appendix **??**.

4.1 Synthetic Data

Consider the least squares problem in (7). The aim of this problem is given a matrix $\vec{\boldsymbol{X}}$, set of initial weights $\vec{\boldsymbol{w}}$, and a vector of outputs $\vec{\boldsymbol{y}}$, to gradually update the weights to minimise the loss of our problem (i.e. the residual).

Let us first introduce another algorithm we shall employ, **AMSGrad**, in the same manner as done in [13]; AMSGrad has also been shown to converge with assumptions on knowledge of the Lipschitz constant in e.g. [25] and we merely present it in order to further discuss alternative methods against SGD. The scheme is defined as

$$\textbf{AMSGrad:} \qquad \vec{\boldsymbol{x}}_{t+1} = \vec{\boldsymbol{x}}_t - \frac{\eta_t}{\sqrt{\vec{\boldsymbol{B}}_{t+1}}}\vec{\boldsymbol{m}}_{t+1}, \qquad \begin{cases} \vec{\boldsymbol{m}}_{t+1} = \beta_1 \vec{\boldsymbol{m}}_t + (1-\beta_1)\vec{\boldsymbol{g}}_t, \\ \vec{\boldsymbol{b}}_{t+1} = \beta_2 \vec{\boldsymbol{b}}_t + (1-\beta_2)\vec{\boldsymbol{g}}_t^2, \\ \vec{\boldsymbol{B}}_{t+1} = \max\left(\vec{\boldsymbol{B}}_t, \vec{\boldsymbol{b}}_{t+1}\right), \end{cases} \tag{48}$$

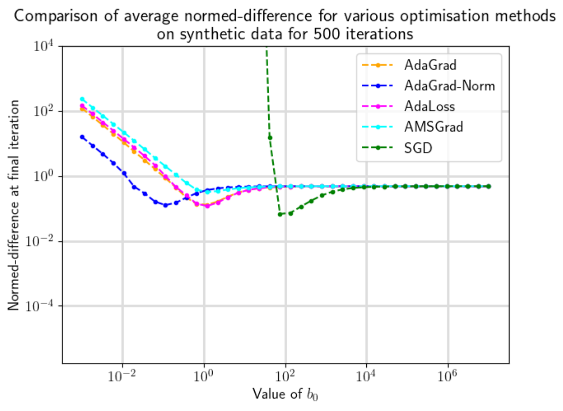
with $\{\eta_t\}_0^{N-1}$ being pre-defined as $\eta_t := \eta/\sqrt{t}$. Furthermore we define $\beta_1, \beta_2 \in (0,1]$ with $\beta_1 < \sqrt{\beta_2}$; we choose to initialise $\beta_1 = 0.2, \beta_2 = 0.3$ to fulfil this condition. We also initialise $\vec{\boldsymbol{m}}_0 = \vec{\boldsymbol{b}}_0 = \vec{\boldsymbol{B}}_0 = 0$.

We define an initial matrix $\vec{\boldsymbol{X}} \in \mathbb{R}^{1000\times 101}$ which is populated in every entry barring the first column by Gaussians with $\mu = 0, \sigma^2 = 1$ (the initial matrix is generated $\mathbb{R}^{1000\times 100}$, however, we then add a column such that $\vec{\boldsymbol{X}}_{i,0} = 1$ to account for bias terms which do not depend on values of $\vec{\boldsymbol{w}}$). We also generate a (seeded) random vector $\vec{\boldsymbol{w}}^* \in \mathbb{R}^{101}$ which represents the "true" values of the weights, and compute $\vec{\boldsymbol{y}} := \vec{\boldsymbol{X}}\vec{\boldsymbol{w}}^* \in \mathbb{R}^{1000}$. We also generate a seeded random vector $\vec{\boldsymbol{w}} \in \mathbb{R}^{101}$ which represents our initial guess for the weights. For our problem, the "free" parameters we have are b_0 and η, or in particular their ratio, which we set as the initial learning rate for *all* methods for a fair comparison. By setting $\eta = 1$ for all tests and choosing a range of b_0, we should be able to get an idea of how these methods perform in different regimes.

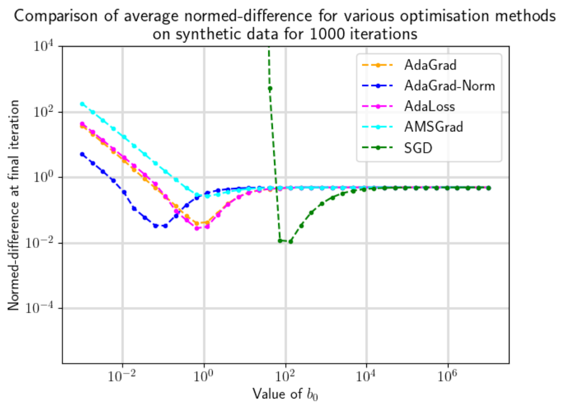
We define our b_0 values as[10] $b_{0,k} := 10^k$ for $k := \left\{-3 + j\left(\frac{10}{39}\right)\right\}, j = 0, \cdots 39$ such that we may test a wide range of b_0. We run for 3000 iterations and sample the average

[10]What this is saying is that we have simply run a `linspace` command to generate 40 equidistant values for k between -3 and 7.

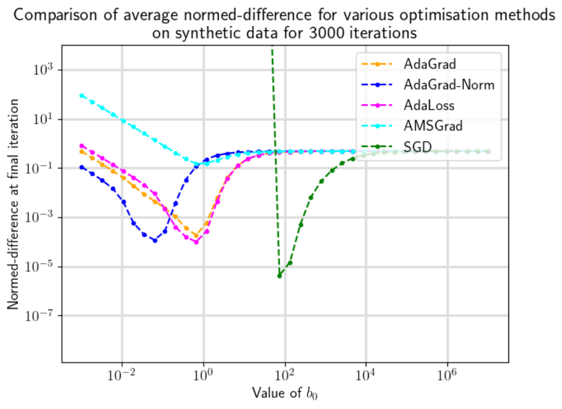
normed-difference[11] at iterations of 500, 1000, and 3000. For comparison, we run *AdaGrad, AdaGrad-Norm, AdaLoss, AMSGrad*, and *SGD*. We present the synthetic results in Figure 1.



(a) Average normed-difference after $t = 500$ iterations.



(b) Average normed-difference after $t = 1000$ iterations.



(c) Average normed-difference after $t = 3000$ iterations.

Figure 1: Plots of least-squares average normed-difference on synthetic data after t iterations of various optimisation algorithms as a function of initial parameter b_0, for initial stepsize $\eta = 1$.

Note immediately how ill-parameterisation for SGD presents disastrous results: for values of $b_0 \lessapprox 10^2$ we see final average normed-differences which rapidly blow up to

[11] I.e. $\frac{1}{2m}\left\|\vec{X}\vec{w} - \vec{y}\right\|$. We use this metric out of computational convenience as this normed-difference is already called within our weight-generating functions. Ordinary least squares losses would just be quadratic scalings of the losses we plot; the aim of the plots we produce are more to draw comparison against SGD and other methods than for benchmarking against existing values, particularly as we run bespoke parameterisation for both synthetic and real life data.

greater than 10^{100}. We see that in general, the methods we have discussed perform well with AdaGrad, AdaGrad-Norm, and AdaLoss performing better than AMSGrad overall, and particularly after 3000 iterations as seen in Figure 1(c). We also note that while AdaLoss in general outperforms AdaGrad-Norm, this is not the case when $b_0 \lessapprox 10^{-1}$; below this point AdaGrad-Norm seems to outperform the other methods by at least an order of magnitude which is best seen in Figure 1(a).

Importantly we observe that the two methods we discussed in detail in the linear regression setting, AdaGrad-Norm and AdaLoss, are indeed robust to initialisation; achieving small ($< 10^0$) average normed-differences after 3000 iterations for virtually all values of b_0. AMSGrad however seems to not show much improvement as iterations increase which perhaps suggests further study might be needed on AMSGrad in the stochastic linear regression regime.

4.2 Housing Dataset

We now move on to the *California Housing Dataset* [17] which contains data on census block groups, or CBGs (CBGs being the smallest geographical grouping in the U.S. census), from the 1990 Californian census. This dataset has data from $20,640$ CBGs corresponding to $20,640$ rows each with 8 different features per CBG: *Median Household Income, House Age, Average number of Rooms, Average number of Bedrooms, Population in CBG, Average number of Household Members, Latitude*, and *Longitude*. This data is all represented in the matrix we call $\vec{X}$, and the *target*, or $\vec{y}$, is the median household price per CBG. The weights $\vec{w}$ are just the weightings we apply to the features listed above to try and determine a formula for accurately predicting the median house price. This dataset was chosen for several reasons: it is a large, pre-processed, and numerical dataset with no missing entries[12]. There is much room for discussion on optimal pre-processing of real data for machine learning (such as via imputing missing values) but this clearly falls beyond the scope of this report and thus we have opted to focus on analysing pre-processed data.

Upon importing the data we immediately perform a $80\% : 20\%$ training-to-testing split on the data such that we leave an "untouched" set of data which we can use to check our model once we have our final weights derived from the various optimisation methods. We also scale our features $\vec{X}$, by subtracting the mean for each feature and normalising the variance to 1; applying the training dataset's scaling to the testing dataset. We

[12]There is an alternative version of this dataset which contains a non-numerical feature which is the proximity to water, but the version used (via `scikit-learn`) does not have such a feature by default.

scale to ensure gradient searches are not skewed heavily in one specific feature which might have massively different scales on data; for example in our dataset we expect roughly similar values for latitude with far less variance than median income, or median population. More concretely, when updating gradients, $\vec{w}_t$ might oscillate if we have wildly varying feature scales, leading to erratic jumps in the gradient; particularly for co-ordinate based methods such as AdaGrad. Also of note is the key fact that we scale the testing sample using the same parameters gleaned from the training sample; such that we do not incorporate any information on the testing sample values when we evaluate our generated weights.

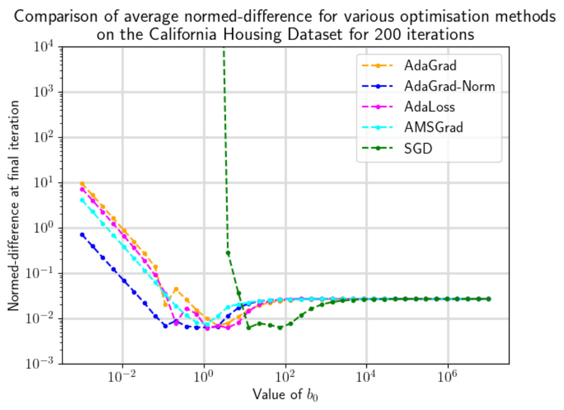
After scaling the data we train our models for $t = \{200, 500, 800\}$ iterations, for the same range of b_0 values as in our synthetic data[13]. The results for this implementation can be seen in Figure 2.

Again we see the large sensitivity to initial conditions that is present in SGD; for all iterations sampled we see divergence when b_0 is less than about 10^1. AdaGrad-Norm appears to perform the best for small values of b_0, but AdaLoss quickly catches up to attain comparable average normed-difference values at lower and lower values of b_0, from around 10^0 at 200 iterations to around 10^{-1} at 800 iterations. AMSGrad performs poorly for small values of b_0 (particularly in the same regime where AdaLoss performs notably worse than AdaGrad-Norm), but attains similar results as other methods for higher values of b_0, beyond $\approx 10^2$. Further, we see that AdaGrad outperforms AdaLoss but not AdaGrad-Norm in the regime of $b_0 < 10^{-1}$, but gives performance worse than AdaLoss and better than AdaGrad-Norm for values of b_0 roughly beyond this range.

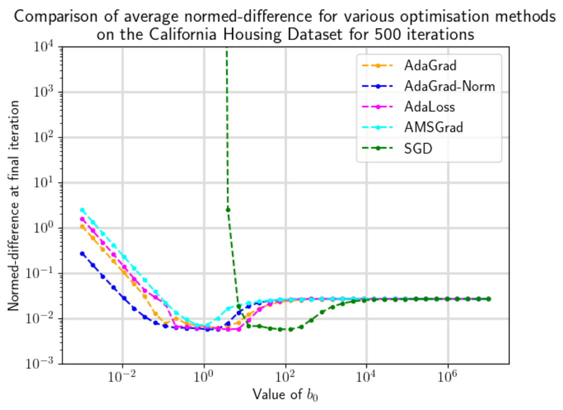
We can zoom into the region $b_0 \in [10^{-0.1}, 10^{0.1}]$ (which appears to be the region with the best values for average normed-difference for non-SGD methods), again with 40 equidistant points sampled in this region (but this time we ignore any SGD information given the large average normed-differences this method attains). We present the results in Figure 2(d).

Here we can see how AdaLoss and AdaGrad-Norm perform quite similarly. Noting that `scikit-learn SGDRegressor`, which uses a modified SGD with a varying stepsize of $\eta/t^{1/4}$ for a limit of 1000 iterations [16], produces average normed-differences of $O(10^{-2})$, we observe that this result underperforms our model by an order of magnitude in this scenario. AMSGrad and AdaGrad in this regime both slightly underperform both AdaGrad-Norm and AdaLoss, but the difference is negligible when considering these methods all out-perform `SGDRegressor`, as well as our implementation of SGD.

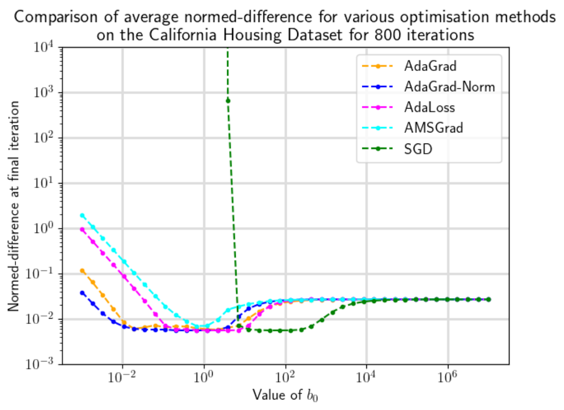
[13]That is to say, $b_{0,k} := 10^k$ for $k := \left\{-3 + j\left(\frac{10}{39}\right)\right\}, j = 0, \cdots 39$ for a total of 40 different values tested.



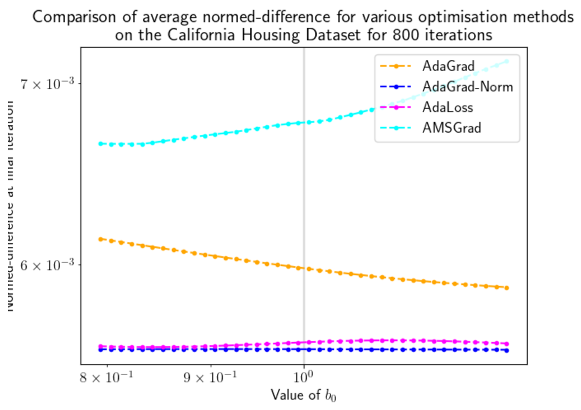
(a) Average normed-difference after $t = 200$ iterations.



(b) Average normed-difference after $t = 500$ iterations.



(c) Average normed-difference after $t = 800$ iterations.



(d) Zoom on average normed-difference after $t = 800$ iterations, no SGD plotted.

Figure 2: Plots of average normed-difference on the Housing Dataset after t iterations of various optimisation algorithms as a function of initial parameter b_0, for initial stepsize $\eta = 1$.

Of course, these results are for a small range of b_0; however we have shown experimentally that both AdaGrad-Norm and AdaLoss still seem to converge regardless of choice of initial parameters and so in industrial contexts running a wide initial array of values for a basic hyperparameter search might be in order. Certainly, we have shown that the results in “extreme” initial parameters (referring to both incredibly large and incredibly small b_0) lead to far better results than one might achieve with SGD for this dataset, with the exception of a small range of $b_0 \in [10^1, 10^4]$. This constraint on b_0 for SGD is highly suboptimal in industrial contexts as it is impossible to know *a priori* what range of values guarantees convergence. A parameter sweep might be brought up for a possible rectification to guarantee convergence in which case it is true that SGD presents lower minimum average normed-differences for both the methods we have discussed, but clearly it a far more reliable choice to simply use AdaGrad-Norm or AdaLoss which on average presents far better results.

5 Conclusion

In this report, we have looked at two adaptive stepsize methods, AdaLoss and AdaGrad-Norm, in both theoretical and practical contexts for the setting of stochastic linear regression. We have shown in theory that not only are these two methods robust to initial parameterisation, but also demonstrated scenarios in which AdaLoss potentially has lower limits on iterations needed for convergence than AdaGrad-Norm. Further, we have explored these methods in experimental contexts for both synthetically generated data, as well as on a real-world dataset in order to back up the claims we made in theory. These findings present further exploration of the application of these two methods to those found in literature and bolster the claims that these methods might prove useful in further industrial contexts for machine learning.

There are clear extensions demonstrated in this report. Primarily, there is very little literature comparing the methods we have discussed (AdaGrad-Norm, AdaLoss, AMSGrad) in the stochastic linear regression setting which encapsulates a discussion of practical implementation. As stated earlier in the report, while there are theoretical bounds for the convergence rates for all these methods, these bounds do not translate directly into measures of performance. We have only looked at these in two simple cases; further exploration is necessary in order to examine if these methods might serve to replace SGD (or related methods such as those implemented in `SGDRegressor`) as standard tools.

6 References

[1] Anderson Ang. *Some Special Classes of Function in Optimization - L.S.C, Convex, Strongly Convex, Lipschitz, Smooth, Etc.* Nov. 9, 2022. URL: `https://angms.science/doc/CVX/CVX_alphabeta.pdf` (visited on 11/25/2022).

[2] Phil Blunsom and Matthew Fellows. *University of Oxford, Department of Computer Science: Machine Learning - Michaelmas Term 2022*. Lectures. URL: `https://www.cs.ox.ac.uk/teaching/materials22-23/ml/lectures/` (visited on 11/19/2022).

[3] Soham De, Anirbit Mukherjee, and Enayat Ullah. *Convergence guarantees for RMSProp and ADAM in non-convex optimization and an empirical comparison to Nesterov acceleration*. Number: arXiv:1807.06766. Nov. 20, 2018. DOI: `10.48550/arXiv.1807.06766`. arXiv: `1807.06766[cs,math,stat]`. URL: `http://arxiv.org/abs/1807.06766` (visited on 11/20/2022).

[4] Alexandre Défossez, Léon Bottou, Francis Bach, and Nicolas Usunier. *A Simple Convergence Proof of Adam and Adagrad*. Number: arXiv:2003.02395. Oct. 17, 2022. arXiv: `2003.02395[cs,stat]`. URL: `http://arxiv.org/abs/2003.02395` (visited on 11/22/2022).

[5] John Duchi, Elad Hazan, and Yoram Singer. “Adaptive Subgradient Methods for Online Learning and Stochastic Optimization”. In: *Journal of Machine Learning Research* 12.61 (2011), pp. 2121–2159. URL: `http://jmlr.org/papers/v12/duchi11a.html`.

[6] Saeed Ghadimi and Guanghui Lan. *Stochastic First- and Zeroth-order Methods for Nonconvex Stochastic Programming*. Number: arXiv:1309.5549. Sept. 21, 2013. DOI: `10.48550/arXiv.1309.5549`. arXiv: `1309.5549[cs,math,stat]`. URL: `http://arxiv.org/abs/1309.5549` (visited on 11/23/2022).

[7] Jean Lafond, Nicolas Vasilache, and Léon Bottou. *Diagonal Rescaling For Neural Networks*. Number: arXiv:1705.09319. May 25, 2017. DOI: `10.48550/arXiv.1705.09319`. arXiv: `1705.09319[cs,stat]`. URL: `http://arxiv.org/abs/1705.09319` (visited on 11/19/2022).

[8] Kfir Levy. “Online to Offline Conversions, Universality and Adaptive Minibatch Sizes”. In: *Advances in Neural Information Processing Systems*. Vol. 30. Curran Associates, Inc., 2017. URL: `https://proceedings.neurips.cc/paper/2017/`

hash/ce5140df15d046a66883807d18d0264b-Abstract.html (visited on 11/19/2022).

[9] Xiaoyu Li and Francesco Orabona. *On the Convergence of Stochastic Gradient Descent with Adaptive Stepsizes*. Number: arXiv:1805.08114. Feb. 26, 2019. arXiv: 1805.08114[cs,math,stat]. URL: http://arxiv.org/abs/1805.08114 (visited on 11/22/2022).

[10] Nicolas Loizou, Sharan Vaswani, Issam Hadj Laradji, and Simon Lacoste-Julien. "Stochastic Polyak Step-size for SGD: An Adaptive Learning Rate for Fast Convergence". In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics*. International Conference on Artificial Intelligence and Statistics. ISSN: 2640-3498. PMLR, Mar. 18, 2021, pp. 1306–1314. URL: https://proceedings.mlr.press/v130/loizou21a.html (visited on 11/20/2022).

[11] Sriram Pemmaraju. *CS:5360 Randomized Algorithms*. CS:5360 Randomized Algorithms - Lecture 4. Sept. 2019. URL: https://homepage.cs.uiowa.edu/~sriram/5360/fall18/ (visited on 11/27/2022).

[12] Tran Thi Phuong and Le Trieu Phong. "On the Convergence Proof of AMSGrad and a New Version". In: *IEEE Access* 7 (2019), pp. 61706–61716. ISSN: 2169-3536. DOI: 10.1109/ACCESS.2019.2916341. arXiv: 1904.03590[cs,math,stat]. URL: http://arxiv.org/abs/1904.03590 (visited on 11/22/2022).

[13] Sashank J. Reddi, Satyen Kale, and Sanjiv Kumar. *On the Convergence of Adam and Beyond*. Number: arXiv:1904.09237. Apr. 19, 2019. DOI: 10.48550/arXiv.1904.09237. arXiv: 1904.09237[cs,math,stat]. URL: http://arxiv.org/abs/1904.09237 (visited on 11/21/2022).

[14] Herbert Robbins and Sutton Monro. "A Stochastic Approximation Method". In: *The Annals of Mathematical Statistics* 22.3 (1951). Publisher: Institute of Mathematical Statistics, pp. 400–407. ISSN: 0003-4851. URL: https://www.jstor.org/stable/2236626 (visited on 11/22/2022).

[15] Sebastian Ruder. *An overview of gradient descent optimization algorithms*. Number: arXiv:1609.04747. June 15, 2017. DOI: 10.48550/arXiv.1609.04747. arXiv: 1609.04747[cs]. URL: http://arxiv.org/abs/1609.04747 (visited on 11/21/2022).

[16] *sklearn.linear_model.SGDRegressor*. scikit-learn. URL: https://scikit-learn/stable/modules/generated/sklearn.linear_model.SGDRegressor.html (visited on 11/30/2022).

[17] *The California housing dataset — Scikit-learn course*. URL: https://inria.github.io/scikit-learn-mooc/python_scripts/datasets_california_housing.html (visited on 11/21/2022).

[18] Cheik Traoré and Edouard Pauwels. "Sequential convergence of AdaGrad algorithm for smooth convex optimization". In: *Operations Research Letters* 49.4 (July 1, 2021), pp. 452–458. ISSN: 0167-6377. DOI: 10.1016/j.orl.2021.04.011. URL: https://www.sciencedirect.com/science/article/pii/S0167637721000651 (visited on 11/19/2022).

[19] Rachel Ward, Xiaoxia Wu, and Leon Bottou. *AdaGrad stepsizes: Sharp convergence over nonconvex landscapes*. Number: arXiv:1806.01811. Apr. 18, 2021. DOI: 10.48550/arXiv.1806.01811. arXiv: 1806.01811[cs,stat]. URL: http://arxiv.org/abs/1806.01811 (visited on 11/19/2022).

[20] Ashia C. Wilson, Rebecca Roelofs, Mitchell Stern, Nathan Srebro, and Benjamin Recht. *The Marginal Value of Adaptive Gradient Methods in Machine Learning*. Number: arXiv:1705.08292. May 21, 2018. DOI: 10.48550/arXiv.1705.08292. arXiv: 1705.08292[cs,stat]. URL: http://arxiv.org/abs/1705.08292 (visited on 11/21/2022).

[21] Xiaoxia Wu, Simon S. Du, and Rachel Ward. *Global Convergence of Adaptive Gradient Methods for An Over-parameterized Neural Network*. Number: arXiv:1902.07111. Oct. 19, 2019. DOI: 10.48550/arXiv.1902.07111. arXiv: 1902.07111[cs,math,stat]. URL: http://arxiv.org/abs/1902.07111 (visited on 11/20/2022).

[22] Xiaoxia Wu, Rachel Ward, and Léon Bottou. *WNGrad: Learn the Learning Rate in Gradient Descent*. Number: arXiv:1803.02865. Nov. 19, 2020. DOI: 10.48550/arXiv.1803.02865. arXiv: 1803.02865[cs,math,stat]. URL: http://arxiv.org/abs/1803.02865 (visited on 11/19/2022).

[23] Xiaoxia Wu, Yuege Xie, Simon Shaolei Du, and Rachel Ward. "AdaLoss: A Computationally-Efficient and Provably Convergent Adaptive Gradient Method". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 36.8 (June 28, 2022). Number: 8, pp. 8691–8699. ISSN: 2374-3468. DOI: 10.1609/aaai.v36i8.20848. URL: https://ojs.aaai.org/index.php/AAAI/article/view/20848 (visited on 11/19/2022).

[24] Yuege Xie, Xiaoxia Wu, and Rachel Ward. *Linear Convergence of Adaptive Stochastic Gradient Descent*. Number: arXiv:1908.10525. Mar. 6, 2020. DOI: 10.48550/

arXiv.1908.10525. arXiv: 1908.10525[cs,math,stat]. URL: http://arxiv.org/abs/1908.10525 (visited on 11/19/2022).

[25] Dongruo Zhou, Jinghui Chen, Yuan Cao, Yiqi Tang, Ziyan Yang, and Quanquan Gu. *On the Convergence of Adaptive Gradient Methods for Nonconvex Optimization*. Number: arXiv:1808.05671. Oct. 19, 2020. DOI: 10.48550/arXiv.1808.05671. arXiv: 1808.05671[cs,math,stat]. URL: http://arxiv.org/abs/1808.05671 (visited on 11/20/2022).