Accelerated Convergence of Gradient Descent Using Adaptive Parameters

A Thesis
Presented in Partial Fulfillment of the Requirements for the
Degree of Master of Science
with a
Major in Mathematics
in the
College of Graduate Studies
University of Idaho
by
Matthew Mills

Approved by:
Major Professor: Fuchang Gao, Ph.D.
Committee Members: Lyudmyla Barannyk, Ph.D.; Linh Nguyen, Ph.D.
Department Administrator: Hirotachi Abo, Ph.D.

May 2022

Abstract

The Nesterov gradient descent algorithm serves as a performance benchmark for convex optimization problems. Like many other gradient-based methods, the Nesterov algorithm requires choosing a constant step size before optimization begins, and the performance of the algorithm heavily depends on the step size. Here, we propose three novel adaptive algorithms which adaptively determine the step size based on the searching history. The new adaptive methods were tested alongside the original Nesterov algorithm on a list of commonly-used optimization test functions in a range of dimensions. The numerical experiments showed that they consistently outperformed the Nesterov algorithm by a wide margin. We also discuss ways that these adaptive methods could be improved.

Acknowledgments

I would like to acknowledge my advisor, Dr. Fuchang Gao, for his insight, effort, and direction on this project and his willingness to create a successful research environment.

Dedication

This thesis is dedicated to my wife, who supported me through many long days and nights, as well as my parents, who always encouraged me to pursue my interest in Mathematics.

Contents

Chapter 1: Introduction

This thesis explores solutions to optimization problems using gradient-based methods. An optimization problem is formally stated as follows:

Problem 1. *Let f be a function defined on some set I, find*

$$\text{argmin}_{x \in I} f(x).$$

Optimization can also refer to finding maximum function values as well, but in the context of machine learning, the function f usually is an indication of error within a model or approximated solution, so it is customary to let the term "optimization" refer to finding minimum values. In particular, we are concerned with optimization problems with a convex target function. Convex functions appear in many different applications, so it is natural to study optimization techniques which apply specifically to convex functions. Recall that for differentiable functions, convexity is defined as follows ([18], p52):

Definition 1. *A continuously differentiable function f is said to be convex on $\mathbb{R}^n$ if for any $x, y \in \mathbb{R}^n$, it follows that*

$$f(y) \geq f(x) + \langle \nabla f(x), y - x \rangle. \tag{1}$$

The inequality (1) can be derived from another more common definition of convexity ([18], p54):

Definition 2. *A function f in $\mathbb{R}^n$ is called convex if for all $x, y \in \mathbb{R}^n$, and all $p \in [0, 1]$,*

$$f(px + (1-p)y) \leq pf(x) + (1-p)f(y). \tag{2}$$

In fact, for differentiable functions, the two definitions are equivalent. Now, this second definition can be readily used to prove the following important property about convex functions which guarantees that an algorithmic scheme can achieve a minimal function value [10]:

Proposition 1. *Let f be a convex function on $\mathbb{R}^n$, and suppose f has a local minimum at x^*. Then f also has a global minimum at x^*.*

Proof. Suppose x^* is a local minimum of f. Then for any $y \in \mathbb{R}^n$, we can choose $p \in (0, 1]$ small enough, so that $z := (1-p)x^* + py$ is in the neighborhood of x^* such that $f(x^*) \leq f(z)$. It follows that

$$f(x^*) \leq f((1-p)x^* + py) \leq (1-p)f(x^*) + pf(y).$$

Thus, $f(x^*) \leq f(y)$. So f has a global minimum at x^*. □

There are many existing methods to perform convex optimization [1], and an example which is consistently regarded as a benchmark for algorithm performance is the Nesterov gradient descent method ([18], p80). However, the Nesterov gradient descent algorithm is non-adaptive, meaning that it uses some parameters that are predetermined by the user, and the step sizes are computed separately from the given function and error values produced by the algorithm. Consequently, the performance of the algorithm heavily depends on the initial parameters. As such, it is a non-trivial task to choose the right initial parameters, which often requires a brute-force search.

Thus, the goal of this research project is to create a convex optimization algorithm with better performance by using adaptive techniques. The new algorithm's successful performance improvement is measured by running itself and the Nesterov algorithm on a set of commonly-used optimization test functions, with a wide range of dimensions, to compare the number of iterations and function evaluations required for each algorithm to converge. To introduce the new algorithms, it will first be necessary to explore some of the most common gradient-based optimization methods, then the new adaptive methods will be presented along with the specific test cases that were used to verify their performance improvement.

Chapter 2: Optimization Methods

There are many different methods which have been discovered to solve optimization problems. For general reference, we recommend the excellent book [1]. Many of these methods rely on computing and using a function's gradient, since it contains valuable information about extreme values. In this chapter, we explain four of the most common optimization methods currently in use: Gradient Descent, Mirror Descent, Coordinate Descent, and the Ellipsoid method. This will help setup the motivation for a new method.

1. Gradient Descent

For a given continuous function f, the gradient descent method begins by picking an initial point $(x_0, f(x_0))$, and a step size. Then, to find the nearest minimal function value, new x values are iteratively computed by finding the gradient at the current x value, scaling the gradient by some constant determined by the step size, and then going in the direction opposite to the gradient by subtracting from the current x value. This is because the direction opposite the gradient indicates the path of steepest descent towards the nearest minimal function value [6]. Also, the constant gradient scalar is normally referred to as the step size, but in the context of machine learning, is usually referred to as the learning rate. This process can be formally stated as follows:

Algorithm 1. Gradient Descent *For a convex function f, an initial point x_0, and a given constant step size η, iteratively compute $x_{n+1} = x_n - \eta \nabla f(x)$ for $n = 1, 2, \ldots$.*

Now, a question that naturally arises is whether or not this process will converge to some minimum value of f? Proving this result requires setting up and stating some preliminary definitions and lemmas (See e.g. [10]). First, let $\|\cdot\|$ denote the Euclidean norm defined by

$$\|x\| = \left(\sum_{i=1}^{n} x_i^2 \right)^{1/2}, \ x = (x_1, x_2, \ldots, x_n).$$

Definition 3. *Let $f : \mathbb{R}^n \mapsto \mathbb{R}$ be a convex, continuous objective function with respect to a given norm $\|\cdot\|$, then f is defined to be L-Lipschitz if for all $x, y \in \mathbb{R}^n$,*

$$\|f(x) - f(y)\| \leq L\|x - y\|.$$

Furthermore, f is defined to be L-smooth if the gradient $\nabla f := (\partial f/\partial x_1, \ldots, \partial f/\partial x_n)$ is L-Lipschitz. In other words, for any x, y in the domain of f,

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\|.$$

The following result ensures the convergence of the gradient descent algorithm [10]:

Theorem 1. *Let f be convex and L-smooth on $\mathbb{R}^n$, and let $x^* = \text{argmin} f(x)$. Then the gradient descent with $\eta = \frac{1}{L}$ satisfies*

$$f(x_n) - f(x^*) \leq \frac{2L\|x_1 - x^*\|^2}{n - 1}.$$

The proof of this theorem contains the ideas that are essential to the development of the new algorithms. To this end, it is necessary to first prove two lemmas:

Lemma 1. *Let f be convex and L-smooth on $\mathbb{R}^n$. Then for any $x, y \in \mathbb{R}^n$, it follows that*

$$|f(x) - f(y) - \nabla f(y)^T(x - y)| \leq \frac{L}{2}||x - y||^2.$$

Proof. First, express $f(x) - f(y)$ as an integral, then apply the Cauchy-Schwartz inequality and the definition of L-smooth:

$$\begin{aligned}
|f(x) - f(y) - \nabla(y)^T(x - y)| &= \left| \int_0^1 \nabla f(y + t(x - y))^T(x - y)dt - \nabla(y)^T(x - y) \right| \\
&\leq \int_0^1 \|\nabla f(y + t(x - y)) - \nabla f(y)\| \cdot \|x - y\| dt \\
&\leq \int_0^1 Lt\|x - y\|^2 dt \\
&= \frac{L}{2}\|x - y\|^2.
\end{aligned}$$

□

Lemma 1 implies that if f is convex and L-smooth, then for any $x, y \in \mathbb{R}^n$,

$$0 \leq f(x) - f(y) - \nabla f(y)^T(x - y) \leq \frac{L}{2}\|x - y\|^2. \tag{3}$$

Thus, apply the iterative step from gradient descent $y = x - \frac{1}{L}\nabla f(x)$ to get the next result:

$$f(x - \frac{1}{L}\nabla f(x)) - f(x) \leq -\frac{1}{2L}\|\nabla f(x)\|^2. \tag{4}$$

The inequality (3) is often considered to be an alternative definition of smoothness [10], and will be used in the development of the new algorithms.

Lemma 2. *Suppose a function f satisfies (3), then for any $x, y \in \mathbb{R}^n$, it follows that*

$$f(x) - f(y) \leq \nabla f(x)^T (x - y) - \frac{1}{2L} \|\nabla f(x) - \nabla f(y)\|^2.$$

Proof. Let $z = y - \frac{1}{L}(\nabla f(x) - \nabla f(y))$. Then by convexity, we have

$$\begin{aligned} f(x) - f(y) =& f(x) - f(z) + f(z) - f(y) \\ \leq& \nabla f(x)^T (x - z) + \nabla f(y)^T (z - y) + \frac{L}{2} \|z - y\|^2 \\ =& \nabla f(x)^T (x - y) + (\nabla f(x) - \nabla f(y))^T (y - z) + \frac{1}{2L} \|\nabla f(x) - \nabla f(y)\|^2 \\ =& \nabla f(x)^T (x - y) - \frac{1}{2L} \|\nabla f(x) - \nabla f(y)\|^2. \end{aligned}$$

□

With the above results, we now turn to the proof of Theorem 1 [10]:

Proof. By inequality (4) and the iterative step of gradient descent, it is true that

$$f(x_{n+1}) - f(x_n) \leq -\frac{1}{2L} \|\nabla f(x_n)\|^2.$$

Now, define a term that denotes the distance from the value of f at the current point x_n to the function minimum as follows:

$$\delta_n = f(x_n) - f(x^*).$$

Thus, by applying the above definition, it follows that

$$\delta_{n+1} \leq \delta_n - \frac{1}{2L} \|\nabla f(x_n)\|^2.$$

Also, by convexity it is true that

$$\delta_n \leq \nabla f(x_n)^T (x_n - x^*) \leq \|x_n - x^*\| \cdot \|\nabla f(x_n)\|. \tag{5}$$

Now, (5) implies that

$$\delta_{n+1} \leq \delta_n - \frac{1}{2L \|x_n - x^*\|^2} \delta_n^2.$$

If it is shown that $\|x_n - x^*\|$ decreases as n increases, then this would imply

$$\delta_{n+1} \leq \delta_n - \frac{1}{2L \|x_1 - x^*\|^2} \delta_n^2. \tag{6}$$

By Lemma 2, we have

$$f(x) - f(y) \leq \nabla f(x)^T (x - y) - \frac{1}{2L} \|\nabla f(x) - \nabla f(y)\|^2.$$

Switching x and y, we have

$$f(y) - f(x) \leq \nabla f(y)^T (y - x) - \frac{1}{2L} \|\nabla f(y) - \nabla f(x)\|^2.$$

Summing the two inequalities above gives

$$(\nabla f(x) - \nabla f(y))^T (x - y) \geq \frac{1}{L} \|\nabla f(x) - \nabla f(y)\|^2.$$

Now, $\nabla f(x^*) = 0$, so

$$\begin{aligned}
\|x_{n+1} - x^*\|^2 =& \|x_n - \frac{1}{L} \nabla f(x_n) - x^*\|^2 \\
=& \|x_n - x^*\|^2 - \frac{2}{L} \nabla f(x_n)^T (x_n - x^*) + \frac{1}{L^2} \|\nabla f(x_n)\|^2 \\
\leq& \|x_n - x^*\|^2 - \frac{1}{L^2} \|\nabla f(x_n)\|^2 \\
\leq& \|x_n - x^*\|^2.
\end{aligned}$$

Now, let $\omega = \frac{1}{2L\|x_1 - x^*\|^2}$, then $\omega \delta_n^2 + \delta_{n+1} \leq \delta_n$ if and only if

$$\omega \frac{\delta_n}{\delta_{n+1}} + \frac{1}{\delta_n} \leq \frac{1}{\delta_{n+1}}.$$

This implies that

$$\frac{1}{\delta_{n+1}} - \frac{1}{\delta_n} \geq \omega,$$

which implies that

$$\frac{1}{\delta_n} \geq \omega(t - 1).$$

So

$$f(x_n) - f(x^*) = \delta_n \leq \frac{1}{\omega(t - 1)},$$

and this concludes the proof. □

Applications in Machine Learning

The version of gradient descent that has been discussed so far involves minimizing a general function. Now, a common problem in machine learning is to minimize a specific kind of function called a loss function, which indicates the error of a given machine learning model. In this context, a machine learning model is associated with various parameters, and the goal is to configure these parameters so that the model's error is minimal. In general, the loss functions considered in machine learning are not always convex. The minimization is achieved by a process called "training", which involves systematically updating these parameters after evaluating the model on training data, which contains a set of input values matched with

expected output values. After training, the model's accuracy is determined by evaluating it on test data, which is also a set of input values with previously known output values, and determining the error. Naturally, the model is considered to be trained successfully when the error of expected outputs compared with predicted outputs by the model in the test data is minimal.

Thus, it is common to think of training a machine learning model as an optimization problem, in which the loss function is the objective function needing to be minimized. Then it is possible to use gradient descent to find the data values and parameters that produce the least error. There are several main methods that use gradient decent to minimize a loss function which are discussed below.

Stochastic Gradient Descent

When gradient descent is used to minimize error across a data set with many pairs of input values matched to expected output values, a popular and effective technique is to approximate the gradient at each step by a randomly selected individual pair of input/output values in the training data, or by averaging gradients of several values from the training data [13]. It has been shown that the stochastic gradient descent method is nearly optimal when minimizing the loss of a convex function [13].

Now, in order to guarantee that the stochastic gradient descent method converges, the step size of the gradient must occasionally be updated according to a decreasing sequence. This can cause the method to take longer to converge than traditional gradient descent, since decreasing the step size usually increases the amount of time it takes the algorithm to finish [15]. However, there are certain conditions that will guarantee stochastic gradient descent converges at a rate comparable to traditional gradient descent. Suppose f is the function being minimized, ∇f is the true gradient, and ∇f_i is the stochastic gradient chosen at the i-th step, then for all x in the domain of f, and for all i and for some constant $C \in \mathbb{R}$, if

$$\max_i \{\|\nabla f_i(x)\|\} \leq C\|\nabla f(x)\|,$$

then the stochastic gradient descent method will converge as quickly as traditional gradient descent [15].

Batch Gradient Descent

Generally, batch gradient descent is a method of minimizing the loss function that involves updating the parameters of a machine learning model after each value in the training data has been evaluated using the current set of parameters. An

epoch is defined to be the process of updating the model parameters after the loss function has been evaluated on a complete set of training data. In the batch method of gradient descent, the training set in each epoch is the entire training data set. In the on-line method, the training set in each epoch is a single value from the training data set. Finally, in the mini-batch method, the training set is a small, randomly selected portion of the training data set [17].

Momentum

There are many techniques which have been developed to improve the performance of gradient based methods. Most often, these techniques are implemented as a variation of stochastic gradient descent in particular. One such technique is to use a decaying sum to stabilize the direction of the gradient from one iteration to the next [8], [9]. This concept is called *momentum*, and it can be formally stated as a modification to the gradient descent method. Let μ be the momentum factor given, and let m_k denote the momentum in the k-th iteration, then modify gradient descent as follows to incorporate momentum:

Algorithm 2. Gradient Descent with Momentum *For a convex function f, an initial point x_0, and a given constant step size η. Let $m_0 = \nabla f(x_0)$. Do the following until the stopping condition is met:*

$$m_{k+1} = \mu m_k + \eta \nabla(x_k),$$
$$x_{k+1} = x_k - m_{k+1}.$$

This gives rise to another performance enhancing gradient descent modification called adaptive momentum estimation, or *adam* for short [9]. The process that this technique calls for in each iteration of the gradient descent is outlined below:

$$m_{k+1} = \mu m_k + (1-\mu)\nabla f(x_k)$$
$$x_{k+1} = x_k - \eta \frac{m_{k+1}}{1-\mu^k}.$$

There are other more recent methods that have been proven to successfully improve gradient descent. For example, one of these methods is called *adagrad*, in which the learning rate is set to be

$$\frac{\eta}{\sqrt{\Sigma_{i=1}^{k} \nabla(f(x_i))^2}}.$$

A similar method called *adadelta* performs the same process, with the exception that a fixed number of previous gradients are summed in the RMS term during each iteration. This avoids dividing by a value so large that the learning rate vanishes to zero [13].

2. Coordinate Descent

The coordinate descent method of optimization can be described as a gradient descent process in which dimensionality reduction is applied to the gradient step. Thus, in each iterative step, rather than descending by a scaled gradient vector, one or several components of the actual gradient are selected and scaled, and the descent from the current point occurs only along the coordinates of the selected components. Coordinate descent has recently become a competitive alternative to mainstream optimization methods for certain problems. This advancement has been made possible by the improvement of computational statistical methods of selecting the gradient coordinate's dimensionality reduction. We now state an important definition that sets up the discussion of coordinate descent [16].

Definition 4. *Let X, Y be sets and F be a field, then a function $f : X \times Y \mapsto F$ is said to be separable if $f(x, y) = \sum_{i=1}^{n} g_i(x)h_i(y)$, such that for all i, $g_i, h_i : X \times Y \mapsto F$.*

In the context of coordinate descent, it is common practice to separate the function $f : \mathbb{R}^n \mapsto \mathbb{R}$ into smooth and non-smooth portions as follows:

$$\min_{x} \; h(x) + \lambda\Omega(x)$$

where h is smooth and convex, and Ω may be convex, non-smooth, and separable by assumption; and $\lambda \in \mathbb{R}^+$ is a regularization constant. If Ω is separable, then

$$\Omega(x) = \Sigma_{i=1}^{n}\Omega_i(x_i),$$

where $\Omega_i : \mathbb{R} \mapsto \mathbb{R}$ for all i [3]. A basic algorithmic scheme of coordinate descent is provided below. In this implementation, a single component from the gradient is selected in each iteration, and the corresponding coordinate of the current point x is adjusted in the direction opposite the gradient's component [3]:

Algorithm 3. Coordinate Descent *Given index $k = 0$, $x_0 \in \mathbb{R}^n$, and learning rate λ, do the following until the stopping condition is met:*

1. *Choose gradient component index $i_k \in \{1, 2, ..., n\}$*
2. *Compute $x_{k+1} = x_k - \lambda[\nabla f(x_k)]_{i_k} e_{i_k}$*
3. *Set $k = k + 1$*

It is worth noting that in the basic coordinate scheme, the index i_k can be chosen in a cyclic manner, where $i_k = (i_{k-1} \bmod n) + 1$. As stated above, another

method of choosing the index i_k is to randomly select it. This is a more effective coordinate descent method, and a proof of its convergence is provided. It is first necessary to state some definitions.

Definition 5. *A function f is said to be strongly convex with a modulus of convexity $\mu > 0$ if for all x, y:*

$$f(y) - f(x) \geq \nabla f(x)^T (y - x) + \frac{\mu}{2}\|y - x\|^2.$$

Now, a small variation of the Lipschitz definition states that f is said to be L-smooth if for all $x, d \in \mathbb{R}^n$, it follows that

$$\|\nabla f(x + d) - \nabla f(x)\| \leq L\|d\|.$$

In this case, L is said to be the Lipschitz constant. In the context of coordinate descent, it is useful to apply this concept to each dimensional component. This gives rise to the following definition:

Definition 6. *Let f be an L-smooth function, then the component Lipschitz constants are defined to be the set of constants L_i for $i = 1, 2, ..., n$ such that for all $x \in \mathbb{R}^n$ and $t \in \mathbb{R}$,*

$$|[\nabla f(x + te_i)]_i - [\nabla f(x)]_i| \leq L_i|t|.$$

If this is true for all i, then f is said to be uniformly Lipschitz continuous. Also, the coordinate Lipschitz constant L_{max} is defined to be

$$L_{max} = \max_{i=1,2,...,n} L_i.$$

Now, before proving convergence of the random coordinate descent method, it is necessary to formally state the notation of a required concept:
Notation: Let E_{i_k} denote the expectation of the single random index i_k, and let E denote the expectation of all random variables $i_1, i_2, ..., i_n$ [3]. We now proceed with a proof of convergence for the random coordinate descent method [3].

Theorem 2. *Let f be a convex, uniformly Lipschitz continuous function that attains a minimum value $f^* = f(x^*)$ where x^* is in some set S. Suppose there is some finite constant R_0 such that $\max_{x^* \in S} \max_x \{\|x - x^*\| : f(x) \leq f(x_0)\} \leq R_0$. Suppose also that $\lambda = \frac{1}{L_{max}}$. Then for all $k > 0$, it is true that*

$$E(f(x_k)) - f^* \leq \frac{2nL_{max}R_0^2}{k}.$$

Furthermore, if f is strongly convex with $\mu > 0$, it follows that

$$E(f(x_k)) - f^* \leq \left(1 - \frac{\mu}{nL_{max}}\right)^k (f(x_0) - f^*).$$

Proof. By Taylor's theorem and definition of uniform Lipschitz continuity, it follows that

$$\begin{aligned} f(x_{k+1}) =& f(x_k - \lambda[\nabla f(x_k)]_{i_k} e_{i_k}), \\ \leq& f(x_k) - \lambda[\nabla f(x_k)]_{i_k}^2 + \frac{1}{2}\lambda^2 L_{i_k}[\nabla f(x_k)]_{i_k}^2, \\ \leq& f(x_k) - \lambda\left(1 - \frac{L_{max}}{2}\lambda\right)[\nabla f(x_k)]_{i_k}^2, \\ =& f(x_k) - \frac{1}{2L_{max}}[\nabla f(x_k)]_{i_k}^2. \end{aligned}$$

when $\lambda = \frac{1}{L_{max}}$.

Now, take the expectation of both sides over the random index i_k to get the following:

$$\begin{aligned} E_{i_k}(f(x_{k+1})) \leq& f(x_k) - \frac{1}{2nL_{max}}\sum_{i=1}^{m}[\nabla f(x_k)]_i^2 \\ =& f(x_k) - \frac{1}{2nL_{max}}\|\nabla f(x_k)\|^2. \end{aligned}$$

Note that i_k was chosen from $\{1, 2, ..., n\}$ with equal probability. Thus,

$$E_{i_k}(f(x_{k+1})) - f^* \leq E_{i_k}(f(x_k)) - f^* - \frac{1}{2nL_{max}}\|\nabla f(x_k)\|^2.$$

Now, let

$$\phi_k = E(f(x_k)) - f^*,$$

and take the expectation of both sides with respect to all indices to get

$$\begin{aligned} \phi_{k+1} \leq& \phi_k - \frac{1}{2nL_{max}}E(\|\nabla f(x_k)\|^2), \\ \leq& \phi_k - \frac{1}{2nL_{max}}[E(\|\nabla f(x_k)\|)]^2 \end{aligned}$$

by Jensen's inequality.

Now, f is convex, so for any $x^* \in S$, it is true that

$$\begin{aligned} f(x_k) - f(x^*) \leq& \nabla f(x_k)^T(x_k - x^*), \\ \leq& \|\nabla f(x_k)\| \cdot \|x_k - x^*\|, \\ \leq& R_0\|\nabla f(x_k)\|. \end{aligned}$$

by assumption. Thus, take the expectation of both sides to get

$$\frac{\phi_k}{R_0} \leq E(\|\nabla f(x_k)\|).$$

Substitute this bound into the previous result to get the following:

$$\phi_k - \phi_{k+1} \geq \frac{1}{2nL_{max}} \frac{1}{R_0^2} \phi_k^2.$$

Thus, it follows that

$$\begin{aligned} \frac{1}{\phi_{k+1}} - \frac{1}{\phi_k} =& \frac{\phi_k - \phi_{k+1}}{\phi_k \phi_{k+1}}, \\ \geq& \frac{\phi_k - \phi_{k+1}}{\phi_k^2}, \\ \geq& \frac{1}{2nL_{max}R_0^2}. \end{aligned}$$

By recursively summing the above result as a telescoping sum, we get the following result:

$$\frac{1}{\phi_k} - \frac{1}{\phi_0} \geq \frac{k}{2nL_{max}R_0^2}.$$

Thus, it is true that

$$\frac{1}{\phi_k} \geq \frac{k}{2nL_{max}R_0^2}.$$

So then

$$E(f(x_k)) - f^* = \phi_k \leq \frac{2nL_{max}R_0^2}{k},$$

as stated in the first part of the claim.

Now it remains to prove the part of the claim involving strong convexity. Assume that for some $\mu > 0$ and for all x, y:

$$f(y) - f(x) \geq \nabla f(x)^T (y - x) + \frac{\mu}{2} \|y - x\|^2.$$

Now, let $f^* = f(y)$ and $x = x_k$ and apply this to the statement of strong convexity:

$$f^* \geq f(x_k) - \frac{1}{2\mu} \|\nabla f(x_k)\|^2.$$

By using this inequality to bound the squared norm of the gradient, we modify the previous statement of expected values:

$$\begin{aligned} \phi_{k+1} \leq& \phi_k - \frac{\mu}{nL_{max}} \phi_k, \\ =& \left(1 - \frac{\mu}{nL_{max}}\right) \phi_k. \end{aligned}$$

Recursively substituting this expression results in the following inequality:

$$\phi_k \leq \left(1 - \frac{\mu}{nL_{max}}\right)^k \phi_0$$

So substituting back in the definition of ϕ,

$$E(f(x_k)) - f^* \leq \left(1 - \frac{\mu}{nL_{max}}\right)^k (f(x_0) - f^*),$$

which satisfies the claim. □

This proof establishes the fact that the coordinate descent scheme will eventually converge to the minimal function value, on the order of $\frac{1}{k}$.

Accelerated Coordinate Descent

It is possible to derive an algorithmic scheme that accelerates coordinate descent. This is achieved by applying the Nesterov acceleration technique to the coordinate descent scheme, taking care to integrate the randomized gradient coordinate selection into the accelerated scheme. Here is a formal statement of the algorithm [3]:

Algorithm 4. Accelerated Coordinate Descent *For a strongly convex function f with convexity modulus $\mu > 0$, let $x_0 \in \mathbb{R}^n$, $k = 0$, $v_0 = x_0$, and $\gamma_{-1} = 0$. Do the following until the stopping condition is met:*

1. *Choose γ_k to be the larger root by applying the quadratic formula to the following expression: $\gamma_k^2 + \frac{\mu\gamma_{k-1}^2 - 1}{n}\gamma_k - \gamma_{k-1}^2 = 0$*
2. *Set $\alpha_k = \frac{n - \gamma_k\mu}{\gamma_k(n^2 - \mu)}$, and $B_k = 1 - \frac{\gamma_k\mu}{n}$*
3. *Set $y_k = \alpha_k v_k + (1 - \alpha_k)x_k$*
4. *Choose index $i_k \in \{1, 2, ..., n\}$ with uniform probability*
5. *Set $x_{k+1} = y_k - \frac{1}{L_{i_k}}[\nabla f(y_k)]_{i_k} e_{i_k}$*
6. *Set $v_{k+1} = B_k v_k + (1 - B_k)y_k - \frac{\gamma_k}{L_{i_k}}[\nabla f(y_k)]_{i_k} e_{i_k}$*
7. *Set $k = k + 1$*

Let $S_0 = \sup_{x^* \in S} L_{max}\|x_0 - x^*\|^2 + \frac{f(x_0)-f^*}{n^2}$. Then the following theorem ensures that Algorithm 4 converges on the order of $\frac{1}{k^2}$ [3]:

Theorem 3. *For Algorithm 4, suppose the conditions of Theorem 2 are true, then for all $k \geq 0$,*

$$E(f(x_k)) - f^* \leq S_0 \left(\frac{n}{k+1}\right)^2.$$

The Nesterov acceleration achieves a faster rate of convergence by forward substituting accurate approximations of the objective function in several intermediate series. More information about this technique is provided in a later chapter.

3. Mirror Descent

General gradient descent methods rely on analyzing function behavior at local points within a vector space. The mirror descent algorithm was motivated by the fact that information can be gathered by looking at the objective function's dual space as well [11]. The following definitions and assumptions lay the foundation for the formal statement of the mirror descent algorithm [10].

Definition 7. *The function g on $\mathbb{R}^n$ is said to be a subgradient of f at $x \in X$ if for any $y \in X$, it is true that*

$$f(x) - f(y) \leq g^T(x - y).$$

The set of subgradients of f is denoted as $\partial(f)$.

Now, assume that the function f has a minimum, and a subgradient of f at $x \in X$ is computable and denoted by $f'(x)$.
Then let

$$\Psi : X \mapsto \mathbb{R}$$

be a continuously differentiable and strongly convex function with modulus of convexity denoted as μ.
Finally, the conjugate of Ψ is defined to be

$$\Psi^*(y) = \max_{x \in X}\{\langle x, y\rangle - \Psi(x)\},$$

which is assumed to be easily computable, and λ is considered to be the chosen step size.

Thus, a formal statement of the Mirror Descent algorithm is provided below [11].

Algorithm 5. Mirror Descent *Let the assumptions above hold true, and let* $x_0, y_0 \in X$. *Do the following until stopping condition is met:*

1. $x_k = \nabla\Psi^*(y_k)$
2. $y_{k+1} = \nabla\Psi(x_k) - \lambda f'(x_k)$
3. $x_{k+1} = \nabla\Psi^*(y_{k+1})$

A natural consequence of the above algorithmic scheme is that the subgradient iterative definition for x_{k+1} is a linearization of the more general process [11], [7]:

$$x_{k+1} \in \arg\min_{x\in X}\{\langle x, f'(x_k)\rangle + \frac{1}{2\lambda}\|x - x_k\|^2\}.$$

This gives rise to another important variation of the mirror descent method called the Subgradient Algorithm with Nonlinear Projections (SANP), for which a formal statement is given [11].

Algorithm 6. SANP *Assume the conditions of the previous algorithm (Algorithm 5).*

1. *Define a new distance-like function,* $B_\Psi : X\times\text{int}(X) \mapsto \mathbb{R}$ *such that* $B_\Psi(x, y) = \Psi(x) - \Psi(y) - \langle x - y, \nabla\Psi(y)\rangle$
2. *Do the following until the stopping condition is met:*

$$x_{k+1} = \arg\min_{x\in X}\{\langle x, f'(x_k)\rangle + \frac{1}{\lambda}B_\Psi(x, x_k)\}$$

There is a deep relationship between the mirror descent algorithm and the SANP algorithm. This resemblance is due to the fact that the optimality condition of B_Ψ gives rise to the formula for x_{k+1} in mirror descent. This is formally stated in the following theorem which along with an additional lemma gives rise to a proof that the SANP algorithm converges. Proof of this corollary and lemma is omitted [11].

Theorem 4. *The sequence* $\{x_k\} \subseteq X$ *generated by mirror descent corresponds exactly to the sequence generated by SANP.*

Lemma 3. *Let $S \subset \mathbb{R}^n$ be an open set with closure $\overline{S}$ and let $\Psi : \overline{S} \mapsto \mathbb{R}$ be continuously differentiable on S. Then for any three points $a, b \in S$ and $c \in \overline{S}$, the following holds true:*

$$B_\Psi(c, a) + B_\Psi(a, b) - B_\Psi(c, b) = \langle \nabla\Psi(b) - \nabla\Psi(a), c - a\rangle.$$

Now, define $\|z\|_* = \max_x\{\langle x, z\rangle | x \in \mathbb{R}^n, \|x\| \leq 1\}$ to be the dual conjugate norm, and assume the sequence $\{x_k\}$ produced by SANP is well defined. Then consider the following theorem [11]:

Theorem 5. *Suppose that the assumptions listed above for the mirror descent algorithm hold true, and that $x_0 \in int(X)$. Then for $k \geq 1$, it is true that*

$$\min_{1\leq s\leq k} f(x_s) - \min_{x\in X} f(x) \leq \frac{B_\Psi(x^*, x_1) + 2\mu^{-1}\Sigma_{s=1}^k \lambda_s^2 \|f'(x_s)\|_*^2}{\Sigma_{s=1}^k \lambda_s}.$$

In particular, the sequence $\min_{1\leq s\leq k} f(x_s) - \min_{x\in X} f(x)$ converges as $k \to \infty$ provided that $\lambda_s \to 0$ and

$$\sum_s \lambda_s \to \infty.$$

Proof. Let x^* be a minimal solution. Then this implies that for all $x \in X$,

$$\langle x - x_{k+1}, \lambda_k f'(x_k) + \nabla\Psi(x_{k+1}) - \nabla\Psi(x_k)\rangle \geq 0.$$

Thus, let $x = x^*$ to get the following inequality:

$$\langle x^* - x_{k+1}, \nabla\Psi(x_k) - \nabla\Psi(x_{k+1}) - \lambda_k f'(x_k)\rangle \geq 0.$$

Now, let

$$\begin{aligned} s_1 =& \langle x^* - x_{k+1}, \nabla\Psi(x_k) - \nabla\Psi(x_{k+1}) - \lambda_k f'(x_k)\rangle, \\ s_2 =& \langle x^* - x_{k+1}, \nabla\Psi(x_{k+1}) - \nabla\Psi(x_k)\rangle, \\ s_3 =& \langle x_k - x_{k+1}, \lambda_k f'(x_k)\rangle. \end{aligned}$$

Then the subgradient inequality for convex functions gives rise to the next inequality:

$$\begin{aligned} 0 \leq& \lambda_k(f(x_k) - f(x^*)), \\ \leq& \lambda_k\langle x_k - x^*, f'(x_k), \\ =& s_1 + s_2 + s_3. \end{aligned}$$

Now, by the above inequality, it follows that $s_1 \leq 0$. Furthermore, by Lemma 2.9, we have that

$$s_2 = B_\Psi(x^*, x_k) - B_\Psi(x^*, x_{k+1}) - B_\Psi(x_{k+1}, x_k).$$

Also, a general property of the convexity modulus μ is that for all $a, b \in \mathbb{R}^n$,

$$\langle a, b\rangle \leq \frac{1}{2\mu}\|a\|^2 + \frac{\mu}{2}\|b\|^2.$$

Thus, we have that

$$s_3 \leq \frac{1}{2\mu}\lambda_k^2\|f'(x_k)\|_*^2 + \frac{\mu}{2}\|x_k - x_{k+1}\|^2.$$

Recall that B_Ψ is strongly convex, meaning that

$$-B_\Psi(x_{k+1}, x_k) + \frac{\mu}{2}\|x_k - x_{k+1}\|^2 \leq 0.$$

Thus, it follows that

$$\begin{aligned}\lambda_k(f(x_k) - f(x^*)) =& s_1 + s_2 + s_3 \\ \leq& B_\Psi(x^*, x_k) - B_\Psi(x^*, x_{k+1}) + \frac{1}{2\mu}\lambda_k^2\|f'(x_k)\|_*^2.\end{aligned}$$

Thus, by summing the above inequality over $k = 1, ..., s$, we get the following expression:

$$\Sigma_{k=1}^s \lambda_k(f(x_k) - f(x^*)) \leq B_\Psi(x^*, x_1) - B_\Psi(x^*, x_{s+1}) + \frac{1}{2\mu}\Sigma_{k=1}^s\lambda_k^2\|f'(x_k)\|_*^2.$$

Since B_Ψ is a nonnegative function, the last inequality gives rise to the following result:

$$\min_{1\leq s\leq k} f(x_s) - \min_{x\in X} f(x) \leq \frac{B_\Psi(x^*, x_1) + \frac{1}{2\mu}\Sigma_{s=1}^k\lambda_s^2\|f'(x_s)\|_*^2}{\Sigma_{s=1}^k\lambda_s}.$$

Note that this result proves the overall convergence of SANP, so the first part of the claim is satisfied. Also, if $\lambda_k \to 0$ and $\Sigma_{k=1}^\infty \lambda_k = \infty$ as assumed, it follows from this last inequality as well that

$$\min_{1\leq s\leq k} f(x_s) - \min_{x\in X} f(x) \to 0.$$

So this proves the second part of the claim. □

4. Ellipsoid Method

The Ellipsoid method is an optimization algorithm that starts out with an ellipsoid containing a portion of a function with the desired minimal point in $\mathbb{R}^n$. It progresses by using the gradient to find a half-space that cuts the function into portions with minimal points, and those without minimal points, and then creates a new, smaller ellipsoid around the portion of the function with the minimal points [4]. Notice that this is not a method of gradient descent. So each time an approximal minimizer x_k is generated in each algorithmic iteration, $f(x_k)$ is added to a list, and the proposed solution f^* is said to be $f^* = \min_{k=1,2,\dots} f(x_k)$. The algorithm converges when f^* falls below a given error threshold. It has been proven that this method does indeed converge in quadratic time [4]. Before formally stating the ellipsoid method's algorithmic scheme, a definition for ellipsoid is required.

Definition 8. *A set $E \subseteq \mathbb{R}^n$ is an ellipsoid if there exists a vector $a \in \mathbb{R}^n$ and a positive definite $n \times n$ matrix A such that*

$$E = E(A, a) = \{x \in \mathbb{R}^n | (x - a)^T A^{-1} (x - a) \leq 1\}.$$

This gives rise to the following theorem ([5], p69):

Theorem 6. *For every convex body $K \subset \mathbb{R}^n$ there exists a unique ellipsoid E of minimal volume containing K. Moreover, K contains the ellipsoid obtained from E by shrinking E from its center by a factor of n.*

Another definition is required to state the ellipsoid algorithm [20].

Definition 9. *Let A be a matrix, then the encoding length of A, denoted as $\langle A \rangle$, is the number of bits required to encode the entries of A in binary form. For example, if A is an $m \times n$ integer-matrix, then $\langle A \rangle = \sum_{i=1}^{m} \sum_{j=1}^{n} 1 + \lceil log_2(|a_{i,j}| + 1) \rceil$. Also, if b is a vector in $\mathbb{R}^n$, then $\langle A, b \rangle = \langle A \rangle + \langle b \rangle$.*

Now we are ready to state the ellipsoid algorithm. Consider the parameters $\rho = \frac{1}{n+1}$, $\sigma = \frac{n^2}{n^2-1}$, $\tau = \frac{2}{n+1}$, and $\xi = 1$. The ellipsoid algorithm may be speed up to quadratic time by increasing these parameters ([5], p82).

Algorithm 7. Ellipsoid Method *Given an $m \times n$ inequality system $Cx \leq d$, and let $P = \{x \in \mathbb{R}^n | Cx \leq d\}$. Let $k = 0$, $N = 2n((2n+1)\langle C\rangle + n\langle d\rangle - n^3)$, $A_0 = R^2 I_n$, for $R \leq \sqrt{}(n)2^{\langle C,d\rangle - n^2}$, and $a_0 = 0$.*
While $a_k \notin P$ and $k < N$:

1. *Choose a linear inequality $c^T x \leq \gamma$ from $Cx \leq d$ that is false when $x = a_k$*
2. *Set $b = \frac{1}{\sqrt{c^t A_k c}} A_k c$*
3. *Set $a_{k+1} = a_k - \rho b$*
4. *Set $A_{k+1} = \xi\sigma(A_k - \tau b b^T)$*

Chapter 3: Nesterov Acceleration

There is a method of improving the performance of gradient descent that was pioneered by Nesterov which was also proven to be very effective. An outline of this method is provided. All of the theorems and proofs from this chapter come directly from Nesterov's book ([18], p71-80). Now, assume that f is an L-Lipschitz function that is strongly convex with μ as its modulus of convexity. Also, let f have a minimum f^* at $x = x^*$. Then consider the following definition:

Definition 10. *A pair of sequences $\phi_k(x)$ and λ_k is said to be an estimate sequence of f if*

$$\lambda_k \to 0,$$

and for any $x \in \mathbb{R}^n$ and $k \geq 0$,

$$\phi_k(x) \leq (1 - \lambda_k)f(x) + \lambda_k\phi_0(x).$$

The concept of an estimate sequence is useful because it shows that f converges to a minimum value at the same rate that λ_k converges to 0. This is proven in the following Lemma.

Lemma 4. *If for some sequence x_k it is true that $f(x_k) \leq \phi_k^* = \min_{x\in\mathbb{R}^n} \phi_k(x_k)$, then*

$$f(x_k) - f^* \leq \lambda_k[\phi_0(x^*) - f^*] \to 0.$$

Proof. By assumption and definition of estimate sequences,

$$\begin{aligned} f(x_k) \leq & \min_{x\in\mathbb{R}^n} \phi_k(x_k), \\ \leq & \min_{x\in\mathbb{R}^n} (1 - \lambda_k)f(x) + \lambda_k\phi_0(x), \\ \leq & (1 - \lambda_k)f(x^*) + \lambda_k\phi_0(x^*). \end{aligned}$$

□

Now, we provide a recursive definition for an estimate sequence and prove that it satisfies the definition.

Lemma 5. *Let $\phi_0(x)$ be an arbitrary function on $\mathbb{R}^n$ and let y_k be an arbitrary sequence on $\mathbb{R}^n$. Also, let a_k be a sequence with terms strictly between 0 and 1 such that $\sum_{k=1}^{\infty} a_k = \infty$ and $\lambda_0 = 1$. Then (λ_k, ϕ_k) defined as follows satisfy the conditions of an estimate sequence:*

$$\begin{aligned}\lambda_{k+1} =& (1-a_k)\lambda_k,\\ \phi_{k+1}(x) =& (1-a_k)\phi_k(x) + a_k[f(y_k) + \langle \nabla f(y_k), x - y_k\rangle + \frac{\mu}{2}\|x-y_k\|^2].\end{aligned}$$

Proof. By definition, we have that

$$\phi_{k+1}(x) = (1-a_k)\phi_k(x) + a_k[f(y_k) + \langle \nabla f(y_k), x - y_k\rangle + \frac{\mu}{2}\|x-y_k\|^2].$$

Also, by strong convexity, we have that

$$f(x) \geq f(y_k) + \langle \nabla f(y_k), x - y_k\rangle + \frac{\mu}{2}\|x-y_k\|^2.$$

Thus, since $\phi_k(x) \leq (1-\lambda_k)f(x) + \lambda_k\phi_0(x)$, it follows that

$$\begin{aligned}\phi_{k+1}(x) \leq& (1-a_k)\phi_k(x) + a_k f(x),\\ =& (1-(1-a_k)\lambda_k)f(x) + (1-a_k)(\phi_k(x) - (1-\lambda_k)f(x)),\\ \leq& (1-(1-a_k)\lambda_k)f(x) + (1-a_k)\lambda_k\phi_0(x),\\ =& (1-\lambda_{k+1})f(x) + \lambda_{k+1}\phi_0(x).\end{aligned}$$

Now, notice that since $\sum_{k=1}^{\infty}(1-\lambda_k)$ diverges, the sequence λ_k must converge. Thus, the claim is true. □

Nesterov went on to prove an explicit definition of the ϕ functions of an estimate sequence for f. Note that ϕ_0 can be any function, so for simplicity's sake, it is usually chosen to be a quadratic function.

Lemma 6. *Let (λ_k, ϕ_k) be an estimate sequence of f. Then it is possible to define ϕ_k as follows:*

$$\phi_k(x) = \phi_k^*(x) + \frac{\gamma_k}{2}\|x - v_k\|^2,$$

where ϕ_k^*, γ_k, and v_k are sequences with the following definitions:

$\gamma_{k+1} = (1-a_k)\gamma_k + a_k\mu$,

$v_{k+1} = \frac{1}{\gamma_{k+1}}[(1-a_k)\gamma_k v_k + a_k\mu y_k - a_k\nabla f(y_k)]$,

$\phi_{k+1}^* = (1-a_k)\phi_k + a_k f(y_k) - \frac{a_k^2}{2\gamma_{k+1}}\|\nabla f(y_k)\|^2 + \frac{a_k(1-a_k)\gamma_k}{\gamma_{k+1}}(\frac{\mu}{2}\|y_k - v_k\|^2 + \langle \nabla f(y_k), v_k - y_k\rangle)$.

Now we are ready to derive the accelerated algorithmic scheme. Suppose the assumption of Lemma 3.1 holds true, in that

$$\phi_k^* \geq f(x_k).$$

Then substituting this inequality into the explicit definition of ϕ_k^* gives rise to the following inequality:

$$\begin{aligned}\phi_{k+1}^* \geq (1-a_k)f(x_k) + a_k f(y_k) - \frac{a_k^2}{2\gamma_{k+1}}\|\nabla f(y_k)\|^2 \\ + \frac{a_k(1-a_k)\gamma_k}{\gamma_{k+1}}\left(\frac{\mu}{2}\|y_k - v_k\|^2 + \langle \nabla f(y_k), v_k - y_k\rangle\right).\end{aligned}$$

Also, by convexity, $f(x_k) \geq f(y_k) + \langle \nabla f(y_k), x_k - y_k\rangle$, which when substituted into the previous expression gives rise to the next inequality:

$$\phi_{k+1}^* \geq f(y_k) - \frac{a_k^2}{2\gamma_{k+1}}\|\nabla f(y_k)\|^2 + (1-a_k)\langle \nabla f(y_k), \frac{a_k\gamma_k}{\gamma_{k+1}}(v_k - y_k) + x_k - y_k\rangle.$$

The goal is to construct an algorithm for which $\phi_{k+1} \geq f(x_{k+1})$ since that would guarantee $f(x_k) - f^*$ converges according to some sequence λ_k.
Notice from result (4) in Chapter 2, the following inequality holds true:

$$f(y_k) - \frac{1}{2L}\|\nabla f(y_k)\|^2 \geq f(x_{k+1}).$$

Thus, let $\gamma_{k+1} = La_k^2 = (1-a_k)\gamma_k + a_k\mu$, so then

$$\begin{aligned}\phi_{k+1}^* \geq & f(y_k) - \frac{1}{2L}\|\nabla f(y_k)\|^2 + (1-a_k)\langle \nabla f(y_k), \frac{a_k\gamma_k}{\gamma_{k+1}}(v_k - y_k) + x_k - y_k\rangle, \\ \geq & f(x_{k+1}) + (1-a_k)\langle \nabla f(y_k), \frac{a_k\gamma_k}{\gamma_{k+1}}(v_k - y_k) + x_k - y_k\rangle.\end{aligned}$$

Now, we can solve a_k from the quadratic equation above, and this is satisfactory since we have freedom of choice for a_k. We also have freedom of choice for y_k, so we choose y_k such that $\frac{a_k\gamma_k}{\gamma_{k+1}}(v_k - y_k) + x_k - y_k = 0$, so

$$y_k = \frac{a_k\gamma_k v_k + \gamma_{k+1}x_k}{a_k\gamma_k + \gamma_{k+1}} = \frac{a_k\gamma_k v_k + \gamma_{k+1}x_k}{\gamma_k + a_k\mu}.$$

This gives rise to the algorithmic scheme:

Algorithm 8. Nesterov Accelerated Gradient Descent, Scheme 1
Let $x_0 \in \mathbb{R}^n$ and $\gamma_0 > 0$ and set $v_0 = x_0$
Do the following until the stopping condition is met:

1. *Compute $a_k \in (0,1)$ from the equation $La^2 = (1-a_k)\gamma_k + a_k\mu$*

2. *Set* $\gamma_{k+1} = (1 - a_k)\gamma_k + a_k\mu$

3. *Set* $y_k = \frac{a_k\gamma_k v_k + \gamma_{k+1}x_k}{\gamma_k + a_k\mu}$

4. *Compute* $x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k)$

5. *Set* $v_{k+1} = \frac{1}{\gamma_{k+1}}[(1 - a_k)\gamma_k v_k + a_k\mu y_k - a_k\nabla f(y_k)]$

Thus, these terms were derived from the inequality which serves as the condition for Lemma 3.1, so it follows from that lemma that $f(x_k)$ will converge to a minimum as λ_k converges to 0. This scheme can be simplified by substituting the definition of v_k into the formula for y_k, and setting $B_k = \frac{a_{k+1}\gamma_{k+1}(1-a_k)}{a_k(\gamma_{k+1}+a_{k+1}\mu)}$, to get the following definition of y_k:

$$y_k = x_{k+1} + B_k(x_{k+1} - x_k).$$

This makes the update step for v_k unnecessary, since it is handled in place by the new definition of y_k. Now, we substitute in the equation $\gamma_{k+1} = La_k^2$ and simplify to get the following algorithmic scheme:

Algorithm 9. Nesterov Accelerated Gradient Descent, Scheme 2
Let $x_0 \in \mathbb{R}^n$, $y_0 = x_0$, *and* $a_0 \in (0, 1)$
Do the following until the stopping condition is met:

1. *Compute* $x_{k+1} = y_k - \frac{1}{L}\nabla f(y_k)$

2. *Compute* $a_k \in (0, 1)$ *from the equation* $a_{k+1}^2 = (1 - a_{k+1})a_k^2 + a_{k+1}\frac{\mu}{L}$

3. *Set* $B_{k+1} = \frac{a_k(1-a_k)}{a_k^2+a_{k+1}}$

4. *Set* $y_{k+1} = x_{k+1} + B_{k+1}(x_{k+1} - x_k)$

In order to prove the rate at which this scheme converges, it is necessary to first state and prove the following lemma.

Lemma 7. *Nesterov Accelerated Gradient Descent, Scheme 2 is equivalent to the following gradient descent scheme:*

$$\begin{aligned} y_{k+1} &= x_k - \frac{1}{L}\nabla f(x_k), \\ \lambda_{k+1} &= \frac{1 + \sqrt{1 + 4\lambda_k^2}}{2}, \\ \gamma_{k+1} &= \frac{1 - \lambda_k}{\lambda_{k+1}}, \\ x_{k+1} &= y_{k+1} + \gamma_{k+1}(y_k - y_{k+1}). \end{aligned}$$

Proof. For simplicity, let $\mu = 0$ and use the quadratic formula to see that $a_{k+1} = \frac{1}{2}(\sqrt{a_k^4 + 4a_k^2} - a_k^2)$. So we must first establish the relationship between λ_k and a_k. Notice that $\frac{1}{a_{k+1}} = \frac{2}{\sqrt{a_k^4+4a_k^2}-a_k^2}$ and multiply by the conjugate to get the following expression:

$$\begin{aligned}\frac{1}{a_{k+1}} =& \frac{1}{2}\left(1 + \sqrt{\frac{a_k^4 + 4a_k^2}{a_k^4}}\right), \\ =& \frac{1 + \sqrt{1 + 4\frac{1}{a_k^2}}}{2}.\end{aligned}$$

Thus, the recursive structure of each sequence is preserved by the following relationship: $\lambda_k = \frac{1}{a_k}$. Now, this can be used to show the relationship between γ_k and B_k. If it can be shown that $\gamma_k = -B_k$, then this would complete the proof. Notice that

$$\gamma_{k+1} = \frac{1 - \lambda_k}{\lambda_{k+1}} = \frac{1 - \frac{1}{a_k}}{\frac{1}{a_{k+1}}}.$$

Thus, $\gamma_{k+1} = -B_{k+1}$ if and only if $\frac{-a_{k+1}}{a_k}(1 - a_k) = \frac{-a_k}{a_k^2+a_{k+1}}(1 - a_k)$
which is true if and only if $\frac{a_{k+1}}{a_k} = \frac{a_k}{a_k^2+a_{k+1}}$, and this is true if and only if $a_k^2 = a_{k+1}^2 + a_k^2 a_{k+1}$. We evaluate the right hand side:

$$a_{k+1}^2 + a_k^2 a_{k+1} = \frac{a_k^4 + 4a_k^2 - 2a_k^2\sqrt{a_k^4 + 4a_k^2} + a_k^4}{4} + \frac{2a_k^2\sqrt{a_k^4 + 4a_k^2} - 2a_k^4}{4} = a_k^2.$$

Thus, the claim is true.

□

This result will now enable us to prove the rate at which Nesterov Accelerated Gradient Descent, Scheme 2 converges [10].

Theorem 7. *Let f be convex and L-Lipschitz, the Nesterov Accelerated Gradient Descent, Scheme 2 satisfies the following:*

$$f(y_t) - f(x^*) \leq \frac{2L\|x_1 - x^*\|^2}{t^2}.$$

Proof. Notice that by gradient descent and Lemma 2.3,

$$f(y_{k+1}) - f(y_k) \leq f(x_k) - f(y_k) \leq \nabla f(x_k)^T(x_k - y_k) - \frac{1}{2L}\|\nabla f(x_k)\|^2.$$

Then by using the definition of y_{k+1} above, it is clear that the right hand side directly above equals

$$L(x_k - y_{k+1})(x_k - y_k) - \frac{L}{2}\|x_k - y_{k+1}\|^2.$$

Thus, if follows that

$$f(y_{k+1}) - f(x^*) \leq L(x_k - y_{k+1})(x_k - x^*) - \frac{L}{2}\|x_k - y_{k+1}\|^2.$$

Now, multiply $(\lambda_k - 1)$ to the inequality above the last inequality, and add that to the last inequality. Then, by setting a new variable $\sigma_k = f(y_k) - f(x^*)$ we get the following results:

$$\begin{aligned}\lambda_k\sigma_{k+1} - (\lambda_k - 1)\sigma_k \leq & L(x_k - y_{k+1})^T(\lambda_k x_k - (\lambda_k - 1)y_k - x^*) \\ & - \frac{L}{2}\lambda_k\|x_k - y_{k+1}\|^2.\end{aligned}$$

Multiply this expression by λ_k to get the following:

$$\begin{aligned}\lambda_k^2\delta_{s+1} - \lambda_{k-1}^2\delta_k \leq & \frac{L}{2}\left(2\lambda_k(x_k - y_{k+1})^T(\lambda_k x_k - (\lambda_k - 1)y_k - x^*)\right), \\ & - \|\lambda_k(y_{k+1} - x_k)\|^2, \\ = & \frac{L}{2}\left(\|\lambda_k x_k - (\lambda_k - 1)y_k - x^*\|^2 - \|\lambda_k y_{k+1} - (\lambda_k - 1)y_k - x^*\|^2\right),\end{aligned}$$

which follows from the fact that $\lambda_{k-1}^2 = \lambda_k^2 - \lambda_k$ and for any vectors u and v, $2v^Tu - \|v\|^2 = \|u\|^2 - \|u - v\|^2$.

Now, by definition, the following equations hold true:

$$\begin{aligned}x_{k+1} = & y_{k+1} + \gamma_{k+1}(y_k - y_{k+1}), \\ \lambda_{k+1}x_{k+1} = & \lambda_{k+1}y_{k+1} + (1 - \lambda_k)(y_k - y_{k+1}), \\ \lambda_{k+1}x_{k+1} - (\lambda_{k+1} - 1)y_{k+1} = & \lambda_k y_{k+1} - (\lambda_k - 1)y_k.\end{aligned}$$

Let $w_k = \lambda_k x_k - (\lambda_k - 1)y_k - x^*$. Then combining the last two previous results gives rise to the following:

$$\lambda_k^2\sigma_{k+1} - \lambda_{k-1}^2\sigma_k^2 \leq \frac{L}{2}(\|u_k\|^2 - \|u_{k+1}\|^2).$$

Thus, form a telescoping sum by adding up both sides of the inequality from $k = 1$ to $k = t - 1$ to get the following result:

$$\sigma_t \leq \frac{L}{2\lambda_{t-1}^2}\|u_1\|^2.$$

Then by induction, it is clear to see that $\lambda_{t-1} \geq \frac{t}{2}$. Thus, the claim is true.

□

Now, a_k and B_k from Nesterov Accelerated Gradient Descent, Scheme 2 can be estimated by appropriate constants $a_k = \sqrt{\frac{\mu}{L}}$ and $B_k = \frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}$ to produce the following algorithmic scheme:

Algorithm 10. Nesterov Accelerated Gradient Descent, Scheme 3
Let $x_0 \in \mathbb{R}^n$, $y_0 = x_0$
Do the following until the stopping condition is met:

1. *Compute* $y_{k+1} = x_k - \frac{1}{L}\nabla f(x_k)$
2. *Set* $x_{k+1} = y_{k+1} + \frac{\sqrt{L}-\sqrt{\mu}}{\sqrt{L}+\sqrt{\mu}}(y_{k+1} - y_k)$

By setting $\kappa = \frac{L}{\mu}$, we may prove this scheme converges according to exponential decay with the following theorem [10].

Theorem 8. *Let f be strongly convex with μ modulus of convexity and L-Lipschitz, then the Nesterov acceleration converges as follows:*

$$f(y_k) - f^* \leq \frac{L+\mu}{2}\|x_1 - x^*\|^2 e^{-\frac{k-1}{\sqrt{\kappa}}}.$$

Proof. Consider the following recursive function definitions:

$$\begin{aligned}\phi_1(x) =& f(x_1) + \frac{\mu}{2}\|x - x_1\|^2;\\ \phi_{k+1}(x) =& \left(1 - \frac{1}{\sqrt{\kappa}}\right)\phi_k(x) + \frac{1}{\sqrt{k}}\left(f(x_k) + \nabla f(x_k)^T(x - x_k) + \frac{\mu}{2}\|x - x_k\|^2\right).\end{aligned}$$

Thus, by strong convexity, as k increases, ϕ_k will get closer to f according to the following inequality:

$$\phi_{k+1}(x) \leq f(x) + \left(1 - \frac{1}{\sqrt{\kappa}}\right)^k (\phi_1(x) - f(x)).$$

Now, it is proven by induction that $f(y_k) \leq \min_{x\in\mathbb{R}^n} \phi_k(x)$. The details of this proof are meticulous, so they are omitted. Therefore, it follows that

$$\begin{aligned}f(y_k) - f^* \leq& \phi_k(x^*) - f(x^*),\\ \leq& f(x^*) + \left(1 - \frac{1}{\sqrt{\kappa}}\right)^k (\phi_1(x^*) - f(x^*)) - f(x^*),\\ \leq& \left(1 - \frac{1}{\sqrt{\kappa}}\right)^k (\phi_1(x^*) - f(x^*)),\\ \leq& \left(1 - \frac{1}{\sqrt{\kappa}}\right)^{k-1} (f(x_1) + \frac{\mu}{2}\|x^* - x_1\|^2 - f(x^*)).\end{aligned}$$

And by convexity, it follows that

$$\begin{aligned} f(y_k) - f^* &\leq \left(1 - \frac{1}{\sqrt{\kappa}}\right)^{k-1} \left(\frac{L+\mu}{2}\|x^* - x_1\|^2\right), \\ &\leq \frac{L+\mu}{2}\|x^* - x_1\|^2 \left(1 - \frac{1}{\sqrt{\kappa}}\right)^{k-1}, \\ &\leq \frac{L+\mu}{2}\|x^* - x_1\|^2 \left(1 - \frac{1}{\sqrt{\kappa}(k-1)}\right)^{k-1}. \end{aligned}$$

Thus, as k increases, the farthest-right term takes on the form of the exponential, so the claim is true. □

Chapter 4: New Adaptive Methods

The performance of a gradient-based algorithm heavily depends on the choice of step size, or the learning rate. A learning rate that is too small will not only take a long time to converge, but sometimes when coupled with the vanishing gradient, may lead the algorithm to get stuck before reaching the minimum point. Also, a learning rate that is too large may lead the algorithm to diverge. Thus, choosing the right learning rate is the main challenge of these algorithms. Indeed, as we will discuss later in this chapter, a good learning rate is determined by how fast the gradient changes its direction. Furthermore, in real applications, the rate at which the gradient changes its direction changes from location to location, and it is not feasible to have one learning rate that fits universally. For example, consider the two-dimensional Zakharov function:

$$f(x, y) = x^2 + y^2 + (x + 2y)^2 + (x + 2y)^4.$$

The function has a unique minimizer $(0, 0)$. Suppose we start at the initial point $(0.5, 0.5)$. Near $(0.5, 0.5)$, the gradient changes its direction slowly, so a larger step size is desired. As it gets closer to the minimizer at the origin, the gradient changes its direction faster. Consequently, the step size would be better if it were smaller, even though the magnitude of the gradient also gets smaller, as the decrease in magnitude may not be able to catch up to the change of the direction. This case is quite different from the sphere function $g(x, y) = x^2 + y^2$, in which the direction of the gradient does not change, and only its magnitude decreases. Thus, a constant or increasing learning rate is preferable.

The goal of this work is to develop adaptive methods that automatically adjust their learning rates along the way. Figure 1 illustrates this phenomenon for the Zakharov function introduced above. By choosing a learning rate according to the local curvature of the gradient flow curve, one can speed up the process of searching for the minimizer. Without using an adaptive rate, it requires several hundred iterations to reach the minimum, while using an adaptive learning rate, seven steps are enough to get the same level of approximation.

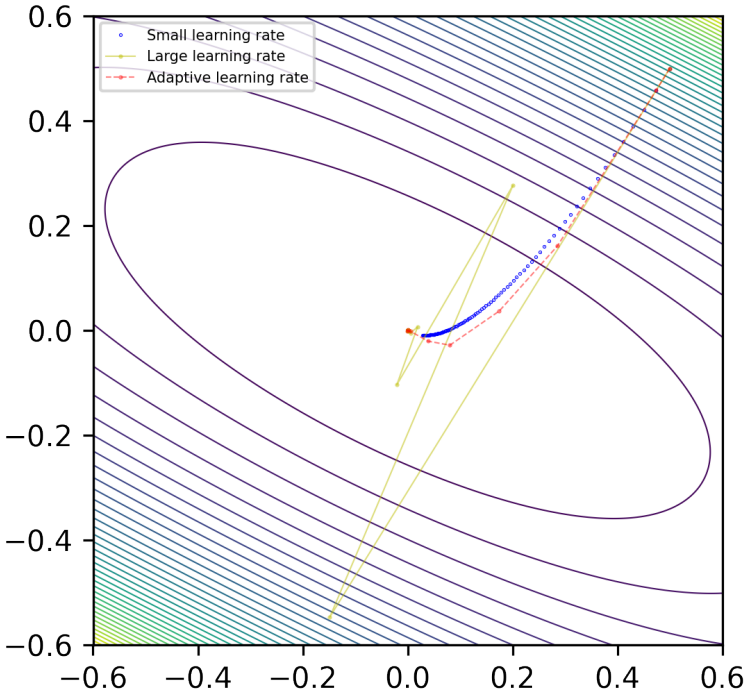


Figure 1: Adaptive learning rate can significantly improve the performance of gradient-based algorithms.

We now discuss the adaptive methods of gradient descent to serve as an alternative to the Nesterov acceleration. We introduce three new methods. Each of these methods relies on update steps which take information about the current run into consideration.

Adaptive Gradient Descent

To introduce the adaptive gradient descent methods, we start with the Taylor expansion of multivariate functions. Suppose f is a n-dimensional function defined on $\mathbb{R}^n$ that has continuous second partial derivatives. For $x = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$ and $y = (y_1, y_2, \ldots, y_n) \in \mathbb{R}^n$, we define $g(t) = f(x + t(y - x))$, $t \in \mathbb{R}$. The second order Taylor approximation of g gives

$$g(t) - g(0) = tg'(0) + \frac{t^2}{2}g''(\xi), \tag{7}$$

for some $\xi \in (0, t)$. By the Chain Rule, we have

$$g'(t) = \sum_{i=1}^{n} \frac{\partial f}{\partial x_i}(y_i - x_i) = (y - x)^T \nabla f(x + t(y - x)),$$

and

$$g''(t) = \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial^2 f}{\partial x_i \partial x_j}(y_i - x_i)(y_j - x_j) = (y-x)^T H(x + t(y-x))(y-x),$$

where H is the Hessian matrix defined by

$$H(z) = \begin{pmatrix} \frac{\partial^2 f(x)}{\partial x_1^2} & \frac{\partial^2 f(x)}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f(x)}{\partial x_2 \partial x_1} & \frac{\partial^2 f(x)}{\partial x_2^2} & \cdots & \frac{\partial^2 f(x)}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \cdots & \vdots \\ \frac{\partial^2 f(x)}{\partial x_n \partial x_1} & \frac{\partial^2 f(x)}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f(x)}{\partial x_n^2} \end{pmatrix}.$$

Thus, (7) can be rewritten as

$$f(x + t(y-x))) - f(x) = t(y-x)^T \nabla f(x) + \frac{t^2}{2}(y-x)^T H(x + \xi(y-x))(y-x).$$

If f is convex, then by (1), we have

$$f(x + t(y-x)) - f(x) \geq t(y-x)^T \nabla f(x),$$

which implies that $(y-x)^T H(x + \xi(y-x))(y-x) \geq 0$. In particular, if we let $y - x = -\nabla f(x)$, then we obtain

$$f(x - t\nabla f(x)) - f(x) = -t\|\nabla f(x)\|^2 + \frac{t^2}{2}(\nabla f(x))^T H(x - \xi \nabla f(x))\nabla f(x). \quad (8)$$

Suppose x, y are close to each other. Then by the assumption that f has continuous second partial derivatives, we have

$$(\nabla f(x))^T H(x - \xi \nabla f(x))\nabla f(x) \approx (\nabla f(x))^T H(x) \nabla f(x),$$

which is independent of t. Thus, to minimize the value $f(x - t\nabla f(x))$, the optimal choice of t would be

$$t \approx \frac{\|\nabla f(x)\|^2}{(\nabla f(x))^T H(x) \nabla f(x)}.$$

However, computing the Hessian matrix is rather costly. In what follows, we need to obtain some estimate for the quantity $(\nabla f(x))^T H(x) \nabla f(x)$. For each $1 \leq i \leq n$, we define

$$h(t) = \frac{\partial f}{\partial x_i}(x + t(y-x)) - \frac{\partial f}{\partial x_i}(x).$$

By the Mean Value Theorem, we have

$$h(t) = h(t) - h(0) = th'(\eta)$$

for some $\eta \in (0,t)$. Also, by the Chain Rule, we have

$$h'(t) = \sum_{j=1}^{n} \frac{\partial^2 f(x)}{\partial x_i \partial x_j}(y_j - x_j),$$

and

$$\frac{\partial f}{\partial x_i}(y) - \frac{\partial f}{\partial x_i}(x) = t \sum_{j=1}^{n} \frac{\partial^2 f(x+\eta(y-x))}{\partial x_i \partial x_j}(y_j - x_j).$$

Thus,

$$\nabla f(y) - \nabla f(x) = (y-x)^T H(x+\eta(y-x)).$$

Consequently,

$$\begin{aligned}(y-x)^T H(x+\eta(y-x))(y-x) =& (y-x)^T(\nabla f(y) - \nabla f(x)) \\ \leq& \|y-x\| \cdot \|\nabla f(y) - \nabla f(x)\|,\end{aligned}$$

by the Cauchy-Schwartz inequality. Now, when x and y are close to each other, by the continuity of the second partial derivatives of f, we have

$$(\nabla f(x))^T H(x) \nabla f(x) \lesssim \|y-x\| \cdot \|\nabla f(y) - \nabla f(x)\|.$$

Thus,

$$t \approx \frac{\|\nabla f(x)\|^2}{(\nabla f(x))^T H(x) \nabla f(x)} \geq \frac{\|y-x\|}{\|\nabla f(y) - \nabla f(x)\|}.$$

This leads to our first adaptive gradient decent algorithm. It is clear that this should yield good results, since $\frac{1}{L}$ was proven to be the optimal step size for convex functions in gradient descent ([18], p69):

Algorithm 11. Adaptive Gradient Descent I. *For a convex function f that has continuous second partial derivatives, the algorithm consists of these steps:*

1. *Pick an initial point x_0, compute $\nabla f(x_0)$;*
2. *Choose a small δ, say $\delta = 10^{-6}$, compute $x_1 = x_0 - \delta \nabla f(x_0)$ and $\nabla f(x_1)$.*
3. *For $n = 1, 2, \ldots,$ until a stopping condition is met, compute*

$$t_n = \frac{\|x_n - x_{n-1}\|}{\|\nabla f(x_n) - \nabla f(x_{n-1})\|},$$

$x_{n+1} = x_n - t_n \nabla f(x_n)$ *and* $\nabla f(x_{n+1})$.

Improved Adaptive Gradient Descent: First Improvement

One advantage of Adaptive Gradient Descent I is that both $\nabla f(x_n)$ and $\nabla f(x_{n-1})$ have been computed in the previous step. Thus, no additional expensive computation is needed. The drawback is that the learning rate estimated by $t_n = \frac{\|x_n - x_{n-1}\|}{\|\nabla f(x_n) - \nabla f(x_{n-1})\|}$ is smaller than the optimal value. This leads to conservative iterative steps. For low dimensional cases, this is not a concern. When the dimension gets higher, and x and y are close, the quantities $(y-x)^T H(x)(y-x)$ and $\|y-x\| \cdot \|\nabla f(y) - \nabla f(x)\|$ may not be close. To get a more accurate estimate of the adaptive learning rate, we propose a direct estimate of the former, under the assumption that the function f has continuous third order partial derivatives.

We reconsider the function $g(t) = f(x + t(y-x)) - f(x)$. If f has continuous third order partial derivatives, then $g'''(t)$ is continuous. By the third order Taylor expansion, we have

$$g(t) = g(0) + tg'(0) + \frac{t^2}{2}g''(0) + \frac{t^3}{6}g'''(\xi), \tag{9}$$

$$g(-t) = g(0) - tg'(0) + \frac{t^2}{2}g''(0) - \frac{t^3}{6}g'''(\eta), \tag{10}$$

for some $\xi \in (0, t)$ and $\eta \in (-t, 0)$. When ξ and η are close, by the continuity of g''', we have $g'''(\xi) \approx g'''(\eta)$. Thus, (9) and (10) implies

$$t^2 g''(0) \approx g(t) + g(-t) - 2g(0).$$

Rewriting it in terms of f, we have

$$t^2 (y-x)^T H(x)(y-x) \approx f(x + t(y-x)) + f(x - t(y-x)) - 2f(x).$$

Consequently, we have

$$\frac{\|y-x\|^2}{(y-x)^T H(x)(y-x)} \approx \frac{\|t(y-x)\|^2}{f(x + t(y-x)) + f(x - t(y-x)) - 2f(x)}.$$

In particular, if y and x are close, then by choosing $t = 1$, we have

$$\frac{\|y-x\|^2}{(y-x)^T H(x)(y-x)} \approx \frac{\|y-x\|^2}{f(y)) + f(2x-y) - 2f(x)}.$$

Thus, we are led to the following substitution

$$s_n := \frac{\|x_n - x_{n-1}\|^2}{f(x_n - 1) + f(2x_n - x_n - 1) - 2f(x_n)}$$

for the learning rate t_n in Adaptive Gradient Decent I.

This estimate is aggressive and risky. If y and x are not close enough, the estimate itself may not be accurate. When y and x are very close to each other,

the denominator on the right-hand side is very close to 0. The estimate of the fraction is then very sensitive to the computational rounding error. If the fraction (which corresponds to the learning rate) is over estimated, then the algorithm may lead to divergence. To avoid this, we use the harmonic average of t_n and s_n as the new learning rate.

Algorithm 12. Adaptive Gradient Descent II. *For a convex function f that has continuous second partial derivatives, the algorithm consists of these steps:*

1. *Pick an initial point x_0, compute $\nabla f(x_0)$;*
2. *Choose a small δ, say $\delta = 10^{-6}$, compute $x_1 = x_0 - \delta \nabla f(x_0)$ and $\nabla f(x_1)$.*
3. *For $n = 1, 2, \ldots,$ until a stopping condition is met, compute*
   - $a_n = \frac{\|\nabla f(x_n) - \nabla f(x_{n-1})\|}{\|x_n - x_{n-1}\|}$
   - $b_n = \frac{f(x_{n-1}) + f(2x_n - x_{n-1}) - 2f(x_n)}{\|x_n - x_{n-1}\|^2}$
   - $x_{n+1} = x_n - \frac{2}{a_n + b_n} \nabla f(x_n)$ *and* $\nabla f(x_{n+1})$.

Note that the estimate of s_n uses three additional function evaluations, two of which are calculated in the previous steps, but the evaluation of $f(2x_n - x_{n-1})$ is an additional calculation. If the gradient of f is given by an analytic formula, and it can be easily evaluated, this extra evaluation is a non-trivial overhead. Otherwise, the numerical calculation of the gradient of a d-dimensional function takes $2d + 1$ function evaluations, so adding one extra function evaluation only increases the computation by $1/(2d + 1)$ fraction of the effort.

Improved Adaptive Gradient Descent: Second Improvement

Assume that the function $g(t)$ defined near 0 has continuous fourth derivatives. Then by the Taylor expansion, we have

$$g(t) = g(0) + tg'(0) + \frac{t^2}{2} g''(0) + g\frac{t^3}{6} g'''(0) + \frac{t^4}{24} g^{(4)}(\xi),$$

$$g(-t) = g(0) - tg'(0) + \frac{t^2}{2} g''(0) - g\frac{t^3}{6} g'''(0) + \frac{t^4}{24} g^{(4)}(\eta),$$

for some $\xi \in (0, t)$ and $\eta \in (-t, 0)$. If ξ and η are close (e.g. when t is small), by the continuity of $g^{(4)}$ we have

$$g'''(0) \approx \frac{3[g(t) - g(-t) - 2tg'(0)]}{t^3}.$$

In particular, if $g(t) = f(x + t(y - x))$ for some x and y that are close to each other, we have

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}(y_i-x_i)(y_j-x_j)(y_k-x_k)\frac{\partial^3 f(x)}{\partial x_i \partial x_j \partial x_k} \approx 3[f(y)-f(2x-y)-(y-x)^T\nabla f(x)].$$

Together with the estimate derived earlier:

$$\frac{\|y-x\|^2}{(y-x)^T H(x)(y-x)} \approx \frac{\|y-x\|^2}{f(y))+f(2x-y)-2f(x)},$$

we now have all the estimates we need for a new and improved adaptive gradient descent. For notational convenience, we denote

$$\begin{aligned} A(x,y) =& f(y)) + f(2x-y) - 2f(x), \\ B(x,y) =& f(y) - f(2x-y) - (y-x)^T\nabla f(x). \end{aligned}$$

Note that for $g(t) = f(x + t(y - x))$, the approximation

$$g(t) = g(0) + tg'(0) + \frac{t^2}{2}g''(0) + \frac{t^3}{6}g'''(0)$$

can be rewritten as

$$f(x + t(y - x)) \approx f(x) + t(y - x)^T\nabla f(x) + \frac{t^2}{2}A(x,y) + \frac{t^3}{2}B(x,y).$$

If x and y are close, then the estimate is sharp, in which case $f(x + t(y - x))$ is minimized when the right-hand side is minimized. This happens when

$$(y - x)^T\nabla f(x) + A(x,y)t + \frac{3}{2}B(x,y)t^2 = 0,$$

from which we can solve for t to get

$$t = \frac{-2(y-x)^T\nabla f(x)}{A(x,y) + \sqrt{[A(x,y)]^2 - 6B(x,y)\cdot(y-x)^T\nabla f(x)}}.$$

This is valid only when $A(x,y)]^2 - 6B(x,y)\cdot(y-x)^T\nabla f(x) \geq 0$. When this condition fails, the right-hand side above does not have a minimum value, in such a case we simply drop the square root term in the denominator.

The above analysis leads us to the following variant of improved adaptive gradient descent:

Algorithm 13. Adaptive Gradient Descent III. *For a convex function f that has continuous third partial derivatives, the algorithm consists of these steps:*

1. *Pick an initial point x_0, compute $\nabla f(x_0)$;*

2. *Choose a small* δ*, say* $\delta = 10^{-6}$*, compute* $x_1 = x_0 - \delta\nabla f(x_0)$ *and* $\nabla f(x_1)$*.*

3. *For* $n = 1, 2, \ldots,$ *until a stopping condition is met,*

   - *Compute* $D = \|x_n - x_{n-1}\|$
   - *Compute* $A = f(x_{n-1}) + f(2x_n - x_{n-1}) - 2f(x_n)$ *and* $a = \frac{A}{D^2}$
   - *Compute* $B = f(x_{n-1}) - f(2x_n - x_{n-1}) - 2(x_n - x_{n-1})^T\nabla f(x_n)$ *and* $b = \frac{B}{D^3}$
   - *Compute* $c = \max\{a^2 - 6b\|\nabla f(x_n)\|, 0\}$
   - *Compute* $\lambda = \frac{2}{a+\sqrt{c}}$
   - *Compute* $x_{n+1} = x_n - \lambda\nabla f(x_n)$ *and* $\nabla f(x_{n+1})$*.*

Adaptive Momentum

We now discuss a new algorithmic scheme called adaptive momentum that is derived using the second order Taylor estimate of the objective function.

As before, we let $g(t) = f(x + t(y - x))$. Recall that we have derived that

$$\begin{aligned} g''(t) =& \Sigma_{i=1}^n \Sigma_{j=1}^n (y_i - x_i)(y_j - x_j)\frac{\partial^2 f}{\partial x_i \partial x_j}(x + t(y - x)) \\ =& (y - x)^T H(x + t(y - x))(y - x). \end{aligned}$$

Let $\lambda \in \mathbb{R}$ and let m_n be a momentum sequence in $\mathbb{R}^n$ such that $m_n = a_n\nabla f(x_n) + b_n m_{n-1}$, where $a_n, b_n \in \mathbb{R}$. Also, define $x_n = x_{n-1} - \lambda m_{n-1}$. Choosing $y = x_{n-1}$ and $x = x_n$, then the second order Taylor expansion of $g(t)$ gives:

$$\begin{aligned} f(x_{n-1}) - f(x_n) - \lambda m_{n-1}^T\nabla f(x_n) =& g(1) - g(0) - \lambda m_{n-1}^T\nabla f(x_n) \\ =& g'(0) + \frac{1}{2}g''(\xi) - \lambda m_{n-1}^T\nabla f(x_n). \end{aligned}$$

Calculate $g'(0)$ and $g''(0)$ and evaluate them using the definition of x_n to get the following expression:

$$\begin{aligned} & f(x_{n-1}) - f(x_n) - \lambda m_{n-1}^T\nabla f(x_n) \\ =& (x_{n-1} - x_n)\nabla f(x_n + 0(x_{n-1} - x_n)) + \frac{1}{2}(y - x)^T H(y - x) - \lambda m_{n-1}^T\nabla f(x_n) \\ =& (\lambda m_{n-1}^T)\nabla f(x_n) + \frac{1}{2}(\lambda m_{n-1}^T)H(\lambda m_{n-1}) - \lambda m_{n-1}^T\nabla f(x_n) \\ =& \frac{1}{2}(\lambda m_{n-1}^T)H(\lambda m_{n-1}) = \frac{1}{2}\lambda^2 A, \end{aligned}$$

where A denotes $(m_{n-1}^T)H(m_{n-1})$.

Now, let $y_n = x_n - \lambda \nabla f(x_n)$. Then by a process of evaluating g' and g'' as above, it follows that

$$\begin{aligned} f(y_n) - f(x_n) + \lambda \|\nabla f(x_n)\|^2 =& \frac{1}{2}\lambda^2 \nabla f(x_n)^T H \nabla f(x_n) \\ =& \frac{1}{2}\lambda^2 B, \end{aligned}$$

where B denotes $\nabla f(x_n)^T H \nabla f(x_n)$.

Next, let $z_n = y_n - \lambda m_{n-1} = x_n - \lambda \nabla f(x_n) - \lambda m_{n-1}$ and by another similar process, we get

$$f(z_n) - f(x_n) + \lambda \|\nabla f(x_n)\|^2 + \lambda m_{n-1}^T \nabla f(x_n) = \frac{1}{2}\lambda^2 B + \frac{1}{2}\lambda^2 A + \lambda^2 C,$$

where C denotes $m_{n-1}^T H \nabla f(x_n)$.

Now, using the fact that $x_{n+1} = x_n - \lambda(a_n \nabla f(x_n) + b_n m_{n-1})$, we get the following equation:

$$\begin{aligned} & f(x_{n+1}) - f(x_n) \\ =& (-\lambda(a_n \nabla f(x_n) + b_n m_{n-1}))\nabla f(x_n) \\ & + \frac{1}{2}(-\lambda(a_n \nabla f(x_n) + b_n m_{n-1}))^T H(-\lambda(a_n \nabla f(x_n) + b_n m_{n-1}) \\ =& - a_n \lambda \|\nabla f(x_n)\|^2 - b_n \lambda m_{n-1}^T \nabla f(x_n) + \frac{1}{2}\lambda^2 a_n^2 B + \frac{1}{2}\lambda^2 b_n^2 A + \lambda^2 a_n b_n C. \end{aligned}$$

Now, the goal is to minimize $f(x_{n+1})$. By the right-hand side above, we can view it as a function of a_n and b_n. This minimum value is attained when $\frac{\partial f_{n+1}}{\partial a_n} = 0$ and $\frac{\partial f_{n+1}}{\partial b_n} = 0$, leading to the following system of linear equations:

$$\begin{aligned} \frac{\partial f_{n+1}}{\partial a_n} &= -\lambda \|\nabla f(x_n)\|^2 + \lambda^2 B a_n + \lambda^2 b_n C = 0, \\ \frac{\partial f_{n+1}}{\partial b_n} &= -\lambda m_{n-1}^T \nabla f(x_n) + \lambda^2 A b_n + \lambda^2 a_n C = 0. \end{aligned}$$

Cramer's rule may be applied to compute the solutions to a_n and b_n:

$$\begin{bmatrix} \lambda B & \lambda C \\ \lambda C & \lambda A \end{bmatrix} \times \begin{bmatrix} a_n \\ b_n \end{bmatrix} = \begin{bmatrix} \|\nabla f(x_n)\|^2 \\ m_{n-1}^T \nabla f(x_n) \end{bmatrix} \tag{11}$$

We now have the necessary information to formalize the terms above into an algorithmic scheme.

Algorithm 14. Adaptive Momentum *For a convex function f that has continuous third partial derivatives, the algorithm consists of these steps:*

1. *Pick an initial point x_0, compute $\nabla f(x_0)$;*

2. *For $n = 1, 2, \ldots$, until a stopping condition is met, do the following:*

- *Set* $x_n = x_{n-1} - \lambda m_{n-1}$
- *Set* $y_n = x_n - \lambda \nabla f(x_n)$
- *Set* $z_= y_n - \lambda m_{n-1}$
- *Compute* $\lambda^2 A = 2(f(x_{n-1}) - f(x_n) - \lambda m_{n-1}^T \nabla f(x_n))$
- *Compute* $\lambda^2 B = 2(f(y_n) - f(x_n) + \lambda \|\nabla f(x_n)\|^2)$
- *Compute* $\lambda^2 C = f(z_n) - f(x_n) + \lambda \|\nabla f(x_n)\|^2 + \lambda m_{n-1}^T \nabla f(x_n) - \frac{1}{2}\lambda^2 B - \frac{1}{2}\lambda^2 A$.
- *Calculate*

$$a_n = \frac{\lambda \|\nabla f(x_n)\|^2 A - \lambda m_{n-1}^T \nabla f(x_n) C}{\lambda^2 AB - \lambda^2 C}$$

$$b_n = \frac{\lambda m_{n-1}^T \nabla f(x_n) B - \lambda \|\nabla f(x_n)\|^2 C}{\lambda^2 AB - \lambda^2 C}$$

- *Set* $m_n = a_n \nabla f(x_n) + b_n m_{n-1}$

Adaptive Nesterov Parameters

Several of the techniques discussed before were combined with new methods to develop a third algorithm. This new algorithm uses the step size parameters κ, λ, γ from both Nesterov Scheme 2 and Scheme 3 from Chapter 3. As stated in that chapter, Scheme 2 converges according to $\frac{2L\|x_1 - x^*\|^2}{k^2}$ while Scheme 3 converges according to $\frac{L+\mu}{2}\|x_1 - x^*\|^2 e^{-\frac{k}{\sqrt{\kappa}}}$, where $\kappa = \frac{L}{\mu}$. Thus, in a given iteration, if $\frac{L+\mu}{2} e^{-\frac{k}{\sqrt{\kappa}}} < \frac{2L}{k^2}$, it follows that the parameters from Scheme 3 will provide a better descent than the parameters from Scheme 2, so in this case, these are the parameters which are chosen to determine the step size. So if $\frac{L+\mu}{2} e^{-\frac{k}{\sqrt{\kappa}}} > \frac{2L}{k^2}$, the parameters from Scheme 2 will provide a better descent, and they are the ones that are chosen.

Now, recall that both Scheme 2 and Scheme 3 use two vectors labeled y_k and x_k that are updated in each iteration. We found that the algorithm occasionally diverged when the direction of descent was chosen to be the gradient evaluated at x_k in each iteration. So we tried choosing the gradient at y_k to be the direction of descent, and the algorithm still would occasionally diverge. We hypothesized that this was due to the fact that in a given iteration, out of all possible directions one could proceed in $\mathbb{R}^n$, the gradients at y_k and x_k are good candidates, but sometimes they can direct the new points towards a steep wall that results in the divergence. We correct this divergence problem by adding an additional check in each iteration that evaluates the function at two points u and v, where $u = x_k - \frac{1}{L}\nabla f(x_k)$ and

$v = x_k - \frac{1}{L}\nabla f(y_k)$, and then compares these values. If $f(u) < f(v)$, then the gradient at x_k is chosen as the direction of descent, and if $f(u) \geq f(v)$, then the gradient at y_k is chosen as the direction of descent. This is motivated by the fact that looking ahead to how the function is behaving in each of these directions helps choose the best gradient direction between x_k and y_k. This look-ahead step requires two additional function evaluations, but the algorithm converges so quickly that this does not hurt its performance overall. A formal statement of this algorithm is provided below:

Algorithm 15. Adaptive Nesterov *Let $\lambda_0 = 0$, $\delta = 1e{-}6$, and f be a convex and L-smooth function, $x_0, y_0 \in \mathbb{R}^n$, and declare new list $P = [\,]$. While the stopping condition is not met:*

1. $L = \frac{\|\nabla f(y_n) - \nabla f(y_{n-1})\|}{\|y_n - y_{n-1}\|}$
2. *Append L to P*
3. *Let P_0 be P if $|P| \leq 10$ or the last ten elements of P otherwise.*
4. $L_k = max(P_0)$, $\mu_k = min(P_0)$
5. $\kappa = \frac{L_k}{\mu_k}$, *and* $\theta = \frac{\sqrt{\kappa}-1}{\sqrt{\kappa}+1}$
6. $\lambda = \frac{1+\sqrt{1+4\lambda^2}}{2}$ *and* $\gamma = \frac{2(1-\lambda)}{1+\sqrt{1+4\lambda^2}}$
7. $\eta = min\{\theta, -\gamma\}$
8. *If* $f(x_k - \frac{1}{L_k}\nabla f(x_k)) < f(x_k - \frac{1}{L_k}\nabla f(y_k))$*:*

$$y_{k+1} = x_k - \frac{1}{L_k}\nabla f(x_k)$$

   Else:

$$y_{k+1} = x_k - \frac{1}{L_k}\nabla f(y_k)$$

9. $x_{k+1} = (1+\eta)y_{k+1} - \eta \cdot y_k$

Chapter 5: Numerical Experiments

The following commonly-used test functions were used in a series of numerical tests to determine how well some of the new methods performed in comparison to the original Nesterov acceleration:

- Norm squared function: $f(x) = \|x\|^2$
- Sum of Squares (Sum2): $f(x) = \Sigma_{i=1}^{n} i x_i^2$
- Sphere function: $f(x) = \Sigma_{i=1}^{n} x_i^2$
- Perm-β function ($\beta = 3$): $f(x) = \Sigma_{i=1}^{n} (\Sigma_{j=1}^{n} (j+\beta)(x_j^i - \frac{1}{j^i}))^2$
- Ellipse function: $f(x) = \Sigma_{i=1}^{n} \Sigma_{j=1}^{i} x_j^2$
- Sum of Different powers function: $f(x) = \Sigma_{i=1}^{n} |x_i|^{i+1}$
- Trid function: $f(x) = \Sigma_{i=1}^{n} (x_i - 1)^2 - \Sigma_{i=2}^{n} x_i x_{i-1}$
- Zakharov function: $f(x) = \Sigma_{i=1}^{n} x_i^2 + (\Sigma_{i=1}^{n} \frac{1}{2} i x_i)^2 + (\Sigma_{i=1}^{n} \frac{1}{2} i x_i)^4$
- Powell function:

  $f(x) = \Sigma_{i=1}^{\lfloor \frac{1}{4} n \rfloor} (x_{4i-3} + 10 x_{4i-2})^2 + 5(x_{4i-1} - x_{4i})^2 + (x_{4i-2} - 2x_{4i-1})^4 + 10(x_{4i-3} - x_{4i})^2$

- Brown function: $f(x) = \Sigma_{i=1}^{d-1} (x_i^2)^{x_{i+1}^2 + 1} + (x_{i+1}^2)^{x_i^2 + 1}$
- Dixon-Price function: $f(x) = (x_1 - 1)^2 + \Sigma_{i=2}^{d} i(2x_i^2 - x_{i-1})^2$
- Exponential function: $f(x) = -\exp(-0.5 \Sigma_{i=1}^{d} x_i^2)$
- Schwefel 2.23 function: $f(x) = \Sigma_{i=1}^{d} x_i^1 0$
- Xin-She Yang N.3 function:

$$f(x) = \exp(-\Sigma_{i=1}^{n} (x_i/\beta)^{2m}) - 2\exp(\Sigma_{i=1}^{n} (x_i^2)) \prod_{i=1}^{n} \cos^2(x_i).$$

Now, in order to implement the new adaptive methods, it is necessary to compute the objective function's gradient. Rather than finding the analytical formula for each function's gradient, we used a function that computes a numerical gradient approximation instead. A formal statement of our numerical gradient function is given below, given that $x \in \mathbb{R}^n$ and $h = 1e^{-8}$:

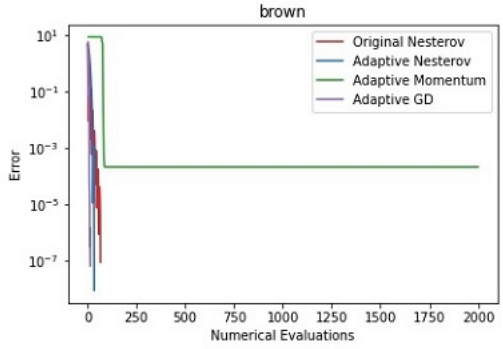
$$\nabla f(x) \approx \frac{1}{2h} \cdot [f(x + h \cdot e_1) - f(x - h \cdot e_1), \ldots, f(x + h \cdot e_n) - f(x - h \cdot e_n]$$

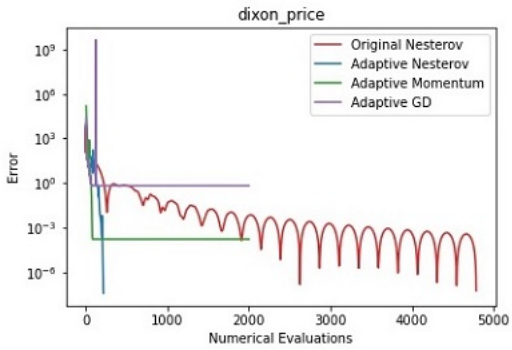
.

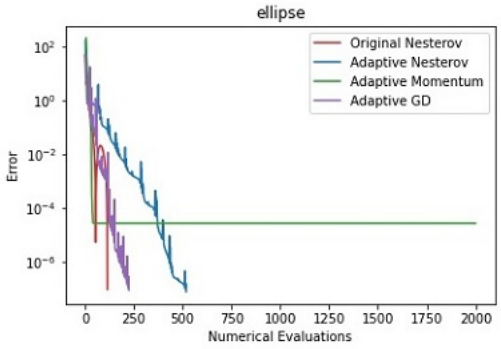
The Adaptive Gradient Descent, Adaptive Momentum, and Adaptive Nesterov algorithms were tested on the list of functions provided above, along with the original Nesterov Scheme 2 algorithm. Scheme 2 was chosen as the comparison since the non-adaptive version of Scheme 3 did not perform as well. Each algorithm was given the same initial x value, and performance was determined based on the number of numerical derivative evaluations required for each method to converge below a given error threshold. For Adaptive Gradient Descent and Adaptive Momentum, the number of numerical derivative evaluations equals the number of iterations during run-time, and for Adaptive Nesterov, the number of numerical derivative evaluations is twice the number of iterations. These tests were run in 5, 20, and 50 dimensions. They were also run on many other randomly chosen start values, with consistent results.

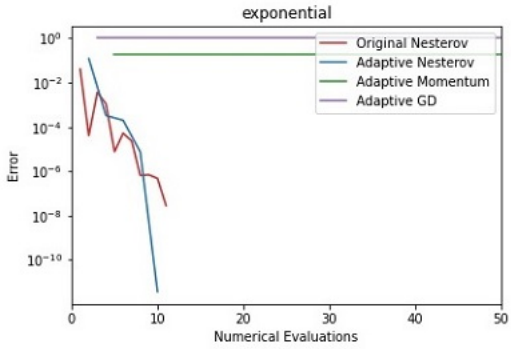
The learning curves from one of these battery of tests are shown below. In these curves, the error of an algorithm is mapped to by the number of numerical derivative evaluations taken by the algorithm. Error is computed by the difference between the test function's value at the current iteration and the test function's global minimum value. Also, the original Nesterov method requires choosing a step size, so before this battery of tests was run, a separate set of tests was performed for the original Nesterov method to find the step size that yielded its best performance for each function and dimension. This was done to guarantee that it would have a fair competition with the new adaptive methods.

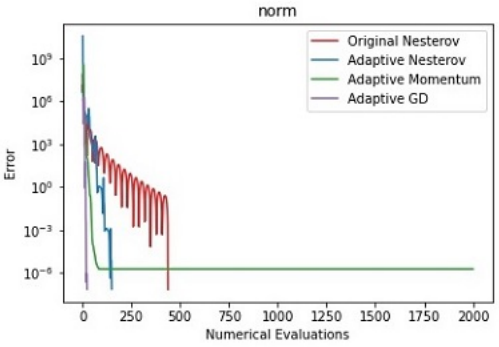
Dimension: 5

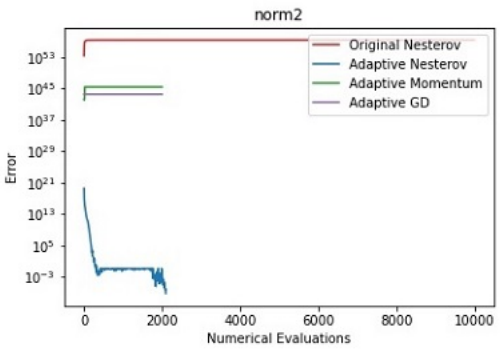

brown
Original Nesterov
Adaptive Nesterov
Adaptive Momentum
Adaptive GD
Error
Numerical Evaluations

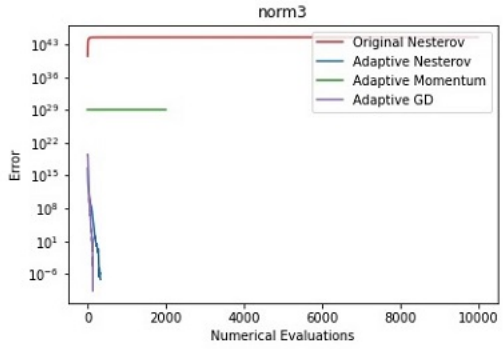

dixon_price
Original Nesterov
Adaptive Nesterov
Adaptive Momentum
Adaptive GD
Error
Numerical Evaluations

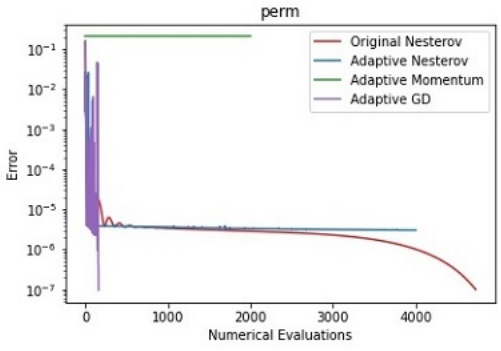

ellipse
Original Nesterov
Adaptive Nesterov
Adaptive Momentum
Adaptive GD
Error
Numerical Evaluations


exponential
Original Nesterov
Adaptive Nesterov
Adaptive Momentum
Adaptive GD
Error
Numerical Evaluations

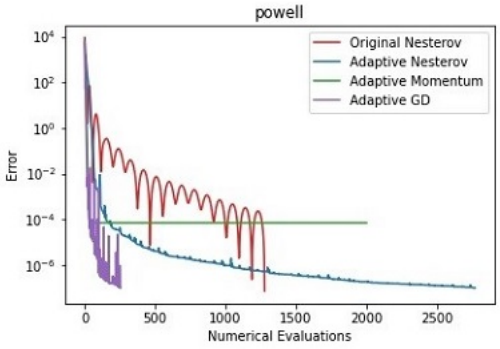

norm
Original Nesterov
Adaptive Nesterov
Adaptive Momentum
Adaptive GD
Error
Numerical Evaluations

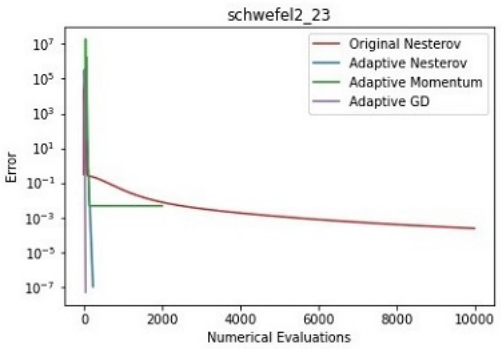

norm2
Original Nesterov
Adaptive Nesterov
Adaptive Momentum
Adaptive GD
Error
Numerical Evaluations

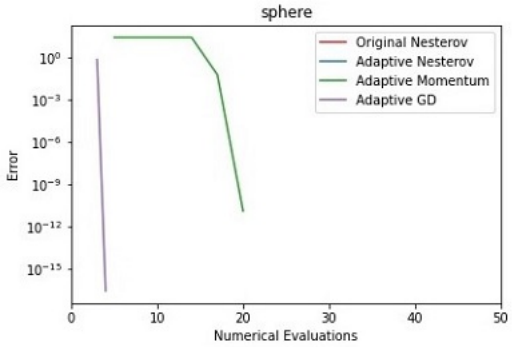

norm3
Original Nesterov
Adaptive Nesterov
Adaptive Momentum
Adaptive GD
Error
Numerical Evaluations

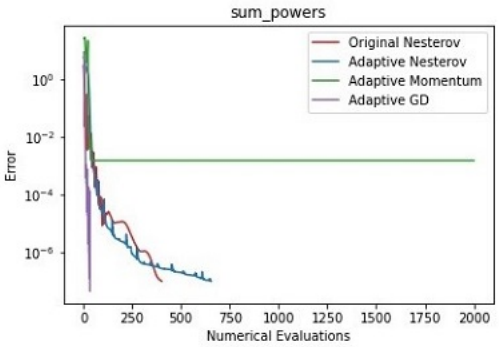

perm
Original Nesterov
Adaptive Nesterov
Adaptive Momentum
Adaptive GD
Error
Numerical Evaluations

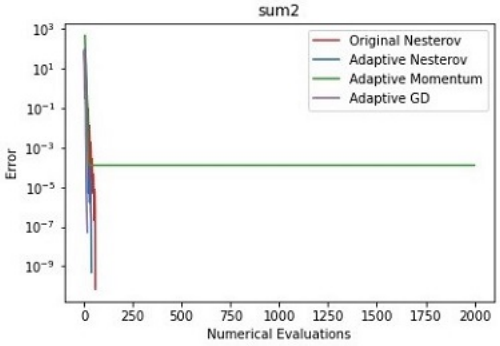
Dimension: 5

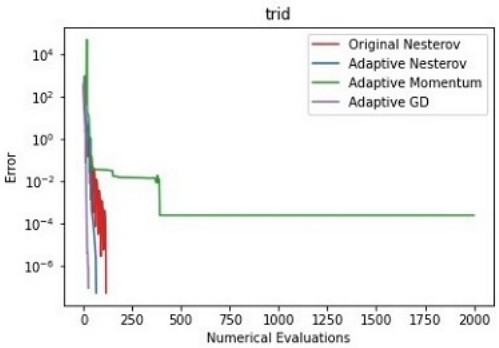

powell
Original Nesterov
Adaptive Nesterov
Adaptive Momentum
Adaptive GD
Error
Numerical Evaluations

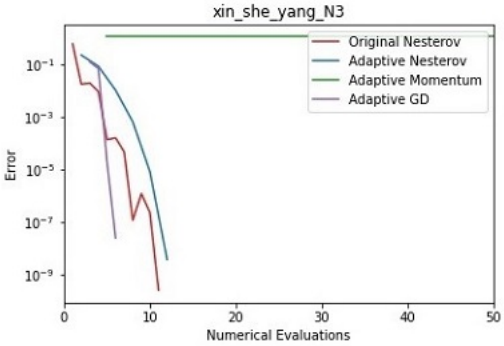

schwefel2_23
Original Nesterov
Adaptive Nesterov
Adaptive Momentum
Adaptive GD
Error
Numerical Evaluations

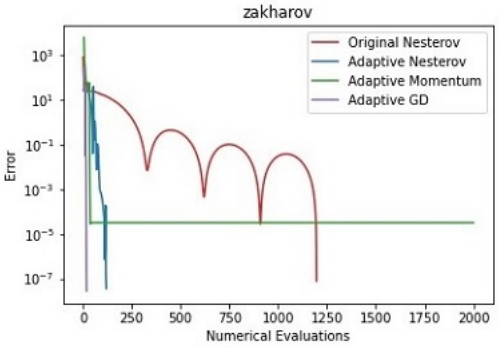

sphere
Original Nesterov
Adaptive Nesterov
Adaptive Momentum
Adaptive GD
Error
Numerical Evaluations


sum_powers
Original Nesterov
Adaptive Nesterov
Adaptive Momentum
Adaptive GD
Error
Numerical Evaluations

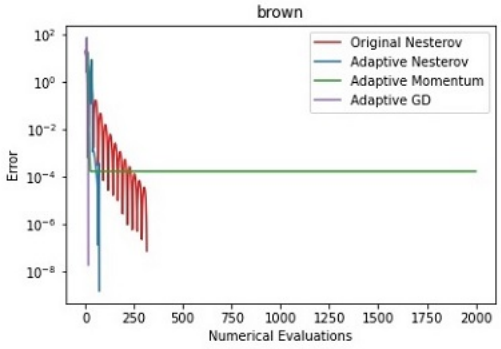

sum2
Original Nesterov
Adaptive Nesterov
Adaptive Momentum
Adaptive GD
Error
Numerical Evaluations

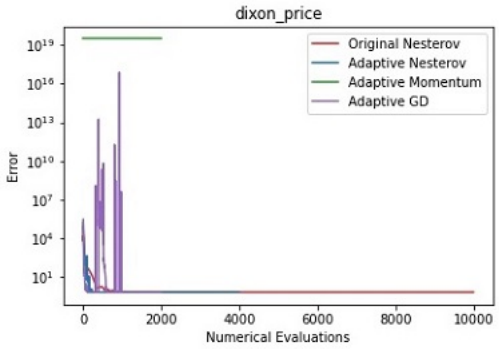

trid
Original Nesterov
Adaptive Nesterov
Adaptive Momentum
Adaptive GD
Error
Numerical Evaluations

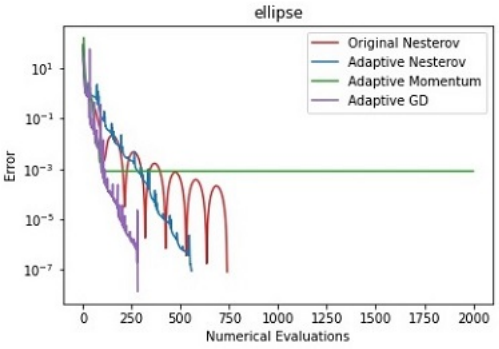

xin_she_yang_N3
Original Nesterov
Adaptive Nesterov
Adaptive Momentum
Adaptive GD
Error
Numerical Evaluations

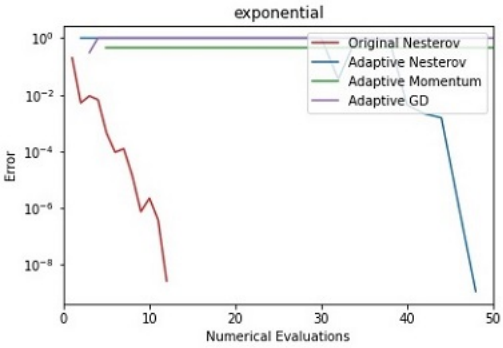

zakharov
Original Nesterov
Adaptive Nesterov
Adaptive Momentum
Adaptive GD
Error
Numerical Evaluations

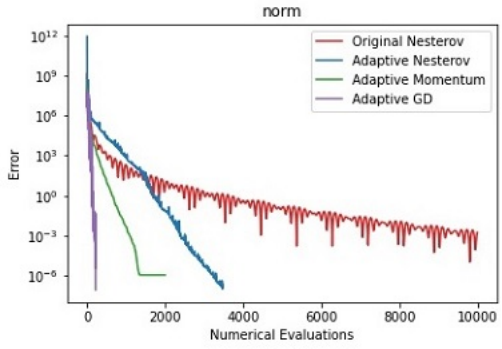
Dimension: 20

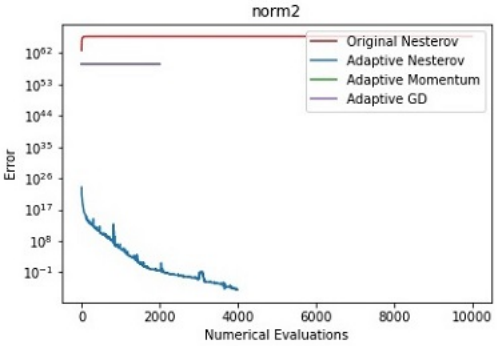


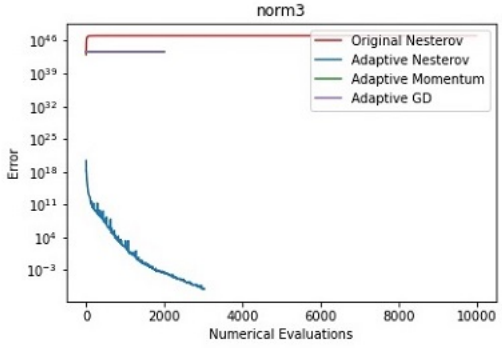


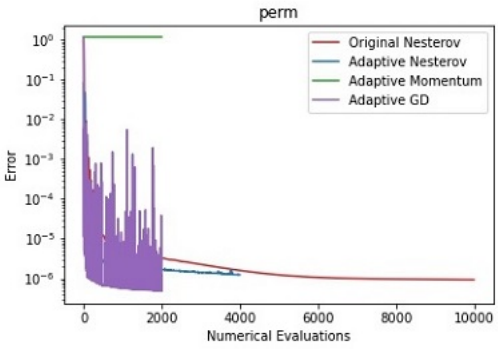




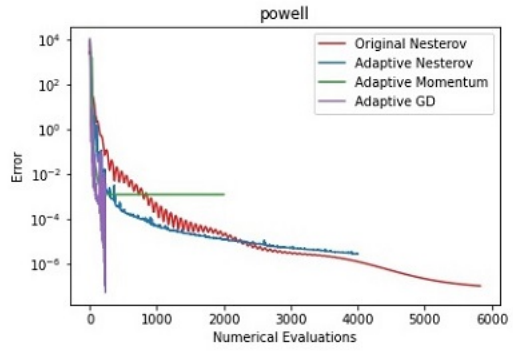


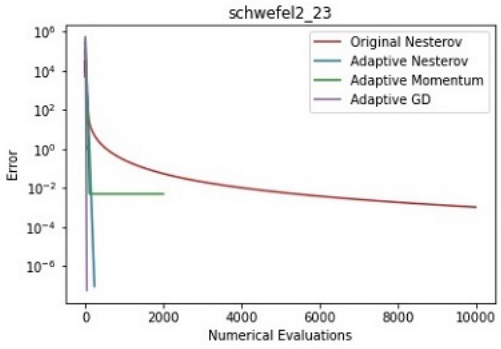


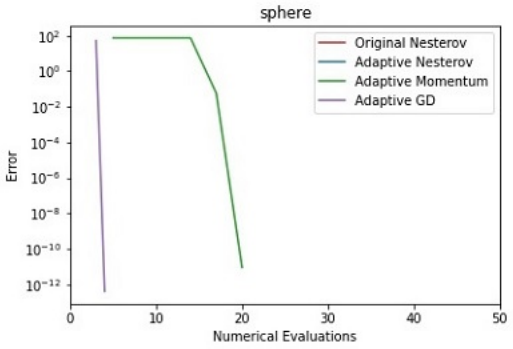


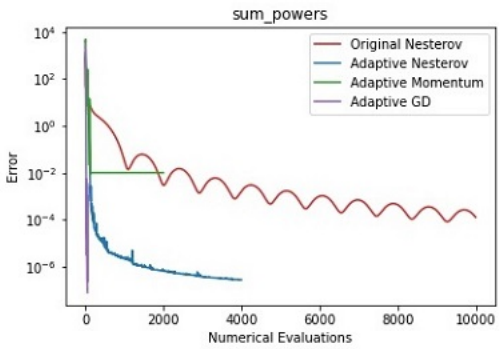


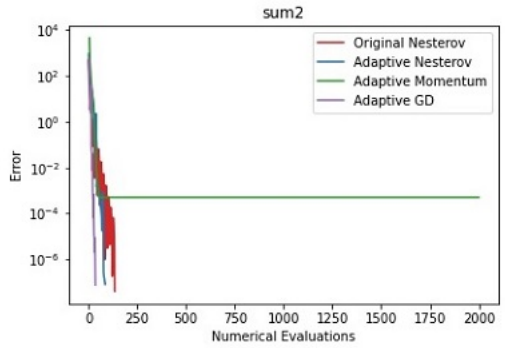
Dimension: 20

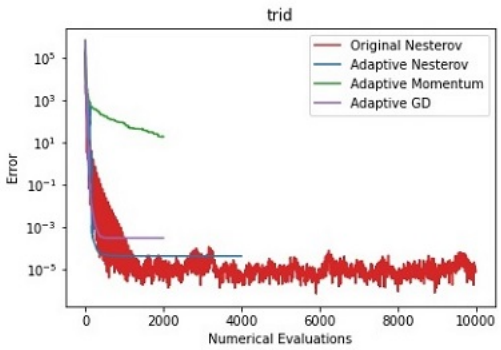


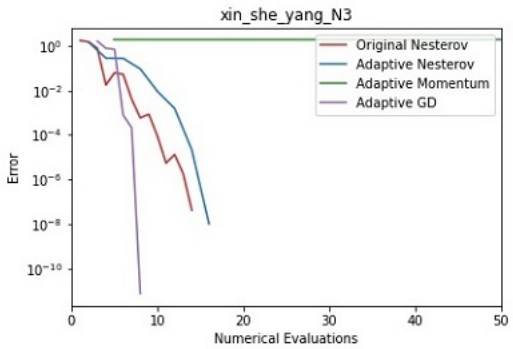


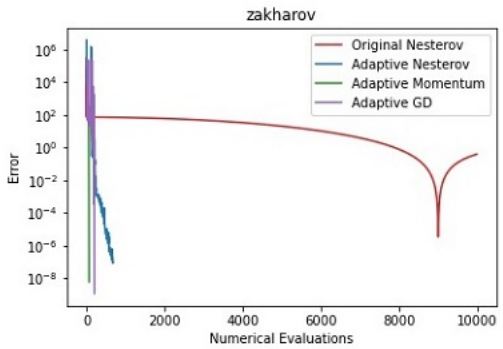




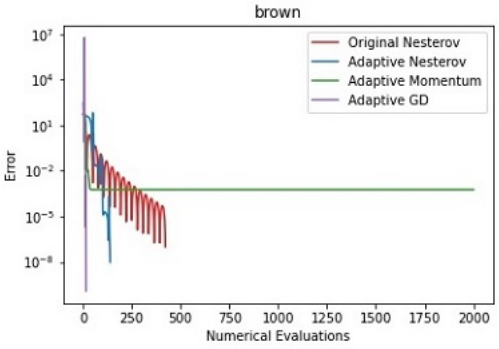


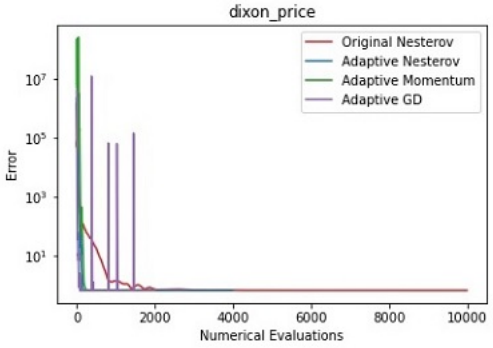


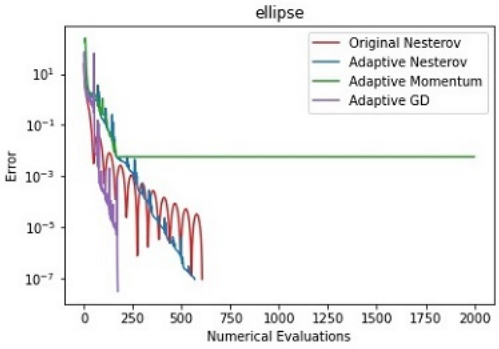


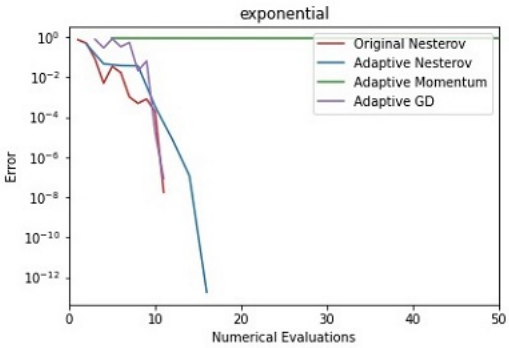


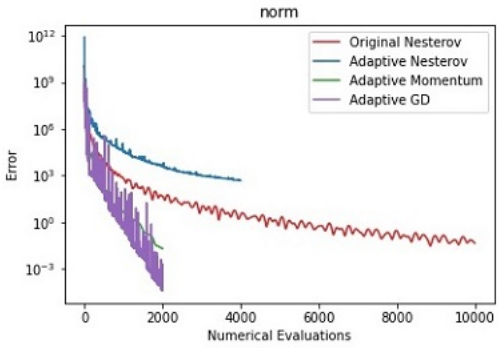
Dimension: 50

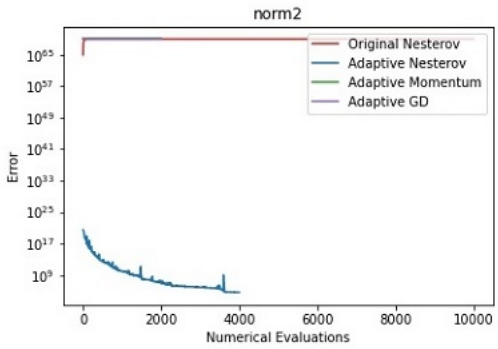


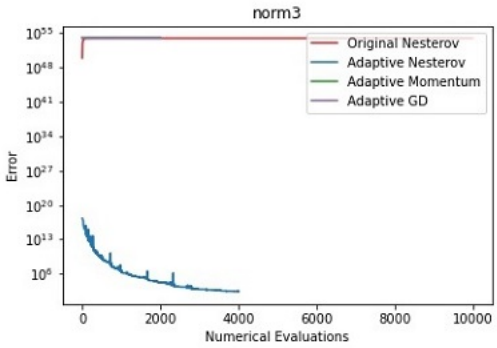


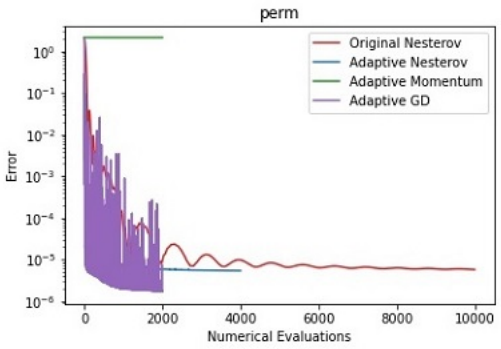




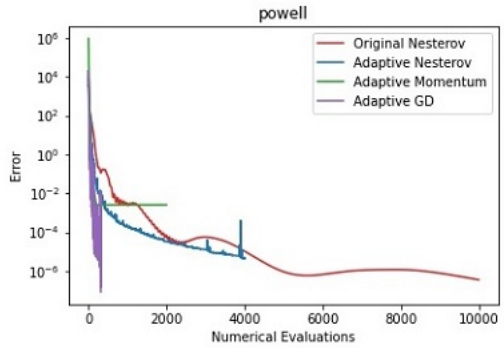


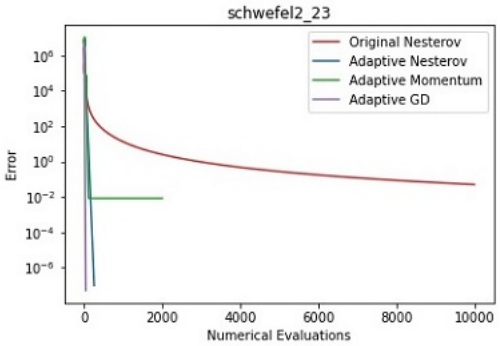


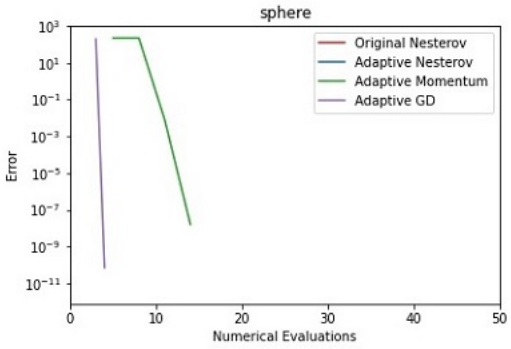


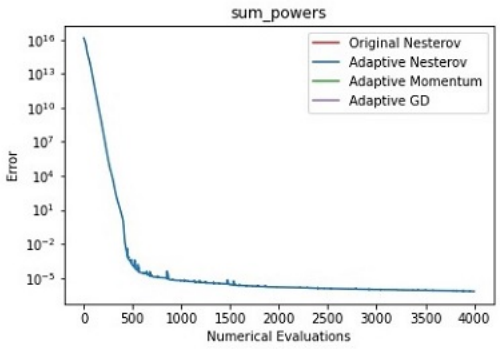


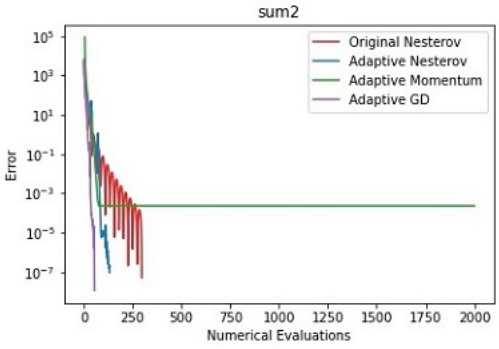
Dimension: 50

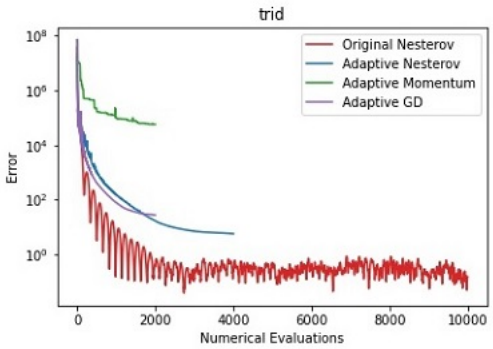


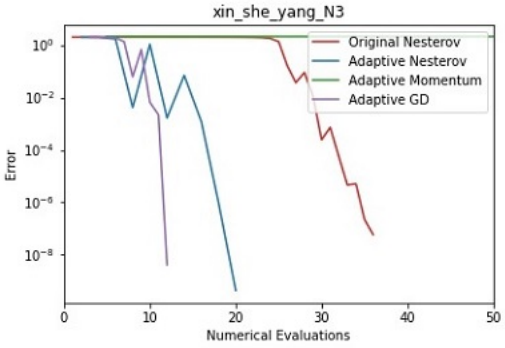


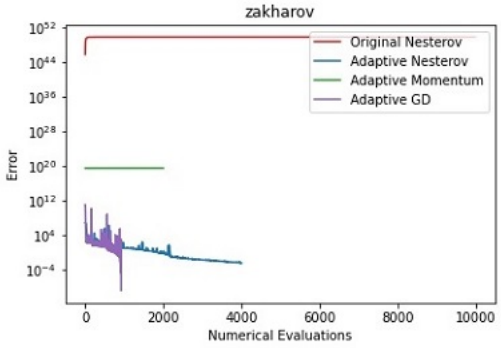












Chapter 6: Conclusion and Future Work

The goal of this research was to modify the Nesterov technique by including adaptive capability to improve its performance. The tests show that the algorithms derived previously do indeed succeed at outperforming the non-adaptive Nesterov method. The Adaptive Gradient Descent algorithm converges most quickly on most functions, but it diverges in a few examples. Adaptive Momentum behaves similarly, but it converges a little more slowly than Adaptive Gradient Descent. The Adaptive Nesterov algorithm converges more slowly than these two methods, but it never diverges, which is an improvement in its own right.

All three of these methods outperformed the original, non-adaptive Nesterov scheme in the majority of tests cases. This is significant, because unlike the original Nesterov method, they do not require a predetermined learning rate. Choosing a learning rate can be a very difficult question, and often times it is only answered by running many tests and choosing the learning rate that produced the best results. Thus, the adaptive methods presented here outperformed the original method without needing to be assigned any predetermined values. Another observation is that the learning curves of all methods oscillate between higher and lower levels of error until convergence is achieved. We conjecture that this is because the step sizes slightly overshoot the ideal descent, and thus the descent occurs in the wrong direction temporarily until it gets corrected. Naturally, the adaptive methods seem to correct faster than the non-adaptive method, which might explain their improved performance.

There are still some open questions regarding the effectiveness of the adaptive methods. One function they did not perform well on was the Dixon-Price function. This function is very flat at a small distance from the origin, and grows sharply towards infinity in the shape of a U curve. It is hypothesized that the adaptive methods can effectively crawl down the sharp wall of the U curve, but they stop progressing to smaller values when they enter the flat region of the function close to

the origin. This is because the gradient approaches zero in this region and becomes negligible, so the algorithm stops producing new points. This was not a problem unique to the adaptive methods, since the non-adaptive Nesterov method experienced the same issue, but it was thought that taking into account the curvature of this function would have allowed the descent to continue, and this was not the case.

Another interesting task would be to compare these new methods with other methods. For instance, Nesterov pioneered another optimization technique called Fast Iterative Soft Thresholding Algorithm (FISTA) that appears to take into consideration information from the Lipschitz constant. It would be meaningful to compare these methods, especially since the convergence rate of FISTA is not clearly understood [19]. It would also be beneficial to study the behavior of these new methods on non-Lipschitz functions. If a function is not Lipschitz, the L value becomes infinite. However, when it is being approximated, we conjecture that it can be reduced by dividing the learning rate by the magnitude of the gradient found by numerical approximation in order to allow the gradient descent to work. So in summary, the main result from exploring the three new methods derived in Chapter 4 is that the Lipschitz constant is an important quantity when determining the learning rate of gradient descent. Therefore, by relying on the information it provides about the current location of an objective function, it is possible to drastically improve an algorithm's rate of convergence while simultaneously removing the requirement of finding a predetermined learning rate.

Bibliography

[1] Boyd, S., & Vandenberghe, L. (2004). Convex Optimization. Cambridge: Cambridge University Press. doi:10.1017/CBO9780511804441

[2] Zeiler, M. D. (2012). Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701.

[3] Wright, S. J. (2015). Coordinate descent algorithms. Mathematical Programming, 151(1), 3-34.

[4] Boyd, S. (2018). Ellipsoid Method. Notes for EE364b.

[5] Grötschel, M., Lov'asz, L., Schrijver, A. (1993). The ellipsoid method. In Geometric Algorithms and Combinatorial Optimization (pp. 64-101). Springer, Berlin, Heidelberg.

[6] Haji, S. H., Abdulazeez, A. M. (2021). Comparison of optimization techniques based on gradient descent algorithm: A review. PalArch's Journal of Archaeology of Egypt/Egyptology, 18(4), 2715-2743.

[7] Raskutti, G., Mukherjee, S. (2015). The information geometry of mirror descent. IEEE Transactions on Information Theory, 61(3), 1451-1457.

[8] Botev, A. G. Lever, Barber, D. (2017). Nesterov's accelerated gradient and momentum as approximations to regularised update descent. 2017 International Joint Conference on Neural Networks (IJCNN). 1899-1903 doi: 10.1109/IJCNN.2017.7966082.

[9] Dozat, T. (2016). Incorporating nesterov momentum into adam.

[10] Bubeck, S. (2015). Convex optimization: Algorithms and complexity. Foundations and Trends in Machine Learning, 8(3-4), 231-357.

[11] Beck, A., Teboulle, M. (2003). Mirror descent and nonlinear projected subgradient methods for convex optimization. Operations Research Letters, 31(3), 167-175.

[12] Allen-Zhu, Z., Orecchia, L. (2014). Linear coupling: An ultimate unification of gradient and mirror descent. arXiv preprint arXiv:1407.1537.

[13] Hardt, M., Recht, B., Singer, Y. (2016, June). Train faster, generalize better: Stability of stochastic gradient descent. In International conference on machine learning (pp. 1225-1234). PMLR.

[14] Hallen, H. (2017). A Study of Gradient-Based Algorithms http://lup.lub.lu.se/student-papers/record/8904399

[15] Schmidt, M., Roux, N. L. (2013). Fast convergence of stochastic gradient descent under a strong growth condition. arXiv preprint arXiv:1308.6370.

[16] Rosenberg, M. (1969). Separable Functions and the Generalization of Matricial Structure. Mathematics Magazine 42:4, 175-186, DOI: 10.1080/0025570X.1969.11975955

[17] Wilson, D. R., Martinez, T. R. (2003). The general inefficiency of batch training for gradient descent learning. Neural networks, 16(10), 1429-1451.

[18] Nesterov, Y. (2003). Introductory lectures on convex optimization: A basic course (Vol. 87). Springer Science Business Media.

[19] Chambolle, Dossal, C. (2015). On the convergence of the iterates of" FISTA". Journal of Optimization Theory and Applications, 166(3), 25.

[20] Rebennack, S., Floudas, C. A., & Pardalos, P. M. (2009). Ellipsoid Method.