Convergence Analysis of Deterministic and Stochastic Methods for Convex Optimization

by

Riley Brooks

A final project submitted to the

Department of Applied Mathematics

in partial fulfillment of the

requirements for the degree of

Master's of Mathematics (MMath)

Department of Applied Mathematics

University of Waterloo

August 2017

Waterloo Ontario

Abstract

Numerous scientific problems can be formulated in terms of optimizing a convex objective function. As such, the study and developement of algorithms designed to solve these problems is of considerable importance. This project attempts to act as an introduction to a selected number of well known optimization algorithms by examining convergence results of deterministic and stochastic algorithms. The concept of momentum is discussed and shown to have significant impact on convergence. Throughout, it should become clear that deterministic methods which have been known for decades are of fundamental importance for developing methods which noticably reduce the computational complexity while preserving fast convergence. Several methods are shown to achieve known optimal rates for a certain classes of algorithms. This is concluded with a variety of numerical examples to illustrate the comparative performance of the algorithms presented.

Acknowledgments

I would like to extend my deepest thank you and appreciation to my supervisor Dr. Jun Liu for his guidance and support during my time as a graduate student. I am grateful to have been given the opportunity to advance my academic career. It has been a pleasure to work in such an interdisciplinary environment, and what I have learned will be invaluable for my future career path. A special thank you to Dr. Xinzhi Liu for contributing his time and expertise to reviewing this paper.

Table of Contents

List of Figures

Chapter 1

Introduction

1.1 Motivation

Optimization problems arise throughout a diverse range of fields within applied science, from controller and filter design in control and electrical engineering to optimizing stochastic processes emanating from economics and finance. In particular, many fields of machine learning revolve around minimizing a convex function for efficient model estimation. In this scenario, the most convenient problem formulation is

$$\min_{x \in \mathcal{X}} f(x), \tag{1.1}$$

where $\mathcal{X} \in \mathcal{R}^d$ is a convex space and $f : \mathcal{X} \to \mathcal{R}$ is a continuously differentiable convex function. The solution of this problem is called the minimizer of f and is denoted $x^* \in \mathcal{X}$.

Since the time of Euler, problem (1.1) has been solvable using a method, known today as, the gradient descent algorithm. Gradient descent is in a class of optimizaition methods known as first order methods. This is due to the parameter update only requiring information about the objective and its gradient as can be seen by the gradient descent update below

$$x_{k+1} = x_k - \alpha \nabla f\left(x_k\right), \tag{1.2}$$

The simplicity of gradient descent is quite attractive, however it has been shown that it converges with rate $O\left(1/k\right)$ which is sub-optimal for first order methods [1]. In addition, full gradients are required for each update which can become problematic for large systems due to high computational cost. These issues have led researchers to design algorithms that either speed up convergence, lower computational complexity, or both.

Research on the theory and implementation of *accelerated* first order methods has come into the spotlight since the 1980s. The gradient descent algorithm is the basis for several methods of this type. In particular, in celebrated work Nesterov introduced his Accelerated Gradient (NAG) algorithm in 1983 [2]. This scheme relies on an additional momentum term that does not appear in gradient descent. This added "momentum" accelerates the algorithm towards the minimizer, effectively enhancing the convergence rate as compared to gradient descent. It was also shown by Nesterov that this method converges at the optimal rate of $O\left(1/k^2\right)$ for first order methods optimizing a general convex function.

More recently, the study and application of stochastic methods have allowed for increased tractibility of optimization problems. The goal of these schemes is to decrease computational complexity while retaining similar convergence rates to deterministic methods. In [3] and [4], Randomized Coordinate Gradient Descent (RCGD) was introduced to handle high dimensional problems with a lower computational complexity than gradient descent. Shortly after, an accelerated version of RCGD, called Accelerate Randomized Coordinate Gradient (ARCG) was proposed in [5]. The proposal of these algorithms displayed how well known deterministic methods can be used to design stochastic ones.

The most recent developements in optimization stretch this idea to use statistical moment estimation to solve huge, sparse problems. Two such methods of this type are Adam and Adagrad which were first introduced in [6] and [7] respectively. They use estimations of the mean and uncentered variance of the gradient in place of the using the full gradient. As with RCGD and ARCD, the goal is to lower the computational complexity while also holding on to quick convergence.

1.2 Outline

This project will proceed as follows, Chapter 2 will cover basic definitions, such as Lipschitz continuity and convexity, that will be required for all subsequent chapters. Chapter 3 will cover the algorithms and convergence analysis of the most well known deterministic methods. In particular, convergence results for Gradient Descent, Nesterov Accelerated Gradient and Newton's Method will be established for weakly and strongly convex objective function. Chapter 4 covers several stochastic algorithms and proceeds in the same manner as chapter 3. Finally, numerical tests for all methods covered will be presented and analyzed. The goal is to present a unified survey of several deterministic and stochastic optimization algorithms commonly used or discussed in the literature.

Chapter 2

Definitions and Inequalities

2.1 Lipschitz Continuity

This property is extremely useful for convergence analysis as it provides access to several convenient inequalities, especially when paired with convexity. It will be assumed throughout that $f(x)$, and in the case of stochastic optimization, each sub-function $f_i\left(x\right)$ will have Lipschitz continuous gradient.

Definition 2.0.1. A function $f : \mathbb{R}^d \to \mathbb{R}$ is *Lipschitz Continuous* with Lipschitz factor L if and only if the following inequalities hold for all $x, y \in \mathbb{R}^d$,

$$\|f\left(x\right) - f\left(y\right)\| \leq L\|x - y\|, \tag{2.1}$$

$$f\left(y\right) \leq f\left(x\right) + \left\langle \nabla f\left(x\right), y - x \right\rangle + \frac{L}{2}\|y - x\|^2, \tag{2.2}$$

The two inequalities above are equivalent, so either can be used to represent Lipschitz continuity. If, in addition, $f(x)$ is twice differentiable then it is also true that $\nabla^2 f(x) \leq L\mathbb{I}_d$, where $\mathbb{I}_d$ is the $d \times d$ identity matrix.

2.2 Convexity

If a function is known to be convex then it satisfies several inequalities that are practical when deriving convergence rates for optimization algorithms. Since this project's focus is convex optimization these inequalities are listed below for convenience. As mentioned previously, the combination of convexity and Lipschitz continuity implies a function satisfies a wide selection of inequalities. The definition for convexity and subsequent inequalities now follows.

Definition 2.0.2. Let $\chi \subseteq \mathbb{R}^d$ be a convex set, a function $f : \chi \to \mathbb{R}$ is convex if $\forall x, y \in \chi$, $\forall \alpha \in [0, 1]$

$$f(\alpha x + (1-\alpha) y) \leq \alpha f(x) + (1-\alpha) f(y). \tag{2.3}$$

This definition simply implies that a line segment connecting any two points of a function lies above or on the graph of the function. This is especially important for convex optimization since if f is differentiable then all stationary points of f are global minima. In this case, two additional inequalities arise which are equivalent to f being convex.

$$f(y) \geq f(x) + \nabla f(x) \cdot (y - x) \tag{2.4}$$

$$(\nabla f(x) - \nabla f(y)) \cdot (x - y) \geq 0 \tag{2.5}$$

In the case where f is twice differentiable, $\nabla^2 f(x) \succeq 0$, is equivalent to the definition above. Combining the definitions of convexity and Lipschitz continuity of the gradient yields several practical inequalities which will be used in later chapters and are listed below. As in the individual definitions for convexity and Lipschitz continuity, if f is twice differentiable then all of the above are equivalent to $0 \preceq \nabla^2 f(x) \succeq L\mathbb{I}_d$.

$$0 \leq f(y) - f(x) - \langle \nabla f(x), y-x \rangle \leq \frac{L}{2}\|y-x\|^2 \tag{2.6}$$

$$f(y) \geq f(x) + \langle \nabla f(x), y-x \rangle + \frac{1}{2L}\|\nabla f(y) - \nabla f(x)\|^2 \tag{2.7}$$

$$0 \leq \langle \nabla f(x) - \nabla f(y), x-y \rangle \leq L\|y-x\|^2 \tag{2.8}$$

$$\langle \nabla f(y) - \nabla f(x), y-x \rangle \geq \frac{1}{L}\|\nabla f(y) - \nabla f(x)\|^2 \tag{2.9}$$

Definition 2.0.3. A function $f : \mathbb{R}^d \to \mathbb{R}$ is *strongly convex* with convexity parameter μ if and only if the following hold.

$$f(y) - f(x) - \nabla f(x) \cdot (y-x) - \frac{\mu}{2}\|y-x\|^2 \geq 0 \tag{2.10}$$

$$(\nabla f(y) - \nabla f(x)) \cdot (y-x) - \mu\|y-x\|^2 \geq 0 \qquad \forall x, y \in \mathbb{R} \tag{2.11}$$

Once again, if f is twice differentiable there is another inequality, namely $\nabla^2 f(x) \succeq \mu\mathbb{I}_d$.

2.3 Expectation

Also known as expected value, the expectation of a discrete random variable is the probability weighted average of all possible values. In the section on stochastic optimization the discrete random variable will be denoted i_k where i corresponds to an element of the gradient vector (a particular dimension where $i \in 1, \ldots, d$). The subscript k represents the iteration count, as such i_k represents the set of randomly selected coordinates during the k^{th} iteration of the algorithm. Alternatively, one could find the expectation with respect to i_k over all $k \in 1, \ldots, K$ which is denoted as ξ_{k-1} for convenience. A short-hand that is used for this form of the expectation is $\phi_k = \mathbb{E}_{\xi_{k-1}} \left[f \left(x_k \right) \right]$.

In all cases, the expectation with respect to the k^{th} iteration (i.e. expectation w.r.t i_k) is considered first. It then follows that to complete the convergence inequality one must take the expectation over all iterations, hence the need for the notation ξ_{k-1}. It is important to note that the probability of selecting a particular coordinate is uniformly distributed so the probability weighted average will be

$$\mathbb{E}_{i_k} \left[f \left(x_k \right) \right] = \sum_{i=1}^{n} p^{(i)} \cdot f_i \left(x_k \right), \qquad p^{(i)} = \frac{1}{n}.$$

Chapter 3

Deterministic Methods

3.1 Gradient Descent

In all likelihood, Gradient Descent was the first known method for finding optimal values of a function. Whether or not this is the case, gradient descent is the foundation for most determinsitic optimization methods as well as many well known stochastic schemes. As such, it is appropriate to begin any general discussion of optimization algorithms with the gradient descent method. The method is described by the iterative scheme below where α_k is the time step and is optimally chosen as $\alpha = \frac{1}{L}$:

$$x_{k+1} = x_k - \alpha_k \nabla f\left(x_k\right), \tag{3.1}$$

Every algorithm discussed in this project, except Adam and Adagrad, will in some way include this type of iterative step. Expectedly, the analysis for GD is much simpler than all susbequent algorithms but will still provide useful intuition for when it comes time to analyze the remaining methods. Since the main purpose of this report is to review methods for convex optimization we begin with considering the case where the objective f is a general convex function.

3.1.1 General Convexity

A simple manipulation of the definition for Lipschitz continuity, given by (2.2), through the gradient descent update rule yields another helpful inequality for this section.

$$\begin{aligned} f\left(x_{k+1}\right) - f\left(x_k\right) &\leq -\frac{1}{L}\left(\nabla f\left(x_k\right), \nabla f\left(x_k\right)\right) + \frac{1}{2L}\left|\left|\nabla f\left(x_k\right)\right|\right|^2 \\ &= -\frac{1}{2L}\left|\left|\nabla f\left(x_k\right)\right|\right|^2 \end{aligned} \tag{3.2}$$

The algorithm, which the proceeding analysis is based upon, is given below for both the weakly and strongly convex scenarios. The analysis follows a procedure which is analogous to the original proof published by Nesterov in [2].

Algorithm 1: Gradient Descent: Weak & Strong Convexity

```
**Data**: $\alpha = 1/L$: Stepsize (Weakly Convex)
**Data**: $\alpha = 1/\left(\mu + L\right)$: Stepsize (Strongly Convex)
**Data**: $x_0 = y_1$: Initial parameter vector
initialization;
while $x_k$ not converged do
    $k \rightarrow k + 1$;
    $x_{k+1} = x_k - \alpha \cdot \nabla f\left(x_k\right)$    (Update parameters)
end
**Result**: $x_k$ (Resulting parameters)
```

Theorem 3.1. *Assume f is convex and has L-Lipschitz continuous gradient. Then the gradient descent method with step size $\alpha = \frac{1}{L}$ has the following global convergence rate*

$$f\left(x_k\right) - f^* \leq \frac{L}{2k}\|x_0 - x^*\|^2. \tag{3.3}$$

Proof. We begin by subtracting f^* from both sides of (3.2) and take advantage of the convexity of f.

$$\begin{aligned} f\left(x_{k+1}\right) - f^* &\leq f\left(x_k\right) - f^* - \frac{1}{2L}\|\nabla f\left(x_k\right)\|^2 \\ &\leq \langle \nabla f\left(x_k\right), x_k - x^* \rangle - \frac{1}{2L}\|\nabla f\left(x_k\right)\|^2 \end{aligned}$$

Now, use the GD update rule with a time step of $\frac{1}{L}$ and rearrange the resulting inequality.

$$\begin{aligned} f\left(x_{k+1}\right) - f^* &\leq -L\langle x_k - x^*, x_{k+1} - x_k \rangle - \frac{L}{2}\|x_{k+1} - x^*\|^2 \\ &= -\frac{L}{2}\left[x_{k+1} - x^* + x_k - x^*\right] \cdot \left[x_{k+1} - x^* - x_k + x^*\right] \\ &= \frac{L}{2}\left[\|x_k - x^*\|^2 - \|x_{k+1} - x^*\|^2\right] \end{aligned}$$

Sum the inequality resulting from the last line from 0 to $k-1$.

$$\begin{aligned} \sum_{k=0}^{k-1}\left(f\left(x_{k+1}\right) - f^*\right) &\leq \frac{L}{2}\left[\|x_0 - x^*\|^2 - \|x_k - x^*\|^2\right] \\ f\left(x_k\right) - f^* &\leq \frac{L}{2k}\|x_0 - x^*\|^2 \end{aligned}$$

Thus the expected sublinear convergence rate given by (3.3) is obtained. □

3.1.2 Strong Convexity

The method shown here is primarily taken from Nesterov's book on convex optimization [1]. To begin, it is convenient to derive an inequality to use as a starting point for the proof.

$$\begin{aligned}\|x_{k+1}-x^*\|^2 &= \|x_k - x^* - \alpha\nabla f(x_k)\|^2 \\ &= \|x_k-x^*\|^2 - 2\alpha\langle\nabla f(x_k), x_k - x^*\rangle + \alpha^2\|\nabla f(x_k)\|^2 \\ &\le \|x_k-x^*\|^2 - \frac{2\alpha\mu L}{\mu+L}\|x_k-x^*\|^2 + \left(\alpha^2 - \frac{2\alpha}{\mu+L}\right)\|\nabla f(x_k)\|^2 \qquad (3.4)\end{aligned}$$

where the last inequality is due to

$$\langle\nabla f(x_k), x_k - x^*\rangle \ge \frac{\mu L}{\mu+L}\|x_k-x^*\|^2 + \frac{1}{\mu+L}\|\nabla f(x_k)\|^2$$

Theorem 3.2. *Assume f is convex with strong convexity parameter μ. Additionally, f has L-Lipschitz continuous gradient and consider using a step size $\alpha = \frac{2}{\mu+L}$. Then GD convereges linearly as*

$$\|x_{k+1}-x^*\|^2 \le \left(\frac{\mu-L}{\mu+L}\right)^2\|x_k-x^*\|^2. \qquad (3.5)$$

Proof. Setting the step size in (3.4) as $\alpha = \frac{2}{\mu+L}$ leads to

$$\begin{aligned}\|x_{k+1} - x^*\|^2 &\leq \left(1 - \frac{4\mu L}{(\mu + L)^2}\right) \|x_k - x^*\|^2 \\ &\leq \left(\frac{\mu - L}{\mu + L}\right)^2 \|x_k - x^*\|^2.\end{aligned}$$

This implies GD converges linearly with rate $\frac{\mu-L}{\mu+L}$. □

3.2 Nesterov Accelerated Gradient

In recent years, *momentum* methods have come to play a powerful role in optimization. Intuitively, momentum arises by adding a weighted version of the previously iterated step of an algorithm to the current one. Applied to the gradient descent scheme, whose continuous time counterpart is represented by the gradient flow dynamics $\dot{x} = \nabla f(x)$, momentum leads to a second-order DE when transitioning to continuous time. In a sense, one can interpret this physically as a massive particle moving in a potential well as noted in [8]. This system exhibits acceleration due to the presence of a second derivative term. As such momentum methods are said to accelerate an algorithm towards the minimizer and are therefore expected to obtain faster convergence than methods without momentum. In fact, for first order methods the optimal convergence rate is $O(1/k^2)$ which is indeed quicker than gradient descent which achieves $O(1/k)$ convergence. With the publishing of Nesterov's 1983 paper [2], NAG became the only known algorithm to achieve the optimal convergence rate for first order optimization methods. This algorithm relies strongly on gradient descent but rather than simply taking the most recently iterated solution, NAG uses a convex combination of the current and previous solution vectors. In essence, this added step provides the algorithm *momentum* which accelerates the process towards finding the minimizer. The iterative scheme involved in this process is given below:

$$x_{k+1} = y_k - \alpha_k \nabla f(y_k)$$

$$y_{k+1} = x_{k+1} + \beta_k (x_{k+1} - x_k) \tag{3.6}$$

The value for β_k depends on whether the objective is weakly or strongly convex. In [9], $\beta_k = \frac{k-1}{k+2}$ for f weakly convex and in [1], $\beta_k = \frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}$ for strongly convex. In both cases $\alpha_k \leq \frac{1}{L}$ where the optimal choice is $\alpha_k = \frac{1}{L}$.

Algorithm 2: NAG: Weak Convexity

```
Data: $\alpha = 1/L$: Stepsize
Data: $x_0 = y_1$: Initial parameter vector
Data: $a_1 = 1$: initial convexity factor
initialization;
while $x_k$ not converged do
    $k \to k+1$;
    $x_k \to y_k - \alpha \nabla f(y_k)$    (Update parameters;
    $a_{k+1} \to \left(1 + \sqrt{4a_k^2 + 1}\right)/2$    (Update convexity factor)
    $y_{k+1} \to x_k + (a_k - 1)(x_k - x_{k-1})/a_{k+1}$    (Update acceleration step);
end
Result: $x_k$ (Resulting parameters)
```

The algorithm above can be simplified by appropriately choosing a constant time step to use throughout the whole iteration process. Notice that the convexity factor, $\beta_k = \frac{a_k - 1}{a_{k+1}}$, between x_k and x_{k+1} is defined with a_k being quadratic rather than linear as it was prior to the algorithm. This choice is made for the purpose of acquiring the convergence result in the ensuing subsection.

3.2.1 General Convexity

Theorem 3.3. *Assume $f : \mathbb{R}^d \to \mathbb{R}$ is convex, has L-Lipschitz continuous gradient and has a non-empty solution set. Then Nesterov's accelerated gradient descent algorithm satisfies*

$$f\left(x_k\right) - f^* \leq \frac{4L\|y_0 - x^*\|^2}{\left(k+2\right)^2} \tag{3.7}$$

Proof. We begin by mentioning an inequality that can be obtained in the same manner as (3.2) while not choosing a constant time step.

$$f\left(y_k\right) - f\left(x_k\right) \geq \frac{1}{2}\alpha_k\left(2 - \alpha_k L\right)\|\nabla f\left(y-k\right)\|^2 \tag{3.8}$$

Next, let $z_k = \left(a_k - 1\right)\left(x_{k-1} - x_k\right)$. Then $y_{k+1} = x_k - \frac{1}{a_{k+1}}z_k$, which implies $z_{k+1} - x_{k+1} = z_k - x_k + a_{k+1}\alpha_{k+1}\nabla f\left(y_k + 1\right)$. Given this, the following holds

$$\begin{aligned}
\|z_{k+1} - x_{k+1} + x^*\|^2 &= \|z_k - x_k + x^* + x^* + a_{k+1}\alpha_{k+1}\nabla f\left(y_{k+1}\right)\|^2 \\
&= \|z_k - x_k + x^*\|^2 + 2a_{k+1}\alpha_{k+1}\langle\nabla f\left(y_{k+1}\right), z_k - x_k + x^*\rangle | a_{k+1}^2\alpha_{k+1}^2\|\nabla f\left(y_{k+1}\right)\|^2 \\
&= \|z_k - x_k + x^*\|^2 + 2a_{k+1}\alpha_{k+1}\langle\nabla f\left(y_{k+1}\right), z_k\rangle \\
&\quad + 2a_{k+1}\alpha_{k+1}\langle\nabla f\left(y_{k+1}\right), x^* - x_k\rangle + a_{k+1}^2\alpha_{k+1}^2\|\nabla f\left(y_{k+1}\right)\|^2 \\
&= \|z_k - x_k + x^*\|^2 + 2\left(a_{k+1} - 1\right)\alpha_{k+1}\langle\nabla f\left(y_{k+1}\right), z_k\rangle \\
&\quad + 2a_{k+1}\alpha_{k+1}\langle\nabla f\left(y_{k+1}\right), x^* - y_{k+1}\rangle + a_{k+1}^2\alpha_{k+1}^2\|\nabla f\left(y_{k+1}\right)\|^2.
\end{aligned} \tag{3.9}$$

By the definition of convexity and the index condition given by the algorithm there are two more inequalities that can be put to use.

$$\langle \nabla f(y_{k+1}), y_{k+1} - x^* \rangle \geq f(x_{k+1}) - f^* + \frac{\alpha_{k+1}}{2} \|\nabla f(y_{k+1})\|^2 \tag{3.10}$$

$$\frac{\alpha_{k+1}}{2} \|\nabla f(y_{k+1})\|^2 \leq f(y_{k+1}) - f(x_{k+1}) \leq f(x_k) - f(x_{k+1}) - \frac{1}{a_{k+1}} \langle \nabla f(y_{k+1}), z_k \rangle \tag{3.11}$$

Substituting these two inequalities into (3.9) yields

$$\begin{aligned}
\|z_{k+1} - x_{k+1} + x^*\|^2 - \|z_k - x_k + x^*\|^2 &\leq 2(a_{k+1} - 1)\alpha_{k+1} \langle \nabla f(y_{k+1}), z_k \rangle \\
&\quad - 2a_{k+1}\alpha_{k+1}(f(x_{k+1}) - f^*) + \left(a_{k+1}^2 - a_{k+1}\right)\alpha_{k+1}^2 \|\nabla f(y_{k+1})\|^2 \\
&\leq -2a_{k+1}\alpha_{k+1}(f(x_{k+1}) - f^*) + 2\left(a_{k+1}^2 - a_{k+1}\right)\alpha_{k+1}(f(x_k) - f(x_{k+1})) \\
&= 2\alpha_{k+1}a_k^2 f(x_k) - f^*\left(2\alpha_{k+1}a_k^2 - 2\alpha_{k+1}a_{k+1}^2\right) - 2\alpha_{k+1}a_{k+1}^2 f(x_{k+1}) \\
&= 2\alpha_{k+1}a_k^2(f(x_k) - f^*) - 2\alpha_{k+1}a_{k+1}^2(f(x_{k+1}) - f^*) \\
&= 2\alpha_k a_k^2(f(x_k) - f^*) - 2\alpha_{k+1}a_{k+1}^2(f(x_{k+1}) - f^*). \qquad (3.12)
\end{aligned}$$

Finally, the convergence result can be obtained by using (3.12) as shown below.

$$\begin{aligned}
2\alpha_{k+1}a_{k+1}^2(f(x_{k+1}) - f^*) &\leq 2\alpha_{k+1}a_{k+1}^2(f(x_{k+1}) - f^*) + \|z_{k+1} - x_{k+1} + x^*\|^2 \\
&\leq 2\alpha_k a_k(f(x_k) - f^*) + \|z_k - x_k + x^*\|^2 \\
&\leq 2\alpha_0 a_0(f(x_0) - f^*) + \|p_0 - x_0 + x^*\|^2 \\
&\leq \|y_0 - x^*\|^2 \qquad (3.13)
\end{aligned}$$

(3.7) is obtained by noting that the index condition given by the algorithm is satisfied when $\alpha_k \leq \frac{1}{L}$, so the time step will not decrease after this point. Thus, one can assume $\alpha_k \geq \frac{1}{2L}$. Also notice, from the definition for a_{k+1} one has $a_{k+1} \leq a_k \leq 0.5 \leq a_0 + \sum_{k=1}^{k} 0.5 = 1 + 0.5\,(k+1)$. $\square$

3.2.2 Strong Convexity

In constrast to the analysis for weak convexity, a lyapunov style approach is taken here that follows the work done in [8]. The idea was to find a general differential equation for momentum methods and then chose an appropriately discretization so as to obtain an algorithm that is equivalent to Nesterov's original algorithm for strongly convex functions in [1]. The derivation of the DE, which was originally published in [9], is not covered since the interest is to study discrete time convergence. For this purpose only the Lyapunov function and the algorithm are required. A brief overview of Lyapunov functions is covered below including the derivation of the Lyapunov candidate function associated to the dynamics below.

$$Z\,(t) = X\,(t) + \frac{1}{\dot{\beta}\,(t)}\dot{X}\,(t)$$

$$\mu\dot{Z}\,(t) = -\mu\dot{X}\,(t) - \dot{\beta}\,(t)\,\nabla f\,(X)$$

Definition 3.3.1. A function V is called a *Lyapunov candidate function* if it satisfies the following two properties

$$V\,(t) > 0, \qquad \dot{V}\,(t) \leq 0 \qquad \forall t \geq 0$$

Notice that these conditions are listed in continuous time. The implication though is that the Lyapunov function can be discretized appropriately and that these conditions still hold. The purpose of using discretized Lyapunov functions is to show that the error between Lyapunov function at consecutive time steps is bounded and non-positive. The derivation of the Lyapunov candidate function is now presented and is attributed to the work [8]. It is worth noting that the $\dot{\beta}(t)$ and τ_k are related as $\tau_k = \delta\dot{\beta}$ where δ is some chosen discretization spacing. The Lyapunov function under consideration is then

$$V(t) = f(X) - f(x^*) + \frac{\mu}{2}\|x^* - Z(t)\|^2$$

Clearly this function is positive definite since $f(X) \geq f(x^*)$. It remains to show $\dot{V}(t) \leq 0$. This begins by differentiating and using the second dynamics equation.

$$\begin{aligned}\dot{V} &= \langle \nabla f(X), \dot{X}\rangle - \mu\langle \dot{Z}, x^* - X - \frac{1}{\dot{\beta}}\dot{X}\rangle \\ &= \langle \nabla f(X), \dot{X}\rangle + \langle \mu\dot{X} + \dot{\beta}\nabla f(X), x^* - X - \frac{1}{\dot{\beta}}\dot{X}\rangle\end{aligned}$$

Rearranging and using the strong convexity condition gives,

$$\begin{aligned}&= \dot{\beta}\langle \nabla f(x), x^* - X\rangle + \mu\langle \dot{X}, x^* - X - \frac{1}{\dot{\beta}}\dot{X}\rangle \\ &\leq -\dot{\beta}\left(f(X) - f(x^*) + \frac{\mu}{2}\|x^* - X\|^2\right) + \mu\langle \dot{X}, x^* - X - \frac{1}{\dot{\beta}}\dot{X}\rangle\end{aligned}$$

Expanding the term $\|x^* - X\|^2$ and using the first dynamics equation returns the expected result.

$$\begin{aligned}&= -\dot{\beta}\left(f(X) - f(x^*) + \frac{\mu}{2}\|x^* - X - \frac{1}{\dot{\beta}}\dot{X}\|^2\right) - \mu\langle \dot{X}, x^* - X - \frac{1}{\dot{\beta}}\dot{X}\rangle \\ &-\frac{\mu}{2\dot{\beta}}\|\dot{X}\|^2 + \mu\langle \dot{X}, x^* - X - \frac{1}{\dot{\beta}}\dot{X}\langle \\ &\leq -\dot{\beta}\left(f(X) - f(x^*) + \frac{\mu}{2}\|x^* - Z\|^2\right)\end{aligned}$$

Discretizing the Lyapunov function and replacing x_k with y_k, which can be done since the conditions for a Lyapunov function will hold as long as $y_k \in \chi$, yields the function given by (3.14).

Prior to presenting the strongly convex NAG algorithm it is important to point out that in order to ensure this error is bounded an additional sequence y_k is needed in the algorithm. Henceforth, y_k will be known as the *error bounding sequence.*

Algorithm 3: NAG: Strong Convexity

```
Data: $\tau_k = \tau = 1/\sqrt{\kappa}$: Stepsize
Data: $z_0 = x_0$: Initial parameter vector
initialization;
while $x_k$ not converged do
    $y_{k+1} \to x_k - \frac{1}{L}\nabla f(x_k)$    (Update error bounding sequence);
    $k \to k+1$;
    $x_{k+1} \to \frac{1}{1+\tau} y_k + \frac{\tau}{1+\tau} z_k$    (Parameter Update);
    $z_{k+1} \to z_k + \tau\left(x_{k+1} - z_k - \frac{1}{\mu}\nabla f(x_{k+1})\right)$    (Update parameter vector associated with
    derivative discretization);
end
Result: $x_k$ (Resulting parameters)
```

Theorem 3.4. *Using the following Lyapunov function*

$$\tilde{V}_k = f(y_k) - f(x^*) + \frac{\mu}{2}||z_k - x^*||^2 \tag{3.14}$$

it can be shown

$$\tilde{V}_{k+1} - \tilde{V}_k = -\tau_k \tilde{V}_k + \epsilon_{k+1} \tag{3.15}$$

where the error is expressed as

$$\epsilon_{k+1} = \left(\frac{\tau_k^2}{2\mu} - \frac{1}{2L}\right)||\nabla f(x_k)||^2 + \left(\frac{\tau_k L}{2} - \frac{\mu}{2\tau_k}\right)||x_k - y_k||^2 \tag{3.16}$$

Proof. The error is required to be non-positive in order for $\tilde{V}_k$ to satisfy the conditions of the Lyapunov function. This results in the time step to be $\tau_k \leq \frac{1}{\sqrt{\kappa}}$, where equality yields the convergence rate originally obtained by Nesterov for strongly convex functions.

$$\begin{aligned}
\tilde{V}_{k+1} - \tilde{V}_k &= f\left(y_{k+1}\right) - f\left(y_k\right) - \mu\langle z_{k+1} - z_k, x^* - z_{k+1}\rangle - \frac{\mu}{2}||z_{k+1} - z_k||^2 \\
&= f\left(y_{k+1}\right) - f\left(x_k\right) + f\left(x_k\right) - f\left(y_k\right) - \mu\langle z_{k+1} - z_k, x^* - z_k\rangle + \frac{\mu}{2}||z_{k+1} - z_k||^2 \\
&\leq f\left(y_{k+1}\right) - f\left(x_k\right) + \langle\nabla f\left(x_k\right), x_k - y_k\rangle - \frac{\mu}{2}||x_k - y_k||^2 - \mu\langle z_{k+1} - z_k, x^* - z_k\rangle + \frac{\mu}{2}||z_{k+1} - z_k||^2 \\
&= f\left(y_{k+1}\right) - f\left(x_k\right) + \langle\nabla f\left(x_k\right), x_k - y_k\rangle - \frac{\mu}{2}||x_k - y_k||^2 + \frac{1}{\sqrt{\kappa}}\langle\nabla f\left(x_k\right) - \mu\left(x_k - z_k\right), x^* - z_k\rangle \\
&\quad + \frac{\mu}{2}||z_{k+1} - z_k||^2 \\
&= f\left(y_{k+1}\right) - f\left(x_k\right) + \frac{1}{\sqrt{\kappa}}\langle\nabla f\left(x_k\right), x^* - x_k\rangle - \frac{\mu}{2}||x_k - y_k||^2 - \frac{1}{\sqrt{\kappa}}\mu\langle x_k - z_k, x^* - z_k\rangle \\
&\quad + \frac{\mu}{2}||z_{k+1} - z_k||^2 \\
&\leq -\frac{1}{\sqrt{\kappa}}\left(f\left(x_k\right) - f\left(x^*\right) + \frac{\mu}{2}||x^* - x_k||^2\right) + f\left(y_{k+1}\right) - f\left(x_k\right) - \frac{\mu}{2}||x_k - y_k||^2 \\
&\quad - \frac{1}{\sqrt{\kappa}}\mu\langle x_k - z_k, x^* - z_k\rangle + \frac{\mu}{2}||z_{k+1} - z_k||^2
\end{aligned}$$

Both inequalities above come from using the strong convexity property of f. The first equality arises from the final line of Algorithm 3 while the last equality is due to the update for the error bounding sequence.

In the succeeding analysis the first line is due to a simple manipulation of norms while the subsequent two equalities come about due to the final line of Algorithm 3 and the parameter update rule respectively. Both inequalities are a result of (2.6) which is based on the convexity and

Lipschitz continuity of the objective.

$$\begin{aligned}
\tilde{V}_{k+1}-\tilde{V}_k =& -\frac{1}{\sqrt{\kappa}}\left(f\left(x_k\right)-f\left(x^*\right)+\frac{\mu}{2}||x^*-z_k||^2\right)+f\left(y_{k+1}\right)-f\left(x_k\right)-\frac{\mu}{2}||x_k-y_k||^2+\frac{\mu}{2}||z_{k+1}-z_k||^2 \\
& -\frac{\mu}{2\sqrt{\kappa}}||x_k-z_k||^2 \\
=& -\frac{1}{\sqrt{\kappa}}\left(f\left(x_k\right)-f\left(x^*\right)+\frac{\mu}{2}||x^*-z_k||^2\right)+f\left(y_{k+1}\right)-f\left(x_k\right)-\frac{\mu}{2}||x_k-y_k||^2 \\
& +\frac{\mu}{2}||\frac{1}{\sqrt{\kappa}}\left(x_k-z_k\right)||^2-\frac{1}{\sqrt{\kappa}}\langle\nabla f\left(x_k\right),\frac{1}{\sqrt{\kappa}}\left(x_k-z_k\right)\rangle+\frac{1}{2\mu\kappa}||\nabla f\left(x_k\right)||^2-\frac{\mu\sqrt{\kappa}}{2}||x_k-y_k||^2 \\
=& -\frac{1}{\sqrt{\kappa}}\left(f\left(x_k\right)-f\left(x^*\right)+\frac{\mu}{2}||x^*-z_k||^2\right)+f\left(y_{k+1}\right)-f\left(x_k\right)-\frac{1}{\sqrt{\kappa}}\langle\nabla f\left(x_k\right),y_k-x_k\rangle \\
& +\frac{1}{2\mu\kappa}||\nabla f\left(x_k\right)||^2-\frac{\mu\sqrt{\kappa}}{2}||x_k-y_k||^2 \\
\leq& -\frac{1}{\sqrt{\kappa}}\left(f\left(x_k\right)-f\left(x^*\right)+\frac{\mu}{2}||x^*-z_k||^2\right)+f\left(y_{k+1}\right)-f\left(x_k\right)-\frac{1}{\sqrt{\kappa}}\left(f\left(y_k\right)-f\left(x_k\right)\right) \\
& +\frac{1}{2\mu\kappa}||\nabla f\left(x_k\right)||^2+\left(\frac{L}{2\sqrt{\kappa}}-\frac{\mu\sqrt{\kappa}}{2}\right)||x_k-y_k||^2 \\
=& -\frac{1}{\sqrt{\kappa}}\left(f\left(y_k\right)-f\left(x^*\right)+\frac{\mu}{2}||x^*-z_k||^2\right)+f\left(y_{k+1}\right)-f\left(x_k\right)+\frac{1}{2\mu\kappa}||\nabla f\left(x_k\right)||^2 \\
& +\left(\frac{L}{2\sqrt{\kappa}}-\frac{\mu\sqrt{\kappa}}{2}\right)||x_k-y_k||^2 \\
\leq& -\frac{1}{\sqrt{\kappa}}\left(f\left(y_k\right)-f\left(x^*\right)+\frac{\mu}{2}||x^*-z_k||^2\right)+\left(\frac{1}{2\mu\kappa}-\frac{1}{2L}\right)||\nabla f\left(x_k\right)||^2 \\
& +\left(\frac{L}{2\sqrt{\kappa}}-\frac{\mu\sqrt{\kappa}}{2}\right)||x_k-y_k||^2
\end{aligned}$$

Notice that the last two terms are zero since $\kappa=\frac{L}{\mu}$. Therefore the following relation holds

$$\tilde{V}_{k+1} \leq \left(1-\frac{1}{\sqrt{\kappa}}\right)\tilde{V}_k \tag{3.17}$$

Hence, the algorithm has the convergence rate previously mentioned and it matches that of

Nesterov's accelerated scheme for strongly convex functions. □

3.3 Newton's Method

Unlike Gradient Descent and NAG, Newton's method is a second order optimization scheme. This means that information regarding the hessian of the objective is required for computation within the algorithm. Although it can be shown that Newton's method can converge quite quickly near the optimizer, the fact that the Hessian is needed greatly increases the number of operations required to perform the optimization. The iterative scheme looks quite similar to gradient descent, with the timestep from GD seemingly being replaced by the inverse of the hessian.

$$x_{k+1} = x_k - \gamma \nabla f\left(x_k\right) / \nabla^2 f\left(x_k\right) \tag{3.18}$$

The parameter γ allows for control over the step size, however it is normally set to one.

It is important to realize that in general Newton's method is not a global method; convergence is only achieved if the initial parameter vector is "close enough" to the minimizer. Thus, before procceding to problems involving weakly convex objectives, a proof of local quadratic convergence for this method is presented.

Theorem 3.5. *Let f be L-Lipschitz continuous and the smallest eigenvalue of its Hessian is bounded as $|\lambda_{min}| > 0$. Then, as long as $\|x_0 - x^*\|$ is sufficiently small, the sequence generated by Newton's method converges quadratically to x^*.*

Proof. The result is easily obtained by inspecting $\|x_{k+1} - x^*\|$ and using the Lipschitz condition.

$$\begin{aligned}
\|x_{k+1} - x^*\| &= \|x_k - x^* - \nabla^2 f\left(x_k\right)^{-1} \cdot \nabla f\left(x_k\right)\| \\
&= \|\nabla^2 f\left(x_k\right)^{-1} \cdot \left(\nabla f\left(x_k\right) - \nabla^2 f\left(x_k\right)\left(x_k - x^*\right)\right)\| \\
&\leq \|\nabla^2 f\left(x_k\right)^{-1}\| \cdot \|\nabla f\left(x_k\right) - \nabla f\left(x^*\right) - \nabla^2 f\left(x_k\right)\left(x_k - x^*\right)\| \\
&\leq \beta \|\nabla^2 f\left(x_k\right)^{-1}\| \cdot \|x_k - x^*\|^2 \leq \frac{\beta}{\lambda_{min}} \|x_k - x^*\|^2
\end{aligned}$$

When $\frac{\beta}{\lambda_{min}}\|x_0 - x^*\| < 1$ then quadratic converge occurs:

$$\frac{\beta}{\lambda_{min}}\|x_{k+1} - x^*\| \leq \left(\frac{\beta}{\lambda_{min}}\|x_k - x^*\|\right)^2 \tag{3.19}$$

□

3.3.1 General Convexity

Theorem 3.6. *Assume $f : \mathbb{R}^d \to \mathbb{R}$ is convex and meets a local Lipschitz condition such that*

$$\|\nabla f\left(y\right) - \nabla f\left(x\right) - \nabla^2 f\left(x\right) \cdot \left(y - x\right)\| \leq L\left(y - x\right)^T \cdot \nabla^2 f\left(x\right) \cdot \left(y - x\right) \tag{3.20}$$

$\forall x, y \in \mathbb{R}^d$ and $L \geq 1$. Then Newton's method applied to the optimization problem

$$x\left(\lambda\right) = argmin_{x \in \mathbb{R}^d} f\left(x\right) + \frac{\lambda}{2}\|x\|^2 \tag{3.21}$$

converges linearly.

Proof. The proof proceeds by way of induction. Since Newton's method is being applied, the problem listed above is equivalent to

$$\nabla f\left(x\right)+\lambda x=0. \tag{3.22}$$

Thus, the error in the current iteration of the solution is simply $\|\nabla f\left(x_k\right)+\lambda_k x_k\|$. Assume that the approximate error for the k^{th} iteration satisfies

$$\|\nabla f\left(x_k\right)+\lambda_k x_k\|\leq\frac{1}{2L}\lambda_k,$$

which can be assumed to hold for $k=0$ by appropriately choosing λ. Now the goal is to show the error relation above holds for the $(k+1)^{th}$ iteration, where $0\leq\lambda_{k+1}\leq\lambda_k$.

Let $\lambda_{k+1}=\left(1-\eta\right)\lambda_k$ for $\eta\in(0,1]$, and apply Newton's method to the problem

$$\begin{aligned}
\left(x_{k+1}-x_k\right)\cdot\nabla^2 f\left(x_k\right)+\left(1-\eta\right)\lambda_k\left(x_{k+1}-x_k\right) &= -\nabla f\left(x_k\right)-\left(1-\eta\right)\lambda_k x_k\\
\|\left(x_{k+1}-x_k\right)\cdot\nabla f\left(x_k\right)+\left(1-\eta\right)\lambda_k\left(x_{k+1}-x_k\right)\| &= \|-\nabla f\left(x_k\right)-\lambda_k x_k+\eta\lambda_k x_k\|\\
&\leq \|-\nabla f\left(x_k\right)-\lambda_k x_k\|+\eta\lambda_k\|x_k\|\\
&\leq \frac{1}{2L}\lambda_k+\eta\lambda_k\|x_k\|.
\end{aligned}$$

Notice also that

$$\begin{aligned}
\|\left(x_{k+1}-x_k\right)\cdot\nabla^2 f\left(x_k\right)+\left(1-\eta\right)\lambda_k\left(x_{k+1}-x_k\right)\|^2 &= \|\left(x_{k+1}-x_k\right)\cdot\nabla^2 f\left(x_k\right)\|^2\\
&+2\left(1-\eta\right)\lambda_k\left(x_{k+1}-x_k\right)^T\cdot\nabla^2 f\left(x_k\right)\cdot\left(x_{k+1}-x_k\right)+\left(\left(1-\eta\right)\lambda_k\right)^2\|x_{k+1}-x_k\|^2,
\end{aligned}$$

which, by convexity, implies

$$(x_{k+1} - x_k)^T \cdot \nabla^2 f(x_k) \cdot (x_{k+1} - x_k) \leq \frac{1}{2(1-\eta)} \left(\frac{1}{2L} + \eta \|x_k\| \right)^2 \lambda_k.$$

Using the Newton update rule and the convexity and Lipschitz relation one can determine

$$\begin{aligned}
&\|\nabla f(x_{k+1}) + (1-\eta)\lambda_k x_{k+1}\| \\
&= \|\nabla f(x_{k+1}) - \big(\nabla f(x_k) + (x_{k+1} - x_k) \cdot \nabla^2 f(x_k)\big) + \big(\nabla f(x_k) + (x_{k+1} - x_k) \cdot \nabla^2 f(x_k)\big) + (1-\eta)\lambda x_{k+1}\| \\
&= \|\nabla f(x_{k+1}) - \nabla f(x_k) + (x_{k+1} - x_k) \cdot \nabla^2 f(x_k)\| \\
&\leq L (x_{k+1} - x_k)^T \cdot \nabla^2 f(x_k) \cdot (x_{k+1} - x_k) \leq \frac{L}{2(1-\eta)} \left(\frac{1}{2L} + \eta \|x_k\| \right)^2 \lambda_k
\end{aligned}$$

Choosing $\eta = \frac{1}{2L(1+\|x_k\|)}$, one obtains the expected result of

$$\|\nabla f(x_{k+1}) + \lambda_{k+1} x_{k+1}\| \leq \frac{1}{2L} \lambda_{k+1}, \tag{3.23}$$

which confirms the linear convergence rate of Newton's method in this setting. □

Notice that this differs from the local quadratic convergence previously attained for Newton's method. This occurs because the problem setting for weak convexity applies for all $x \in \mathbf{dom} f$, hence it is a globally convergent setting for Newton's method.

Chapter 4

Stochastic Optimization

The algorithms presented in this chapter all rely on a probabilistic procedure to ensure convergence. For this reason, the analysis differs slightly from the chapter on deterministic methods since the concept of *expectation* arises when studying stochastic algorithms. The main goal remains the same, find an upper bound on the difference between the function value at the k^{th} iteration and the minimal value. Since the algorithms call for randomly selecting elements of the parameter vector to update one most consider not the actual value of the objective but the expected value. As in the previous chapter the focus will be on objective functions which satisfy weak and strong convexity conditions. For the final two methods, Adam and Adagrad, only general convexity is considered for the convergence analysis.

4.1 Stochastic Gradient Descent

We begin with one of the simplest stochastic variants of Gradient Descent, hence the name. Stochastic gradient descent involves taking a gradient descent step during each iteration but instead of using the full gradient the algorithm randomly selects a subgradient of the objective. This will decrease the complexity as compared to gradient descent but hopefully retain the same convergence rate.

SGD follows a simple iterative procedure similar to gradient descent. The difference lies in the objective since it will be assumed that $f(x) = \sum_{i=1}^{d} f_i(x)$. From here subgradients of the objective can be defined as $\nabla f_i(x)$ which leads to the following iteration procedure.

$$x_{k+1} = x_k - \alpha_k \nabla f_i(x_k) \tag{4.1}$$

The index i is the aforementioned random variable which can take values in the set $i \in \{1, 2, \ldots, d\}$. The SGD algorithm for both weak and strong convex scenarios is listed below and the resulting analysis will be mainly based on the work done in [10]. The projection operator present in the algorithm becomes useful when investigating the convergence for strongly convex objective functions.

Algorithm 4: SGD: Weak & Strong Convexity

```
Data: $\alpha_0$: Initial Stepsize
Data: $x_0$: Initial parameter vector
initialization;
while $x_k$ not converged do
    Select $g_k$ (Subgradient chosen for current iteration);
    $x_{k+1} = \prod_{\chi}(x_k - \alpha_k g_k)$    (Update parameters) $k \to k+1$;
end
Result: $x_k$ (Resulting parameters)
```

4.1.1 General Convex

Theorem 4.1. *Assume f is convex with unbiased subgradient $\tilde{g}_k$ such that $\mathbb{E}\left[\tilde{g}_k | x_k\right] = g_k \in \partial f\left(x_k\right)$ also,*

$$\mathbb{E}\left[||g_k||^2\right] \leq C^2 \qquad \mathbb{E}\left[||x_1 - x^*||^2\right] \leq R^2 \qquad \forall k$$

and that $\alpha_k \geq 0$, $\sum_{k=1}^{\infty} \alpha_k^2 = ||\alpha||^2 < \infty$ and $\sum_{k=1}^{\infty} \alpha_k = \infty$ which is taken to imply $\alpha_k = \mathcal{O}\left(1/k\right)$. Then,

$$\min_{i=1,\ldots,k} \mathbb{E}\left[f\left(x_i\right) - f^*\right] \leq \frac{R^2 + C^2 \left||\alpha|\right|^2}{4 \sum_{i=1}^{k} \alpha_i} \tag{4.2}$$

Proof. Taking the expectation of the squared norm of the difference, $(x_{k+1} - x^*)$ and substituting the update rule one obtains;

$$\begin{aligned}
\mathbb{E}\left[||x_{k+1} - x^*||^2 | x_k\right] &= \mathbb{E}\left[||x_k - \alpha_k \tilde{g}_k - x^*|| \, | x_k\right] \\
&= ||x_k - x^*||^2 - 2\alpha_k \mathbb{E}\left[\tilde{g}_k^T \left(x_k - x^*\right) | x_k\right] + \alpha_k^2 \mathbb{E}\left[||\tilde{g}_k||^2 | x_k\right] \\
&= ||x_k - x^*||^2 - 2\alpha_k \mathbb{E}\left[\tilde{g}_k | x_k\right]\left(x_k - x^*\right) + \alpha_k^* \mathbb{E}\left[||\tilde{g}_k||^2 | x_k\right] \\
&\leq ||x_k - x^*||^2 - 2\alpha_k \left(f\left(x_k\right) - f^*\right) + \alpha_k^2 \mathbb{E}\left[||\tilde{g}_k||^2 | x_k\right]
\end{aligned}$$

The third line uses the multiplicative property of the conditional expectation and the inequality arises from the convexity of f. Taking the expectation of both sides then yields;

$$\mathbb{E}\left[||x_{k+1} - x^*||^2\right] \leq \mathbb{E}\left[||x_k - x^*||^2\right] - 2\alpha_k \mathbb{E}\left[f\left(x_k\right) - f^*\right] + \alpha_k^2 \mathbb{E}\left[||\tilde{g}_k||^2\right]$$

If one applys the above inequality recursively and uses the assumption from the theorem that $\mathbb{E}\left[||\tilde{g}_k||^2\right] \leq C^2$ then;

$$0 \leq \mathbb{E}\left[||x_1 - x^*||^2\right] - 2\sum_{i=1}^{k} \alpha_i \mathbb{E}\left[f(x_i) - f^*\right] + C^2 \sum_{i=1}^{k} \alpha_i^2$$

Simply rearranging the last inequality one achieves the desired result. □

4.1.2 Strong Convexity

As previously menitoned, the work presented in this section is primarily based on the analysis in [10] which shows that convergence of SGD for strongly convex objectives achieves the optimal convergence of $O(1/k)$. Before continuing to the theorem, the definition of λ-smoothness is presented here in the form it will be used in what follows.

Definition 4.1.1. A function F is λ-smooth with respect to x^* over a convex set $\mathcal{X}$ if $\forall x \in \mathcal{X}$,

$$F(x) - F^* \leq \frac{\lambda}{2} ||x - x^*||^2 \tag{4.3}$$

The theorem regarding the convergence rate for strongly convex objectives and its resulting proof are now presented.

Theorem 4.2. *Suppose f is μ-strongly convex and λ smooth with respect to x^* over a convex set $\mathcal{X}$ and $\mathbb{E}\left[||\tilde{g}_k||^2\right] \leq C^2$. Then choosing $\alpha_k = \frac{1}{\mu k}$ the following holds $\forall k$*

$$\mathbb{E}\left[f\left(x_k\right) - f^*\right] \leq \frac{2\lambda C^2}{\mu^2 k}$$

To begin, one can prove $\mathbb{E}\left[x_k - x^*\right] \leq \frac{4C^2}{\mu^2 k}$, holds and then apply the definition of λ-smoothness. This inequality can be proved by induction.

Proof. To confirm the base case ($k = 1$) holds, take the squared norm of both sides of the strong convexity condition as shown below.

$$||g_1||^2 \, ||x_1 - x^*||^2 \geq ||\langle g1, x_1 - x^*\rangle||^2 \geq \frac{\mu^2}{4} \, ||x_1 - x^*||^4$$

$$||g_1||^2 \geq \frac{\mu^2}{4} \, ||x_1 - x^*||^2$$

Using the triangle inequality one can easily show that $\mathbb{E}\left[||\tilde{g}_1||^2\right] \geq \mathbb{E}\left[||g_1||^2\right]$. Thus, taking the expectation of both sides and using the assumption from the theorem;

$$\mathbb{E}\left[||x_1 - x^*||^2\right] \leq \frac{4}{\mu^2}\mathbb{E}\left[||\tilde{g}_1||^2\right] \leq \frac{4C^2}{\mu^2}$$

So the inequality holds for the base case. Now assume it holds for some $k = m$ and consider $k = m + 1$.

$$\begin{aligned}\mathbb{E}\left[||x_{m+1}-x^*||^2\right] &= \mathbb{E}\left[||\Pi_{\mathcal{X}}\left(x_m-\alpha_m\tilde{g}_m\right)-x^*||^2\right]\\ &\leq \mathbb{E}\left[||x_m-\alpha_m\tilde{g}_m-x^*||^2\right]\end{aligned}$$

The first line uses the update rule where the notation $\Pi_{\mathcal{X}}\left(y\right)$ denotes the projection of y onto $\mathcal{X}$. The inequality is results from the convexity of $\mathcal{X}$ ensuring $||\Pi_{\mathcal{X}}\left(y\right)-x||\leq||y-x||$.

Expanding the inequality above and using $\mathbb{E}\left[\tilde{g}_m\right]=g_m=\mathbb{E}\left[g_m\right]$ along with strong convexity gives;

$$\begin{aligned}&= \mathbb{E}\left[||x_m-x*||^2\right]-2\alpha_m\mathbb{E}\left[\langle g_m,x_m-x^*\rangle\right]+\alpha_m^2\mathbb{E}\left[||\tilde{g}_m||^2\right]\\ &\leq \mathbb{E}\left[||x_m-x*||^2\right]-2\alpha_m\mathbb{E}\left[f\left(x_m\right)-f^*+\frac{\mu}{2}||x_m-x^*||^2\right]+\alpha_m^2C^2\\ &\leq \mathbb{E}\left[||x_m-x^*||^2\right]-2\alpha_m\mathbb{E}\left[\frac{\mu}{2}||x_m-x^*||^2+\frac{\mu}{2}||x_m-x^*||^2\right]+\alpha_m^2C^2\\ &= \left(1-2\alpha_m\mu\right)\mathbb{E}\left[||x_m-x^*||^2\right]+\alpha_m^2C^2\end{aligned}$$

Choosing the step size as $\alpha_m=\frac{1}{\lambda m}$, and using the induction assumption gives,

$$\mathbb{E}\left[||x_{m+1}-x^*||^2\right]\leq\left(1-\frac{2}{m}\right)\mathbb{E}\left[||x_m-x^*||^2\right]+\frac{C^2}{\mu^2m^2}\qquad\leq\frac{4C^2}{\mu^2m}-\frac{8C^2}{\mu^2m}+\frac{C^2}{\mu^2m^2}\leq\frac{4C^2}{\mu^2m}$$

Using the smoothness condition then gives the required result. □

4.2 Randomized Coordinate Gradient Descent

The convergence analysis to be discussed in the two subsequent sections is prominently based on work done by Nesterov in [3]. As suggested by the name, Randomized Coordinate Descent (RCD) and its accelerated counterpart (ARCD) are applied coordinate-wise on the objective. Rather than updating all parameters using a random subgradient, coordinate descent methods select a dimension of the gradient and descend only in that direction during the current iteration. Thus, a general form of the $k+1$ iterate of this process is

$$x_{i_{k+1}} = x_{i_k} - \alpha_k \nabla f_{i_k}\left(x_k\right) \tag{4.4}$$

Here, i_k denotes the i^{th} corrdinate of the k^{th} iteration. Due to the clear depenedence on corrdinate-wise operations of RCGD a definition of Lipschitz continuity that fits this setting will be quite useful.

Definition 4.2.1. Coordinate-wise Lipschitz continuity A function $f : \mathbb{R}^d \to \mathbb{R}$ has coordinate-wise Lipschitz continuous gradient with respect to coordinate i if

$$\|f_i'\left(x + \alpha e_i\right) - f_i\left(x\right)\| \leq L_i \|\alpha e_i\|, \tag{4.5}$$

or equivalently

$$f\left(x + \alpha e_i\right) \leq f\left(x\right) + \alpha\langle f_i'\left(x\right), e_i\rangle + \frac{\alpha L_i}{2}\|e_i\|^2 \tag{4.6}$$

$$f\left(x\right) - f\left(y_i\right) \geq \frac{1}{2L_i}\|f_i'\left(x\right)\|^2 \tag{4.7}$$

where $y_i\left(x\right) = x - \frac{1}{L_i} f_i'\left(x\right)$.

In [3] a random counter $R_\alpha, \alpha \in \mathbb{R}$ is defined which generates an integer $i \in \{1, \dots, n\}$ with probability that depends on the probability distribution assigned by α. For simplicity, $\alpha = 0$ is the distribution considered and corresponds to a uniform distribution. The random variable associated to the sequence of integers is denoted as $\xi_k = \{i_0, \dots, i_k\}$ and the expected value is written as $\phi_k = \mathbb{E}_{\xi_{k-1}} f(x_k)$.

Algorithm 5: RCD: Randomized Coordinate Descent

```
Data: $x_0$: Initial parameter vector
initialization;
while $x_k$ not converged do
    $k \to k+1$;
    $i_k \to R_0$    (Randomly select coordinate to "descend");
    $x_{k+1} \to x_k - \frac{1}{L_{i_k}} U_{i_k} f'_{i_k}(x_k)$    (Update parameters)
end
Result: $x_k$ (Resulting parameters)
```

Access to coordinate-wise Lipschitz constants is usually not available, instead a constant time step is used in practice which is equivalent to replacing L_{i_k} with an upper bound on all Lipschitz factors.

4.2.1 General Convexity

Theorem 4.3. *For $f : \mathbb{R}^d \to \mathbb{R}$ convex, with coordinate-wise Lipschitz continuous gradient and a non-empty solution set the RCD method converges in the following manner for any $k \geq 0$.*

$$\phi_k - f^* \leq \frac{2n}{k+4} R_1^2(x_0), \tag{4.8}$$

where $R_1(x_0) = \max_x\{\max_{x^* \in \chi^*} \|x - x^*\|_1 : f(x) \leq f(x_0)\}$.

Proof. If a sequence of integers i_k is generated by the RCD method then

$$\begin{aligned} f\left(x_k\right)-\mathbb{E}_{i_k}\left(f\left(x_{k+1}\right)\right) &= \mathbb{E}_{i_k}\left[f\left(x_k\right)-f\left(x_{k+1}\right)\right] \\ &= \sum_{i=1}^{n} p^{(i)}\cdot\left[f\left(x_k\right)-f\left(T_i\left(x_k\right)\right)\right] \\ &\geq \sum_{i=1}^{n}\frac{p^{(i)}}{2L_i}\|f_i'\left(x_k\right)\|_1^2 = \frac{1}{2n}\|\nabla f\left(x_k\right)\|_1^2 \end{aligned} \tag{4.9}$$

where $p^{(i)}$ denotes the probability of occurrence for each i_k. The first inequality is due to cooredinate-wise Lipschitz continuity and the last line is a result of the probability distribution being uniform. Now, by convexity, the Cauchy Schwarz inequality and $f\left(x_k\right)\leq f\left(x_0\right)$ one has

$$f\left(x_k\right)-f^*\leq\left\langle\nabla f\left(x_k\right),x_k-x^*\right\rangle\leq\|\nabla f\left(x_k\right)\|_1 R_1\left(x_0\right), \tag{4.10}$$

substituting this inequality into previous expression yields,

$$f\left(x_k\right)-\mathbb{E}_{i_k}\left(f\left(x_{k+1}\right)\right)\geq\frac{1}{2nR_1^2\left(x_0\right)}\left(f\left(x_k\right)-f^*\right)^2. \tag{4.11}$$

Take the expectation of both sides with respect to ξ_{k-1}

$$\phi_k-\phi_{k+1}\geq\frac{1}{2nR_1^2\left(x_0\right)}\mathbb{E}_{\xi_{k-1}}\left[\left(f\left(x_k\right)-f^*\right)^2\right]\geq\frac{1}{2nR_1^2\left(x_0\right)}\left(\phi_k-f^*\right)^2.$$

This implies the following

$$\frac{1}{\phi_{k+1}-f^*}-\frac{1}{\phi_k-f^*}=\frac{\phi_k-\phi_{k+1}}{\left(\phi_{k+1}-f^*\right)\left(\phi_k-f^*\right)}\geq\frac{\phi_k-\phi_{k+1}}{\left(\phi_k-f^*\right)^2}\geq\frac{1}{2nR_1^2\left(x_0\right)}. \tag{4.12}$$

Applying this relation recursively and noting that $\phi_0=f\left(x_0\right)$ and $f\left(x_0\right)-f^*\leq\frac{1}{2n}\|x_0-x^*\|_1^2$,

$$\frac{1}{\phi_k-f^*}\geq\frac{1}{\phi_0-f^*}+\frac{k}{2nR_1^2\left(x_0\right)}\geq\frac{k+4}{2nR_1^2\left(x_0\right)} \tag{4.13}$$

□

4.2.2 Strong Convexity

The strongly convex case is very much related to general convexity. The difference between the two is similar to the difference between strong and weak convexity for gradient descent. For this reason the analysis begins from an inequality shown for weak convexity as mentioned in the proof.

Theorem 4.4. *Let $f : \mathbb{R}^d \to \mathbb{R}$ be strongly convex with convexity parameter $\mu > 0$, have coordinate-wise Lipschitz continuous gradient and non-empty solution set. Then RCD satisfies the convergence relation*

$$\phi_k - f^* \leq \left(1 - \frac{\mu}{n}\right)^k (f(x_0) - f^*) \tag{4.14}$$

Proof. The first inequality below is do to equation 4.9 from the weak convex case. To obtain a tighter bound use the strong convexity condition $f(x_k) - f^* \leq \frac{1}{2\mu}\|\nabla f(x_k)\|^2$.

$$f(x_k) - \mathbb{E}[f(x_{k+1})] \geq \frac{1}{2n}\|\nabla f(x_k)\|_1^2 \geq \frac{\mu}{n}(f(x_k) - f^*)$$

Take the expectation with respect to ξ_{k+1} and subtract f^* frm both sides.

$$\begin{aligned} \phi_{k+1} - f^* &\leq \left(1 - \frac{\mu}{n}\right)(\phi_k - f^*) \\ \phi_k - f^* &\leq \left(1 - \frac{\mu}{n}\right)^k (\phi_0 - f^*) = \left(1 - \frac{\mu}{n}\right)^k (f(x_0) - f^*) \end{aligned} \tag{4.15}$$

The last line arises from applying the first line recursively. □

4.3 Accelerated Randomized Coordinate Gradient Descent

As one may deduce, the difference between RCGD and ARCD is anagolous to the difference between gradient descent and NAG. Transitioning from RCGD requires adding *momentum* to the algorithm in the same manner as in Nesterov's original accelerated gradient method. Notice however that accelerating RCD forces the method of estimate sequences, originally proposed in [1], to appear in the algorithm. Convergence of NAG in the strongly convex case was proved to converge without such a technique in [9], however it requires using discretized Lyapunov functions which are not accessible in the stochastic domain. The iterative scheme is essentially contained in Algorithm 6 through the computation of y_k, v_{k+1} and x_{k+1}.

A final note on notation used in [3] is in order before proceeding to the algorithm. The symbol $s^{\#}$ is defined by

$$s^{\#} \in \underset{x}{\operatorname{argmax}} \left[\langle s, x \rangle - \frac{1}{2\|x\|^2} \right]$$

For large problems one need not restrict the selection to one single coordinate. Algorithm 6 includes this case by considering a decomposition of $\mathbb{R}^d$ as done by [3].

$$\mathbb{R}^d = \otimes_{i=1}^{n} \mathbb{R}^{n_i} \qquad\qquad d = \sum_{i=1}^{n} n_i$$

A partition of the unit matrix along with any vector x can be written as follows.

$$\mathbb{I}_d = (U_1, \dots, U_n) \in \mathbb{R}^{d\times d} \qquad x = \sum_{i=1}^{n} U_i x^{(i)} \qquad x^{(i)} \in \mathbb{R}^{n_i} \qquad U_i \in \mathbb{R}^{d\times n_i} \qquad i = 1, \dots, n$$

The final piece needed is to define the optimal coordinate step, which is again defined in [3] as

$$T_i\left(x\right) = x - \frac{1}{L_i} U_i f_i'\left(x\right)^{\#}$$

Algorithm 6: ARCD: Accelerated Randomized Coordinate Descent

```
Data: $v_0 = x_0$: Initial parameters
Data: $a_0 = 1/n$, $b_0 = 2$
initialization;
while $x_k$ not converged do
    $k \to k+1$;
    $\gamma_k \to \left(1 - \sigma\gamma_{k-1} \pm \sqrt{\left(1-\sigma\gamma_{k-1}\right)^2 - 4n^2\gamma_{k-1}}\right) / (2n) \quad (\gamma_k \geq 1/n)$;
    $\alpha_k \to (n - \gamma_k\sigma) / (\gamma_k (n^2 - \sigma))$ (Coefficient of estimate sequence);
    $\beta_k \to 1 - (\gamma_k\sigma)/n$ (Coefficient of estimate sequence);
    $y_k \to \alpha_k v_k + (1-\alpha_k) x_k$ (Compute accelerated parmeter vector);
    $i_k \to R_0$ (Randomly selected coordinate to descend);
    $v_{k+1} \to \beta_k v_k + (1-\beta_k) y_k - \frac{\gamma_k}{L_{i_k}} U_{i_k} f_i'(y_k)^{\#}$ (Update estimating sequence);
    $x_{k+1} \to y_k - \frac{1}{L_{i_k}} U_{i_k} f_{i_k}'(x_k)^{\#}$ (Update parameters);
    $b_{k+1} = \frac{b_k}{\sqrt{\beta_k}}$, $a_{k+1} = \gamma_k b_{k+1}$ (Update coefficients for estimating sequence)
end
Result: $x_k$ (Resulting parameters)
```

It is important to note that one is required to know both μ and all coordinate -wise Lipschitz factors for the gradients. As one might expect, this is not usually the case. Fortunately, for certain problems μ is not too difficult to determine and one can use the largest Lipschitz factor to implement a constant time step in practice.

The analysis shown below incorporates both the weak and strong convex cases into one theorem. To obtain the convergence result for weak convexity simply set $\mu = 0$. In addition, a relation between the coefficients a_k and b_k that will be useful to prove the theorem below is now introduced. By definitions of a_k and b_k from the algorithm one has,

$$b_k^2 = \beta_k b_{k+1}^2 = \left(1 - \frac{\sigma}{n}\gamma_k\right) b_{k+1}^2 = \left(1 - \frac{\sigma}{n}\frac{a_{k+1}}{b_{k+1}}\right) b_{k+1}^2$$

which implies $b_{k+1} \geq b_k + \frac{\sigma}{2n} a_k$. Also

$$\frac{a_{k+1}^2}{b_{k+1}^2} - \frac{a_{k+1}}{nb_{k+1}} = \frac{\beta_k a_k^2}{b_k^2} = \frac{a_k^2}{b_{k+1}^2},$$

similarly this leads to $a_{k+1} \geq a_k + \frac{1}{2n} b_k$. Now it is possible to prove by induction that

$$a_k \geq \frac{1}{\sqrt{\sigma}} \left(H_1^{k+1} - H_2^{k+1} \right) \qquad b_k \geq H_1^{k+1} + H_2^{k+1},$$

where $H_1 = 1 + \frac{\sqrt{\sigma}}{2n}$ and $H_2 = 1 - \frac{\sqrt{\sigma}}{2n}$.

Obviously these relations hold for $k = 0$ by the definitions given in the algorithm. Now assume they hold for $k = m$ and prove they hold true for $k = m + 1$.

$$a_{m+1} \geq a_m + \frac{1}{2n} b_m = \frac{1}{\sqrt{\sigma}} \left(H_1^{m+1} - H_2^{m+1} \right) + \frac{1}{\sqrt{\sigma}} \left(H_1^{m+1} - H_2^{m+1} \right) = \frac{1}{\sqrt{\sigma}} \left(H_1^{m+2} - H_2^{m+2} \right)$$

$$b_{m+1} \geq b_m + \frac{\sigma}{2n} a_m = H_1^{m+1} + H_2^{m+1} + \frac{\sigma}{2n} \frac{1}{\sqrt{\sigma}} \left(H_1^{m+1} - H_2^{m+1} \right) = H_1^{m+2} + H_2^{m+2}$$

Theorem 4.5. *Let $f : \mathbb{R}^d \to \mathbb{R}$ have coordinate-wise Lipschitz continuous gradient, a non-empty solution set and has convexity parameter $\mu \geq 0$. Then the ARCD method converges with the following rate*

$$\phi_k - f^* \leq \left(\frac{n}{k+1} \right)^2 \left[2\|x_0 - x^*\|^2 + \frac{1}{n^2} \left(f\left(x_0\right) - f^* \right) \right] \tag{4.16}$$

Proof. Assume x_k and v_k are the implementations of corresponding random variables generated by ARCD. Let

$$r_k^2 = \|v_k - x^*\|_1^2 \qquad v_k = y_k + \frac{1 - \alpha_k}{\alpha_k} \left(y_k - x_k \right),$$

where y_k is defined by the ARCD algorithm. Then consider

$$\begin{aligned}
r_{k+1}^2 &= \|\beta_k v_k + (1-\beta_k)\, y_k - \frac{\gamma_k}{L_{i_k}} U_{i_k} f_i'(y_k) - x^*\|^2 \\
&= \|\beta_k v_k + (1-\beta_k)\, y_k - x^*\|_1^2 + \frac{\gamma_k^2}{L_{i_k}} \|f_{i_k}(y_k)\|_{i_k}^2 + \frac{2\gamma_k}{L_{i_k}} \langle f_{i_k}'(y_k), (x^* - \beta_k v_k - (1-\beta_k)\, y_k)^{(i_k)} \rangle \\
&\le \|\beta_k v_k + (1-\beta_k)\, y_k - x^*\|_1^2 + 2\gamma_k^2 \left(f(y_k) - f(T_{i_k}(y_k))\right) \\
&+ 2\gamma_k \langle f_{i_k}'(y_k), \left(x^* - y_k + \frac{\beta_k(1-\alpha_k)}{\alpha_k}(x_k - y_k)\right)^{(i_k)} \rangle
\end{aligned}$$

The last equality arises by replacing the 1-norm with the norm with respect to the random variable i_k. The final line is a result of the convexity condition noted at the beginning of the RCD section. Taking the expectation with respect to i_k and then using the strong convexity condition gives,

$$\begin{aligned}
\mathbb{E}_{i_k}\left[r_{k+1}^2\right] &\le \beta_k r_k^2 + (1-\beta_k)\|y_k - x_*\|_1^2 + 2\gamma_k^2 \left(f(y_k) - \mathbb{E}_{i_k}\left[f(x_{k+1})\right]\right) \\
&+ \frac{2\gamma_k}{n} \rangle \nabla f(y_k), x^* - y_k + \frac{\beta_k(1-\alpha_k)}{\alpha_k}(x_k - y_k) \rangle \\
&\le \beta_k^2 r_k^2 + (1-\beta_k)\|y_k - x^*\|^2 + 2\gamma_k^2 \left(f(y_k) - \mathbb{E}_{i_k}\left[f(x_{k=1})\right]\right) \\
&+ \frac{2\gamma_k}{n}\left[f^* - f(y_k) - \frac{\mu}{2}\|y_k - x^*\|_1^2 + \frac{\beta_k(1-\alpha_k)}{\alpha_k}\left(f(x_k) - f(y_k)\right)\right]
\end{aligned}$$

$$\begin{aligned}
\mathbb{E}_{i_k}\left[r_{k+1}^2\right] &= \beta_k r_k^2 - 2\gamma_k^2 \mathbb{E}_{i_k}\left[f(x_{k+1})\right] + \left(1 - \beta_k - \frac{\mu\gamma_k}{n}\right)\|y_k - x^*\|_1^2 \\
&+ \left(2\gamma_k^2 - \frac{2\gamma_k}{n} + \frac{2\gamma_k}{n}\frac{\beta_k(1-\alpha_k)}{\alpha_k}\right) f(y_k) + \frac{2\gamma_k}{n}\left[f^* + \frac{\beta_k(1-\alpha_k)}{\alpha_k} f(x_k)\right] \\
&= \beta_k r_k^2 - 2\gamma_k^2 \mathbb{E}_{i_k}\left[f(x_{k+1})\right] + \frac{2\gamma_k}{n}\left(f^* + \frac{\beta_k(1-\alpha_k)}{\alpha_k} f(x_k)\right)
\end{aligned}$$

The last line is due to $\beta_k = 1 - \frac{\sigma\gamma_k}{n}$, $\gamma_k^2 - \frac{\gamma_k}{n} = \frac{\beta_k\gamma_k(1-\alpha_k)}{n\alpha_k}$, both of which can be derived from the algorithm. Also from the algorithm

$$b_{k+1}^2 = \frac{b_k^2}{\beta_k}, \qquad a_{k+1}^2 = \gamma_k^2 b_{k+1}^2, \qquad \frac{a_k^2}{b_{k+1}^2} = \frac{\gamma_k \beta_k (1-\alpha_k)}{n\alpha_k}.$$

Using these relations, multiply both sides of the previous inequality by b_{k+1}^2 and take the expectation in ξ_{k-1}.

$$\begin{aligned} b_{k+1}^2 \mathbb{E}_{i_k}\left[r_{k+1}^2\right] &\leq b_k^2 r_k^2 - 2a_{k+1}^2 \left(\mathbb{E}_{i_k}\left[f\left(x_{k+1}\right)\right] - f^*\right) + 2a_k^2\left(f\left(x_k\right) - f^*\right) \\ 2a_{k+1}^2\left(\phi_{k+1} - f^*\right) &\leq 2a_k^2\left(\phi_k - f^*\right) + b_k^2 r_k^2 \\ &\leq 2a_0^2\left(f\left(x_0\right) - f^*\right) + b_0^2 \|x_0 - x^*\|_1^2 \end{aligned}$$

Using the relation between a_{k+1}^2,a_0^2 and b_0^2 at the beginning of the section the expected convergence result is easily obtained. □

4.4 ADAM

Adam, which stands for Adaptive Moment Estimation, is a stochastic optimization developed by Kingma and Ba [6]. Instead of using the gradient to update the parameters, Adam uses a ratio of estimated bias-corrected moments of a randomly chosen subgradient of f. The moments that are used are the mean (first moment) and the uncentered variance (second raw moment). Like all other methods discussed in this report, the process is iterative and the parameters are updated by

$$x_k = x_{k-1} - \alpha \frac{\hat{m}_k}{\sqrt{\hat{v}_k} + \epsilon}, \tag{4.17}$$

where $\hat{m}_k$ and $\hat{v}_k$ denote the bias corrected first and second raw moments respectively.

Algorithm 7: Adam: Adaptive Moment Estimation

```
Data: $\alpha$: Stepsize
Data: $\beta_1, \beta_2 \in [0.1)$: Exponential decay rates for the moment estimates
Data: $f(x)$: Stochastic objective function with parameters $x$
Data: $x_0$: Initial parameter vector
initialization;
while $x_k$ not converged do
    $k \to k + 1$;
    $g_k \to \nabla_x f_k(x_{k-1})$    (Get gradients w.r.t stochastic objective at timestep k);
    $m_k \to \beta_1 \cdot m_{k-1} + (1 - \beta_1) \cdot g_k$    (Update biased first moment estimate)
    $v_k \to \beta_2 \cdot v_{k-1} + (1 - \beta_2) \cdot g_k^2$    (Update biased second raw moment estimate);
    $\hat{m}_k \to m_k / \left(1 - \beta_1^k\right)$    (Compute bias-corrected first moment estimate);
    $\hat{v}_k \to v_k / \left(1 - \beta_2^k\right)$    (Compute bias-corrected second raw moment estimate);
    $x_k \to x_{k-1} - \alpha \cdot \hat{m}_k / \left(\sqrt{\hat{v}_k} + \epsilon\right)$    (Update parameters);
end
Result: $x_k$ (Resulting parameters)
```

The algorithm and subsequent theorem for convergence are a product of [6]. It is worth noting that all operations in the algorithm are done element-wise on the vectors. As such, the convergence result for Adam is taken with respect to the regret since expected values of the gradients, g_k, of the stochastic objective is difficult to compute. The regret is defined as

$$R(K) = \sum_{k=1}^{K} \left[f_k(x_k) - f_k(x^*)\right]. \tag{4.18}$$

Two useful lemmas are listed below which are taken from [6]. The same notation previously used for gradients is once again applied (i.e. $g_k = \nabla f_k(x_k)$). Additionally, the i^{th} element of the vector g_k is denoted $g_{k,i}$ and the vector containing the i^{th} element over all iterations is written as $g_{1:K,i} = [g_{1,i}, g_{2,i}, \ldots, g_{k,i}]$.

Lemma 4.6. *Assume the gradients g_k are bounded such that $\|g_k\|_2 \leq G$ and $\|g_t\|_\infty \leq G_\infty$ then*

$$\sum_{k=1}^{K} \sqrt{\frac{g_{k,i}^2}{k}} \leq 2G_\infty \|g_{1:K,i}\|_2 \tag{4.19}$$

Proof. The proof follows by way of induction over K.

For $K = 1$, one has $\sqrt{g_{1,i}^2} \leq 2_\infty \|g_{1,i}\|_2$. Now assume the relation holds for $K = k - 1$, then

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

From, $\|g_{1:K,i}\|_2^2 - g_{K,i}^2 + \frac{g_{K,i}^4}{4\|g_{1:K,i}\|_2^2} \geq \|g_{1:K,i}\|_2^2 - g_{K,i}^2$, and taking the square root of both sides

$$\begin{aligned}\sqrt{\|g_{1:K,i}\|_2^2 - g_{K,i}^2} &\leq \|g_{1:T,i}\|_2 - \frac{g_{K,i}^2}{2\|g_{1:K,i}\|_2} \\ &\leq \|g_{1:K,i}\|_2 - \frac{g_{K,i}^2}{2\sqrt{KG_\infty^2}}\end{aligned}$$

Rearranging the inequality and substituting for the left hand side term,

$$2G_\infty \sqrt{\|g_{1:K,i}\|_2^2 - g_K^2} + \sqrt{\frac{g_{K,i}^2}{K}} \leq 2G_\infty \|g_{1:K,i}\|_2 \tag{4.20}$$

□

Lemma 4.7. *Assume the gradients are bounded as above. Let* $\gamma = \frac{\beta_1^2}{\sqrt{\beta_2}}$ *for* $\beta_1, \beta_2 \in [0, 1)$ *such that* $\gamma < 1$. *Then*

$$\sum_{k=1}^{K} \frac{\hat{m}_{k,i}^2}{\sqrt{k\hat{v}_{k,i}^2}} \leq \frac{2}{1-\gamma} \frac{1}{\sqrt{1-\beta_2}} \|g_{1:K,i}\|_2 \tag{4.21}$$

Proof. Under the assumption, $\frac{\sqrt{1-\beta_2^k}}{\left(1-\beta_1^k\right)^2} \leq \frac{1}{(1-\beta_1)^2}$. Expand the last term in the summation using the update rules from the Adam algorithm.

$$\begin{aligned}
\sum_{k=1}^{K} \frac{\hat{m}_{k,i}^2}{\sqrt{k\hat{v}_{k,i}}} &= \sum_{k=1}^{K-1} \frac{\hat{m}_{k,i}^2}{\sqrt{k\hat{v}_{k,i}}} + \frac{\sqrt{1-\beta_2^T}}{\left(1-\beta_1^T\right)^2} \frac{\left(\sum_{n=1}^{K} (1-\beta_1) \beta_1^{K-n} g_{n,i}\right)^2}{\sqrt{K \sum_{j=1}^{K} (1-\beta_2) \beta_2^{K-j} g_{j,i}^2}} \\
&\leq \sum_{k=1}^{K-1} \frac{\hat{m}_{k,i}^2}{\sqrt{k\hat{v}_{k,i}}} + \frac{\sqrt{1-\beta_2^T}}{\left(1-\beta_1^T\right)^2} \sum_{n=1}^{K} \frac{K\left((1-\beta_1) \beta_1^{K-n} g_{n,i}\right)^1}{\sqrt{K \sum_{j=1}^{K} (1-\beta_2) \beta_2^{K-j} g_{j,i}^2}} \\
&\leq \sum_{k=1}^{K-1} \frac{\hat{m}_{k,i}^2}{\sqrt{k\hat{v}_{k,i}}} + \frac{\sqrt{1-\beta_2^T}}{\left(1-\beta_1^T\right)^2} \sum_{n=1}^{K} \frac{K\left((1-\beta_1) \beta_1^{K-n} g_{n,i}\right)^1}{\sqrt{K (1-\beta_2) \beta_2^{K-n} g_{n,i}^2}} \\
&\leq \sum_{k=1}^{K-1} \frac{\hat{m}_{k,i}^2}{\sqrt{k\hat{v}_{k,i}}} + \frac{\sqrt{1-\beta_2^T}}{\left(1-\beta_1^T\right)^2} \frac{(1-\beta_1)^2}{\sqrt{K(1-\beta_2)}} \sum_{n=1}^{K} K\left(\frac{\beta_1^2}{\sqrt{\beta_2}}\right)^{K-n} \|g_{n,i}\|_2 \\
&\leq \sum_{k=1}^{K-1} \frac{\hat{m}_{k,i}^2}{\sqrt{k\hat{v}_{k,i}}} + \frac{K}{\sqrt{K(1-\beta_1)}} \sum_{n=1}^{K} \gamma_{K-n} \|g_{n,i}\|_2
\end{aligned}$$

Similarly, the rest of the terms in the summation can be upper bounded.

$$\begin{aligned}
\sum_{k=1}^{K} \frac{\hat{m}_{k,i}^2}{\sqrt{k\hat{v}_{k,i}}} &\leq \sum_{k=1}^{K} \frac{\|g_{k,i}\|_2}{\sqrt{k(1-\beta_2)}} \sum_{j=0}^{K-k} k\gamma^j \\
&\leq \sum_{k=1}^{K} \frac{\|g_{k,i}\|_2}{\sqrt{k(1-\beta_2)}} \sum_{j=0}^{K} k\gamma^j
\end{aligned}$$

For $\gamma, 1$, using the upper bound on the arithmetic-geometric series, $\sum_k k\gamma^k < \frac{1}{(1-\gamma)^2}$

$$\sum_{k=1}^{K} \frac{\|g_{k,i}\|_2}{\sqrt{k\left(1-\beta_2\right)}} \sum_{j=0}^{K} k\gamma^j \leq \frac{1}{\left(1-\gamma\right)^2 \sqrt{1-\beta_2}} \sum_{k=1}^{K} \frac{\|g_{k,i}\|_2}{\sqrt{k}}$$

Applying Lemma 4.6 one obtains,

$$\sum_{k=1}^{K} \frac{\hat{m}_{k,i}^2}{\sqrt{k\hat{v}_{k,i}}} \leq \frac{2G_\infty}{\left(1-\gamma\right)^2 \sqrt{1-\beta_2}} \|g_{1:K,i}\|_2$$

□

Theorem 4.8. *Assume f_k has bounded gradients, $\|\nabla f_k\left(x\right)\|_2 \leq G$, $\|\nabla f_k\left(x\right)\|_\infty \leq G_\infty$ for all $x \in \mathbb{R}^d$ and the distance between any x_k generated by Adam is bounded, $\|x_n - x_m\|_2 \leq D$, $\|x_n - x_m\|_\infty \leq D_\infty$ for any $m, n \in \{1, \ldots, K\}$ and $\beta_1, \beta_2 \in [0, 1)$ saitsfy $\frac{\beta_1}{\sqrt{\beta_2}} < 1$. Let $\alpha_k = \frac{\alpha_k}{\sqrt{k}}$ and $\beta_{1,k} = \beta_1 \lambda^{k-1}$,$\lambda \in (0, 1)$. Adam achieves the following guarantee for all $K \geq 1$.*

$$R\left(K\right) \leq \sum_{i=1}^{d} \left[\frac{D^2}{2\alpha\left(1-\beta_1\right)}\sqrt{K\hat{v}_{K,i}} + \frac{\alpha\left(\beta_1+1\right)G_\infty}{\left(1-\beta_1\right)\sqrt{1-\beta_2}\left(1-\gamma\right)^2}\|g_{1:K,i}\|_2 + \frac{D_\infty^2 G_\infty \sqrt{1-\beta_2}}{2\alpha\left(1-\beta_1\right)\left(1-\lambda\right)^2}\right] \tag{4.22}$$

Proof. From the convexity of each f_k one has,

$$f_k\left(x_k\right) - f_k\left(x^*\right) \leq g_k^T\left(x_k - x^*\right) = \sum_{i=1}^{d} g_{k,i}\left(x_{k,i} - x_{,i}^*\right)$$

Now, in its full form the Adam update rule can be written as

$$x_{k+1} = x_k - \frac{\alpha_k}{1-\beta_1^k}\left(\frac{\beta_{1,k}}{\sqrt{\hat{v}_k}} m_{k-1} - \frac{\left(1-\beta_{1,k}\right)}{\sqrt{\hat{v}_k}} g_k\right)$$

Since all of the operation from the Adam algorithm are performed element-wise, the focus now lies on the i^{th} dimension of x_k. From here subtract $x^*_{,i}$ from both sides of the update rule above and then square the resulting equation. This gives,

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

Since the goal is to analyze the regret, it makes sense to rearrange the above equation for $g_{k,i}\left(x_{k,i}-x^*_{,i}\right)$. An upper bound on the first relation below is available by assigning

$$a=\left(\frac{\beta_{1,k}\sqrt{\hat{v}_{k-1,i}}}{\alpha_k\left(1-\beta_{1,k}\right)}\right)^{1/2}\cdot\left(x^*_{,i}-x_{k,i}\right),\qquad b=\left(\frac{\beta_{1,t}\alpha_{k-1}}{\left(1-\beta_{1,t}\right)\sqrt{\hat{v}_{k-1,i}}}\right)^{1/2}m_{k-1,i},$$

and using the the inequality $a^2+b^2\geq 2ab$.

$$\begin{aligned}g_{k,i}\left(x_{k,i}-x^*_{,i}\right)&=\frac{\left(1-\beta_1^k\right)\sqrt{\hat{v}_{k,i}}}{2\alpha_k\left(1-\beta_{1,k}\right)}\left[\left(x_{k,i}-x^*_{,i}\right)^2-\left(x_{k+1,i}-x^*_{,i}\right)^2\right]+a\cdot b+\frac{\alpha_k\left(1-\beta_1^k\right)\sqrt{\hat{v}_{k,i}}}{2\left(1-\beta_{1,k}\right)}\left(\frac{\hat{m}_{k,i}}{\sqrt{\hat{v}_{k,i}}}\right)^2\\&\leq\frac{1}{2\alpha_k\left(1-\beta_1\right)}\left[\left(x_{k,i}-x^*_{,i}\right)^2-\left(x_{k+1,i}-x_{,i}\right)^2\right]\sqrt{\hat{v}_{k,i}}\\&+\frac{\beta_{1,k}}{2\alpha_{k-1}\left(1-\beta_{1,k}\right)}\left(x^*_{,i}-x_{k,i}\right)^2\sqrt{\hat{v}_{k-1,i}}+\frac{\beta_1\alpha_{k-1}}{2\left(1-\beta_1\right)}\frac{m^2_{k-1,i}}{\sqrt{\hat{v}_{k-1,i}}}+\frac{\alpha_k}{2\left(1-\beta_1\right)}\frac{\hat{m}^2_{k,i}}{\sqrt{\hat{v}_{k,i}}}\end{aligned}$$

The regret can be found by using inequality from Lemma 4.7 and summing the previous inequality over all dimensions (i.e. $i=1,\ldots,d$).

$$\begin{aligned}R\left(K\right)\leq&\sum_{i=1}^{d}\left[\frac{1}{2\alpha_1\left(1-\beta_1\right)}\left(x_{1,i}-x^*_{,i}\right)^2\sqrt{\hat{v}_{1,i}}+\sum_{k=2}^{K}\frac{1}{2\left(1-\beta_1\right)}\left(x_{k,i}-x^*_{,i}\right)^2\left(\frac{\sqrt{\hat{v}_{k,i}}}{\alpha_k}-\frac{\sqrt{\hat{v}_{k-1,i}}}{\alpha_{k-1}}\right)\right]\\&+\sum_{i=1}^{d}\left[\frac{\alpha\left(1+\beta_1\right)G_\infty}{\left(1-\beta_1\right)\sqrt{1-\beta_2}\left(1-\gamma\right)^2}\|g_{1:K,i}\|_2+\sum_{k=1}^{K}\frac{\beta_{1,k}}{2\alpha_k\left(1-\beta_{1,k}\right)}\left(x^*_{,i}-x_{k,i}\right)^2\sqrt{\hat{v}_{k,i}}\right]\end{aligned}$$

$$\leq \sum_{i=1}^{d}\left[\frac{D^2}{2\alpha\left(1-\beta_1\right)}\sqrt{K\hat{v}_{K,i}}+\frac{\alpha\left(1+\beta_1\right)G_\infty}{\left(1-\beta_1\right)\sqrt{1-\beta_2}\left(1-\gamma\right)^2}\|g_{1:K,i}\|_2+\frac{D_\infty^2}{2\alpha}\sum_{k=1}^{k}\frac{\beta_{1,k}}{1-\beta_{1,k}}\sqrt{k\hat{v}_{k,i}}\right]$$

$$\leq \sum_{i=1}^{d}\left[\frac{D^2}{2\alpha\left(1-\beta_1\right)}\sqrt{K\hat{v}_{K,i}}+\frac{\alpha\left(1+\beta_1\right)G_\infty}{\left(1-\beta_1\right)\sqrt{1-\beta_2}\left(1-\gamma\right)^2}\|g_{1:K,i}\|_2+\frac{D_\infty^2 G_\infty\sqrt{1-\beta_2}}{2\alpha}\sum_{k=1}^{k}\frac{\beta_{1,k}}{1-\beta_{1,k}}\sqrt{k}\right]$$

The last inequality is due to the bound on the 2-norm and ∞-norm of the parameter vector. One arrives at the final result once it is noticed that the final term is upper bounded as laid out below. Remember $\beta_{1,k}=\beta_1\lambda^{k-1}$ for $\lambda\in(0,1)$.

$$\sum_{k=1}^{k}\frac{\beta_{1,k}}{1-\beta_{1,k}}\sqrt{k}\leq\sum_{k=1}^{k}\frac{1}{1-\beta_1}\lambda^{k-1}\sqrt{k}\leq\sum_{k=1}^{k}\frac{1}{1-\beta_1}\lambda^{k-1}t\leq\frac{1}{\left(1-\beta_1\right)\left(1-\lambda\right)^2}$$

□

This bound on the regret can be verified by numerical results in practice. However, to do so would require keeping track of all iterations of the parameter vector and gradients which is not practical when dealing with very large problems. This result simply shows that the regret for Adam obeys a $O\left(\sqrt{k}\right)$ bound [6]. In particular, the summation in the theorem can be bounded in a stronger sense if the input data is sparse since $g_{1:K,i}$ and $\hat{v}_{K,i}$ will expectedly be sparse and thus can be bounded above by G_∞.

4.5 AdaGrad

Introduced in 2011 by Duchi et al [7], Adagrad is simple variant of Adam which is useful when working with sparse gradients. When bias-correction is applied to Adam, as it was above, Adagrad corresponds to $\beta_1 = 0$ and $\beta_2 = 1 - \epsilon$. This coincides to an update rule of

$$x_{k+1} = x_k - \alpha_k \frac{g_k}{\sqrt{\sum_{j=1}^{k} g_j^2}} \tag{4.23}$$

This update arises from taking the limit of $\hat{v}_k$ from Adam as β_2 approaches 1.

$$\lim_{\beta_2 \to 1} \hat{v}_k = \lim_{\beta_2 \to 1} \frac{(1-\beta_2)}{\left(1-\beta_2^k\right)} \sum_{j=1}^{k} \beta_2^{k-j} \cdot g_j^2 = \frac{1}{k} \sum_{j=1}^{k} g_j^2 \tag{4.24}$$

Theorem 4.9. *Assume the distance between any x_k generated by Adagrad is bounded $\|x_n - x_m\|_2 \leq D$,$\|x_n - x_m\|_\infty \leq D_\infty$ for any $m, n \in \{1, \ldots, K\}$. Suppose f is convex with non-empty solution set, then Adagrad achieves the following guarantee for all $K \geq 1$ and $\alpha_k = \frac{\alpha}{\sqrt{k}}$*

$$R(K) \leq \frac{1}{2\alpha} \sum_{i=1}^{d} \sqrt{K g_{1:K,i}^2} + \sum_{i=1}^{d} \sum_{k=1}^{K} \frac{\alpha_k}{2} \frac{g_{k,j}^2}{\sqrt{\sum_{j=1}^{k} g_{j,i}^2}} \tag{4.25}$$

Proof. First note that convexity of f implies

$$f_k(x_k) - f_k(x^*) \leq \sum_{i=1}^{d} g_{k,i} \left(x_{k,i} - x_{,i}^*\right) \tag{4.26}$$

Next subtract $x^*_{,i}$ from both sides of the update rule and square the inequality.

$$\left(x_{k+1,i} - x^*_{,i}\right)^2 = \left(x_{k,i} - x^*_{,i}\right)^2 - \frac{2\alpha_k g_{k.i}}{\sqrt{\sum_{j=1}^{k} g_{j,i}^2}} \left(x_{k,i} - x^*_{,i}\right) + \alpha_k^2 \left(\frac{g_{k,i}}{\sqrt{\sum_{j=1}^{k} g_{j,i}^2}}\right)^2$$

By using the convexity condition given above, this equation can be rearranged to give the regret by adding over all $k = 1, \ldots, K$

$$\begin{aligned} g_{k,i}\left(x_{k,i} - x^*_{,i}\right) &= \frac{1}{2\alpha_k}\sqrt{\sum_{j=1}^{k} g_{j,i}^2}\left[\left(x_{k,i} - x^*_{,i}\right)^2 - \left(x_{k+1,i} - x^*_{,i}\right)^2\right] + \frac{\alpha_k}{2}\frac{g_{k,i}^2}{\sqrt{\sum_{j=1}^{k} g_{j,i}^2}} \\ R(K) &\leq \sum_{i=1}^{d} \frac{\sqrt{g_{1,i}^2}}{2\alpha_i}\left(x_{1,i} - x^*_{,i}\right)^2 + \sum_{i=1}^{d}\sum_{k=2}^{K} \frac{1}{2}\left(x_{k,i} - x^*_{,i}\right)^2 \left(\frac{\sqrt{\sum_{j=1}^{k} g_{j,i}^2}}{2\alpha_k} - \frac{\sqrt{\sum_{j=1}^{k-1} g_{j,i}^2}}{2\alpha_{k-1}}\right) \\ &\quad + \sum_{i=1}^{d}\sum_{k=1}^{K} \frac{\alpha_k}{2}\frac{g_{k,i}^2}{\sqrt{\sum_{j=1}^{k} g_{j,i}^2}} \end{aligned} \tag{4.27}$$

□

As with Adam, one would have to have access to all iterations of the parameter vector and the gradients in order to verify the regret bound. For this reason, in the next section, which covers several numerical examples, the convergence will be qualitatively compared to one another rather than individually corroborated.

Chapter 5

Numerical Examples

This chapter will cover several numerical examples in the hopes of providing a pratical understanding of the optimization algorithms discussed in previous chapters. We begin by investigating multiple two-dimensional objective functions in order to illustrate aspects of the algorithms within a relatively simple setting. Not all of these functions are convex over the domain being consider which leads to inefficient convergence for some methods while displaying the flexiblility of others. The functions chosen are motivated by a list of optimization test functions and datasets found in [11]. In particular, test functions are categorized based on the shape of the surface they define; for example, Bowl-shaped, plate-shaped and multiple local minima. These classifications were chosen to attempt to portray varying degrees of difficulty in optimizing test functions.

5.1 Ball Shaped

5.1.1 Sum of Squares Function

This is one of the simplest function to optimize since it simply involves summing up the squares of all elements of the input vector. It is clearly convex and in two dimensions and has a global minimum $f(x^*) = 0$ at $x^* = [0\ 0]$. The intended search domain for this example will be $x_i = [-5\ 5]$. The function for an arbitrary number of dimensions is listed below, however only $d = 2$ will be considered.

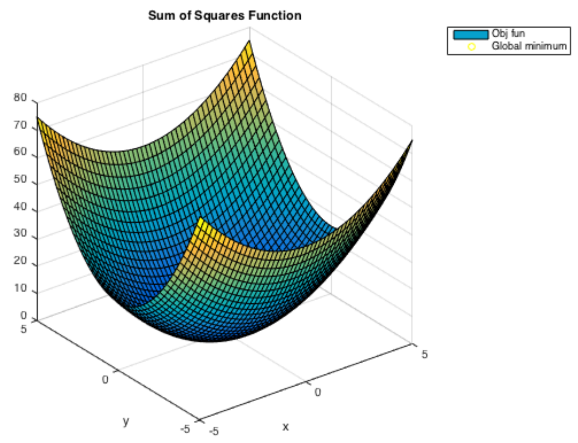
$$f(\vec{x}) = \sum_{i=1}^{d} i x_i^2 \tag{5.1}$$

All discussed algorithms converge to the minimizer as expected. The purpose of including such a simple test function is simply to ensure the supposed results a validated in a straightforward case.

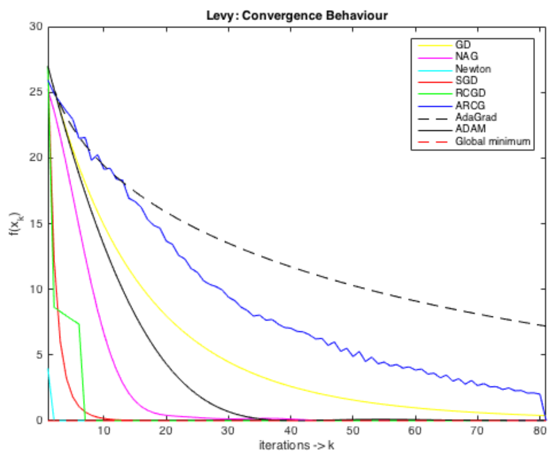
The initial starting point was chosen as $x_0 = [-3, -3]$. From 5.1(b), the first point to notice is Newton converges in one iteration which is the predicted result. For NAG and GD it is challenging to compare the actual rates of convergence (i.e. linear, quadratic) from 5.1(b) but the qualitatively behaviour is the focus and in fact shows that NAG converges in less iterations than GD, which is also the expected result.

Intuitively, SGD should converge more quickly than Adam or Adagrad since the SGD algorithm allows for larger steps than Adagrad and Adam onvex functions. Namely, moment estimation tends to under-estimate the magnitude of the subgradients for this test, hence when the three algorithms use equivalent step sizes SGD will converge faster. Again this is the behaviour observed from 5.1(b).

It is noted here that randomized coordinate descent algorithms are, by construction, meant for high-dimensional problems. Restriction to 2D, which is the approach taken here, voids their effectiveness since convergence can only occur in one direction per iteration.



(a) Sum of Squares Function



(b) Convergence Plot

Figure 5.1: Plots of the Sum of Squares function on the domain $x, y \in [-5, 5]$ along with convergence plot of all algorithms discussed.

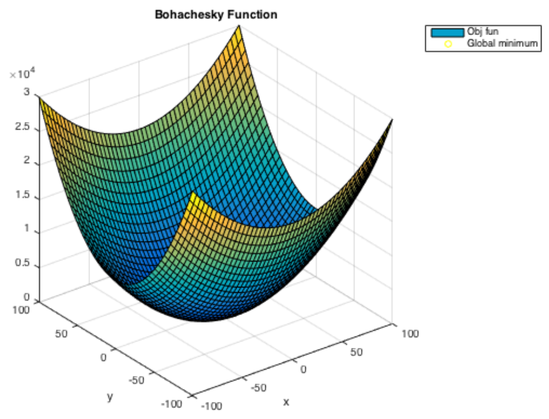
GD	NAG	Newton	SGD	RCGD	ARCG	AdaGrad	Adam
0.01	0.01	1	0.1	0.01	0.01	0.1	0.1

5.1.2 Bohachevsky-1 function

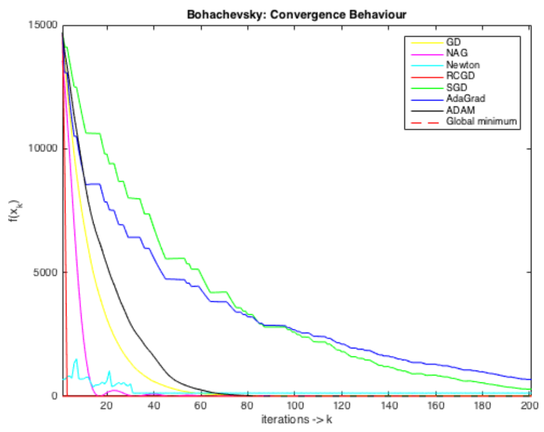
As per the list provided in [11] there are three Bohachevsky functions, however the focus will solely remain on the first of these. The function, shown below, is two-dimensional an bowl shaped. A typical domain to test on is $x_i = [-100 \;\; 100]$ and the global minimum $f(x^*) = 0$ can be found at $x^* = [0 \;\; 0]$. A plot of (5.2) is provided to illustrate the shape of the function.

$$f(\vec{x}) = x_1^2 + 2x_2^2 - 0.3\cos(3\pi x_1) - 0.4\cos(4\pi x_2) + 0.7 \tag{5.2}$$

As with the Sum of Squares function, all algorithms converge to the minimizer (with the exception of ARCG) assuming the step size is chosen small enough. The initial starting point for this test was chosen as $x_0 = [-70, 70]$ simply due to symmetry, while $x_0 = [-15, 15]$ for Newton. In terms of comparison, for the same time step GD and NAG converge as would be expected since the function is convex. Newton requires more than one iteration this time since the function is not quadratic. The stochastic methods appear to converge at a slower rate than the deterministic algorithms for this test function, however choosing larger time steps for SGD, RCGD, and ARCG will result in quicker convergence. Requiring such large step sizes for these convex functions suggests that moment estimation is not as suitable for convergence in this scenario as simple deterministic method would be. The step size values for each algorithm that were used for this test are given below for reference as well.



(a) Bohachevsky Function



(b) Convergence Plot

Figure 5.2: Plots of the Bohachevsky function on the domain $x, y \in [-100, 100]$ along with convergence plot of all algorithms discussed.

GD	NAG	Newton	SGD	RCGD	AdaGrad	Adam
0.02	0.02	1	0.01	0.02	4	2

5.2 Plate Shaped

5.2.1 Booth Function

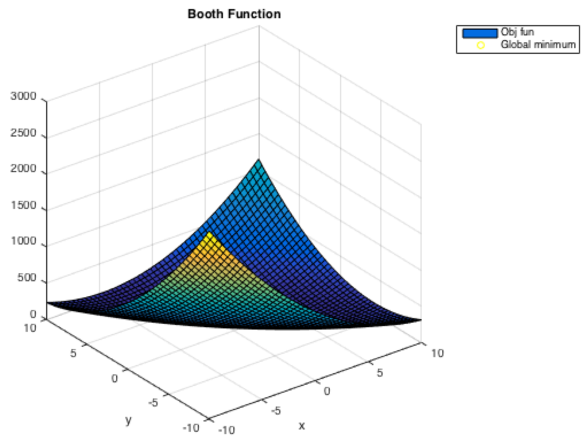
The global minimum of the Booth function is $f(x^*) = 0$ and occurs at the point $x^* = [1\ \ 3]$. The typical domain for evaluation is $x_i \in [-10\ \ 10]$. Both a plot of the function and a table of step sizes for each algorithm is given.

$$f(\vec{x}) = (x_1 + 2x_2 - 7)^2 + (2x_1 + x_2 - 5)^2 \tag{5.3}$$

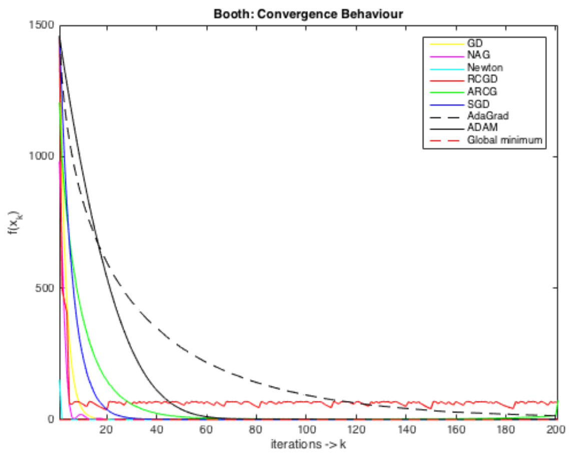
The Booth function is characterized by a valley in the neighbourhood of the line $x = -y$ as can be seen from 5.3(a). It is clear from (5.3) that the Booth function is quadratic, hence Newton's method should converge in one iteration which is what is shown in 5.3(b). Again GD and NAG converge more quickly than the stochastic algorithms with NAG achieving the faster rate.

This test gives the first evidence of a major failing of randomized coordinate descent methods; that being the low dimensionality of the problem only allows updates to the parameter vector in one direction per iteration. This issue is evident from the convergence path of RCGD which approaches the minimum rapidly during early iterations but then gets stuck oscillating between points in either coordinate direction.

Once again Adagrad and Adam require larger step sizes than NAG and GD to achieve convergence in an equivalent number of iterations. Like examples in the previous subsection, this suggests that for functions with only one stationary point moment estimation returns values for the subgradients which are small compared to deterministic methods.



(a) Booth Function



(b) Convergence Plot

Figure 5.3: Plots of the Booth function on the domain $x, y \in [-5, 5]$ along with convergence plot of all algorithms discussed.

GD	NAG	Newton	SGD	RCGD	ARCG	AdaGrad	Adam
0.01	0.01	1	0.01	0.05	0.05	0.5	0.2

5.2.2 McCormick Function

As can be seen from the equation below, the McCormick function is clearly not convex. As a result this is one of the test functions which will prove to be problematic for some of the methods, in particular coordinate descent. The global minimum is $f(x^*) = -1.9133$ which occurs at $x^* =$

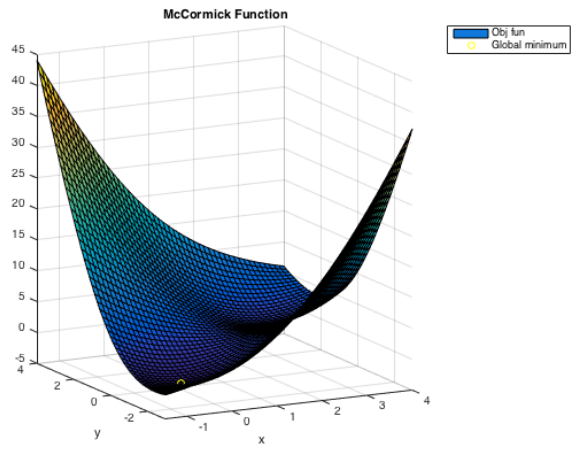
$[-0.54719 \;\; -1.54719]$. The domain of interest is defined as $x_1 \in [-1.5 \;\; 4]$, $x_2 \in [-3 \;\; 4]$.

$$f(\vec{x}) = \sin(x_1 + x_2) + (x_1 - x_2)^2 - 1.5x_1 + 2.5x_2 + 1 \tag{5.4}$$

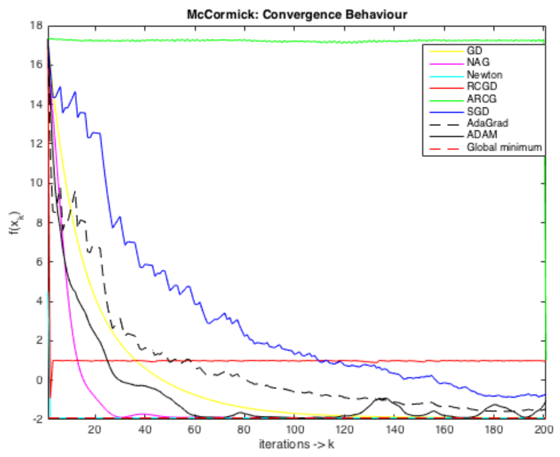
This function is tricky for most deterministic algorithms to optimize due to its non-convexity. Along the ravine running through the function there are multiple stationary points which all effect convergence depending on the initial starting point. Without having the ability to choose the initial point a more reliable approach for optimization would be to determine the step size based on a line search method.

The start point used for this test was $x_0 = [3, -2]$ for all algorithms except Newton which used a start point of $x_0 = [1, -2]$. From this starting point Newton converged in one iteration, however the region over which to chose the initial point is difficult to determine but appears to need to be quadratic.

Once again, this test function shows the ineffectiveness of randomized coordinate descent algorithms in finding the minimum. As well as supporting the idea that moment estimation is better suited for functions with multiple stationary points. This is seen from 5.4(b) where Adam and Adagrad converge at a comparable speed to GD and NAG while using smaller step sizes than previous functions.



(a) McCormick Function



(b) Convergence Plot

Figure 5.4: Plots of the McCormick function on the domain $x, y \in [-5, 5]$ along with convergence plot of all algorithms discussed.

GD	NAG	Newton	SGD	RCGD	ARCG	AdaGrad	Adam
0.01	0.01	1	0.1	0.001	0.001	0.4	0.2

5.3 Functions with Many Local Minima

5.3.1 Ackley Function

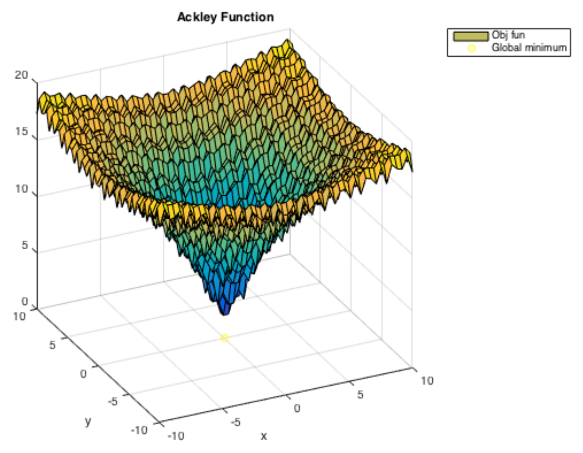
The Ackley function has a comparatively flat outer region but drops off quite drastically near the global minimum $f(\vec{x}) = 0$ at $x^* = [0 \;\; 0]$. The test will be run on the domain $x_i = [-32.768 \;\; 32.768]$ with parameter values of $a = 20$, $b = 0.2$ and $c = 2\pi$. Notice that this function is also clearly not convex over its domain. The purpose of testing this function is to better determine the flexibility of the algorithms. The question is how will local minima and non-convexity effect convergence?

$$f\left(\vec{x}\right) = -a\exp\left(-b\sqrt{\frac{1}{d}\sum_{i=1}^{d}x_i^2}\right) - exp\left(\frac{1}{d}\sum_{i=1}^{d}\cos\left(cx_i\right)\right) + a\exp\left(1\right) \tag{5.5}$$

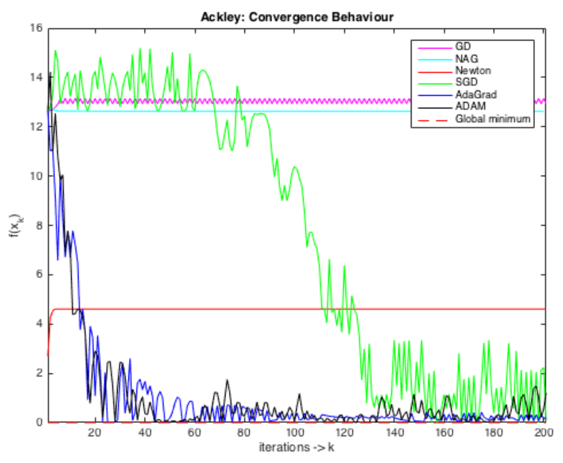
From a first look at 5.5(a) one could correctly deduce that this example will provide difficulty for all algorithms previously discussed. The large number of local minima will almost certainly lead to the deterministic methods converging to a local minimum. The coordinate descent algorithms are left out for this example as the drastic function variability and low dimensionality render them unreliable.

The most reliable methods in this regime are Adam, Adagrad and even SGD appears to be tending towards the global minimum. This illustrates the large scope of test functions that moment estimation schemes can cover.

The starting point was chosen as $x_0 = [-5, 5]$ by simple symmetry of the domain. Overall the purpose of this test was to express the fallibility of deterministic methods when several local minima are present. This particular function was chosen because of the difficulty to place the initial guess within a suitable region of the minimizer to ensure convergence.



(a) Ackley Function



(b) Convergence Plot

Figure 5.5: Plots of the Ackley function on the domain $x, y \in [-5, 5]$ along with convergence plot of all algorithms discussed.

GD	NAG	Newton	SGD	RCGD	ARCG	AdaGrad	Adam
0.05	0.05	1	0.1	0.1	0.1	1	0.5

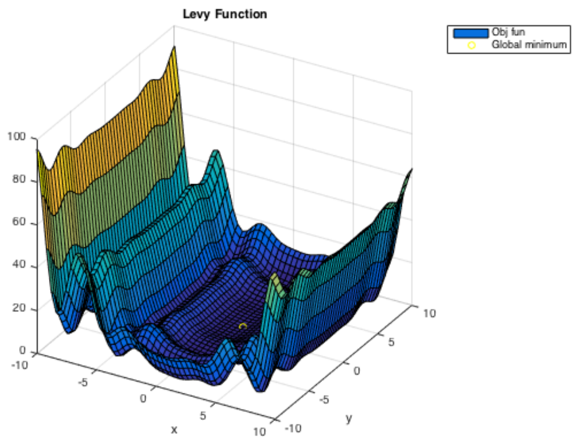
5.3.2 Levy Function

This is another non-convex function which will be used to test the scope of applicability of the alorithms. The domain to be used is $x_i \in [-10 \;\; 10]$ while the global mimimum $f(x^*) = 0$ occurs at $x^* = [1 \;\; 1]$. The variable w_i denotes $w_i = 1 + \frac{x_i - 1}{4}$.

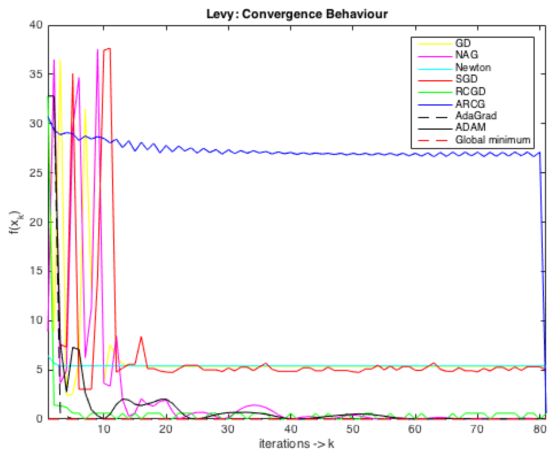
$$f(\vec{x}) = \sin^2(\pi w_1) + \sum_{i=1}^{d-1} (w_i - 1)^2 \left[1 + 10\sin^2(\pi w_i + 1)\right] + (w_d - 1)^2 \left[1 + \sin^2(2\pi w_d)\right] \tag{5.6}$$

Choosing the starting point as $x_0 = [-6, 1]$ allows for convergence of most of the algorithms but requires taking step sizes that are quite large compared to previous examples. In fact, even though convergence is attainable, choosing a constant step size that is too large for this test will result in divergence; hence stochastic methods can be unstable in this sense. Typically, convergence needs to occur quickly for the stochastic methods on the Levy function or else the global minimum will not be reached.

A step size of 0.1 for GD and NAG is close to the maximum allowed value for the starting point chosen. While NAG reaches the global minimizer, GD gets stuck in the trench between x_0 and x^* and so converges to a local minimum. Not all of the results being discussed here can be seen quantitatively from the plots provided. Instead, the discussion revolves around the qualitative aspects of the convergence plot combines with the observed results of the convergence path which is not shown.



(a) Levy Function



(b) Convergence Plot

Figure 5.6: Plots of the Levy function on the domain $x, y \in [-5, 5]$ along with convergence plot of all algorithms discussed.

GD	NAG	Newton	SGD	RCGD	ARCG	AdaGrad	Adam
0.1	0.1	1	0.1	0.5	0.01	6	1.5

5.4 Additional Functions

5.4.1 Beale Function

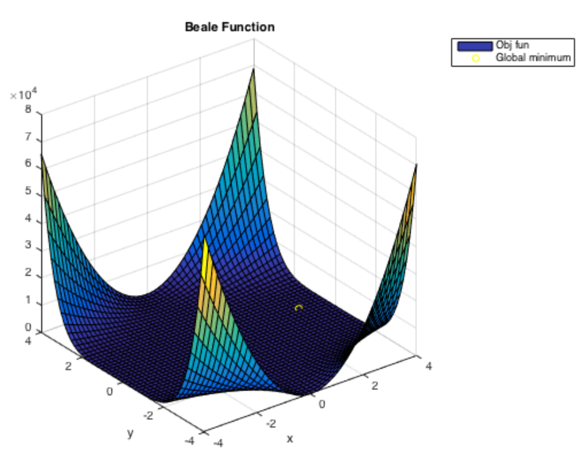
The Beale function is characterized by sharp peaks on the corners of the domain $x_i \in [-4.5 \;\; 4.5]$. There is one global minimum of $f(x^*) = 0$ at $x^* = [3 \;\; 0.5]$. This function simply adds a high order polynomial to the list of test examples.

$$f(\vec{x}) = (1.5 - x_1 + x_1 x_2)^2 + \left(2.25 - x_1 + x_1 x_2^2\right)^2 + \left(2.625 - x_1 + x_1 x_2^3\right)^2 \tag{5.7}$$

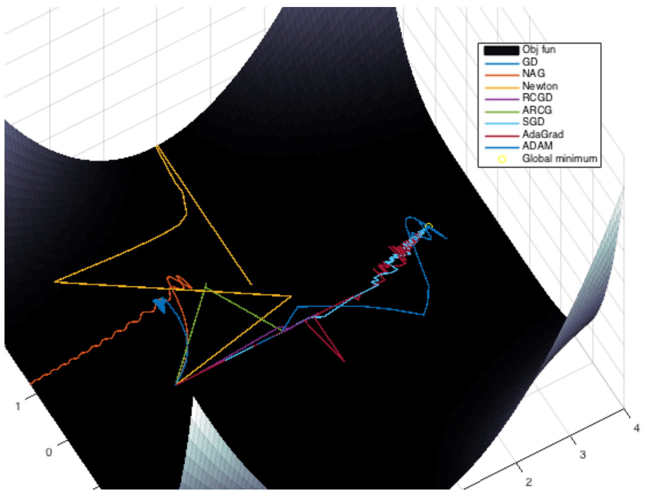
This function was chosen to illustrate that some algorithms will determine that a stationary point is the global minimizer when in fact it is not.

Although it is difficult to see (due to the convergent methods) from 5.7(b) the minimizer is marked by a yellow circle on the far right side of the plot. Only Adam, Adagrad and SGD converge to this point. This further shows the flexbility of these algorithms when multiple stationary points are present and are not as densely populating the surface. **Note:** Even though it is difficult to tell from 5.7(a), the function is known to have multiple stationary points because it is a multivariate sixth order polynomial.

The convergence plot describes the following: GD, Newton and the coordinate descent methods all converge to different stationary points. In fact it is easy to see (after finding the gradient) that $\vec{x} = [0, 1]$ is a stationary point of the function which is the point Newton approaches. NAG on the other hand appears to converge on the same stationary point as GD but actually diverges for the step size chosen in the table below.



(a) Beale Function



(b) Convergence Plot

Figure 5.7: Plots of the Beale function on the domain $x, y \in [-5, 5]$ along with convergence plot of all algorithms discussed.

GD	NAG	Newton	SGD	RCGD	ARCG	AdaGrad	Adam
0.01	0.01	1	0.1	0.01	0.2	1	0.5

5.4.2 Styblinski-Tang 2D

Styblinski-Tang has four local minima in the search domain, one of which is the global minimum. This function is chosen as a test example because it shows both the sensitivity of deterministic methods to the initial parameter vector when local minima are present and also the property that stochastic methods may converge to the global minimizer even if the initial parameter vector is chosen within one of the wells containing a local minimum. The global minimum $f(x^*) = -78.3323$ occurs at $x^* = [-2.903534 \quad -2.903534]$.

$$f(\vec{x}) = \frac{1}{2} \sum_{i=1}^{d} \left(x_i^4 - 16x_i^2 + 5x_i \right) \tag{5.8}$$

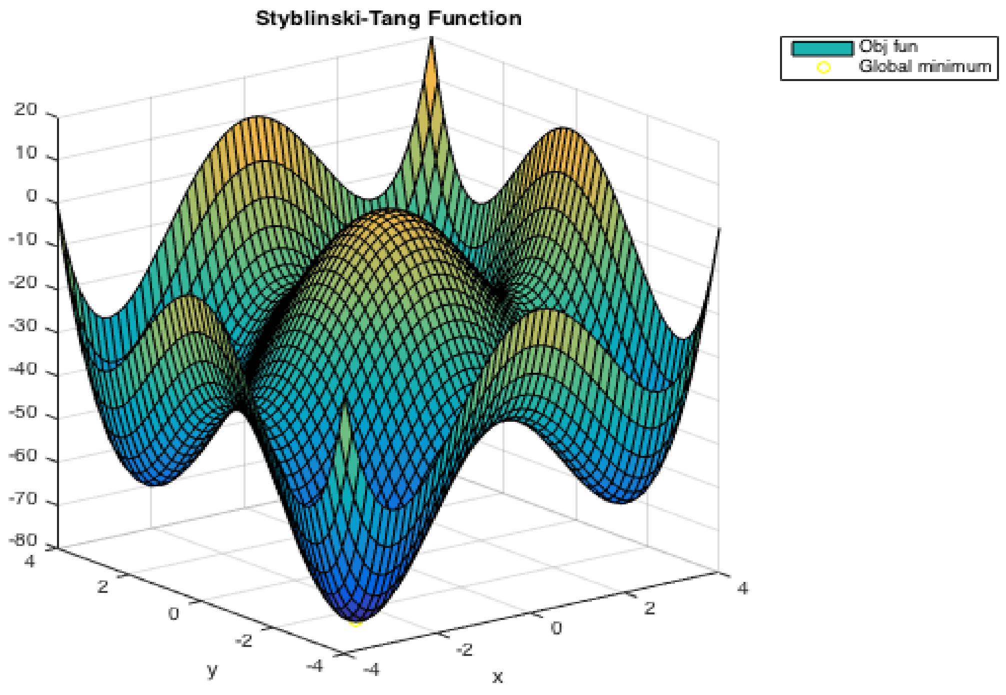


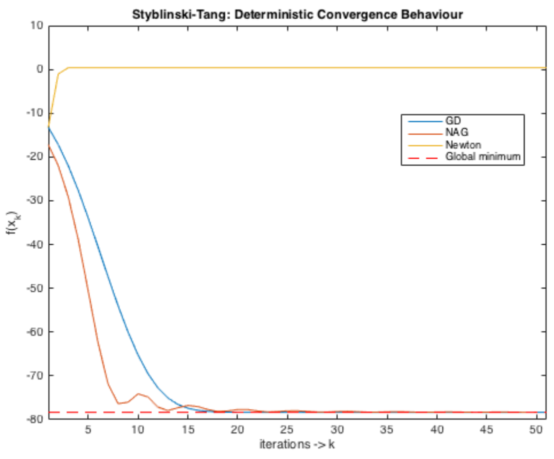
Figure 5.8:

The Styblinski-Tang function, as shown by 5.4.2, has four local minima, a local maxima and four saddle points within the chosen domain. The starting point, $x_0 = [-0.5, -1]$, is chosen inside the well containing the global minimizer so that the deterministic methods have an chance of converging to x^*. Unlike the other examples there are separate convergence plots for deterministic and stochastic methods.

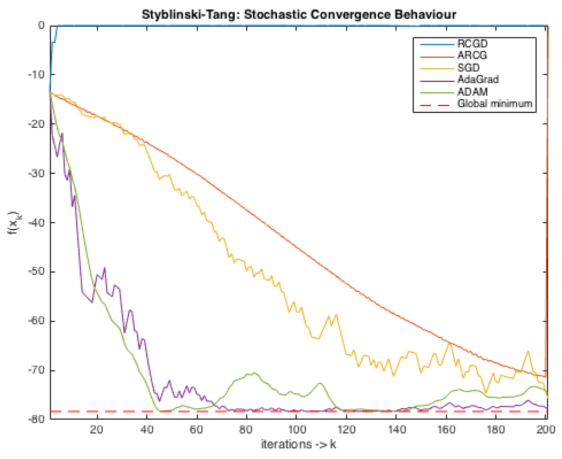
Plot 5.9(a) shows that GD and NAG will converge in very few iterations when matched against stochastic algorithms. This is a product of the initial start point being selected in a region of x^* which is well suited for these two schemes. Although it is not shown in either plot, Newton actually approaches the global maxima unless x_0 is chosen excessively close to x^*.

The stochastic algorithms vary in their applicability for this test. SGD, Adam and Adagrad, when compared to deterministic methods, take an order of magnitude longer to converge. However, what is not shown here is that these methods can in fact converge to x^* even when x_0 is chosen outside the well containing x^*. This reinforces the idea of flexibility for these stochastic methods; they may take longer to converge but the cost per iteration is less and they can converge when deterministic methods do not.

Finally, RCGD actually finds a saddle point of the function while ARCG enters the well containing x^* but has difficulty converging to the minimizer most likely due to previously mentioned flaws of randomized coordinate descent algorithms.



(a) Deterministic Convergence Plot



(b) Stochastic Convergence Plot

Figure 5.9: Plots of convergence results for deterministic and stochastic methods on the Styblinski-Tang function over the domain $x, y \in [-5, 5]$.

GD	NAG	Newton	SGD	RCGD	ARCG	AdaGrad	Adam
0.01	0.01	1	0.002	0.01	0.001	0.3	0.1

Bibliography

[1] Yurii Nesterov. *Introductory Lectures on Convex Optimization: A Basic Course*, volume 87 of *Applied Optimization*. Springer Science+Business Media New York, 2004.

[2] Yurii Nesterov. A method of solving a convex programming problem with convergence rate $o(1/k^2)$. *Soviet Mathematics Doklady*, 27:372–376, 1983.

[3] Yurii Nesterov. Efficiency of coordinate descent methods on huge-scale optimization problems. *SIAM Journal on Optimization*, 22:341–362, 2012.

[4] Z. Lu and L. Xiao. On the complexity analysis of randomized block-coordinate descent methods. *Mathematical Programming*, 152:615–642, 2015.

[5] Qihang Lin, Zhaosong Lu, and Lin Xiao. An accelerated proximal coordinate gradient method. In Z. Ghahramani, M. Welling, C. Cortes, N.d. Lawrence, and K.q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3059–3067. Curran Associates, Inc., 2014.

[6] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[7] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *J. Mach. Learn. Res.*, 12(1532-4435):2121–2159, July 2011.

[8] Ashia C. Wilson, Benjamin Recht, and Michael I. Jordan. A lyapunov analysis of momentum methods in optimization. *CoRR*, abs/1611.02635, 2016.

[9] Weijie Su, Stephen Boyd, and Emmanuel J. Candes. A differential equation for modeling nesterov's gradient method: Theory and insights. *J. Mach. Learn. Res.*, 17:1–43, September 2016.

[10] Alexander Rakhlin and Ohad Shamir. Making gradient descent optimal for strongly convex stochastic optimization. *CoRR*, 2011.

[11] Derek Bingham and Sonja Surjanovic. Virtual library of simulation experiments: Test functions and datasets, 2015. [Online; accessed July-2017].