Notes on Unconstrained Optimization

Wes Cowan
Department of Mathematics, Rutgers University
110 Frelinghuysen Rd., Piscataway, NJ 08854

October 9, 2016

1 Introduction

In this set of notes, we consider the problem of unconstrained optimization. That is, given a function $f : \mathbb{R}^n \mapsto \mathbb{R}$, find an $\underline{x}^* \in \mathbb{R}^n$ such that

$$f(\underline{x}^*) = \min_{\underline{x}} f(\underline{x}). \tag{1}$$

Note, in the case of maximization, we may simply consider the problem of minimizing $-f$. The function f is referred to as the *objective function*.

In general, in the following sections, we consider f to be continuous, and at least twice differentiable over the domain ($\mathbb{R}^n$). We draw a distinction between *local* and *global* minima:

Definition 1 *A **local minimum** occurs at $\underline{x}^*$ if there exists an $\varepsilon > 0$ such that*

$$f(\underline{x}) \geqslant f(\underline{x}^*) \text{ for all } \underline{x} \in \mathbb{R}^n \text{ with } ||\underline{x} - \underline{x}^*|| < \varepsilon. \tag{2}$$

*A **global minimum** occurs at $\underline{x}^*$ if*

$$f(\underline{x}) \geqslant f(\underline{x}^*) \text{ for all } \underline{x} \in \mathbb{R}^n. \tag{3}$$

We say that a minimum is strict if in the above inequalities we have $f(\underline{x}) > f(\underline{x}^*)$ for $\underline{x} \neq \underline{x}^*$.

A local minimum is a point at which the value of the function is less than or equal to all immediately nearby or surrounding function values. A global minimum realizes the smallest possible value of the function over *all* feasible inputs. In general, based purely on local knowledge of a function and its behavior, it can be very difficult to distinguish whether you have discovered a local or a global minimum.

The problem indicated above is to be differentiated from the problem of constrained optimization or non-linear programming, which restricts the set of feasible $\underline{x}$ over which we are interested. This problem will be considered in more detail in future notes and lectures. Both problems have tremendous application in a number of areas in computer science and artificial intelligence, for instance in

model fitting, neural network training, clustering, classification, and optimal control. In low dimension, for instance over one to three variables ($n \leqslant 3$), these problems can frequently be solved using basic calculus techniques, or visualization. In low dimensions, functions can frequently be understood 'holistically' or in their entirety, leading to a direct knowledge of the minima. In application, we frequently run into very high dimensional problems, defying both visualization or global comprehension, for instance training neural networks defined with thousands of weights that need to be effectively tuned. These high dimensional problems often present unique computational difficulties for their scale. (Consider the problem of trying to invert a 10,000 x 10,000 matrix.)

We are largely interested in the following questions:

- When is a minimum of f, and corresponding minimizer $\underline{x}^*$, guaranteed to exist?
- How can minimizers be accurately and efficiently computed?
- What are the benefits and costs of various algorithmic solutions?
- What kind of theoretical guarantees do the various algorithms offer?

Section 2, contains a number of simple mathematical results and tools (for instance, multi-dimensional Taylor expansions) that will repeatedly prove useful. Section 3 contains a number of motivating examples from applications, as well as a worked problem to motivate some of the math to follow. Section 4 derives necessary and sufficient conditions for the existence of minima, based both on arguments from the derivatives of f and arguments from convex analysis. Section 5 presents a systematic way of viewing descent algorithms, algorithms for computing successively improved estimates for a function minimum, and discusses the three main algorithms in this area in detail. Section 6 applies the established descent algorithms to the problem of minimizing high dimensional quadratic functions, which is useful in itself and for motivating the convergence rate results for those algorithms, as presented in Section 7. Section 9 contains a number of related descent methods and ideas that defy easy categorization. Section 8 describes the method of computing the minimizer via conjugate directions, and derives the conjugate gradient algorithm for both quadratic and non-quadratic functions. Section 10 discusses stochastic methods of optimization, a sharp departure from the previous deterministic methods, and how this is applied to optimization over large data sets. Additional mathematical background is contained in the Appendix, as necessary.

2 Mathematical Preliminaries

The following are a number of concepts that will occur repeatedly in the rest of these notes.

The Gradient $\nabla f(\underline{x})$: As a generalization of the derivative of a one dimensional function, we have the gradient, the vector of derivatives with respect to each variable at point $\underline{x}$,

$$\nabla f(\underline{x}) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(\underline{x}) \\ \frac{\partial f}{\partial x_2}(\underline{x}) \\ \cdots\cdots \\ \frac{\partial f}{\partial x_n}(\underline{x}) \end{bmatrix}. \tag{4}$$

The gradient essentially conveys the rate of change of the function f along each of the coordinate axes. This can be utilized to determine the rate of change of the function f *in any direction*, in the following way.

The Directional Derivative $\nabla_{\underline{d}} f(\underline{x})$: We can consider the rate of change of f at $\underline{x}$ *in the direction of* $\underline{d}$ in the following way, utilizing the classical difference quotient, comparing $f(\underline{x})$ to $f(\underline{x}+\alpha\underline{d})$ for small values of α. That is, we define the directional derivative as

$$\nabla_{\underline{d}} f(\underline{x}) = \lim_{\alpha \to 0} \frac{f(\underline{x}+\alpha\underline{d}) - f(\underline{x})}{\alpha} = \nabla f(\underline{x})^{\mathrm{T}} \underline{d}. \tag{5}$$

Note, the value of the limit should be interpreted as the gradient dotted with the direction vector $\underline{d}$, essentially the projection of the gradient in the direction of movement.

Another way to consider the above is to define a function $g(\alpha) = f(\underline{x}+\alpha\underline{d})$. The derivative $g'(\alpha)$ then gives the rate of change in the direction of $\underline{d}$, which can be computed via the chain rule as $g'(\alpha) = \nabla f(\underline{x})^{\mathrm{T}} \underline{d}$.

It will frequently be useful to identify directions in which the function is decreasing, or $\underline{d}$ for which $\nabla f(\underline{x})^{\mathrm{T}} \underline{d} < 0$.

The Hessian $\nabla^2 f(\underline{x})$: As the gradient generalizes the derivative, the Hessian generalizes the second derivative. It is formed of the matrix of all possible second derivatives:

$$\nabla^2 f(\underline{x}) = \begin{bmatrix} \frac{\partial^2 f}{\partial^2 x_1}(\underline{x}) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(\underline{x}) & \frac{\partial^2 f}{\partial x_1 \partial x_3}(\underline{x}) & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(\underline{x}) \\ \frac{\partial^2 f}{\partial x_2 \partial x_1}(\underline{x}) & \frac{\partial^2 f}{\partial x_2^2}(\underline{x}) & \frac{\partial^2 f}{\partial x_2 \partial x_3}(\underline{x}) & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n}(\underline{x}) \\ \frac{\partial^2 f}{\partial x_3 \partial x_1}(\underline{x}) & \frac{\partial^2 f}{\partial x_3 \partial x_2}(\underline{x}) & \frac{\partial^2 f}{\partial x_3^2}(\underline{x}) & \cdots & \frac{\partial^2 f}{\partial x_3 \partial x_n}(\underline{x}) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(\underline{x}) & \frac{\partial^2 f}{\partial x_n \partial x_2}(\underline{x}) & \frac{\partial^2 f}{\partial x_n \partial x_3}(\underline{x}) & \cdots & \frac{\partial^2 f}{\partial x_n^2}(\underline{x}) \end{bmatrix}. \tag{6}$$

The Hessian allows us to discuss the way the gradient changes in any given direction. Note, because $\partial^2 f/(\partial x_i \partial x_j) = \partial^2 f/(\partial x_j \partial x_i$, we have that the Hessian matrix $\nabla^2 f(\underline{x})$ is always symmetric. That is, $\nabla^2 f(\underline{x})^{\mathrm{T}} = \nabla^2 f(\underline{x})$.

Taylor Series for f ***around*** $\underline{x}$: A frequent tool is going to be the approximation of f by simple functions, in particular by polynomials in each of the variables. This leads to the n-dimensional generalization of the Taylor series. In particular, assuming that f is twice differentiable, we have the following expansion of f around a point $\underline{x}$:

$$f(\underline{y}) = f(\underline{x}) + \nabla f(\underline{x})^{\mathrm{T}}(\underline{y}-\underline{x}) + \frac{1}{2}(\underline{y}-\underline{x})^{\mathrm{T}} \nabla^2 f(\underline{x})(\underline{y}-\underline{x}) + o\left(||\underline{y}-\underline{x}||^2\right), \tag{7}$$

where the asymptotic term $o\left(||\underline{y}-\underline{x}||^2\right)$ indicates a term that goes to 0 as $\underline{y} \to \underline{x}$, and goes to 0 faster than $||\underline{y}-\underline{x}||^2$ does, i.e.,

$$\lim_{\underline{y} \to \underline{x}} \frac{o\left(||\underline{y}-\underline{x}||^2\right)}{||\underline{y}-\underline{x}||^2} = 0. \tag{8}$$

Frequently, we will be interested in the expansion of $f(\underline{x}+\alpha\underline{d})$, for fixed direction $\underline{d}$, which leads to the slightly simplified form:

$$f(\underline{x}+\alpha\underline{d}) = f(\underline{x}) + \alpha \nabla f(\underline{x})^{\mathrm{T}} \underline{d} + \frac{1}{2}\alpha^2 \underline{d}^{\mathrm{T}} \nabla^2 f(\underline{x}) \underline{d} + o\left(\alpha^2\right). \tag{9}$$

Additional Mathematical Background: Additional mathematical background, such as some discussion of eigenvalues and eigenvectors, is given in Appendix A

2.1 Example Problems

1 Let A be a square matrix that is not symmetric. Prove that there is a symmetric matrix Q such that $\underline{x}^{\mathrm{T}} A \underline{x} = \underline{x}^{\mathrm{T}} Q \underline{x}$ for all $\underline{x} \in \mathbb{R}^n$.

2 Consider the function $f(\underline{x}) = \frac{1}{2}\underline{x}^{\mathrm{T}} Q \underline{x} - \underline{b}^{\mathrm{T}} \underline{x}$, for Q a symmetric matrix. Prove that $\nabla f(\underline{x}) = Q\underline{x} - \underline{b}$ and $\nabla^2 f(\underline{x}) = Q$.

3 Motivating Examples

This section presents some examples to motivate the types of problems we will consider, as well as motivating some of the math and analysis to follow.

3.1 Model Fitting

A fundamental motivation for considering these types of problems is model fitting. Given input data $\underline{x}_1, \underline{x}_2, \ldots, \underline{x}_N$ and corresponding output data $y_1, y_2, \ldots, y_n$, we want to fit a model to the data that best captures the data. Let $f_{\underline{v}}(\underline{x})$ be a function that, given input $\underline{x}$, produces a predicted output, based on a set of parameters $\underline{v}$. For instance, a linear model might be:

$$f_{a,b,c}(x_1, x_2) = ax_1 + bx_2 + c. \tag{10}$$

The model f might also be a complex neural network, defined in terms of hundreds or thousands of parameters giving the weights between different neurons.

To fit the model to the data, we want to find the values of the parameters that minimize some measure of error. There are many ways of measuring the error of a model, but a classical and frequently applicable one is the sum of the squared error, i.e.,

$$\mathrm{Error}(\underline{v}) = \sum_{i=1}^{N} \left(y_i - f_{\underline{v}}(\underline{x}_i)\right)^2. \tag{11}$$

We are interested then in an optimal set of parameters $\underline{v}^*$ such that

$$\mathrm{Error}(\underline{v}^*) = \min_{\underline{v}} \mathrm{Error}(\underline{v}) = \min_{\underline{v}} \sum_{i=1}^{N} \left(y_i - f_{\underline{v}}(\underline{x}_i)\right)^2. \tag{12}$$

In the case that error is measured in terms of sum squared error, and the model f is a linear model in terms of its variables and parameters, the optimal parameter values $\underline{v}^*$ can be solved for explicitly. This is the basis of *least squares regression*. However, we are frequently interested in introducing non-linear models, for instance neural networks and various kernel methods to capture non-linear structure in the data. In these non-linear cases, solving for the optimal parameters can often be difficult, necessitating many of the algorithms we will consider.

3.2 Classification

Consider two sets of data points, $\underline{x}_1, \ldots, \underline{x}_N$ which are known to belong to Class A, and $\underline{y}_1, \ldots, \underline{y}_M$ which are known to belong to Class Not-A (or Class B, as you prefer). Given a vector of parameters $\underline{v}$ and an additional scalar v_0, we seek to model the probability a data point belongs in Class A as

$$\mathbb{P}(\underline{x} \in A) = \frac{1}{1 + e^{-(\underline{v}^{\mathrm{T}} \underline{x} + v_0)}}. \tag{13}$$

This is a logistic regression model, essentially modeling the probability of belonging to Class A in terms of how far on either side of the hyperplane $\underline{v}^{\mathrm{T}} \underline{z} + v_0 = 0$ the data point lies.

Given this model, for a specific set of parameters we may express the likelihood of the specific observed data in the following way:

$$\begin{aligned} \mathrm{lik}(\underline{v}, v_0) &= \prod_{i=1}^{N} \mathbb{P}(\underline{x}_i \in A) \prod_{j=1}^{M} \mathbb{P}\left(\underline{y}_j \notin A\right) \\ &= \prod_{i=1}^{N} \frac{1}{1 + e^{-(\underline{v}^{\mathrm{T}} \underline{x}_i + v_0)}} \prod_{j=1}^{M} \left(1 - \frac{1}{1 + e^{-(\underline{v}^{\mathrm{T}} \underline{y}_j + v_0)}}\right). \end{aligned} \tag{14}$$

We then seek the parameters $\underline{v}^*, v_0^*$ that make the observed data as likely as possible - which is to say, $\underline{v}^*, v_0^*$ that maximize the likelihood $\mathrm{lik}(\underline{v}, v_0)$:

$$\mathrm{lik}(\underline{v}^*, v_0^*) = \max_{\underline{v}, v_0} \mathrm{lik}(\underline{v}, v_0). \tag{15}$$

However, given the product structure of the objective function here, it is useful not to consider maximizing the likelihood itself, but rather the ln of the likelihood. This transforms an objective function with a product form to an objective function with a sum form, which is frequently easier to deal with computationally. Additionally, it is convenient to consider minimizing the $-\ln$ instead of maximizing the ln, to keep the problem in the relevant form.

$$\begin{aligned} -\ln \mathrm{lik}(\underline{v}^*, v_0^*) &= \min_{\underline{v}, v_0} -\sum_{i=1}^{N} \ln\left(\frac{1}{1 + e^{-(\underline{v}^{\mathrm{T}} \underline{x}_i + v_0)}}\right) - \sum_{j=1}^{M} \ln\left(1 - \frac{1}{1 + e^{-(\underline{v}^{\mathrm{T}} \underline{y}_j + v_0)}}\right) \\ &= \min_{\underline{v}, v_0} \sum_{i=1}^{N} \ln\left(1 + e^{-(\underline{v}^{\mathrm{T}} \underline{x}_i + v_0)}\right) + \sum_{j=1}^{M} (\underline{v}^{\mathrm{T}} \underline{y}_j + v_0) + \sum_{j=1}^{M} \ln\left(1 + e^{-(\underline{v}^{\mathrm{T}} \underline{y}_j + v_0)}\right) \end{aligned} \tag{16}$$

3.3 A Simple Example

Consider the following problem, to motivate a lot of the work to come. Define the following function

$$f(x, y) = x^2 + y^2 + \beta xy + 2x + y, \tag{17}$$

where β is some constant value (for instance, measured by experiment). We are interested in finding x^* and y^* to minimize f.

Considering y as 'fixed', we can consider the 1-variable problem of minimizing with respect to x. In classical form, this leads to setting the derivative with respect to x equal to 0, or

$$2x+\beta y+2=0. \tag{18}$$

Similarly, considering x as 'fixed', we can consider the 1-variable problem of minimizing with respect to y. In classical form, this leads to setting the derivative with respect to y equal to 0, or

$$2y+\beta x+1=0. \tag{19}$$

As x^* must minimize f when y^* is held constant, and vice versa, the above equations must be satisfied for x^* and y^*. Solving the above equations with respect to x and y therefore yields:

$$\begin{aligned} x^* &= -\frac{\beta-4}{\beta^2-4} \\ y^* &= -\frac{2\beta-2}{\beta^2-4} \\ f(x^*,y^*) &= \frac{5-2\beta}{\beta^2-4}. \end{aligned} \tag{20}$$

However, the problem is not solved completely, as we have not verified that the above point is a minimizer. Note, for instance, that the solutions have a singularity when $\beta=\pm 2$. Whenever something like this occurs, points where solutions seem to explode or cease to be, it is usually indicative of some kind of new behavior that is worth exploring.

Note the following: taking $y=x$, we have $f(x,x)=2x^2+\beta x^2+3x=(2+\beta)x^2+3x$. In this case, if $2+\beta\leqslant 0$, we can force the value of f as low as we want by choosing an appropriate x. Similarly, taking $y=-x$, we have $f(x,-x)=(2-\beta)x^2+x$. In this case, if $2-\beta\leqslant 0$, we can again force the value of f as low as we want by choosing an appropriate f. It therefore seems that if $\beta\leqslant -2$ or $2\leqslant\beta$, no minimizer exists.

We can see this in another way. Consider expressing the function f in the following vector form:

$$f(\underline{x})=\frac{1}{2}\underline{x}^{\mathrm{T}}M\underline{x}+\underline{b}^{\mathrm{T}}\underline{x}, \tag{21}$$

where

$$\begin{aligned} M &= \begin{bmatrix} 2 & \beta \\ \beta & 2 \end{bmatrix}, \\ \underline{b} &= \begin{bmatrix} 2 \\ 1 \end{bmatrix}. \end{aligned} \tag{22}$$

Let $\underline{v}$ be an eigenvector of M with eigenvalue λ. Evaluating f at $\alpha\underline{v}$, we have

$$f(\alpha\underline{v})=\frac{1}{2}\lambda\alpha^2+\alpha\underline{b}^{\mathrm{T}}\underline{v}. \tag{23}$$

From the above, it is clear that if $\lambda\leqslant 0$, the value of f can be as made as low as we want, by choosing the appropriate (potentially negative) value of α. It is clear then that for a minimizer to

exist, we must have all eigenvalues of M be strictly greater than 0, which is to say M must be *positive definite*. Computing the eigenvalues of M as the roots of $\det(\lambda \mathrm{I} - M) = 0$, we have eigenvalues of $2+\beta$ and $2-\beta$. Restricting these to be strictly positive gives us the requirement that $-2 < \beta < 2$. This agrees with the previous observation, that for β outside of this range, it is possible to force f as low as we want.

We can extend this in the following way. Let $\underline{x}^*$ be the vector specified by the point (x^*, y^*). Note that as x^* and y^* were chosen to make the derivatives 0, we have $\nabla f(\underline{x}^*) = 0$. Observing as well that $\nabla^2 f(\underline{x}) = M$ for all $\underline{x}$, the Taylor polynomial for f centered at $\underline{x}^*$ is therefore

$$f(\underline{x}) = f(\underline{x}^*) + \frac{1}{2}(\underline{x} - \underline{x}^*)^{\mathrm{T}} M (\underline{x} - \underline{x}^*). \tag{24}$$

Note, the above representation of f is exact, with no asymptotic term, since f is a quadratic form.

If $-2 < \beta < 2$, we have that M is positive definite, and hence from the above form we have for $\underline{x} \neq \underline{x}^*$ that $f(\underline{x}) > f(\underline{x}^*)$. Hence $\underline{x}^*$ is a global minimum of f. We will see in the next section that in the general case for non-quadratic f, the Hessian being positive definite at a stationary point (i.e., one where all derivatives are 0) implies that the point is at least a local minimum.

3.4 Example Problems

1 Consider the function $f(x,y) = x^2 + y^2 + \beta xy + 2x + y$. For a given value of y, we can minimize with respect to x by setting the derivative with respect to x equal to 0 and solving for a new value of x, which gives $x_{\text{new}} = -1 - \frac{\beta}{2} y$. Similarly, for a given x, we can minimize with respect to y by setting the derivative with respect to y equal to 0 and solving for a new value of y, which gives $y_{\text{new}} = -\frac{1}{2} - \frac{\beta}{2} x$.

This suggests the following algorithm for computing a minimizer. Let (x_0, y_0) be an initial guess, and then compute new guesses based on the formula:

$$\begin{aligned} x_{k+1} &= -1 - \frac{\beta}{2} y_k \\ y_{k+1} &= -\frac{1}{2} - \frac{\beta}{2} x_k. \end{aligned} \tag{25}$$

For $-2 < \beta < 2$, show that (x_k, y_k) converges to (x^*, y^*) and that $f(x_k, y_k)$ decreases to $f(x^*, y^*)$ as $k \to \infty$. What happens for β outside of this range?

4 Existence of Minimizers: Necessary and Sufficient Conditions

In this section, we consider the problem of characterizing minimizers. Given a point $\underline{x} \in \mathbb{R}^n$, how can we know whether or not $\underline{x}$ achieves a minimum value of f? We consider characterization from three perspectives: *necessary conditions* that must be satisfied by any minimizer, *sufficient conditions* that imply a given point is a minimizer, and *convex analysis*, which will allow us to infer such things as whether a given local minimum is in fact a global minimum.

4.1 Necessary Conditions

The following theorem describes properties that must be satisfied by a point $\underline{x}^*$ in order for $f(\underline{x}^*)$ to be a local minimum. These properties do not guarantee that $\underline{x}^*$ is a local minimum, but if these properties are *not* satisfied, then the point is not a minimum.

Theorem 1 (Necessary Conditions) *If $\underline{x}^*$ is a local minimum of f, then*

$$\nabla f(\underline{x}^*) = 0 \tag{26}$$

*and $\nabla^2 f(\underline{x}^*)$ is **positive semi-definite**.*

Proof. Let $\underline{x}^*$ be a local minimum of f. In that case, given any direction $\underline{d}$, we have that for all sufficiently small $\alpha > 0$,

$$f(\underline{x} + \alpha \underline{d}) - f(\underline{x}) \geqslant 0. \tag{27}$$

Note, the restriction to 'sufficiently small' α is due to this being a *local* minimizer. In the case of a global minimizer, the above would hold for *all* $\alpha > 0$.

In this case, we may utilize the Taylor expansion in Eq. (9) to yield

$$0 \leqslant f(\underline{x}^* + \alpha \underline{d}) - f(\underline{x}^*) = \alpha \nabla f(\underline{x}^*)^T \underline{d} + o(\alpha). \tag{28}$$

or, dividing through by α,

$$0 \leqslant \nabla f(\underline{x}^*)^T \underline{d} + \frac{o(\alpha)}{\alpha}. \tag{29}$$

Taking the limit as $\alpha \to 0$, we have from the above that

$$0 \leqslant \nabla f(\underline{x}^*)^T \underline{d}. \tag{30}$$

As the above must hold for *all* direction $\underline{d}$, it must hold for the direction $-\underline{d}$. We have therefore that $0 \leqslant \nabla f(\underline{x}^*)^T \underline{d}$ and $0 \leqslant \nabla f(\underline{x}^*)^T (-\underline{d})$ or $0 \leqslant -\nabla f(\underline{x}^*)^T \underline{d}$, which may be combined to yield

$$0 = \nabla f(\underline{x}^*)^T \underline{d}. \tag{31}$$

The only way the above can hold for all $\underline{d}$ is if the gradient is the zero vector, $\nabla f(\underline{x}^*) = 0$. In particular, taking $\underline{d} = \nabla f(\underline{x}^*)$, we have $\nabla f(\underline{x}^*)^{\mathrm{T}} \nabla f(\underline{x}^*) = ||\nabla f(\underline{x}^*)||^2 = 0$, which implies that the gradient is the zero vector.

Note: It is tempting to argue that the gradient must be zero at a local minimum, because the derivative with respect to any one of the variables must be zero (thinking of it as a 1-dimensional problem). However, this would essentially be restricting to varying the function along any one of the single coordinate axes. The above proof, however, makes it clear that the derivative of the function along *any direction* must be zero.

It remains to show that the Hessian must be positive definite. Having conclude that the gradient must be zero, we have the following expansion of f from Eq. (9):

$$f(\underline{x}^* + \alpha \underline{d}) = f(\underline{x}^*) + \frac{1}{2} \alpha^2 \underline{d}^{\mathrm{T}} \nabla^2 f(\underline{x}^*) \underline{d} + o\left(\alpha^2\right). \tag{32}$$

As $\underline{x}^*$ is a local minimum of f, we have for all sufficiently small $\alpha > 0$

$$0 \leqslant f(\underline{x}^* + \alpha \underline{d}) - f(\underline{x}^*) = \frac{1}{2}\alpha^2 \underline{d}^{\mathrm{T}} \nabla^2 f(\underline{x}^*) \underline{d} + o\left(\alpha^2\right), \tag{33}$$

or, dividing through by α^2,

$$0 \leqslant \frac{1}{2} \underline{d}^{\mathrm{T}} \nabla^2 f(\underline{x}^*) \underline{d} + \frac{o\left(\alpha^2\right)}{\alpha^2}. \tag{34}$$

Taking the limit as $\alpha \to 0$, and dropping the positive $1/2$ factor, we have that $0 \leqslant \underline{d}^{\mathrm{T}} \nabla^2 f(\underline{x}^*) \underline{d}$. As this holds for all directions $\underline{d}$, it follows that the Hessian matrix $\nabla^2 f(\underline{x}^*)$ is positive semi-definite, i.e., has all non-negative eigenvalues. □

The best way to interpret these results is the following: at a local minimum $\underline{x}^*$, the directional derivative must be 0 in all directions, otherwise there is a direction in which the function increases; similarly, the *second derivative* in any direction must be non-negative, otherwise over a small step in that direction, the derivative of the function would be decreasing from zero, become negative, and therefore decrease the value of the function.

4.2 Sufficient Conditions

The following theorem describes a set of *sufficient conditions* for $\underline{x}^*$ to be a local minimum of f. If they are satisfied, then $\underline{x}^*$ is a local minimum, though there may be some local minima for which they are not satisfied.

Theorem 2 (Sufficient Conditions) *At a point $\underline{x}^*$, if $\nabla f(\underline{x}^*) = 0$ and $\nabla^2 f(\underline{x}^*)$ is **positive definite**, then $\underline{x}^*$ is a strict local minimum of f.*

Proof. Taking the condition that $\nabla f(\underline{x}^*) = 0$, we have that for any $\underline{x}$, taking the Taylor expansion form Eq. (7),

$$f(\underline{x}) = f(\underline{x}^*) + \frac{1}{2}(\underline{x} - \underline{x}^*)^{\mathrm{T}} \nabla^2 f(\underline{x}^*)(\underline{x} - \underline{x}^*) + o\left(||\underline{x} - \underline{x}^*||^2\right). \tag{35}$$

Note, because $\nabla^2 f(\underline{x}^*)$ is real and symmetric, it has n-real eigenvalues. Let $\lambda_{\min}$ be the smallest eigenvalue of $\nabla^2 f(\underline{x}^*)$. By assumption, since $\nabla^2 f(\underline{x}^*)$ is positive definitely, $\lambda_{\min} > 0$. From Eq. (169), we have

$$f(\underline{x}) - f(\underline{x}^*) \geqslant \frac{1}{2}\lambda_{\min} ||\underline{x} - \underline{x}^*||^2 + o\left(||\underline{x} - \underline{x}^*||^2\right). \tag{36}$$

Dividing through by $||\underline{x} - \underline{x}^*||^2$, we have

$$\frac{f(\underline{x}) - f(\underline{x}^*)}{||\underline{x} - \underline{x}^*||^2} \geqslant \frac{1}{2}\lambda_{\min} + \frac{o\left(||\underline{x} - \underline{x}^*||^2\right)}{||\underline{x} - \underline{x}^*||^2}. \tag{37}$$

Note that $o\left(||\underline{x} - \underline{x}^*||^2\right) / ||\underline{x} - \underline{x}^*||^2$ goes to 0 as $||\underline{x} - \underline{x}^*|| \to 0$. Hence, for all $\underline{x}$ that are sufficiently close to $\underline{x}^*$, i.e., for which $||\underline{x} - \underline{x}^*||$ is sufficiently small, we have that $o\left(||\underline{x} - \underline{x}^*||^2\right) / ||\underline{x} - \underline{x}^*||^2 \geqslant$

$-\lambda_{\min}/4$. Hence, for all $\underline{x}$ sufficiently close to $\underline{x}^*$:

$$\frac{f(\underline{x})-f(\underline{x}^*)}{||\underline{x}-\underline{x}^*||^2} \geqslant \frac{1}{2}\lambda_{\min}-\frac{1}{4}\lambda_{\min}=\frac{1}{4}\lambda_{\min}>0. \tag{38}$$

From this, we have that when $||\underline{x}-\underline{x}^*||$ is sufficiently small, $f(\underline{x})-f(\underline{x}^*)>0$, which proves that $\underline{x}^*$ is a strict local minimum. □

As a related result to the above theorem, following almost immediately from the above proof, we have that in the case that $\nabla^2 f(\underline{x}^*)$ is positive definite, for $\underline{x}$ near $\underline{x}^*$, f behaves like a quadratic function centered on $\underline{x}^*$.

Theorem 3 (A Lower Bound Near $\underline{x}^*$) *If $\nabla f(\underline{x}^*)=0$, and $\nabla^2 f(\underline{x}^*)$ is positive definite, then there is an $\varepsilon>0$ and $\gamma>0$ such that for all $\underline{x}$ such that $||\underline{x}-\underline{x}^*||<\varepsilon$, we have*

$$f(\underline{x} \geqslant f(\underline{x}^*)+\frac{1}{2}\gamma||\underline{x}-\underline{x}^*||^2 \tag{39}$$

Proof. This can be proven by example from the proof of the previous theorem, from which we have that for all $\underline{x}$ sufficiently close to $\underline{x}^*$, i.e., $||\underline{x}-\underline{x}^*||$ sufficiently small,

$$f(\underline{x}) \geqslant f(\underline{x}^*)+\frac{1}{4}\lambda_{\min}||\underline{x}-\underline{x}^*||^2. \tag{40}$$

4.3 Convex Analysis

The results of the previous two subsections, Theorem 1 and Theorem 2, characterize *local* minima. Having verified that a given $\underline{x}^*$ is a local minimum, it can be difficult to determine whether or not it is a *global* minimum, without additional knowledge about the function f. One possibility would be to solve for all possible local minima, for instance solving for all possible solutions to $\nabla f(\underline{x}^*)=0$, and determining which gave the smallest possible value of f. However, it can be difficult to know in advance how many possible solutions to $\nabla f(\underline{x}^*)=0$ there may be, if there are even finitely many (see the example problems after this section).

Assuming that f has certain structure, however, we can ensure that any local minimum is in fact a global minimum, or even that a given minimum is unique.

Definition 2 *A function f is convex over $\mathbb{R}^n$ if it satisfies the following inequality, for any $\underline{x},\underline{y}\in\mathbb{R}^n$: for all $0\leqslant\lambda\leqslant 1$*

$$f(\lambda\underline{x}+(1-\lambda)\underline{y}) \leqslant \lambda f(\underline{x})+(1-\lambda)f(\underline{y}). \tag{41}$$

The function f is strictly convex if the above inequality is strict for all $0<\lambda<1$.

A function f is concave if $-f$ is convex.

An important geometric notion here is that the set of points $\underline{z} = \lambda \underline{x} + (1-\lambda)\underline{y}$ for $0 \leqslant \lambda \leqslant 1$ describes the line segment connecting the points $\underline{x}$ and $\underline{y}$ in space, or the set of weighted averages of the two points. A function is convex then if along the line connecting any two points in space, the value of the function is at most the weighted average of the function at the end points.

An alternative way of considering convexity of a function: f of the weighted average of two points is at most the weighted average of f of each point.

Another way of considering convexity of a function: along the line between $\underline{x}$ and $\underline{y}$, the function f falls below the line connecting the points $(\underline{x}, f(\underline{x}))$ and $(\underline{y}, f(\underline{y}))$.

The following is another important property of convex functions, essentially stating that at any point, the plane tangent to the function (i.e., the linear approximation of f) lies entirely below the function itself.

Proposition 1 *If f is convex over $\mathbb{R}^n$, then for any $\underline{x}, \underline{y} \in \mathbb{R}^n$:*

$$f(\underline{y}) \geqslant f(\underline{x}) + \nabla f(\underline{x})^T (\underline{y} - \underline{x}). \tag{42}$$

Additionally, if f is strictly convex over $\mathbb{R}^n$, then the above holds as a strict inequality for all $\underline{y} \neq \underline{x}$.

Proof. Let $\underline{z} = \lambda \underline{x} + (1-\lambda)\underline{y} = \underline{x} + (1-\lambda)(\underline{y} - \underline{x})$ for $0 < \lambda < 1$. Then we have, thinking about the directional derivative from $\underline{x}$ in the direction of $\underline{y} - \underline{x}$,

$$\frac{f(\underline{z}) - f(\underline{x})}{1-\lambda} \leqslant \frac{\lambda f(\underline{x}) + (1-\lambda) f(\underline{y}) - f(\underline{x})}{1-\lambda} = f(\underline{y}) - f(\underline{x}). \tag{43}$$

Taking the limit of the left hand side as $\lambda \to 1$ (i.e., $\underline{z} \to \underline{x}$), the limit of the above ratio yields the directional derivative at $\underline{x}$ in the direction of $\underline{y} - \underline{x}$:

$$\nabla f(\underline{x})^{\mathrm{T}} (\underline{y} - \underline{x}) \leqslant f(\underline{y}) - f(\underline{x}). \tag{44}$$

Rearranging the above yields the result. □

The assumption that f is convex is a powerful tool for understanding the nature of the minima of f. In particular, we have the following results:

Theorem 4 (Uniqueness of Convex Minima) *If f is convex over $\mathbb{R}^n$, then the minimal function value is unique. That is, if $\underline{x}^*$ and $\underline{y}^*$ are local minima of f, we have $f(\underline{x}^*) = f(\underline{y}^*)$.*

Proof. The proof proceeds by contradiction. Assume, without loss of generality, that $f(\underline{x}^*) < f(\underline{y}^*)$. We have, by convexity of f, that for any $0 < \lambda < 1$:

$$f(\lambda \underline{x}^* + (1-\lambda)\underline{y}^*) \leqslant \lambda f(\underline{x}^*) + (1-\lambda) f(\underline{y}^*) < \lambda f(\underline{x}^*) + (1-\lambda) f(\underline{x}^*) = f(\underline{y}^*), \tag{45}$$

in particular it is worth emphasizing,

$$f(\lambda \underline{x}^* + (1-\lambda)\underline{y}^*) < f(\underline{y}^*). \tag{46}$$

Choosing λ sufficiently close to 0, we may construct a point $\underline{z} = \lambda \underline{x}^* + (1-\lambda)\underline{y}^*$ that is arbitrarily close to $\underline{y}^*$, but has $f(\underline{z}) < f(\underline{y}^*)$. This violates the definition of $\underline{y}^*$ as a local minimum of f! Therefore, we must have $f(\underline{x}^*) = f(\underline{y}^*)$. □

Additionally, if the function f is *strictly* convex, the above result may be strengthened to show that not only is the minimal function value unique, but that there is a single unique minimizer $\underline{x}^*$.

Theorem 5 (Uniqueness of the Minimizer Under Strict Convexity) *If f is strictly convex over $\mathbb{R}^n$, and $\underline{x}^*$ is a local minimum of f, then $\underline{x}^*$ is a unique global minimum of f. That is, for all $\underline{x} \neq \underline{x}^*$,*

$$f(\underline{x}) > f(\underline{x}^*). \tag{47}$$

Proof. We proceed by contradiction. Let $\underline{x}^*$ and $\underline{y}^*$ be local minima of f, with $\underline{x}^* \neq \underline{y}^*$. By Theorem 4, we have that $f(\underline{x}^*) = f(\underline{y}^*)$. However, for $0 < \lambda < 1$, we have by the strict convexity of f that

$$f(\lambda \underline{x}^* + (1-\lambda)\underline{y}^*) < \lambda f(\underline{x}^*) + (1-\lambda) f(\underline{y}^*) = f(\underline{x}^*). \tag{48}$$

For any such λ then, taking $\underline{z} = \lambda \underline{x}^* + (1-\lambda)\underline{y}^*$, we have $f(\underline{z}) < f(\underline{x}^*)$. This violates the uniqueness of the minimal value of f for convex functions. Hence, no such $\underline{y}^* \neq \underline{x}^*$ can exist. □

Given the above results, it is clear that convexity is a powerful tool for characterizing the minima of functions. Convexity is frequently a very natural property of objective functions of interest. For instance, norms are naturally convex: for $0 \leqslant \lambda \leqslant 1$, by the triangle inequality we have

$$||\lambda \underline{x} + (1-\lambda)\underline{y}|| \leqslant ||\lambda \underline{x}|| + ||(1-\lambda)\underline{y}|| \leqslant \lambda ||\underline{x}|| + (1-\lambda)||\underline{y}||. \tag{49}$$

Additionally, sums of convex functions, $f_1 + f_2$ where f_1, f_2 are convex, are also convex (see example problems following this section).

The following propositions give additional properties of interest for convex functions over $\mathbb{R}^n$:

Proposition 2 *If f is convex over $\mathbb{R}^n$, then $\nabla^2 f(\underline{x})$ is positive semi-definite for all $\underline{x} \in \mathbb{R}^n$. Additionally, if f is strictly convex over $\mathbb{R}^n$, then $\nabla^2 f(\underline{x})$ is positive definite for all $\underline{x} \in \mathbb{R}^n$.*

Proof. For $\underline{x}, \underline{y} \in \mathbb{R}^n$ and $0 < \lambda < 1$, let $\underline{z} = (1-\lambda)\underline{x} + \lambda \underline{y} = \underline{x} + \lambda(\underline{y} - \underline{x})$. We then have

$$f(\underline{z}) = f(\underline{x}) + \lambda \nabla f(\underline{x})^{\mathrm{T}}(\underline{y} - \underline{x}) + \frac{1}{2}\lambda^2 (\underline{y} - \underline{x})^{\mathrm{T}} \nabla^2 f(\underline{x})(\underline{y} - \underline{x}) + o\left(\lambda^2\right). \tag{50}$$

Hence,

$$\frac{f(\underline{z}) - f(\underline{x})}{\lambda} = \nabla f(\underline{x})^{\mathrm{T}}(\underline{y} - \underline{x}) + \frac{1}{2}\lambda (\underline{y} - \underline{x})^{\mathrm{T}} \nabla^2 f(\underline{x})(\underline{y} - \underline{x}) + o(\lambda). \tag{51}$$

Additionally, by Proposition 1,

$$f(\underline{z}) - f(\underline{x}) \geqslant \nabla f(\underline{x})^{\mathrm{T}}(\underline{z} - \underline{x}) = \lambda \nabla f(\underline{x})^{\mathrm{T}}(\underline{y} - \underline{x}). \tag{52}$$

Combining these two results,

$$\nabla f(\underline{x})^{\mathrm{T}}(\underline{y} - \underline{x}) \leqslant \frac{f(\underline{z}) - f(\underline{x})}{\lambda} = \nabla f(\underline{x})^{\mathrm{T}}(\underline{y} - \underline{x}) + \frac{1}{2}\lambda (\underline{y} - \underline{x})^{\mathrm{T}} \nabla^2 f(\underline{x})(\underline{y} - \underline{x}) + o(\lambda), \tag{53}$$

or

$$0 \leqslant \frac{1}{2}\lambda(\underline{y}-\underline{x})^{\mathrm{T}}\nabla^2 f(\underline{x})(\underline{y}-\underline{x}) + o(\lambda). \tag{54}$$

Dividing through by λ (which is positive) and taking the limit as $\lambda \to 0$, we have

$$0 \leqslant \frac{1}{2}(\underline{y}-\underline{x})^{\mathrm{T}}\nabla^2 f(\underline{x})(\underline{y}-\underline{x}). \tag{55}$$

The above holds for any $\underline{y} \in \mathbb{R}^n$. Taking $\underline{y} = \underline{x} + \underline{d}$ for some $\underline{d} \in \mathbb{R}^n$, we have from the above that

$$0 \leqslant \underline{d}^{\mathrm{T}}\nabla^2 f(\underline{x})\underline{d}. \tag{56}$$

As the above must hold for all such $\underline{d}$, we have that the Hessian matrix $\nabla^2 f(\underline{x})$ is positive semi-definite, and that this must hold for all $\underline{x} \in \mathbb{R}^n$.

Note, in the case of f being *strictly* convex, the above can be tightened to $\nabla^2 f(\underline{x})$ is positive definite for all $\underline{x} \in \mathbb{R}^n$. □

The above result on the Hessian of convex functions leads to the following property:

Theorem 6 *If f is strictly convex, then $\underline{x}^*$ is a global minimum of f if and only if $\nabla f(\underline{x}^*) = 0$.*

Proof. Since f is strictly convex, we have that any minimum is the unique global minimum. If $\underline{x}^*$ is a minimizer of f, we have from Theorem 1 that $\nabla f(\underline{x}^*) = 0$. In the other direction, since f is strictly convex, by Proposition 2, we have that $\nabla^2 f(\underline{x}^*)$ is positive definite, since the Hessian matrix is positive definite everywhere. By Theorem 2, if we additionally have that $\nabla f(\underline{x}^*) = 0$. This completes the proof. □

The result of this is that when confronted with the prospect of minimizing a strictly convex objective function, it suffices to consider $\underline{x}^*$ where the gradient is zero.

4.4 Example Problems

1 Characterize the minima of the function $f(x,y) = (x^2+y^2-r^2)^2$ taking r to be a constant. In general, what can be said about the minima (and maxima) of $f_p(x,y) = |x^2+y^2-r^2|^p$?

2 Prove that the following functions are convex:

- Prove that $f(\underline{x}) = \underline{x}^{\mathrm{T}}\underline{x}$ is convex.
- If $f : \mathbb{R}^n \mapsto \mathbb{R}$ is convex and $g : \mathbb{R} \mapsto \mathbb{R}$ is convex, prove that

$$h(\underline{x}) = g(f(\underline{x})) \tag{57}$$

  is convex.
- If $f_1 : \mathbb{R}^n \mapsto \mathbb{R}$ and $f_2 : \mathbb{R}^n \mapsto \mathbb{R}$ are convex, prove that $f(\underline{x}) = f_1(\underline{x}) + f_2(\underline{x})$ is convex.
- Prove or disprove: if f_1 is convex and f_2 is convex, then $f_1 - f_2$ is convex.

5 Descent Algorithms

The results of the previous sections suggest that in general, if we wish to locate the minimizers of a function f, it suffices to solve the system of equations given by

$$\nabla f(\underline{x}) = 0. \tag{58}$$

If all solutions $\underline{x}^*$ can be found, the various $f(\underline{x}^*)$ can be compared to identify which gives the smallest value of f, and is therefore the global minimizer. However, the system of equations resulting from $\nabla f(\underline{x}) = 0$ can frequently be highly non-linear, admitting many (and occasionally infinitely many) solutions. These systems can be difficult, if not impossible, to solve directly, and without deeper analysis it can be difficult to know that all solutions have been found.

These concerns lead to a class of algorithms known as iterative descent algorithms. These algorithms compute a sequence of successive points $\underline{x}_0, \underline{x}_1, \underline{x}_2, \ldots \in \mathbb{R}^n$ such that $f(\underline{x}_k) \geqslant f(\underline{x}_{k+1})$ and $\underline{x}_k$ converges (in some sense) to a local minimum $\underline{x}^*$. In general, without stronger assumptions on f (such as convexity), it can be difficult to know in advance whether the algorithm will converge on a local or global minimum.

Essentially, descent algorithms proceed by starting at some point $\underline{x}$, and computing a new point $\underline{x}' = \underline{x} + \alpha \underline{d}$, that is moving from point $\underline{x}$ in direction $\underline{d}$ by a stepsize α. The direction and step size are taken to be such that the value of the function at the new point is decreased from the value of the function at the old point, $f(\underline{x}') \leqslant f(\underline{x})$. This reduces a multi-dimensional optimization problem, which can be difficult, to a sequence of one dimensional optimization problems, which can often be treated via elementary means.

Descent algorithms follow the same basic structure:

- Let $\underline{x}_0$ be an initial guess, set $k := 0$.
- While $\underline{x}_k$ does not satisfy some *Termination Condition*:
  - Compute a descent direction $\underline{d}_k$.
  - Compute a stepsize α_k.
  - Compute a new iterate
  $$\underline{x}_{k+1} := \underline{x}_k + \alpha_k \underline{d}_k. \tag{59}$$
  - Update time $k := k + 1$.

There are three main questions that need to be answered to build a descent algorithm:

- How should you compute a descent direction $\underline{d}_k$?
- How should you compute a good stepsize α_k?
- When should the algorithm terminate?

The remainder of this section is concerned with providing answers to these three questions. Additionally, an important question with regards to descent algorithms is:

- Do the iterates $\{\underline{x}_k\}$ converge to a local minimum $\underline{x}^*$?
- How fast is the convergence?

These questions are considered in more detail in Section 7.

5.1 Descent Directions

There are a couple of standard rules frequently utilized for determining descent directions. Any of these rules may be utilized at any step in the algorithm, and it may frequently be useful to combine these rules in 'phases'. The most common algorithms use either **steepest descent**, **Newton's method**, or **coordinate descent**.

Arbitrary Descent Directions: The most immediate answer to 'what direction to use' is simply, 'any direction in which the function decreases'. Here we may utilize the notion of the directional derivative. The rate of change of the function in direction $\underline{d}_k$ is

$$\nabla_{\underline{d}_k} f(\underline{x}_k) = \nabla f(\underline{x}_k)^{\mathrm{T}} \underline{d}_k. \tag{60}$$

Hence, any direction $\underline{d}_k$ that satisfies

$$\nabla f(\underline{x}_k)^{\mathrm{T}} \underline{d}_k < 0 \tag{61}$$

will do. The remaining rules below give more specific answers, providing rules for computing directions that satisfy the above.

Gradient-Derived Directions: The following rule produces descent directions by systematically modifying the gradient vector to produce a new direction vector.

Let D_k be a positive definite matrix, and define the direction

$$\underline{d}_k = -D_k \nabla f(\underline{x}_k). \tag{62}$$

This will result in a descent direction, as $\nabla f(\underline{x}_k)^{\mathrm{T}} \underline{d}_k = -\nabla f(\underline{x}_k)^{\mathrm{T}} D_k \nabla f(\underline{x}_k) < 0$ by the positive definiteness of D_k. Note, if $-\nabla f(\underline{x}_k)^{\mathrm{T}} D_k \nabla f(\underline{x}_k) = 0$, we must have that $\nabla f(\underline{x}_k) = 0$, in which case a minimizer has been found.

Steepest Descent: This rule is simply that the direction is always the direction from $\underline{x}_k$ in which the function has the steepest rate of decrease, i.e.,

$$\underline{d}_k = -\nabla f(\underline{x}_k). \tag{63}$$

To see that this is the direction of steepest descent, note that the directional derivative is given by $\nabla_{\underline{d}} f(\underline{x}_k) = \nabla f(\underline{x}_k)^{\mathrm{T}} \underline{d}$, and that thinking of this as a dot product, it will be minimized by vectors $\underline{d}$ in the opposite direction to $\nabla f(\underline{x}_k)$. (Note here, the scale of the direction does not matter - such concerns will be handled considering the step size α_k separately.)

Note additionally, this can be viewed as a Gradient-Derived direction as above, taking $D_k = \mathrm{I}$, the identity matrix.

Scaled Descent: It may occur that in some objective functions, some variables are weighted differently than others in a way that vastly affects their relative importance. Consider for instance the

objective function $f(x,y) = 1000x^2 + y^2$, in which a unit change of the variable x has a much more dramatic effect than a unit change of the variable y. This can negatively affect the convergence rates of descent algorithms.

To combat this, we may consider scaling the variables in the following way, taking a gradient-derived descent step where

$$D_k = \begin{bmatrix} d_{k,1} & 0 & 0 & \dots & 0 \\ 0 & d_{k,2} & 0 & \dots & 0 \\ 0 & 0 & d_{k,3} & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & d_{k,n} \end{bmatrix} \tag{64}$$

where $d_{k,i}$ is a positive constant (which guarantees positive definiteness). A standard choice, where applicable, is $d_{k,i} \approx (\partial^2 f / \partial x_i^2(\underline{x}_k))^{-1}$. The reasoning for basing the scaling purely on the second order variation is that near a minimum, we expect the function to behave essentially like a quadratic function with respect to the variables. Such a scaling therefore attempts the normalize the resulting quadratic form.

Newton's Method Descent: The previous methods utilized local information to compute a descent direction only in terms of the first derivatives, the gradient. If we are willing to utilize additional computational effort, it is frequently useful to consider additional information from the second derivatives in the form of the Hessian matrix. This leads to the following direction,

$$\underline{d}_k = -\left[\nabla^2 f(\underline{x}_k)\right]^{-1} \nabla f(\underline{x}_k), \tag{65}$$

when the Hessian matrix $\nabla^2 f(\underline{x}_k)$ is positive definite. When the Hessian is positive definite, its inverse is also positive definite, and the above can be recognized as a gradient-derived direction with $D_k = \left[\nabla^2 f(\underline{x}_k)\right]^{-1}$.

As motivation for this, consider the idea of approximating f by a quadratic function, centered on $\underline{x}_k$. This yields, for $\underline{x}$ near $\underline{x}_k$,

$$f(\underline{x}) \approx f_k(\underline{x}) = f(\underline{x}_k) + \nabla f(\underline{x}_k)^{\mathrm{T}}(\underline{x} - \underline{x}_k) + \frac{1}{2}(\underline{x} - \underline{x}_k)^{\mathrm{T}} \nabla^2 f(\underline{x}_k)(\underline{x} - \underline{x}_k). \tag{66}$$

Attempting to minimize $f_k(\underline{x})$ then, we have that $\nabla f_k(\underline{x}) = \nabla^2 f(\underline{x}_k)(\underline{x} - \underline{x}_k) + \nabla f(\underline{x}_k) = 0$, which is solved by

$$\underline{x} = \underline{x}_k - \left[\nabla^2 f(\underline{x}_k)\right]^{-1} \nabla f(\underline{x}_k). \tag{67}$$

Because the formula above was derived by approximating f by a quadratic form, there is no reason to think that the $\underline{x}$ above should minimize f. However, if $\underline{x}_k$ is close to a minimum, $\underline{x}$ as above may be closer. This motivates thinking about descent in the direction of $-\left[\nabla^2 f(\underline{x}_k)\right]^{-1} \nabla f(\underline{x}_k)$.

Note, when $\nabla^2 f(\underline{x}_k)$ is positive definite, its inverse exists, and is also positive definite. However it may not always be the case that $\nabla^2 f(\underline{x}_k)$ is positive definite. In that case, the Newton's method defined direction may not be a descent direction at all. It is frequently useful to consider directions of the form

$$\underline{d}_k = -\left[\nabla^2 f(\underline{x}_k) + \varepsilon_k \mathrm{I}\right]^{-1} \nabla f(\underline{x}_k), \tag{68}$$

where ε_k is chosen to be a positive constant sufficiently large to make $\nabla^2 f(\underline{x}_k) + \varepsilon_k \mathrm{I}$ positive definite.

It is worth noting that computing both the Hessian and the inverse of the Hessian can incur considerable computational expense. However, the tradeoff is that Newton's method type steps tend to converge to minimizers at a much faster rate than other methods.

Coordinate Descent: The idea behind coordinate descent is to pick one of the variables, and minimize from the current point with respect to that variable. Let the vector $\underline{e}_i$ be the vector of all 0s and a 1 in the i-th coordinate.

At time k, if $\partial f/\partial x_i(\underline{x}_k) \neq 0$ for some $i = 1, 2, \ldots, n$, then the value of the function f can be decreased by increasing or decreasing the value of the i-th coordinate of $\underline{x}_k$. If $\partial f/\partial x_i(\underline{x}_k) = 0$ for each i, then $\nabla f(\underline{x}_k) = 0$ and we are at a minimizer (or, potentially, a maximizer - it is worth checking). Hence, at time k, we take

$$\underline{d}_k = -\left(\frac{\partial f}{\partial x_i}(\underline{x}_k)\right)\underline{e}_i \text{ for } i \text{ such that } \frac{\partial f}{\partial x_i}(\underline{x}_k) \neq 0. \tag{69}$$

Note, the $-$ sign in front is to ensure that the direction is a *descent* direction. If the derivative with respect to a certain variable is positive, then the function can be decreased by *reducing* the value of that variable. If the derivative with respect to that variable is negative, the function can be decreased by *increasing* the value of that variable.

As a standard rule of thumb, at time k the coordinate i is chosen to be the i with the largest value of $|\partial f/\partial x_i(\underline{x}_k)|$, i.e., the derivative with greatest magnitude. This maximizes the amount f can be reduced relative to any slight change in a single variable.

A standard coordinate descent algorithm will loop through the variables, minimizing with respect to one variable at a time, until a minimum is reached.

5.2 Propitious Step Sizes

Having determined a descent direction $\underline{d}_k$, it remains to determine a stepsize α_k such that $f(\underline{x}_k + \alpha_k \underline{d}_k) < f(\underline{x}_k)$ and descent is achieved. All the rules of the previous subsection yield directions in which descent is possible for sufficiently small stepsizes, but we would like systematic rules for determining effective stepsizes, and with any luck, stepsizes that yield as great a descent as possible.

Constant Step Size: The simplest possible rule is to take a constant stepsize, $\alpha_k = \alpha > 0$. However, if α is too large, this can result in overshoot - going too far in direction $\underline{d}_k$ may result in the function actually increasing, and the algorithm never settles down into a local minimum. On the other side of this, if α is too small, this can result in very slow convergence of the algorithm. It can be difficult to know in advance what an appropriate stepsize is. One possibility might be to proceed in phases of a certain stepsize, decreasing the size of the step whenever descent is violated.

Armijo's Rule: Armijo's Rule can be derived from the following observation, that

$$\lim_{\alpha \to 0} \frac{f(\underline{x}_k + \alpha \underline{d}_k) - f(\underline{x}_k)}{\alpha} = \nabla f(\underline{x}_k)^{\mathrm{T}} \underline{d}_k, \tag{70}$$

and hence

$$\lim_{\alpha \to 0} \frac{f(\underline{x}_k + \alpha \underline{d}_k) - f(\underline{x}_k)}{\alpha \nabla f(\underline{x}_k)^{\mathrm{T}} \underline{d}_k} = 1. \tag{71}$$

Hence, for sufficiently small α, the above ratio can be made arbitrarily close to 1.

Armijo's rule proceeds in the following way: starting with an initial stepsize guess $s_{k,0} = s$, a reduction factor $0 < \beta < 1$, and a stopping threshold $0 < \sigma < 1$, iteratively reduce the initial stepsize by a factor of β ($s_{k,t} = \beta s_{k,t-1}$) until

$$\sigma \leqslant \frac{f(\underline{x}_k + s_{k,t}\underline{d}_k) - f(\underline{x}_k)}{s_{k,t}\nabla f(\underline{x}_k)^{\mathrm{T}}\underline{d}_k}, \tag{72}$$

or equivalently

$$\sigma \leqslant \frac{f(\underline{x}_k + s\beta^t\underline{d}_k) - f(\underline{x}_k)}{s\beta^t\nabla f(\underline{x}_k)^{\mathrm{T}}\underline{d}_k}, \tag{73}$$

At this point, stop the iteration, and take the stepsize $\alpha_k = s_{k,t} = s\beta^t$.

The final result of this computational subroutine is guaranteed to be a descent step, as $\nabla f(\underline{x}_k)^{\mathrm{T}}\underline{d}_k) < 0$ (taking $\underline{d}_k$ as a descent direction), and hence

$$f(\underline{x}_k + \alpha_k\underline{d}_k) - f(\underline{x}_k) \leqslant \alpha_k\nabla f(\underline{x}_k)^{\mathrm{T}}\underline{d}_k\sigma < 0. \tag{74}$$

This computation must be re-done every time a new stepsize is required, as the number of necessary reductions by β may change with the current $\underline{x}_k, \underline{d}_k$. In general, the closer β is to 0, the fewer iterations will be needed to compute the stepsize, but the cost of this is that the final stepsize will generally be extremely small. Additionally, the closer σ is to 1, the more iterations will likely be necessary to reach a sufficiently small stepsize to meet the termination condition. Good initial stepsize guesses will likely be problem dependent, but could be generated by initial approximations of the function f by simple functions, such as quadratic forms. In general, a set of hyperparameters (s, β, σ) that perform well for a given instance of the problem may not perform well for others.

Minimization Rule: The minimization rule is the most straightforward of the available stepsize rules, simply take α_k to be the stepsize that minimizes the function f in the specified direction $\underline{d}_k$ from $\underline{x}_k$, i.e.,

$$\alpha_k = \arg\min_{\alpha>0} f(\underline{x}_k + \alpha\underline{d}_k). \tag{75}$$

It is reasonable to protest this on the grounds that we have essentially traded one optimization problem (minimizing f with respect to $\underline{x}$) for another (minimizing $f(\underline{x}_k + \alpha\underline{d}_k)$ with respect to α). However, importantly, the above problem is a one dimensional optimization problem. These can frequently be efficiently and effectively solved computationally.

Define the function $g(\alpha) = f(\underline{x}_k + \alpha\underline{d}_k)$. We are interested in computing minimizing α for the function g. In some cases, for instance if g is a quadratic function with respect to α, we may compute the minimizing α explicitly. However in general we must be content with approximation.

The general scheme will be as follows: define **a three point pattern** as a set of value $a < c < b$ where $g(a) > g(c)$ and $g(c) < g(b)$. If we have identified a three point pattern, there must exist a local minimum of g in the interval $[a, b]$ (though the local minimum may or may not be c). By approximating g over this interval with simple polynomials, we may compute another three point pattern over a smaller sub-interval, which must contain a local minimum of g. In this way, the containing interval is steadily shrunk, and a minimizing α computed to arbitrary accuracy. The following methods all follow involve successive approximations of g by quadratic functions. Other

methods exist, for instance approximating g by cubic polynomials, but the improved convergence rates of these methods comes at the cost of increased computational effort. These additional methods are discussed in Appendix B.

For each of the following rules, it is assumed a three point pattern $a < c < b$ has been found (these can frequently be found by direct search). Each method below approximates g by a different quadratic function, and uses this approximation to construct a new narrower three point pattern. These rules can be iterated until sufficient accuracy (sufficiently narrow bounds on the containing interval) has been achieved. Note, if $g'(c) = 0$, the search can stop, as c must be a minimizer of g.

- **Fit to:** $g(a), g'(a), g(b), g'(b)$
  If $g'(b) < g'(a)$, we may fit the following polynomial to $g(a)$, $g'(a)$, $g'(b)$:

$$\tilde{g}(\alpha) = g(a) + g'(a)(\alpha - a) + \frac{1}{2}\frac{g'(b) - g'(a)}{b - a}(\alpha - a)^2. \tag{76}$$

  The above approximation for g is minimized at

$$\alpha' = a - \frac{g'(a)}{\frac{g'(b)-g'(a)}{b-a}} = \frac{ag'(b) - bg'(a)}{g'(b) - g'(a)}. \tag{77}$$

  If $\alpha' < c$ and $g(\alpha') < g(c)$, the three point pattern $a < c < b$ may be replaced with the narrower $a < \alpha' < c$. If $\alpha' < c$ and $g(\alpha') > g(c)$, the three point pattern may be replaced with the narrower $\alpha' < c < b$. If $\alpha' > c$, similar rules of replacement can be worked out.

  Note, if $g'(b) > g'(a)$, a similar polynomial can be fit to $g(b), g'(b), g'(a)$ instead, and a similar analysis performed. In either case, it can be shown that the width of the interval converges to 0 with order ≈ 1.618.

- **Fit to:** $g(a), g(b), g(c)$
  In this case, we attempt to fit a polynomial to $g(a), g(b)$, and $g(c)$. This yields

$$\tilde{g}(\alpha) = \frac{g(c)(\alpha - a)(\alpha - b)}{(c - a)(c - b)} + \frac{g(a)(\alpha - b)(\alpha - c)}{(a - b)(a - c)} + \frac{g(b)(\alpha - c)(\alpha - a)}{(b - c)(b - a)}. \tag{78}$$

  The above approximation for g is minimized at

$$\alpha' = \frac{1}{2}\frac{g(a)(c^2 - b^2) + g(b)(a^2 - c^2) + g(c)(b^2 - a^2)}{g(a)(c - b) + g(b)(a - c) + g(c)(b - a)}. \tag{79}$$

  The above point can be utilized to reduce the three point pattern, as in the previous section: If $\alpha' < c$ and $g(\alpha') < g(c)$, the three point pattern $a < c < b$ may be replaced with the narrower $a < \alpha' < c$. If $\alpha' < c$ and $g(\alpha') > g(c)$, the three point pattern may be replaced with the narrower $\alpha' < c < b$. If $\alpha' > c$, similar rules of replacement can be worked out.

  It can be shown that the width of the interval converges to 0 with order ≈ 1.3.

- **Newton's Method in 1D - Fit to:** $g(a), g'(a), g''(a)$
  In this case, we consider the 1-dimensional version of Newton's method, which is to fit a function to $g(a)$, and the first and second derivatives at a. This yields

$$\tilde{g}(\alpha) = g(a) + g'(a)(\alpha - a) + \frac{1}{2}g''(a)(\alpha - a)^2. \tag{80}$$

The above approximation of g is minimized at

$$\alpha' = a - \frac{g'(a)}{g''(a)}. \tag{81}$$

As in the previous cases, the point α' may be used to reduce the three point pattern. It can be shown that the width of the interval converges to 0 with order at least 2. This is a very fast algorithm - the cost of using it however is the additional computational burden of determining g'' at various points. The second derivative can often be approximated, however, with finite difference formulas.

Limited Minimization Rule: The limited minimization rule is a simple restriction of the minimization rule - instead of absolute minimization with respect to α, we consider α at most some maximal stepsize value s_0, i.e,

$$\alpha_k = \arg\min_{s_0 \geqslant \alpha > 0} f(\underline{x}_k + \alpha \underline{d}_k). \tag{82}$$

This can simplify some of the effort of implementing the rules of the previous section.

5.3 Termination Conditions

Having established rules for determining a descent direction $\underline{d}_k$ and a descending stepsize α_k, it remains to determine when the descent algorithm should terminate. In general, it will be impossible to compute a local minimum $\underline{x}^*$ exactly, and without prior knowledge of $\underline{x}^*$ or f, it can be difficult to know in advance how far a given $\underline{x}_k$ is from $\underline{x}^*$. The following provide several useful heuristics for considering terminating an algorithm:

- Terminate when $||\nabla f(\underline{x}_k)|| < \varepsilon$: We have that at the minimum $\underline{x}^*$, $\nabla f(\underline{x}^*) = 0$. Hence, for $\underline{x}_k$ that are *near* $\underline{x}^*$, it is reasonable to expect that $\nabla f(\underline{x}_k)$ should be *near* the zero vector. Hence, if the norm of the gradient becomes sufficiently close to zero (sufficiently close potentially being problem-defined or defined by the bounds of computational accuracy), it is reasonable to assume that the iterate $\underline{x}_k$ is close to a minimum. In general, it may also be wise to implement checks on the positive definiteness of $\nabla^2 f(\underline{x}_k)$, in case the algorithm is descending into a saddle point.

- Terminate when $||\nabla f(\underline{x}_k)||/||\nabla f(\underline{x}_0)|| < \varepsilon$: This essentially 'rescales' the gradient, to ensure that the gradient is not simply near zero because the relevant variables are near zero.

- Terminate when $||\underline{d}_k|| < \varepsilon$: In the case that the descent direction is derived from the gradient, it may be suitable to look at the scale of the descent direction vector. When it becomes near the zero vector (and thus has gradient near zero), you may be sufficiently close to a stationary point.

- Terminate when any combination of the above conditions are met, or when any problem-specific criteria are met: For the specific problem of interest, there may be specific cost criteria that indicate when to terminate the algorithm, for instance achieving optimization above or below a certain threshold, or exhausting some computational budget.

The following theorem provides some justification or mathematical confidence for considering termination conditions of the above form:

Theorem 7 *Let $\underline{x}^*$ be a local minimum of f. Suppose that there is some constant $m > 0$ and a sphere S centered at $\underline{x}^*$ such that for all $\underline{x} \in S$, the smallest eigenvalue of $\nabla^2 f(\underline{x})$ is at least m. In that case, for every $x \in S$ with $||\nabla f(\underline{x}_k)|| < \varepsilon$, the following holds:*

$$||\underline{x} - \underline{x}^*|| < \frac{\varepsilon}{m} \text{ and } f(\underline{x}) - f(\underline{x}^*) < \frac{\varepsilon^2}{m}. \tag{83}$$

The above result indicates that under certain conditions on the Hessian, being sufficiently close to zero in gradient implies you are estimably close to the minimizer. It is of course difficult to know an appropriate value m in advance, but the above theorem implies we should have some confidence in the utility of our descent algorithms.

5.4 Standard Descent Algorithms

There are a couple of descent algorithms that are somewhat 'standard', as pure forms of the above indicated rules.

Steepest Descent: Steepest descent is often implemented with a constant stepsize $\alpha > 0$, taking the form of

$$\underline{x}_{k+1} = \underline{x}_k - \alpha \nabla f(\underline{x}_k). \tag{84}$$

The constant α is frequently referred to as the 'learning rate'. As noted previously, if α is too small, this can lead to slow convergence of the algorithm. If α is too large, however, it can happen that approaching the minimum $\underline{x}^*$ from certain directions can result in overshoot, and oscillation around the minimum. This is addressed in more detail in Section 7.

Coordinate Descent: There are three traditional approaches to implementing coordinate descent, which vary in terms of the direction taken:

- **Cyclic:** In this case, the function f is minimized relative to the $x_1, x_2, \ldots x_n$ variables successively, in order, before repeating the process again with x_1. An alternative is the Aitken Double Sweep, which first successively minimizes with respect to $x_1, x_2, \ldots, x_n$ then minimizes with respect to $x_{n-1}, x_{n-2}, \ldots, x_2, x_1$, before repeating. An advantage to this approach is that the full gradient need not be computed only the derivative relative to any single variable at a time (which can itself be approximated).

- **The Gauss-Southwell Method:** In this case, which was referenced previously, at time k the coordinate i is chosen with the largest value of $|\partial f / \partial x_i(\underline{x}_k)|$, i.e., the coordinate which can produce the largest decrease in f for the smallest change in x_i.

- **Randomized Choice:** At each time step, a coordinate i is chosen at random from the available n coordinates. Again in this case, the full gradient need not be computed in any single time step.

The actual minimization with respect to a coordinate selected by the above means may be performed by any of the rules recommended for choosing good stepsizes above. In general, coordinate descent has slower convergence properties than steepest descent, but can be easier to implement.

Newton's Method: Newton's method is often implemented in its 'pure' form as:

$$\underline{x}_{k+1} = \underline{x}_k - \left[\nabla^2 f(\underline{x}_k)\right]^{-1} \nabla f(\underline{x}_k), \tag{85}$$

essentially taking the stepsize $\alpha = 1$.

This is essentially justified from Eqs. (66) and (67), approximating the function f as a quadratic form centered on $\underline{x}_k$ and solving for the $\underline{x}_{k+1}$ that minimizes the resulting approximation.

However, there are a couple of things to be mindful of with respect to Newton's Method. For instance, in its pure form as above, it is just as likely to converge to a local maximum of a function as converge to a local minimum. Newton's method essentially iteratively solves the system of equations defined by $\nabla f(\underline{x}) = 0$, which is satisfied both at minima and maxima. In order to keep from mistakenly converging on a local maximum, it is necessary to guarantee that *descent* steps are being taken. If the direction $-\left[\nabla^2 f(\underline{x}_k)\right]^{-1} \nabla f(\underline{x}_k)$ is not a descent direction, consider the direction $\left[\nabla^2 f(\underline{x}_k)\right]^{-1} \nabla f(\underline{x}_k)$.

It can also happen that $\nabla^2 f(\underline{x}_k)$ is not invertible. In such a case, consider modifying the Hessian to something of the form

$$\nabla^2 f(\underline{x}_k) + \varepsilon_k \mathrm{I}, \tag{86}$$

for $\varepsilon_k > 0$ to render the matrix invertible, or alternatively consider an alternative descent method at this point if this becomes an issue.

5.5 Example Problems

1 Implement some descent algorithms (in a language of your choice) and run them on suitable test functions f to compare their performance. Are there any functions where one algorithm is preferable to another? How do the solutions you get compare to available 'black box' numerical optimizers, for instance in Python or Mathematica or Matlab? Look up the documentation of these numerical optimizers, and see what algorithms they are running.

2 For a suitable test function f and various start points $\underline{x}$ and directions $\underline{d}$, implement Amijo's rule to determine an appropriate step size. How do the results vary with the hyperparameters (s, β, σ)? Are there any choices of hyperparameters that seem to perform particularly well?

3 How do termination conditions affect the performance of a descent algorithm?

4 The minimization rule for finding optimal α_k is premised on reducing 'three point patterns', based on intervals that are known to contain minima in one dimension. How can you systematically construct an *initial* three point pattern to start these algorithms with? *Hint: Think about the motivation for Armijo's Rule.*

5 Consider a function of the form $f(\underline{x}) = \frac{1}{2}\underline{x}^{\mathrm{T}} Q \underline{x} - \underline{b}^{\mathrm{T}} \underline{x}$, for positive definite Q. (It will suffice here to consider small dimensions, for instance $n = 2$ or $n = 3$.) How does steepest descent

behave with a constant stepsize $\alpha > 0$ that is 'too large'? Does it behave that way from all initial guesses? How does Q define what 'too large' is? This problem will be considered further in the next section.

6 Examples on General Quadratic Functions

It is useful and illustrative to consider the main descent algorithms as applied to quadratic forms:

$$f(\underline{x}) = \frac{1}{2}\underline{x}^{\mathrm{T}} Q \underline{x} - \underline{b}^{\mathrm{T}} \underline{x}, \tag{87}$$

where Q is taken to be a positive definite symmetric matrix, and $\underline{b}$ an arbitrary vector.

Note, we have that $\nabla f(\underline{x}) = Q\underline{x} - \underline{b}$ and $\nabla^2 f(\underline{x}) = Q$. Hence we have, by Theorem 2 that the minimum of the function occurs at $\underline{x}^* = Q^{-1}\underline{b}$. If Q is sufficiently small, it can be inverted and the minimizer computed directly using this formula. However, for very large Q, computing Q^{-1} can be a very expensive task. This prompts the use of descent algorithms.

Steepest Descent: Taking the standard steepest descent iteration, we have

$$\underline{x}_{k+1} = \underline{x}_k - \alpha_k \nabla f(\underline{x}_k). \tag{88}$$

We are interested in computing a good stepsize α_k. Consider utilizing the minimization rule, taking α_k to realize the following minimum:

$$\begin{aligned}
&\min_{\alpha>0} f(\underline{x}_k - \alpha \nabla f(\underline{x}_k)) \\
&= \min_{\alpha>0} \frac{1}{2} \left(\underline{x}_k - \alpha \nabla f(\underline{x}_k)\right)^{\mathrm{T}} Q \left(\underline{x}_k - \alpha \nabla f(\underline{x}_k)\right) - \underline{b}^{\mathrm{T}} \left(\underline{x}_k - \alpha \nabla f(\underline{x}_k)\right) \\
&= \frac{1}{2}\underline{x}_k^{\mathrm{T}} Q \underline{x}_k - \underline{b}^{\mathrm{T}} \underline{x}_k + \min_{\alpha>0} \left(\frac{1}{2}\alpha^2 \nabla f(\underline{x}_k)^{\mathrm{T}} Q \nabla f(\underline{x}_k) - \alpha \nabla f(\underline{x}_k)^{\mathrm{T}} Q \underline{x}_k + \alpha \underline{b}^{\mathrm{T}} \nabla f(\underline{x}_k) \right) \\
&= f(\underline{x}_k) + \min_{\alpha>0} \left(\frac{1}{2}\alpha^2 \nabla f(\underline{x}_k)^{\mathrm{T}} Q \nabla f(\underline{x}_k) - \alpha \nabla f(\underline{x}_k)^{\mathrm{T}} \left(Q\underline{x}_k - \underline{b}\right) \right) \\
&= f(\underline{x}_k) + \min_{\alpha>0} \left(\frac{1}{2}\alpha^2 \nabla f(\underline{x}_k)^{\mathrm{T}} Q \nabla f(\underline{x}_k) - \alpha \nabla f(\underline{x}_k)^{\mathrm{T}} \nabla f(\underline{x}_k) \right).
\end{aligned} \tag{89}$$

Note, the above minimum is realized taking $\alpha = \nabla f(\underline{x}_k)^{\mathrm{T}} \nabla f(\underline{x}_k) / \nabla f(\underline{x}_k)^{\mathrm{T}} Q \nabla f(\underline{x}_k)$, which yields

$$\min_{\alpha>0} f(\underline{x}_k - \alpha \nabla f(\underline{x}_k)) = f(\underline{x}_k) - \frac{1}{2} \frac{\left(\nabla f(\underline{x}_k)^{\mathrm{T}} \nabla f(\underline{x}_k)\right)^2}{\nabla f(\underline{x}_k)^{\mathrm{T}} Q \nabla f(\underline{x}_k)}. \tag{90}$$

The above clearly represents a descent if $\nabla f(\underline{x}_k) \neq 0$, so we have as an explicit formula for the optimal stepsize,

$$\alpha_k = \frac{\nabla f(\underline{x}_k)^{\mathrm{T}} \nabla f(\underline{x}_k)}{\nabla f(\underline{x}_k)^{\mathrm{T}} Q \nabla f(\underline{x}_k)}. \tag{91}$$

This yields the 'optimal' steepest descent iteration in the case of quadratic f:

$$\underline{x}_{k+1} = \underline{x}_k - \frac{\nabla f(\underline{x}_k)^{\mathrm{T}} \nabla f(\underline{x}_k)}{\nabla f(\underline{x}_k)^{\mathrm{T}} Q \nabla f(\underline{x}_k)} \nabla f(\underline{x}_k), \tag{92}$$

with

$$f(\underline{x}_{k+1}) = f(\underline{x}_k) - \frac{1}{2}\frac{\left(\nabla f(\underline{x}_k)^{\mathrm{T}}\nabla f(\underline{x}_k)\right)^2}{\nabla f(\underline{x}_k)^{\mathrm{T}} Q \nabla f(\underline{x}_k)}. \tag{93}$$

To analyze the convergence, note that the above gives us

$$f(\underline{x}_{k+1}) - f(\underline{x}^*) = f(\underline{x}_k) - f(\underline{x}^*) - \frac{1}{2}\frac{\left(\nabla f(\underline{x}_k)^{\mathrm{T}}\nabla f(\underline{x}_k)\right)^2}{\nabla f(\underline{x}_k)^{\mathrm{T}} Q \nabla f(\underline{x}_k)}. \tag{94}$$

Additionally, note that for any $\underline{x}$, we have (after some algebra)

$$f(\underline{x}) - f(\underline{x}^*) = \frac{1}{2}(Q\underline{x} - \underline{b})^{\mathrm{T}} Q^{-1}(Q\underline{x} - \underline{b}) = \frac{1}{2}\nabla f(\underline{x})^{\mathrm{T}} Q^{-1} \nabla f(\underline{x}). \tag{95}$$

Hence,

$$\begin{aligned} f(\underline{x}_{k+1}) - f(\underline{x}^*) &= \left(1 - \frac{1}{2}\frac{\left(\nabla f(\underline{x}_k)^{\mathrm{T}}\nabla f(\underline{x}_k)\right)^2}{\left[\nabla f(\underline{x}_k)^{\mathrm{T}} Q \nabla f(\underline{x}_k)\right]\left[f(\underline{x}_k) - f(\underline{x}^*)\right]}\right)\left(f(\underline{x}_k) - f(\underline{x}^*)\right) \\ &= \left(1 - \frac{\left(\nabla f(\underline{x}_k)^{\mathrm{T}}\nabla f(\underline{x}_k)\right)^2}{\left[\nabla f(\underline{x}_k)^{\mathrm{T}} Q \nabla f(\underline{x}_k)\right]\left[\nabla f(\underline{x}_k)^{\mathrm{T}} Q^{-1} \nabla f(\underline{x}_k)\right]}\right)\left(f(\underline{x}_k) - f(\underline{x}^*)\right). \end{aligned} \tag{96}$$

At this point we may make use of the **Kantorovich Inequality**, which says that for any symmetric, positive definite matrix Q, with largest eigenvalue $\lambda_{\max}$ and smallest eigenvalue $\lambda_{\min}$, the following inequality holds for all $\underline{v} \in \mathbb{R}^n$ with $\underline{v} \neq 0$:

$$\frac{\left(\underline{v}^T \underline{v}\right)^2}{\left[\underline{v}^{\mathrm{T}} Q \underline{v}\right]\left[\underline{v}^{\mathrm{T}} Q^{-1} \underline{v}\right]} \geqslant \frac{4\lambda_{\max}\lambda_{\min}}{(\lambda_{\max} + \lambda_{\min})^2}. \tag{97}$$

See the example problems regarding the proof. Applying this to the above, taking $\underline{v} = \nabla f(\underline{x}_k)$, we have

$$\begin{aligned} f(\underline{x}_{k+1}) - f(\underline{x}^*) &\leqslant \left(1 - \frac{4\lambda_{\max}\lambda_{\min}}{(\lambda_{\max} + \lambda_{\min})^2}\right)\left(f(\underline{x}_k) - f(\underline{x}^*)\right) \\ &= \left(\frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}}\right)^2 \left(f(\underline{x}_k) - f(\underline{x}^*)\right). \end{aligned} \tag{98}$$

Hence we see that in the case of optimally chosen α_k, applying steepest descent to a quadratic function yields an error in the minimum that decreases in each iteration by a factor of $((\lambda_{\max} - \lambda_{\min})/(\lambda_{\max} + \lambda_{\min}))^2$. Note the importance of the ratio $C(Q) = \lambda_{\max}/\lambda_{\min}$, known as the 'condition number' of Q, since

$$\frac{\lambda_{\max} - \lambda_{\min}}{\lambda_{\max} + \lambda_{\min}} = \frac{C(Q) - 1}{C(Q) + 1} = 1 - \frac{2}{C(Q) + 1}. \tag{99}$$

Hence, if $C(Q) = \lambda_{\max}/\lambda_{\min}$ is very large, the error of steepest descent can contract very slowly. Such matrices are 'ill-conditioned'. Often ill-conditioning can be addressed by changing coordinates from $\underline{x}$ to some alternative coordinate system. In the general case, for non-quadratic f, convergence to a minimum $\underline{x}^*$ will frequently be governed by the condition number of the Hessian $\nabla^2 f(\underline{x}^*)$.

Newton's Method: Consider taking the pure Newton's method iteration, governed by

$$\underline{x}_{k+1} = \underline{x}_k - \left[\nabla^2 f(\underline{x}_k)\right]^{-1} \nabla f(\underline{x}_k). \tag{100}$$

Noting that $\nabla^2 f(\underline{x}_k) = Q$ and $\nabla f(\underline{x}_k) = Q\underline{x}_k - \underline{b}$, we have

$$\underline{x}_{k+1} = \underline{x}_k - Q^{-1}\left(Q\underline{x}_k - \underline{b}\right) = \underline{x}_k - Q^{-1}Q\underline{x}_k + Q^{-1}\underline{b} = Q^{-1}\underline{b}. \tag{101}$$

Noting again that $\underline{x}^* = Q^{-1}\underline{b}$, the above shows that Newton's method solves for the quadratic minimizer in one step, independent of the initial starting point. This is not surprising - Newton's method is premised on approximating the function f by a quadratic form and minimizing that. In the case that f *is* a quadratic form, there is no approximation necessary, the the minimization is exact. **HOWEVER**, if the whole point of utilizing these iterative solution algorithms is to avoid the potentially expensive computation of Q^{-1} for the known optimum $Q^{-1}\underline{b}$, Newton's method is a poor choice.

6.1 Example Problems

1 Consider applying steepest descent with constant stepsize α to the function $f(\underline{x}) = \frac{1}{2}\underline{x}^{\mathrm{T}}Q\underline{x} - \underline{b}^{\mathrm{T}}\underline{x}$, with Q as a symmetric positive definite matrix. Show that if $0 < \alpha < 2/\lambda_{\max}$ where $\lambda_{\max}$ is the largest eigenvalue of Q, then the iteration

$$\underline{x}_{k+1} = \underline{x}_k - \alpha\nabla f(\underline{x}_k) \tag{102}$$

always yields descent. *Hint: If λ is an eigenvalue of Q, then $1-\alpha\lambda$ is an eigenvalue of $I-\alpha Q$. See also the properties in Appendix A.*

2 Show that taking $0 < \alpha < 2/\lambda_{\max}$ in the above problem, the iterates $\underline{x}_k$ will converge to $\underline{x}^* = Q^{-1}\underline{b}$. Consider, as an example, $\alpha = 2/(\lambda_{\max} + \lambda_{\min})$.

3 The analysis of this section shows that for $f(\underline{x}) = (1/2)\underline{x}^{\mathrm{T}}Q\underline{x} - \underline{b}^{\mathrm{T}}\underline{x}$, under steepest descent with optimal stepsize, the function values $f(\underline{x}_k)$ converge *monotonically* to the minimal function value $f(\underline{x}^*)$. Is it true that the $\underline{x}_k$ converge similarly monotonically to $\underline{x}^*$? For instance, is $||\underline{x}_{k+1} - \underline{x}^*|| < ||\underline{x}_k - \underline{x}^*||$? Explain. Show that the Q-norm of the error, $||\underline{v}||_Q = \sqrt{\frac{1}{2}\underline{v}^{\mathrm{T}}Q\underline{v}}$ does converge monotonically.

4 Prove the Kantorovich Inequality, Eq. (97). *Hint: Because Q is real, symmetric, and positive definite, there is an orthonormal set of eigenvectors that forms a basis of $\mathbb{R}^n$. Express $\underline{v}$ in terms of these eigenvectors.*

7 Convergence Results

We are generally concerned with iterative descent algorithms of the form:

$$\underline{x}_{k+1} = \underline{x}_k + \alpha_k \underline{d}_k. \tag{103}$$

There are two central questions with regards to convergence of these algorithms:

- Under what conditions will the iterates $\{\underline{x}_k\}$ converge to a local minimum $\underline{x}^*$?
- At what rate does the convergence occur, i.e., how much does $\underline{x}_{k+1}$ improve over $\underline{x}_k$ as an estimate of $\underline{x}^*$?

7.1 Verifying Convergence

The ideal behavior of these descent algorithms is that the iterates $\{\underline{x}_k\}$ converge to a stationary point $\underline{x}^*$ where $\nabla f(\underline{x}^*) = 0$. Since each direction $\underline{d}_k$ was chosen to be a descent direction (i.e., $\nabla f(\underline{x}_k)^{\mathrm{T}} \underline{d}_k < 0$), this ensures that $\underline{x}^*$ is a local minimum of f (or potentially a saddle point - be mindful of the Hessian!).

However, it might happen for instance that the directions $\underline{d}_k$ are chosen in such a way as to, in the limit, be orthogonal to the gradients $\nabla f(\underline{x}_k)$. In that case, it might happen that the $\underline{x}_k$ converge to a *non*-stationary point $\underline{x}'$ (i.e., $\nabla f(\underline{x}') \neq 0$ and hence, is not a minimum), while simultaneously (due to the hypothetical orthogonality)

$$\lim_{k\to\infty} \nabla f(\underline{x}_k)^{\mathrm{T}} \underline{d}_k = 0. \tag{104}$$

Because the directional derivative of f at time k is given by $\nabla f(\underline{x}_k)^{\mathrm{T}} \underline{d}_k$, the above implies that the method could get trapped at such a point $\underline{x}'$. In order to guarantee convergence of a descent algorithm, it is necessary to ensure that such an event cannot happen. The following results will be described in fairly general terms, but it can be shown that these general results apply to the standard descent algorithms of the previous sections.

In order to characterize the orthogonality property above, we introduce the following definition:

Definition 3 *The sequence of directions $\{\underline{d}_k\}$ corresponding to the sequence of points $\{\underline{x}_k\}$ is **gradient-related** if for any subsequence $\{\underline{x}_{k_n}\}$ that converges to a non-stationary point, the corresponding $\{\underline{d}_{k_n}\}$ are bounded, and*

$$\limsup_n \nabla f(\underline{x}_{k_n})^T \underline{d}_{k_n} < 0. \tag{105}$$

We have the following result, which guarantees that the result of the descent algorithms of the previous sections will be stationary points.

Theorem 8 *Let $\{\underline{x}_k\}$ be the iterates of a descent algorithm generated by*

$$\underline{x}_{k+1} = \underline{x}_k + \alpha_k \underline{d}_k. \tag{106}$$

Let $\{\underline{d}_k\}$ is gradient related, and α_k is chosen by i) Armijo's rule, ii) the minimization rule, or iii) the limited minimization rule. If the $\{\underline{x}_k\}$ converge to some limit $\underline{x}'$, then $\nabla f(\underline{x}') = 0$.

Proof. For simplicity, we verify the result only for the minimization rule, i.e., taking

$$\alpha_k = \arg\min_{\alpha \geqslant 0} f(\underline{x}_k + \alpha \underline{d}_k). \tag{107}$$

Note, taking this absolute minimization rule, we have that the directional derivative at $\underline{x}_{k+1}$ in the direction of $\underline{d}_k$ is 0, i.e.,

$$\nabla f(\underline{x}_{k+1})^{\mathrm{T}} \underline{d}_k = 0. \tag{108}$$

Assume that the iterates $\{\underline{x}_k\}$ have a limit $\underline{x}'$ that is not stationary, i.e., $\nabla f(\underline{x}') \neq 0$. Taking $\{\underline{d}_k\}$ as gradient-related, that the sequence $\{\underline{d}_k\}$ is bounded (i.e., for some $\delta > 0$, $||\underline{d}_k|| < \delta$ for all sufficiently large k), and satisfies

$$\limsup_k \nabla f(\underline{x}_k)^{\mathrm{T}} \underline{d}_k < 0. \tag{109}$$

By the boundedness of $\{\underline{d}_k\}$, we may extract a subsequence $\{\underline{d}_{k_n}\}$ that converges to some finite limit point, $\underline{d}'$. Note, the corresponding subsequence $\{\underline{x}_{k_n}\}$ converges to $\underline{x}'$.

We have additionally by the above equation that

$$\limsup_n \nabla f(\underline{x}_{k_n})^{\mathrm{T}} \underline{d}_{k_n} = \lim_n \nabla f(\underline{x}_{k_n})^{\mathrm{T}} \underline{d}_{k_n} = \nabla f(\underline{x}')^{\mathrm{T}} \underline{d}' < 0. \tag{110}$$

However, we also have that $\nabla f(\underline{x}_{k_n+1})^{\mathrm{T}} \underline{d}_{k_n} = 0$ for all k_n on the subsequence. Taking limits as $k_n \to \infty$, we have $\nabla f(\underline{x}')^{\mathrm{T}} \underline{d}' = 0$. This is a contradiction. Hence, $\nabla f(\underline{x}') = 0$, and the limit is stationary.

The above result guarantees that if a descent algorithm is converging to a point, it is converging to a solution to $\nabla f(\underline{x}) = 0$. In the view of 'descent', this means that the algorithm cannot get stuck on the side of a sloping function before reaching the bottom. The above result can actually be strengthened, to show that even if an absolute limit $\underline{x}'$ of the sequence $\{\underline{x}_k\}$ doesn't exist, any limit point of the sequence must be a stationary point (see Prop. 1.2.1. in *Nonlinear Programming*, Bertsekas).

These result give some confidence in the final output of these descent algorithms. One rule not addressed in the above theorem is that of constant stepsize, $\alpha_k = \alpha$ for some constant value $\alpha > 0$. The difficulty is that in general, convergence cannot be guaranteed if α is too large. For further analysis, see the following subsection on Steepest Descent with Constant Stepsize.

7.2 Convergence Rates

The previous subsection essentially guarantees that the limit of our descent algorithms will be a stationary point. In this section, we consider the question of how quickly the iterates of the algorithm converge to that solution. Essentially, how much computational effort must be spent to achieve a certain degree of accuracy?

Assume that the iterates $\{\underline{x}_k\}$ converge to some local minimum $\underline{x}^*$. Let $\{\underline{e}_k\}$ be the errors, i.e., $\underline{e}_k = \underline{x}_k - \underline{x}^*$. We generally characterize the rate of convergence of the algorithm in terms of the rate of convergence of $\underline{e}_k$ to 0 or the rate of convergence of $f(\underline{x}_k)$ to $f(\underline{x}^*)$. The errors converge *with order p* if for some positive constant c,

$$\limsup_k \frac{||\underline{e}_{k+1}||}{||\underline{e}_k||^p} \leqslant c. \tag{111}$$

The general results will be that the 'typical' convergence of steepest descent is linear or of order 1, i.e., $||\underline{e}_{k+1}||/||\underline{e}_k||$ is bound in the limit by some constant less than 1. The typical convergence of Newton's Method is quadratic, i.e., $||\underline{e}_{k+1}||/||\underline{e}_k||^2$ is bound in the limit by some constant.

However, different behavior can frequently arise in 'singular' circumstances, particularly when the Hessian has eigenvalues at or near 0 in the close vicinity of the minimum $\underline{x}^*$. In such cases, the performances of these algorithms is typically worse. We can generally only guarantee *superlinear* convergence for Newton's method, i.e., $||\underline{e}_{k+1}||/||\underline{e}_k||$ goes to 0 in the limit. For gradient-type methods like steepest descent, we can only guarantee *super-arithmetic* convergence,

$$f(\underline{x}_k) - f(\underline{x}^*) = o\left(\frac{1}{k}\right). \tag{112}$$

The arguments of the following sections should not be taken as rigorous, but explanatory, justifying the typical behavior of the algorithms. We assume here that f is thrice differentiable, and hence we may make use of the following expansion around any stationary limit point $\underline{x}^*$:

$$f(\underline{x}^* + \underline{e}) = f(\underline{x}^*) + \frac{1}{2}\underline{e}^{\mathrm{T}}\nabla^2 f(\underline{x}^*)\underline{e} + O\left(||\underline{e}||^3\right) \tag{113}$$

7.2.1 Steepest Descent with Constant Stepsize

Consider applying steepest descent with constant stepsize to a function f, i.e., for some $\alpha > 0$ we have

$$\underline{x}_{k+1} = \underline{x}_k - \alpha\nabla f(\underline{x}_k). \tag{114}$$

Assume that the iterates $\{\underline{x}_k\}$ converge to some $\underline{x}^*$, with $\nabla^2 f(\underline{x}^*)$ positive definite. Taking $\underline{x}_k = \underline{x}^* + \underline{e}_k$, we have (for large k where the error is small),

$$\underline{x}^* + \underline{e}_{k+1} = \underline{x}^* + \underline{e}_k - \alpha\nabla\left[f(\underline{x}^*) + \frac{1}{2}\underline{e}_k^{\mathrm{T}}\nabla^2 f(\underline{x}^*)\underline{e}_k + O\left(||\underline{e}_k||^3\right)\right], \tag{115}$$

or

$$\begin{aligned}\underline{e}_{k+1} &= \underline{e}_k - \alpha\left[\nabla^2 f(\underline{x}^*)\underline{e}_k + O\left(||\underline{e}_k||^2\right)\right] \\ &= \left(\mathrm{I} - \alpha\nabla^2 f(\underline{x}^*)\right)\underline{e}_k + O\left(||\underline{e}_k||^2\right).\end{aligned} \tag{116}$$

where the asymptotic term should be interpreted as a vector function, with indicated order bounds on the norm. Taking the norm of the above equation, we may apply the triangle inequality to yield

$$\frac{||\underline{e}_{k+1}||}{||\underline{e}_k||} \leqslant \frac{||\left(\mathrm{I} - \alpha\nabla^2 f(\underline{x}^*)\right)\underline{e}_k||}{||\underline{e}_k||} + \frac{O\left(||\underline{e}_k||^2\right)}{||\underline{e}_k||}. \tag{117}$$

Let $\lambda_{\max}$ and $\lambda_{\min}$ respectively be the largest and smallest eigenvalues of $\nabla^2 f(\underline{x}^*)$. Note that the largest and smallest eigenvalues of $\mathrm{I} - \alpha\nabla^2 f(\underline{x}^*)$ will be $1 - \alpha\lambda_{\min}$ and $1 - \alpha\lambda_{\max}$ respectively. From the properties of eigenvalues and eigenvectors (Appendix A), we therefore have

$$\frac{||\underline{e}_{k+1}||}{||\underline{e}_k||} \leqslant \max\left(|1 - \alpha\lambda_{\min}|, |1 - \alpha\lambda_{\max}|\right) + O\left(||\underline{e}_k||\right). \tag{118}$$

Taking the limit as $k \to \infty$, noting that the errors converge to 0, we have

$$\limsup_k \frac{||\underline{e}_{k+1}||}{||\underline{e}_k||} \leqslant \max\left(|1 - \alpha\lambda_{\min}|, |1 - \alpha\lambda_{\max}|\right). \tag{119}$$

However, to guarantee convergence, it is necessary the the constant above be less than 1 (otherwise, the errors might be dilating to infinity, for instance). In order to guarantee this, we must have $0 < \alpha < 2/\lambda_{\max}$. In general, if this is violated (if α is too large), it can happen that convergence to a minimizer fails - the stepsize is essentially too big, and approaching the minimizer from the wrong direction can lead to oscillation. If the stepsize is sufficiently small, we may guarantee convergence - though note that if the stepsize is too small, the constant contraction factor $\max(|1-\alpha\lambda_{\min}|, |1-\alpha\lambda_{\max}|)$ will be very close to 1, giving slow convergence.

Note that because the acceptable range of α depends on the eigenvalues of $\nabla^2 f(\underline{x}^*)$, it can be difficult to know in advance what a good stepsize is. This can be addressed somewhat by experimentation - for instance, computing as an approximation $\nabla^2 f(\underline{x}_0)$ for $\underline{x}_0$ assumed to be near the minimum. In general though, this motivates an interest in stepsizes that decrease (slowly) over time.

7.2.2 Newton's Method

Consider applying Newton's method in the vicinity of a minimizer $\underline{x}^*$ with $\nabla^2 f(\underline{x}^*)$ positive definite. We have

$$\underline{x}_{k+1} = \underline{x}_k - \left[\nabla^2 f(\underline{x}_k)\right]^{-1} \nabla f(\underline{x}_k). \tag{120}$$

Note we have the following expansions:

$$\begin{aligned} \nabla f(\underline{x}^* + \underline{e}) &= \nabla^2 f(\underline{x}^*)\underline{e} + O(||\underline{e}||^2) \\ \nabla^2 f(\underline{x}^* + \underline{e}) &= \nabla^2 f(\underline{x}^*) + O(||\underline{e}||), \end{aligned} \tag{121}$$

where the asymptotic terms above should be interpreted as a vector function and a matrix function of bound norms, respectively. Applying this to the iterates of Newton's method, taking $\underline{x}_k = \underline{x}^* + \underline{e}_k$, we have

$$\begin{aligned} \underline{e}_{k+1} &= \underline{e}_k - \left[\nabla^2 f(\underline{x}^*) + O(||\underline{e}_k||)\right]^{-1} \left(\nabla^2 f(\underline{x}^*)\underline{e}_k + O(||\underline{e}_k||^2)\right). \\ &= \left[\nabla^2 f(\underline{x}^*) + O(||\underline{e}_k||)\right]^{-1} \left(\left[\nabla^2 f(\underline{x}^*) + O(||\underline{e}_k||)\right]\underline{e}_k - \left(\nabla^2 f(\underline{x}^*)\underline{e}_k + O(||\underline{e}_k||^2)\right)\right) \\ &= \left[\nabla^2 f(\underline{x}^*) + O(||\underline{e}_k||)\right]^{-1} \left(\nabla^2 f(\underline{x}^*)\underline{e}_k + O(||\underline{e}_k||)\underline{e}_k - \nabla^2 f(\underline{x}^*)\underline{e}_k - O(||\underline{e}_k||^2)\right) \\ &= \left[\nabla^2 f(\underline{x}^*) + O(||\underline{e}_k||)\right]^{-1} \left(O(||\underline{e}_k||)\underline{e}_k - O(||\underline{e}_k||^2)\right). \end{aligned} \tag{122}$$

Note, the two asymptotic terms above can be combined into a single $O(||\underline{e}_k||^2)$ (vector-function) term,

$$\underline{e}_{k+1} = \left[\nabla^2 f(\underline{x}^*) + O(||\underline{e}_k||)\right]^{-1} O\left(||\underline{e}_k||^2\right). \tag{123}$$

Note from the above, that for sufficiently large k (and therefore sufficiently small error) we have that $\nabla^2 f(\underline{x}^*) + O(||\underline{e}_k||)$ is invertible, and is approximately $\nabla^2 f(\underline{x}^*)$. It is clear then from the above that in the limit, the $k+1$-th error term is on the order of the square of the previous error term, i.e,

$$\limsup_k \frac{||\underline{e}_{k+1}||}{||\underline{e}_k||^2} \tag{124}$$

is at most some finite constant, depending on the Hessian $\nabla^2 f(\underline{x}^*)$ and the higher order behavior of f around $\underline{x}^*$.

7.3 Example Problems

1 Show that Steepest Descent, Coordinate Descent, and Newton's Method all generate gradient related direction sequences $\{\underline{d}_k\}$. *Hint: Assume that some subsequence $\underline{x}_{k_n}$ converges to a non-stationary $\underline{x}'$. What can be said about the corresponding $\underline{d}_k$?*

2 Consider the 1-d problem, looking at $f(x) = -\cos(x)$. How do the iterates of Newton's method converge to the minimum $x^* = 0$? How does this compare to the analysis in the above section? How does this compare to $f(x) = x^4$ or $f(x) = |x|^{1/2}$?

3 Look up coordinate descent methods. How do the rates of convergence compare to steepest descent, or Newton's method? When might you prefer one to the other?

8 Conjugate Direction Methods

This section discusses a method for minimizing quadratic forms, specifically tailed to the algebraic structure of the problem. This leads to an algorithm that can solve for quadratic minimizers in no more than n steps. As (most/many) functions behave essentially quadratically in a neighborhood of a minimum, this method can be generalized to apply to non-quadratic functions (though, in the non-quadratic case, more than n steps are generally required).

Let $f(\underline{x}) = \frac{1}{2}\underline{x}^{\mathrm{T}}Q\underline{x} - \underline{b}^{\mathrm{T}}\underline{x}$, for Q a symmetric, positive definite matrix. Note that since $\nabla f(\underline{x}) = Q\underline{x} - \underline{b}$, we have the explicit formula for the minimizer $\underline{x}^* = Q^{-1}\underline{b}$. However, as discussed previously, if Q is particularly large, inverting it to compute $\underline{x}^*$ can be particularly costly.

The key concept in this section is the notion of Q-conjugate vectors:

Definition 4 *For a positive definite matrix Q, the set of non-zero vectors $\{\underline{d}_0, \underline{d}_1, \ldots, \underline{d}_{n-1}\}$ is **Q-conjugate** if for each $i \neq j$, $\underline{d}_j^T Q \underline{d}_i = 0$.*

The property of being Q-conjugate can be thought of as a sort of Q-mediated orthogonality. There are a number of important properties of Q-conjugate vectors, perhaps most importantly that they are linearly independent. To see this, suppose

$$\psi_0 \underline{d}_0 + \psi_1 \underline{d}_1 + \ldots + \psi_{n-1}\underline{d}_{n-1} = 0. \tag{125}$$

Consider multiplying each side of the above equation by $\underline{d}_i^{\mathrm{T}} Q$. In that case, we have

$$\begin{aligned} \psi_0 \underline{d}_i^{\mathrm{T}} Q \underline{d}_0 + \psi_1 \underline{d}_i^{\mathrm{T}} Q \underline{d}_1 + \ldots + \psi_{n-1}\underline{d}_i^{\mathrm{T}} Q \underline{d}_{n-1} &= 0 \\ \psi_0 0 + \psi_1 0 + \ldots \psi_i \underline{d}_i^{\mathrm{T}} Q \underline{d}_i + \ldots + \psi_{n-1} 0 &= 0 \\ \psi_i \underline{d}_i^{\mathrm{T}} Q \underline{d}_i &= 0. \end{aligned} \tag{126}$$

Since Q is positive definite and $\underline{d}_i$ is non-zero, we have from the above that $\psi_i = 0$. As this must hold for all i, this verifies that all the $\underline{d}_i$ are linearly independent. If there are n such vectors, they necessarily span $\mathbb{R}^n$. This leads to the most important property of Q-conjugate vectors.

The concept of Q-conjugate vectors is useful for the following reason: given a set of Q-conjugate vectors, the minimizer $\underline{x}^* = Q^{-1}\underline{b}$ may be expressed simply and immediately. Let

$$\psi_0 \underline{d}_0 + \ldots + \psi_{n-1}\underline{d}_{n-1} = Q^{-1}\underline{b}. \tag{127}$$

Multiplying each side of the above equation by $\underline{d}_i^{\mathrm{T}} Q$ and simplifying due to Q-conjugacy yields

$$\psi_i \underline{d}_i^{\mathrm{T}} Q \underline{d}_i = \underline{d}_i^{\mathrm{T}} \underline{b}, \tag{128}$$

or

$$\psi_i = \frac{\underline{d}_i^{\mathrm{T}} \underline{b}}{\underline{d}_i^{\mathrm{T}} Q \underline{d}_i}. \tag{129}$$

This leads to a natural representation of the minimizer,

$$\underline{x}^* = \sum_{i=0}^{n-1} \left(\frac{\underline{d}_i^{\mathrm{T}} \underline{b}}{\underline{d}_i^{\mathrm{T}} Q \underline{d}_i} \right) \underline{d}_i. \tag{130}$$

The above representation is particularly computationally attractive, as it requires no expensive operations. Further, each coefficient can be computed with respect to the single $\underline{d}_i$ alone. It is clear from this that if a set of Q-conjugate vectors are available, the minimizer is efficiently computable.

The motivation behind 'conjugate direction methods' is the systematic generation of Q-conjugate vectors. The most common algorithm is the conjugate gradient method: it successively generates new Q-conjugate vectors by decomposing the current gradient of f in terms of established Q-conjugate vectors and generating a new $\underline{d}$ vector from what is left - allowing the method to expand the solution for $\underline{x}$ out of the subspace spanned by the previous Q-conjugate vectors.

The Conjugate Gradient Method

- Let $\underline{x}_0$ be an initial guess for $\underline{x}^*$. Let $\underline{g}_0 = \nabla f(\underline{x}_0) = Q\underline{x}_0 - \underline{b}$. Let $\underline{d}_0 = -\underline{g}_0$.
- At time k, for $k < n$ or while $\underline{g}_k \neq 0$:
  - Set $\underline{x}_{k+1} = \underline{x}_k - \left(\frac{\underline{d}_k^{\mathrm{T}} \underline{g}_k}{\underline{d}_k^{\mathrm{T}} Q \underline{d}_k} \right) \underline{d}_k$: stepsize chosen by minimization rule.
  - Set $\underline{g}_{k+1} = \nabla f(\underline{x}_{k+1}) = Q\underline{x}_{k+1} - \underline{b}$.
  - Set
$$\underline{d}_{k+1} = -\underline{g}_{k+1} + \sum_{j=0}^{k} \left(\frac{\underline{d}_j^{\mathrm{T}} Q \underline{g}_{k+1}}{\underline{d}_j^{\mathrm{T}} Q \underline{d}_j} \right) \underline{d}_j. \tag{131}$$
- Terminating the above at time $n_0 \leqslant n$, we have $\underline{x}^* = \underline{x}_{n_0}$.

Simplifications of the above algorithm are discussed in the Example Problems section. Note, the above method can be interpreted as generating a new Q-conjugate direction $\underline{d}_{k+1}$ by taking $-\underline{g}_{k+1}$, 'Q-projecting' it into the space spanned by the $\underline{d}_0, \ldots, \underline{d}_k$, and taking $\underline{d}_{k+1}$ to be whatever is left. We have the following result:

Theorem 9 *For $f(\underline{x}) = (1/2)\underline{x}^T Q \underline{x} - \underline{b}^T \underline{x}$, with symmetric, positive definite Q, the conjugate gradient method produces the minimizer $\underline{x}^*$ exactly in at most n steps.*

Proof. Note that if the method terminates early, with $\underline{g}_k = 0$, a minimizer has been found since $\nabla f(\underline{x}_k) = 0$ and for such f this uniquely identifies the minimizer, $\underline{x}^* = \underline{x}_k$.

In this proof, we will make extensive use of the simplified formula for the recursion, discussed in the Example Problems section,

$$\underline{d}_{k+1} = -\underline{g}_{k+1} + \left(\frac{\underline{g}_{k+1}^{\mathrm{T}} \underline{g}_{k+1}}{\underline{g}_k^{\mathrm{T}} \underline{g}_k} \right) \underline{d}_k. \tag{132}$$

If the method runs for a full n steps, it suffices to show that: i) any non-zero $\underline{d}_{k+1}$ will be Q-conjugate to the prior $\{\underline{d}_0, \ldots, \underline{d}_k\}$, ii) $\underline{d}_k \neq 0$ for $k < n$, and lastly iii) $\underline{x}_n = \underline{x}^*$.

Assume inductively that the first $\underline{d}_0, \ldots, \underline{d}_k$ are non-zero, and Q-conjugate with each other. For any $i \leqslant k$, we have

$$\begin{aligned} \underline{d}_i^{\mathrm{T}} Q \underline{d}_{k+1} &= -\underline{d}_i^{\mathrm{T}} Q \underline{g}_{k+1} + \sum_{j=0}^{k} \left(\frac{\underline{d}_j^{\mathrm{T}} Q \underline{g}_{k+1}}{\underline{d}_j^{\mathrm{T}} Q \underline{d}_j} \right) \underline{d}_i^{\mathrm{T}} Q \underline{d}_j \\ &= -\underline{d}_i^{\mathrm{T}} Q \underline{g}_{k+1} + \left(\frac{\underline{d}_i^{\mathrm{T}} Q \underline{g}_{k+1}}{\underline{d}_i^{\mathrm{T}} Q \underline{d}_i} \right) \underline{d}_i^{\mathrm{T}} Q \underline{d}_i = 0. \end{aligned} \tag{133}$$

This completes the proof of Q-conjugacy.

Consider the case of $\underline{d}_{k+1} = 0$. In such a case, we have that

$$\underline{g}_{k+1} = \left(\frac{\underline{g}_{k+1}^{\mathrm{T}} \underline{g}_{k+1}}{\underline{g}_k^{\mathrm{T}} \underline{g}_k} \right) \underline{d}_k. \tag{134}$$

as well as, from the recursion for $\underline{x}_{k+1}$ in terms of $\underline{x}_k$,

$$\underline{g}_{k+1} = \underline{g}_k - \left(\frac{\underline{d}_k^{\mathrm{T}} \underline{g}_k}{\underline{d}_k^{\mathrm{T}} Q \underline{d}_k} \right) Q \underline{d}_k. \tag{135}$$

Hence,

$$\underline{g}_{k+1}^{\mathrm{T}} \underline{g}_{k+1} = \left(\frac{\underline{g}_{k+1}^{\mathrm{T}} \underline{g}_{k+1}}{\underline{g}_k^{\mathrm{T}} \underline{g}_k} \right) \left(\underline{d}_k^{\mathrm{T}} \underline{g}_k - \left(\frac{\underline{d}_k^{\mathrm{T}} \underline{g}_k}{\underline{d}_k^{\mathrm{T}} Q \underline{d}_k} \right) \underline{d}_k^{\mathrm{T}} Q \underline{d}_k \right) = 0. \tag{136}$$

Hence, if we have $\underline{d}_{k+1} = 0$ at any point prior to termination ($k+1 = n$), we have $\underline{g}_{k+1} = 0$, and a minimizer has been found. If the loop does not terminate early, we have $\underline{d}_k \neq 0$ for all $k < n$, and $\{\underline{d}_0, \ldots, \underline{d}_{n-1}\}$ form a Q-conjugate basis of $\mathbb{R}^n$. It remains to show that $\underline{x}_n = \underline{x}^*$. Note, from the recursion for $\underline{x}_{k+1}$, we have for $k \geqslant 0$ both

$$\underline{x}_{k+1} = \underline{x}_0 - \sum_{j=0}^{k} \left(\frac{\underline{d}_j^{\mathrm{T}} \underline{g}_j}{\underline{d}_j^{\mathrm{T}} Q \underline{d}_j} \right) \underline{d}_j \tag{137}$$

and

$$\underline{g}_{k+1} = \underline{g}_0 - \sum_{j=0}^{k} \left(\frac{\underline{d}_j^{\mathrm{T}} \underline{g}_j}{\underline{d}_j^{\mathrm{T}} Q \underline{d}_j} \right) Q \underline{d}_j. \tag{138}$$

From the latter, we get that for $k \geqslant 0$, $\underline{d}_{k+1}^{\mathrm{T}} \underline{g}_{k+1} = \underline{d}_{k+1}^{\mathrm{T}} \underline{g}_0$ (by Q-conjugacy), and this simplifies the former to

$$\underline{x}_{k+1} = \underline{x}_0 - \sum_{j=0}^{k} \left(\frac{\underline{d}_j^{\mathrm{T}} \underline{g}_0}{\underline{d}_j^{\mathrm{T}} Q \underline{d}_j} \right) \underline{d}_j \tag{139}$$

Recalling that $\underline{g}_0 = Q\underline{x}_0 - \underline{b}$, the above yields,

$$\underline{x}_n = \underline{x}_0 - \sum_{j=0}^{n-1} \left(\frac{\underline{d}_j^{\mathrm{T}} (Q\underline{x}_0 - \underline{b})}{\underline{d}_j^{\mathrm{T}} Q \underline{d}_j} \right) \underline{d}_j = \underline{x}_0 - \sum_{j=0}^{n-1} \left(\frac{\underline{d}_j^{\mathrm{T}} Q \underline{x}_0}{\underline{d}_j^{\mathrm{T}} Q \underline{d}_j} \right) \underline{d}_j + \sum_{j=0}^{n-1} \left(\frac{\underline{d}_j^{\mathrm{T}} \underline{b}}{\underline{d}_j^{\mathrm{T}} Q \underline{d}_j} \right) \underline{d}_j. \tag{140}$$

Lastly, we observe that taking a basis of $\{\underline{d}_i\}$ we may represent $\underline{x}_0$ in terms of it, in which case

$$\underline{x}_0 = \sum_{j=0}^{n-1} \left(\frac{\underline{d}_j^{\mathrm{T}} Q \underline{x}_0}{\underline{d}_j^{\mathrm{T}} Q \underline{d}_j} \right) \underline{d}_j, \tag{141}$$

and

$$\underline{x}_n = \sum_{j=0}^{n-1} \left(\frac{\underline{d}_j^{\mathrm{T}} \underline{b}}{\underline{d}_j^{\mathrm{T}} Q \underline{d}_j} \right) \underline{d}_j = \underline{x}^*. \tag{142}$$

Hence we see that the method terminates in at most n steps, producing an explicit form of the minimizer.

The following theorem gives three important structural characterizations of the conjugate gradient method. These are important for generalizing the method to non-quadratic f.

Theorem 10 *For the sequences (up to termination) of gradients $\{\underline{g}_k\}$, the gradients are mutually orthogonal ($\underline{g}_i^T \underline{g}_j = 0$ for $i \neq j$), and for $i \leqslant k$,*

$$\underline{g}_{k+1}^T \underline{d}_i = 0. \tag{143}$$

Note, the fact that at every time $k+1$, $\underline{g}_{k+1}^{\mathrm{T}} \underline{d}_i = 0$ for each $i \leqslant k$ implies that at each time k, $\underline{x}_{k+1}$ effectively minimizes the function f with respect to the currently available directions $\{\underline{d}_0, \ldots, \underline{d}_k\}$ from the initial point $\underline{x}_0$, i.e., on the manifold

$$\underline{x}_0 + \mathrm{span}\{\underline{d}_0, \ldots, \underline{d}_k\}. \tag{144}$$

Proof. The last of the three claims ($\underline{g}_{k+1}^{\mathrm{T}} \underline{d}_i = 0$ for $i \leqslant k$) may be proven inductively, utilizing the recursive formula

$$\underline{g}_{k+1} = \underline{g}_k - \left(\frac{\underline{d}_k^{\mathrm{T}} \underline{g}_k}{\underline{d}_k^{\mathrm{T}} Q \underline{d}_k} \right) Q \underline{d}_k, \tag{145}$$

which may be recovered from the recursive formula for $\underline{x}_{k+1}$.

It follows from the construction of the directions from the gradients that for any k up to termination, $\underline{g}_k$ is in the span of $\{\underline{d}_0, \ldots, \underline{d}_k\}$. As such, since each $\underline{d}_i$ is orthogonal to $\underline{g}_{k+1}$ for $i \leqslant k$, we see that $\underline{g}_k$ is orthogonal to $\underline{g}_{k+1}$ - and indeed, any $\underline{g}_i$ is orthogonal to $\underline{g}_{k+1}$ for $i \leqslant k$. Hence the sequence of gradients is mutually orthogonal.

8.1 Generalizing to Non-Quadratic f

For quadratic f, we have that the conjugate gradient method solves for the minimizer exactly in at most n steps. In general, we cannot be so lucky, but we may apply the method fruitfully for general f in the following way:

The General Conjugate Gradient Method:

- Let $\underline{x}_0$ be an initial guess for $\underline{x}^*$. Let $\underline{g}_0 = \nabla f(\underline{x}_0)$. Let $\underline{d}_0 = -\underline{g}_0$.
- At time k, for $k < n$ or until *Restart* or *Termination* Conditions are met:
  - Set $\alpha_k = \arg\min_{\alpha > 0} f(\underline{x}_k + \alpha \underline{d}_k)$: stepsize chosen by minimization rule.
  - Set $\underline{x}_{k+1} = \underline{x}_k + \alpha_k \underline{d}_k$.
  - Set $\underline{d}_{k+1} = -\nabla f(\underline{x}_{k+1}) + \beta_k \underline{d}_k$, where

$$\beta_k = \frac{\nabla f(\underline{x}_{k+1})^{\mathrm{T}} \left[\nabla f(\underline{x}_{k+1}) - \nabla f(\underline{x}_k)\right]}{\nabla f(\underline{x}_k)^{\mathrm{T}} \nabla f(\underline{x}_k)}. \tag{146}$$

- When *Restart* condition is met, begin the algorithm again with the current value of $\underline{x}_k$ as the new initial guess.
- When *Termination* condition is met, output the current value of $\underline{x}_k$ as an approximation of $\underline{x}^*$.

In general, appropriate *Termination* conditions are the same as in the discussion of previous algorithms - namely, when the $\nabla f(\underline{x}_k)$ gradients become sufficiently close to 0. The question of *Restart* conditions is more interesting. In general, we should not expect that utilizing the conjugate gradient method for non-quadratic f for more than n iterations should give anything particularly useful as n generated directions will (barring singular cases) span the whole of $\mathbb{R}^n$.

The characterization of the conjugate gradient method as generating orthogonal gradients in the quadratic case provides another possible restart rule, as a measure of the decay of conjugacy. Namely, $\nabla f(\underline{x}_k)^{\mathrm{T}} \nabla f(\underline{x}_{k-1})$ should ideally be 0 always, so restart whenever that quantity is sufficiently far from 0, for instance taking $0 < \delta < 1$,

$$\frac{\nabla f(\underline{x}_k)^{\mathrm{T}} \nabla f(\underline{x}_{k-1})}{||\nabla f(\underline{x}_{k-1})||^2} > \delta, \tag{147}$$

where the denominator is included as a sort of relative scaling factor.

Note: the direction update formula given by $\underline{d}_{k+1} = -\nabla f(\underline{x}_{k+1}) + \beta_k \underline{d}_k$ above is a variant on the equivalent direction update formula for the quadratic case, given in the Example Problems, Eq. (148). In the case of quadratic f, the formula given above and the formula given in Eq. (148) are equivalent, because $\nabla f(\underline{x}_{k+1})^{\mathrm{T}} \nabla f(\underline{x}_k) = 0$. However, the formula given above is generally more reliable and numerically stable the direct analog of Eq. (148).

8.2 Example Problems

1 Show that the iteration in Eq. (131) can be equivalently replaced by the more efficient

$$\underline{d}_{k+1} = -\underline{g}_{k+1} + \left(\frac{\underline{g}_{k+1}^{\mathrm{T}} \underline{g}_{k+1}}{\underline{g}_k^{\mathrm{T}} \underline{g}_k} \right) \underline{d}_k. \tag{148}$$

2 Complete the proof of Theorem 10.

3 Consider an example Q and $\underline{b}$ of (sufficiently large to be interesting) dimension. Implement the conjugate gradient method. How does the sequence of 'projection coefficients' $\{\underline{d}_j^{\mathrm{T}} \underline{b} / \underline{d}_j^{\mathrm{T}} Q \underline{d}_j\}$ for the minimizer behave? How about the 'residual' gradient norm, $||\nabla f(\underline{x}_k)|| = ||Q\underline{x}_k - \underline{b}||$? What does this suggest for potential approximation schemes?

4 Prove that the direction chosen in the general conjugate gradient method is a descent direction, i.e., $\nabla f(\underline{x}_k)^{\mathrm{T}} \underline{d}_k < 0$. *Hint: α is chosen by the minimization rule. What does this imply?*

5 Implement the general conjugate gradient method on a simple example, such as

$$f(x,y) = x^2 + y^2 + x^4 + y^4. \tag{149}$$

How does the added non-quadratic contribution affect the convergence to the minimizer?

9 Additional Descent Methods and Results

In this section, we consider some descent methods and results that are not easily categorized with the other sections, namely momentum or 'heavy-ball' methods, and various treatments of descent with error. This section is essentially 'miscellaneous'.

- ***Momentum Methods:***
  Momentum methods are essentially based on the idea that deciding where to go next based only where you are now is throwing away potentially useful information from the past. As it was previously advantageous to move from $\underline{x}_{k-1}$ to $\underline{x}_k$, it may be advantageous to continue somewhat in that direction at time k - as though momentum from the previous step keeps you moving in that direction. This leads to the iteration:

$$\underline{x}_{k+1} = \underline{x}_k - \alpha_k \nabla f(\underline{x}_k) + \beta_k \left(\underline{x}_k - \underline{x}_{k-1} \right). \tag{150}$$

  Ideally, such an addition can help keep the iterates from being trapped in shallow local minima - the momentum term effectively carries the iterate out of small valleys that would have

otherwise captured a steepest descent algorithm. As is the case for general steepest descent, there remains a question of choosing effective stepsizes and momentum scaling factors. Numerous adaptive algorithms exist, but it is often effective to simply take constant stepsize and constant momentum factors.

- ***Steepest Descent with Bounded Error:***
  Errors frequently arise in computation, from the limits of computational accuracy, to errors introduced by approximation (for instance, approximating gradients or Hessians with finite difference formulae), to errors introduced by noisy data or measurements. An important property of optimization algorithms is robustness to error - that any introduced errors will, within limits, not dramatically affect the final answers.

  Consider steepest descent in an environment when the gradient (the most computationally taxing component of steepest descent) cannot be computed precisely, but there are errors introduced. We take the errors in the computed gradient to be a sequence of vectors $\{\underline{e}_k\}$ that are bounded, i.e., $||\underline{e}_k|| < \delta$ for some positive constant δ. Hence we have the iteration, taking constant stepsize α,

$$\underline{x}_{k+1} = \underline{x}_k - \alpha(\nabla f(\underline{x}_k + \underline{e}_k). \tag{151}$$

  Under such a framework, it seems unreasonable to expect absolute convergence of $\underline{x}_k$ to some minimizer $\underline{x}^*$, since errors may potentially push the iterates away from the minimizer at every step. However, if the errors are not too large, we should expect that this should settle near the minimizer, for instance having the $\underline{x}_k$ oscillating in some basin near $\underline{x}^*$. This is indeed the case. Consider taking $f(\underline{x}) = (1/2)\underline{x}^{\mathrm{T}} Q \underline{x} - \underline{b}^{\mathrm{T}} \underline{x}$, with symmetric, positive definite Q, having largest and smallest eigenvalues $\lambda_{\max}$ and $\lambda_{\min}$ respectively. In that case, we can show that for sufficiently small stepsize α, we have

$$||\underline{x}_k - \underline{x}^*|| \leqslant \frac{\alpha\delta}{1-q} + q^k ||\underline{x}_0 - \underline{x}^*||, \tag{152}$$

  where

$$q = \max\left(|1 - \alpha\lambda_{\min}|, |1 - \alpha\lambda_{\max}|\right) < 1. \tag{153}$$

  Hence we see that in general, the error cannot be reduced below a certain bound $(\alpha\delta)/(1-q)$, but that bound is reduced with smaller stepsizes (thus reducing the impact of the error), or reducing the error itself. Note: compare the value q to the corresponding factor in the previous analysis of steepest descent with constant stepsize.

- ***Steepest Descent with Random Error:***
  In some situations, the function to be optimized cannot be measured directly - attempting to evaluate the function at some $\underline{x}$ may be subject to some random error, for instance as arises in a lot of physical or experimental applications. Let $F(\underline{x}, \underline{w})$ be the result of taking a measurement at $\underline{x}$, subject to random factors $\underline{w}$. In this case, we can consider the 'true' function to optimize to be the average over all errors, i.e.,

$$f(\underline{x}) = \mathbb{E}\left[F(\underline{x}, \underline{w})\right]. \tag{154}$$

  In this case, we are not looking for the $\underline{x}$ that produces the maximal response, but the $\underline{x}$ that produces the maximal *average* response. Note that for a given input $\underline{x}$, we only observe a

random value of F. By evaluating or measuring multiple times at the same input $\underline{x}$, we can attempt to average out the error and measure the true value of $f(\underline{x})$. However, this can be expensive in terms of cost and computation. We would rather make full use of a single noisy evaluation at $\underline{x}$, to the best of our abilities.

Consider then the following method: when evaluating at $\underline{x}_k$, let $\underline{w}_k$ be the corresponding random error. Let $\nabla_{\underline{x}} F$ be the gradient of F with respect to the $\underline{x}$-variables (which is likely to be a function of the random $\underline{w}$-variables. We may construct the iterates

$$\underline{x}_{k+1} = \underline{x}_k - \alpha_k \nabla_x F(\underline{x}_k, \underline{w}_k). \tag{155}$$

Under the condition that the random variables $\{\underline{w}_k\}$ are independent and have finite variance, convergence can be assured under certain constraints on the stepsizes. Namely, if the stepsizes are too large convergence can fail, as the random errors will perpetually push the iterates away from a minimizer. If the stepsizes are too small, however, the iterates can become stuck, converging to a non-significant point $\underline{x}'$. If we have, however,

$$\begin{aligned} \alpha_k &\to 0 \\ \sum_{k=0}^{\infty} \alpha_k &= \infty \\ \sum_{k=0}^{\infty} \alpha_k^2 &< \infty, \end{aligned} \tag{156}$$

Then it can be shown that with probability 1, either $f(\underline{x}_k)$ diverges to $-\infty$ in which case there is no minimizer, or $\nabla f(\underline{x}_k) \to 0$. The conditions above effectively guarantee that the stepsizes become sufficiently small to guarantee convergence, but become small slowly enough to prevent convergence at a non-significant point.

9.1 Example Problems

1 For quadratic f, what is the general performance of steepest descent with momentum for constant stepsize, constant momentum scaling factor? How big or small do these need to be to guarantee convergence, and what is the rate of convergence? *Hint: It will be useful to be able to solve second order linear recurrence relations.*

2 Prove the result in Eq. (152). How might this generalize to non-quadratic f?

3 Implement the steepest descent with random error algorithm in the 1-D case, with $F(x,W) = (x-W)^2$ where W is taken to be a standard normal random variable. What are good choices of stepsize progression? What happens if the stepsizes go to 0 too quickly?

10 Optimization over Data and Stochastic Gradient Descent

A frequent application of optimization is that of model fitting. A model captures some dependence of predicted output on input, as a function of a set of parameters $\underline{p}$,

$$\text{Model}_{\underline{p}}(\text{input}) = \text{predicted output}. \tag{157}$$

Classical examples include linear models, i.e., $\text{Model}_{\underline{p}}(\underline{x}) = \underline{x}^{\mathrm{T}}\underline{p}$, or more complicated models such as neural networks. In any of these cases, we are usually interested in tuning the model to a set of data. In this case, we have a set of input/output pairs representing known measurements or collected data, $(\underline{x}_1, \underline{y}_1), \ldots, (\underline{x}_N, \underline{y}_N)$. A set of parameters for a given model is generally evaluated in terms of how well it predicts or captures each known data point, according to some loss function:

$$\sum_{i=1}^{N} \text{Loss}(\text{Model}_{\underline{p}}(\underline{x}_i), \underline{y}_i). \tag{158}$$

Traditional loss functions include the square error, $\text{Loss}(\tilde{\underline{y}}, \underline{y}) = ||\tilde{\underline{y}} - \underline{y}||^2$, or various entropy or probability measurements in the case of classification problems, such as the probability of misclassification.

Additionally, it is common to include a 'regularization' term $\text{Regularization}(\text{Model}_{\underline{p}})$ - something that penalizes overly complex models, so as to reduce overfitting to the data. This in general will increase the generalizability of your model, i.e., how well it extends to data it has not seen yet. For instance, in fitting polynomials to the data, preventing any of the coefficients from becoming too large generally reduces overfitting. This leads to a common regularization term, for $\gamma > 0$

$$\text{Regularization}(\text{Model}_{\underline{p}}) = \gamma ||\underline{p}||^2, \tag{159}$$

where γ controls how significant the regularization term is, or how costly overfitting is.

Hence, we are frequently in the position of trying to find $\underline{p}$ to minimize functions of the form

$$\sum_{i=1}^{N} \text{Loss}(\text{Model}_{\underline{p}}(\underline{x}_i), \underline{y}_i) + \text{Regularization}(\text{Model}_{\underline{p}}). \tag{160}$$

Hence, in this section we are particularly concerned with optimization problems minimizing functions of the form

$$f(\underline{x}) = \sum_{i=0}^{N} f_i(\underline{x}) \tag{161}$$

where the component $\{f_i\}$ functions may represent for instance how well a given model fits the i-th block or batch of a known data set. Note, if each i represents the contribution of a data block, we may include a regularization term as f_0 for instance, depending on how you want to index your component functions. We are particularly interested in when N is very large. Note, by linearity we have that

$$\nabla f(\underline{x}) = \sum_{i=0}^{N} \nabla f_i(\underline{x}). \tag{162}$$

Depending on the size of N and the form of the individual f_i, it can be a very expensive task to compute the gradient $\nabla f(\underline{x})$, rendering methods such as steepest descent unwieldy. This motivates the method known as stochastic gradient descent.

Stochastic Gradient Descent:

- Let $\underline{x}_0$ be an initial guess for $\underline{x}^*$.

- At time k, until *Termination Condition* is met:
  - Choose $i \in \{0, 1, 2, \ldots, N\}$ uniformly at random.
  - Set $\underline{x}_{k+1} = \underline{x}_k - \alpha_k \nabla f_i(\underline{x}_k)$.

Essentially, stochastic gradient descent is steepest descent applied to a randomly chosen f_i. It is not guaranteed that any single step of SGD will be a descent direction, however it can be shown that the result of each step is a descent *on average*, interpreting the gradient $\nabla f(\underline{x})$ as a probabilistic sum of randomly chosen $\nabla f_i(\underline{x})$. One advantage of this method is that it can frequently carry the iterates out of shallow local minima that they would otherwise become trapped in - a local minimum for f_j may not be a local minimum for a different f_i, and hence there is some probability that the iterates will leave the j-minimum through descent on f_i. Computationally, this is a tremendously efficient method as it avoids the computational burden of computing the $\nabla f(\underline{x})$.

In the usual way, as we have seen with the other methods discussed, it remains to consider how to choose stepsizes, and how to determine an appropriate termination condition. Termination should be considered through a combination of convergence, and practicality, i.e., how many iterations are feasible to compute. Note, because SGD is a randomized algorithm, convergence in this case will be discussed in terms of expected value. In general, convergence results for stochastic gradient descent with optimal stepsize choice take the form

$$\mathbb{E}\left[f(\underline{x}_k)\right] - \inf_{\underline{x}} f(\underline{x}) \leqslant o\left(\frac{1}{k}\right) \tag{163}$$

with the restriction that f be *strongly convex*, i.e., the smallest eigenvalue of the Hessian $\nabla^2 f(\underline{x})$ is bound $\geqslant \varepsilon > 0$ for some ε. This suggests that, in the worst case, doubling the number of iterations should halve the error.

Additionally, convergence results of this form suggests that convergence (and therefore termination) should be considered *in aggregate*. In short, consider running multiple instances of SGD in parallel, generating simultaneous iterates $\{\underline{x}_k^1, \ldots, \underline{x}_k^m\}$, and consider termination when the corresponding function values become sufficiently close to each other.

In general, the problem of choosing stepsize can be difficult. Numerous algorithms for dynamically and adaptively choosing stepsizes exist. However, a constant stepsize (often referred to as the *learning rate*) generally produces very good results. Note, for constant stepsize, we should not expect convergence for a given sample path if the stepsize is too large - in the usual way, it is reasonable to expect oscillation around the minimum when approaching from certain directions, due to overshoot. However, this is not necessarily a bad thing: computing the exact minimum may result in overfitting of the model to the data, hence inexact minimizers may ultimately more accurately (or at least, flexibly) model future real data.

The following extension of SGD is often utilized in practice, adding a momentum term as in the method of Section 9 (see further discussion there). ***Stochastic Gradient Descent with Momentum:***

- Let $\underline{x}_0$ be an initial guess for $\underline{x}^*$.
- At time k, until *Termination Condition* is met:

- Choose $i \in \{0, 1, 2, \ldots, N\}$ uniformly at random.
- Set $\underline{x}_{k+1} = \underline{x}_k - \alpha_k \nabla f_i(\underline{x}_k) + \beta_k(\underline{x}_k - \underline{x}_{k-1})$.

Applying a momentum term in the case of stochastic gradient descent as opposed to pure steepest descent has an additional advantage. In SGD, in every iteration the algorithm steps only according to information based on the currently chosen f_i. By including a momentum term, SGD-with-Momentum imports information about previous steps, and therefore previously chosen f_j into the current step. Hence, every iteration of SGD-with-Momentum steps based on more information about the total function f than in SGD by itself.

Again, there remains the problem of not only choosing suitable stepsizes, but now suitable momentum scaling factors as well. Again, in practice both are generally taken to be constant, though the specific constants may depend on the specific problem being addressed. As ever, experimentation can be informative.

10.1 Optimizing Over Data Over Time - A Model of Online Learning

In the previous model fitting formulation, we have

$$\sum_{i=1}^{N} \text{Loss}(\text{Model}_{\underline{p}}(\underline{x}_i), \underline{y}_i) + \text{Regularization}(\text{Model}_{\underline{p}}). \tag{164}$$

Note that in this formulation, each data point contributes to the over-all error in the same way - data point 1 is as important in its contribution as data point N. However, it may be the case that more recently collected data points are more significant or salient to the model. For instance, if there is an assumption that the underlying environment can change (such as in natural environments, or economic factors), more recent data may more accurately capture the current state of the world, and should be viewed as therefore more important.

We can capture this mathematically by weighting the contributions of each data point to the total error. Consider a sequence of weights $\omega_0 \geqslant \omega_1 \geqslant \omega_2 \geqslant \ldots \geqslant 0$. At time T, suppose data points $(\underline{x}_1, \underline{y}_1), \ldots, (\underline{x}_T, \underline{y}_T)$ have been collected, sequentially. We may express the total error as

$$\text{Error}_T(\underline{p}) = \sum_{t=1}^{T} \omega_{T-t} \text{Loss}(\text{Model}_{\underline{p}}(\underline{x}_t), \underline{y}_t) + \text{Regularization}(\text{Model}_{\underline{p}}), \tag{165}$$

or consider general optimization problems

$$F_T(\underline{x}) = \sum_{t=1}^{T} \omega_{T-t} f_t(\underline{x}) + f_0(\underline{x}). \tag{166}$$

In both these models, contributions from earlier data points or component functions are weighted away toward 0 as they retreat into the past. Common weighting functions include $\omega_k = \lambda^k$ for $0 < \lambda < 1$ and $\omega_k = 1/(k+1)$. Note that in some applications, for sufficiently old data points or component functions, they may be weighted sufficiently close to 0 that they can effectively be dropped or truncated, taking $\omega_k = 0$ for $k \geqslant K$ for some K.

This formulation can be especially useful when optimization will have to be performed repeatedly, over time, for instance if financial policy needed to be re-optimized every two months, based on newly available data. In this case, the previous solution $\underline{x}_T^*$ may be recycled as an initial guess for the new optimum $\underline{x}_{T+1}^*$ - if the underlying system is not changing overly much over time, the optimal solutions should not change too much either.

The choice of $\omega_k = \lambda^k$ for $0 < \lambda < 1$ has a particularly nice structure in this dynamic optimization case, as we can make use of the following relation, that

$$F_{T+1}(\underline{x}) = \lambda F_T(\underline{x}) + (1-\lambda) f_R(\underline{x}). \tag{167}$$

In general, for particularly large T and general weighting $\{\omega_k\}$, we can consider the following generalization of Stochastic Gradient Descent (we ignore the regularization term for now):

Dynamic Stochastic Gradient Descent over Time:

- Let $\underline{x}_0$ be an initial guess for $\underline{x}_{T_0}^*$ for T_0 initial data points.
- While there is no new data:
  - Choose $t \in \{1, 2, \ldots, T\}$ uniformly at random.
  - Set $\underline{x}_{k+1} = \underline{x}_k - \alpha_k \omega_{T-t} \nabla f_t(\underline{x}_k)$.
- Define $f_{T+1}(\underline{x})$ in terms of the newly collected data.
- Update $T := T + 1$ and restart (recycling the current iterate $\underline{x}_k$ as the new initial guess).

10.2 Example Problems

1 Consider a set of symmetric, positive definite matrices $\{Q_1, \ldots, Q_N\}$ and vectors $\{\underline{b}_1, \ldots, \underline{b}_N\}$. Taking $f_i(\underline{x}) = (1/2)\underline{x}^{\mathrm{T}} Q_i \underline{x} - \underline{b}_i^{\mathrm{T}} \underline{x}$, define

$$f(\underline{x}) = \sum_{i=1}^{N} f_i(\underline{x}). \tag{168}$$

Implement stochastic gradient descent with constant stepsize, and characterize its convergence.

2 Extend the implementation of SGD for the previous problem to allow a momentum term. How does SGD compare to SGD-with-Momentum?

A Mathematical Background

Eigenvalues and Eigenvectors:

Given a square matrix A, an eigenvalue / eigenvector pair $(\underline{v}, \lambda)$ is a vector and scalar such that $A\underline{v} = \lambda \underline{v}$. That is, the action of A on the vector $\underline{v}$ is to stretch it by a factor of λ. An n x n matrix

has at most n eigenvalues and n non-zero eigenvectors. Note, while the eigenvalues of a matrix are uniquely determined, the eigenvectors of a matrix are unique only up to scale, i.e, if $\underline{v}$ is an eigenvector of A, then $\alpha\underline{v}$ is also an eigenvector of A, with the same eigenvalue.

The eigenvalues of a matrix can be solved for as the roots $\{\lambda\}$ of the equation $\det(A-\lambda \mathrm{I}) = 0$. For any eigenvalue λ, the corresponding eigenvector(s) can be solved for as the solution to the equation $(A-\lambda \mathrm{I})\underline{v} = 0$. The following are a number of important results and definitions about eigenvalues and vectors.

- If A is symmetric and real valued, then A has n real-valued eigenvalues and n linearly independent eigenvectors.
- If A is *positive definite*, then all eigenvalues of A are positive. If A is *positive semi-definite*, then all eigenvalues of A are non-negative.
- If the eigenvalues of A are real, with $\lambda_{\min}$ the smallest eigenvalue and $\lambda_{\max}$ the largest eigenvalue, then the following inequality holds for all vectors $\underline{x}$:

$$\lambda_{\min}||\underline{x}||^2 \leqslant \underline{x}^{\mathrm{T}} A \underline{x} \leqslant \lambda_{\max}||\underline{x}||^2, \tag{169}$$

  with equality holding for $\underline{x}$ corresponding only vectors parallel to the eigenvectors corresponding to the smallest and largest eigenvalues, respectively.
- If the eigenvalues of A are real, with $\lambda_{\min}$ the smallest eigenvalue and $\lambda_{\max}$ the largest eigenvalue, then the following inequality holds for all vectors $\underline{x}$:

$$||A\underline{x}|| \leqslant \max\left(|\lambda_{\min}|, |\lambda_{\max}|\right)||x||. \tag{170}$$

Note, the last two results essentially bound the action of A on any vector by dilation by the eigenvalues.

B Line Search Methods

B.1 Golden Mean Search

The Golden Mean line search method is a fairly straightforward algorithm with a linear (geometric) convergence rate, that importantly (and unlike the other methods discussed) *does not rely on the derivative*.

Suppose that a minimum of the function $g : \mathbb{R} \mapsto \mathbb{R}$ is known to lie somewhere on the interval $\alpha \in [a,b]$. An important assumption of this method (though it is interesting to consider the method without it) is that the function g is *unimodal* over this interval, i.e, we need only be concerned with the existence of a single minimum.

Systematically, we will generate two test points a', b' such that $a < a' < b' < b$. If g is unimodal over the interval, and $g(a') < g(b')$, then the minimum must fall in the interval $[a,b']$. If $g(a') > g(b')$, then the minimum must fall in the interval $[a',b]$. In either case, we may reduce the initial interval

to a smaller interval, and the process may be repeated until the resulting containing interval is sufficiently small, i.e., $[a_k, b_k]$ such that $b_k - a_k < \varepsilon$.

Note, under the assumption of unimodality, if $g(a') = g(b')$ with $a' < b'$, then g must be constant over the interval $[a', b']$. In this case, depending on cases, the initial interval may be replaced with $[a, a']$ or $[b', b]$. This case can frequently be ruled out (for instance, if the second derivative is non-zero over the region of interest).

Consider generating a' and b' in the following way: for $0 < \delta < 1$,

$$\begin{aligned} a' &= a + \delta(b-a) \\ b' &= b - \delta(b-a). \end{aligned} \tag{171}$$

In this case, the new interval (either $[a, b']$ or $[a', b]$) will have width $(b-a)(1-\delta)$. We see then that the width of the containing interval shrinks by a factor of $0 < 1 - \delta < 1$ each iteration, achieving linear or geometric convergence to the minimum.

Is there an optimal choice of δ? We need $a' < b'$, which requires that $\delta < 1/2$. Additionally, however, note that if the new interval is $[a, b']$, then the new interval will contain the prior test point a' (similarly, $[a', b]$ contains the prior test point b'). Since the method requires evaluation of the test points $g(a'), g(b')$, if the test points in the current interval could be made to be the test points in the new interval, we could recycle the evaluation of g at the test points, and cut down on the number of required evaluations. Computation will make this more clear.

Suppose that the new interval is $[a, b']$. In this case, we have test points:

$$\begin{aligned} a'' &= a + \delta(b' - a) = a + \delta(1-\delta)(b-a) \\ b'' &= b' - \delta(b' - a) = b - \delta(2-\delta)(b-a). \end{aligned} \tag{172}$$

At this point, we need to evaluate $g(a'')$ and $g(b'')$. If one of these points were a', however, the evaluation of $g(a')$ has already been performed, and hence we could halve the number of evaluations of g at this point. It is impossible to have $a'' = a'$ for δ on this interval, but taking $b'' = a'$, $1 - \delta(2 - \delta) = \delta$. Solving for feasible δ we have

$$\delta^* = \frac{3}{2} - \frac{\sqrt{5}}{2} \approx 0.381967\ldots. \tag{173}$$

It can be shown similarly that if the new interval is $[a', b]$, then taking δ^* as above renders $a'' = b'$, again saving an evaluation of g.

Hence, taking the optimal value of δ as above, we have that the interval at time $[a_k, b_k]$ at time k satisfies

$$b_k - a_k = (b_0 - a_0)(1 - \delta^*)^k = (b_0 - a_0)\left(-\frac{1}{2} + \frac{\sqrt{5}}{2}\right)^k \approx (b_0 - a_0)(0.61803\ldots)^k. \tag{174}$$

Note, the above contraction factor may also be expressed as $1/\phi$ where $\phi = (1 + \sqrt{5})/2$ is the so called Golden Ratio, thus giving the method its name.

Interesting Note: We might consider the following variant on the above strategy. If we have access to the derivative of g, i.e., can evaluate $g'(\alpha)$ efficiently, we can can achieve a linear or geometric convergence that halves the containing interval, through the use of a binary search. Suppose that we have identified an interval $[a,b]$ such that $g'(a) < 0$ and $g'(b) > 0$, and again we assume unimodality over the interval, so that there is a unique minimum. To identify the minimum, it suffices to identify a point α^* such that $g'(\alpha^*) = 0$. We can consider the test point $\alpha' = (a+b)/2$. If $g'(\alpha') < 0$, we reduce the interval to $[\alpha', b]$. If $g'(\alpha') > 0$, we reduce the interval to $[a, \alpha']$. Iterating this reduces the containing interval by half the width each iteration.

This is in some sense more efficient than the Golden Mean search above, yielding a contraction factor of 0.5 compared to ~ 0.618. However, it requires being able to calculate, or at least estimate, the derivative of g, which may not be easily accessible. One of the true advantages of the Golden Mean method is that it relies only on evaluations of g.

B.2 Cubic Interpolation

We consider again the problem of locating a minimum of $g : \mathbb{R} \mapsto \mathbb{R}$. Suppose that a three point pattern a, b, c has been found such that $a < c < b$ and $g(a) > g(c), g(c) < g(b)$, so that the interval $[a,b]$ contains a minimum of g. We will derive a narrower three point pattern a', b', c' that also contains a minimum of g. Iterating this process, the containing interval will steadily shrink until sufficient bounds on the minimum have been established, lying in the interval $[a_k, b_k]$ with $b_k - a_k < \varepsilon$.

In this case, we use cubic interpolation to approximate g, then use the minimum of this approximation to construct a new three point pattern. Given the values $g(a), g'(a), g(b), g'(b)$, we may fit a cubic polynomial to them as

$$\begin{aligned}
\tilde{g}(\alpha) &= A(\alpha - a) + B(\alpha - b) + D(\alpha - a)^2(\alpha - b) + E(\alpha - a)(\alpha - b)^2 \\
A &= \frac{g(b)}{b-a} \\
B &= \frac{g(a)}{a-b} \\
D &= \frac{1}{(b-a)^2}\left(g'(b) - \frac{g(b)-g(a)}{b-a}\right) \\
E &= \frac{1}{(b-a)^2}\left(g'(a) - \frac{g(b)-g(a)}{b-a}\right).
\end{aligned} \tag{175}$$

Note that the derivative will (in general) have two roots, one corresponding to a local maximum and one corresponding to a local minimum. In general, we have that the optima of the cubic polynomial above are given by

$$\alpha'_{\pm} = \frac{(2a+b)D + (a+2b)E \pm \sqrt{(b-a)^2(D^2 + ED + E^2) - 3(A+B)(D+E)}}{3(D+E)}. \tag{176}$$

Note, if $D + E > 0$, then the minimum is given by the larger root. If $D + E < 0$, the minimum is given by the smaller root. Hence, in both case we have that the local minimum is given by $\alpha' = \alpha'_+$.

If $D+E=0$, however, note that the cubic approximation is in fact quadratic, and that may be minimized separately.

In either case, if $g(\alpha') > g(c)$, then the three point pattern may be replaced by the narrower $a < c < \alpha'$ or $\alpha' < c < b$ (depending on if $\alpha' < c$ or $\alpha' > c$). If $g(\alpha') < g(c)$, then the three point pattern may be replaced by $a < \alpha' < c$ or $c < \alpha' < b$ similarly. Iterating this process successively narrows down the interval of containment.

B.3 Error Rates

In previous sections, we have considered algorithms for computing the minimum of a function $g : \mathbb{R} \mapsto \mathbb{R}$. The premise of these methods was that, beginning with some interval $[a,b]$ known to contain the minimum, we might approximate g by simple polynomials and based on minimizing the approximation, generate a new, smaller interval $[a',b']$ which contained the minimum. Iterating this process would eventually reach a point of sufficient accuracy, i.e., $[a_k,b_k]$ such that $b_k - a_k < \varepsilon$ for some sufficiently small error bound ε. In this section, we consider the rate at which convergence occurs.

In what follows, we will assume that there is a unique minimum α^* of g in the interval $[a,b]$, and that $b-a$ is 'sufficiently small'. It is convenient to express the upper and lower bounds in terms of small errors from the minimum, i.e., $a = \alpha^* - \varepsilon_a$ for some $\varepsilon_a > 0$, $b = \alpha^* + \varepsilon_b$ for $\varepsilon_b > 0$. In the case that a value $c \in [a,b]$ is required with $g(a) > g(c)$, $g(c) < g(b)$, we take $c = \alpha^* + \varepsilon_c$ where ε_c is taken to be positive or negative.

B.3.1 Quadratic Fits

Most of the considered algorithms involved fitting parabolas to approximate g over the interval $[a,b]$. Parabolas are a useful choice, as their minimizers may be computed exactly as simple functions of the coefficients.

Note that if g is a quadratic function, than any quadratic fit to g will be exact, and hence the subsequent minimization will be exact, and the algorithm terminates in one step. We may therefore effectively take g to be a cubic polynomial, on the assumption that the interval in question is sufficiently small that the 4-th order variation vanishes. Hence, at least in approximation, we take the series expansion

$$g(\alpha) \approx g(\alpha^*) + \frac{1}{2}(\alpha-\alpha^*)^2 g''(\alpha^*) + \frac{1}{6} g'''(\alpha^*)(\alpha-\alpha^*)^3. \tag{177}$$

Note, the first order term $(\alpha-\alpha^*)g'(\alpha^*)$ vanishes, as α^* is a minimum, and thus $g'(\alpha^*) = 0$.

- **Fit to:** $g(a), g'(a), g(b), g'(b)$

  We consider the case of $g'(b) < g'(a)$, and fit a parabola to $g(a), g'(a), g'(b)$. In the case that $g'(b) > g'(a)$, a similar analysis may be performed fitting to $g(b), g'(b), g'(a)$.

In this case, in Section 5.2, we derived the quadratic fit in Eq. (76), which yielded an approximate minimizer of

$$\alpha' = \frac{ag'(b) - bg'(a)}{g'(b) - g'(a)}. \tag{178}$$

Utilizing the above expansion for g, we have

$$\begin{aligned}
\alpha' &= \frac{(\alpha^* - \varepsilon_a)g'(\alpha^* + \varepsilon_b) - (\alpha^* + \varepsilon_b)g'(\alpha^* - \varepsilon_a)}{g'(\alpha^* + \varepsilon_b) - g'(\alpha^* - \varepsilon_a)} \\
&\approx \frac{(\alpha^* - \varepsilon_a)\left(\varepsilon_b g''(\alpha^*) + \frac{1}{2}g'''(\alpha^*)\varepsilon_b^2\right) - (\alpha^* + \varepsilon_b)\left(-\varepsilon_a g''(\alpha^*) + \frac{1}{2}g'''(\alpha^*)\varepsilon_a^2\right)}{\left(\varepsilon_b g''(\alpha^*) + \frac{1}{2}g'''(\alpha^*)\varepsilon_b^2\right) - \left(-\varepsilon_a g''(\alpha^*) + \frac{1}{2}g'''(\alpha^*)\varepsilon_a^2\right)} \\
&= \alpha^* - \frac{g'''(\alpha^*)\varepsilon_a\varepsilon_b}{2g''(\alpha^*) + g'''(\alpha^*)(\varepsilon_b - \varepsilon_a)}.
\end{aligned} \tag{179}$$

Note that in the limit, the denominator of the above will be dominated by $2g''(\alpha^*)$ when $g''(\alpha^*) > 0$. Hence we see that in the limit, the error on the new point α' will be on the order of $(1/2)(g'''(\alpha^*)/g''(\alpha^*)\varepsilon_a\varepsilon_b$.

- **Fit to:** $g(a), g(b), g(c)$

In this case, in Section 5.2, we derived the quadratic fit in Eq. (78) which yielded an approximate minimizer of

$$\alpha' = \frac{1}{2}\frac{g(a)(c^2 - b^2) + g(b)(a^2 - c^2) + g(c)(b^2 - a^2)}{g(a)(c - b) + g(b)(a - c) + g(c)(b - a)}. \tag{180}$$

A similar analysis to the above yields

$$\alpha' \approx \alpha^* - \frac{(\varepsilon_a\varepsilon_b - \varepsilon_c(\varepsilon_b - \varepsilon_a))g'''(\alpha^*)}{2(3g''(\alpha^*) + (\varepsilon + \varepsilon_b - \varepsilon_a)g'''(\alpha^*)}. \tag{181}$$

Again, in the limit, the denominator will be dominated by the $g''(\alpha^*)$ term when $g''(\alpha^*) > 0$, hence in the limit the error on α' goes like $(1/6)g'''(\alpha^*)/g''(\alpha^*)(\varepsilon_a\varepsilon_b - \varepsilon_c(\varepsilon_b - \varepsilon_a))$.

- **Fit to:** $g(a), g'(a), g''(a)$

In this case of Newton's method, we have from Section 5.2 that

$$\alpha' = a - \frac{g'(a)}{g''(a)}. \tag{182}$$

Taking the same kind of expansion as above, we have

$$\alpha' \approx \alpha^* + \frac{\varepsilon_a^2 g'''(\alpha^*))}{2g''(\alpha^*) - 2\varepsilon_a g'''(\alpha^*)}. \tag{183}$$

Again, in the limit, the denominator is dominated by $g''(\alpha^*)$ when $g''(\alpha^*) > 0$. Hence we have that in the limit, the error on the new point α' goes like $(1/2)g'''(\alpha^*)/g''(\alpha^*)\varepsilon_a^2$.

B.3.2 Cubic Fits

In this subsection, we consider the error associated with the cubic approximation of g of the previous section, fitting a cubic polynomial to $g(a), g(b), g'(a), g'(b)$.

Similar to the previous section, if g is actually a cubic polynomial, the fit is exact. Hence we may assume that g has non-trivial 4-th order variation. In fact, if the interval $[a,b]$ is sufficiently small, we may simply take g as a quartic polynomial (as the higher order terms will effectively vanish around the minimum. Hence we take

$$g(\alpha) \approx g(\alpha^*) + \frac{1}{2}(\alpha-\alpha^*)^2 g''(\alpha^*) + \frac{1}{6}(\alpha-\alpha^*)^3 g'''(\alpha^*) + \frac{1}{24}(\alpha-\alpha^*)^4 g''''(\alpha^*). \tag{184}$$

Taking A, B, D, E as defined in Eq. (175), we have

$$\alpha' = \frac{(2a+b)D + (a+2b)E + \sqrt{(b-a)^2(D^2+ED+E^2) - 3(A+B)(D+E)}}{3(D+E)}. \tag{185}$$

As in the previous sections, approximating A, B, D, E with $a = \alpha^* - \varepsilon_a$ and $b = \alpha^* + \varepsilon_b$ yields

$$(b-a)^2(D^2+ED+E^2) - 3(A+B)(D+E) \approx \frac{1}{4} g''(\alpha^*)^2. \tag{186}$$

The higher order terms (in terms of powers of $\varepsilon_a, \varepsilon_b$) are dropped as non-contributing / dominated by the $1/4 g''(\alpha^*)^2$ term in the limit.

As α^* is a minimizer, taking $g''(\alpha^*) > 0$, this gives

$$\begin{aligned} \alpha' &\approx \frac{(2a+b)D + (a+2b)E + \frac{1}{2} g''(\alpha^*)}{3(D+E)} \\ &= \alpha^* + \frac{(\varepsilon_a^2 - 4\varepsilon_a\varepsilon_b + \varepsilon_b^2) g''''(\alpha^*)}{6(2g'''(\alpha^*) + (\varepsilon_b - \varepsilon_a) g''''(\alpha^*)}. \end{aligned} \tag{187}$$

In the limit, the denominator is dominated by the $g'''(\alpha^*)$ term. If $g'''(\alpha^*) \neq 0$ then, we have in the limit that the error on α' is on the order of $(1/12) g''''(\alpha^*)/g'''(\alpha^*)(\varepsilon_a^2 - 4\varepsilon_a\varepsilon_b + \varepsilon_b^2)$.