Chapter 1

Overview of Numerical Optimisation

This chapter provides a brief overview of the ideas in optimisation used throughout this thesis. [24] is a standard and accesible text on the subject.

First some notation, for the sake of brevity and legibility we will let f_n denote the value of the objective function at $\mathbf{x}_n$, $\mathbf{g}_n$ denote the gradient of the objective function evaluated at the nth iterate $\mathbf{x}_n$, and $\mathbf{H}_n$ denote the Hessian matrix at the nth iteration. In other words:

$$\begin{aligned} f_n &= f(\mathbf{x}_n) \\ \mathbf{g_n} &= \nabla f(\mathbf{x}_n) \\ \mathbf{H}_n &= \nabla\nabla^\top f(x_n) \end{aligned}$$

1.1 Rates Of Convergence[1]

Throughout this thesis, it will sometimes be desireable to compare the rates of convergence of different methods. Suppose there is a vector-valued sequence $\mathbf{x}_n$ that converges to a value $\mathbf{x}^*$.

Linear Convergence: A convergent sequence is said to *converge linearly*[2] to $\mathbf{x}^*$ if there is a $0 < \mu < 1$ such that:

$$\lim_{n\to\infty} \frac{\|\mathbf{x}_{n+1} - \mathbf{x}^*\|}{\|\mathbf{x}_n - \mathbf{x}^*\|} = \mu \tag{1.1}$$

[1] There are slightly different definitions of convegrgence rates from text to text, but all capture the same basic meaning. Refer to [7, 24].

[2] In [24], the case $\mu = 0$ is considered to be a case of linear convergence as well. This definition makes it harder to sharply discriminate between linear and superlinear convergence.

Convergence Class	Example	Iterations until $< 10^{-6}$	Iterations until $< 10^{-12}$
Sublinear	$x_n = \frac{1}{n}$	$10^6 + 1$	$10^{12} + 1$
Linear	$x_n = 2^{-n}$	20	40
Superlinear	$x_n = 2^{-2^n}$	5	6

Table 1.1: Illustrating the different classes of convergence.

If a sequence $\mathbf{x}_n$ converges linearly with constant μ, then $\|\mathbf{x}_{n+1} - \mathbf{x}^*\| \approx \mu\|\mathbf{x}_n - \mathbf{x}^*\|$ for n sufficiently large. A simple example of a linearly converging sequence is given by $1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \ldots$ If plotted on a log scale, the $\|\mathbf{x}_{n+1} - \mathbf{x}^*\|$ terms will tend to lie on a straight line

Sublinear Convergence: A sequence is said to converge *sublinearly* to $\mathbf{x}^*$ if:

$$\lim_{n\to\infty} \frac{\|\mathbf{x}_{n+1} - \mathbf{x}^*\|}{\|\mathbf{x}_n - \mathbf{x}^*\|} = 1$$

Sublinear convergence is very slow. Every reduction in the order of magnitude of the error achieved takes more iterations than the previous reduction. The ur-example of a sublinearly convergent sequence is the reciprocals of the natural numbers: $1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \ldots$.

Superlinear Convergence: A sequence is said to converge superlinearly if:

$$\lim_{n\to\infty} \frac{\|\mathbf{x}_{n+1} - \mathbf{x}^*\|}{\|\mathbf{x}_n - \mathbf{x}^*\|} = 0$$

Usually, one can be more specific about the rate. A sequence is said to converge superlinearly with order p if there exist positive constants $p > 1$ and $\mu > 0$ such that:

$$\lim_{n\to\infty} \frac{\|\mathbf{x}_{n+1} - \mathbf{x}^*\|}{\|\mathbf{x}_n - \mathbf{x}^*\|^p} = \mu \tag{1.2}$$

Note that there is no requirement that $\mu < 1$ in this case. If $p = 2$, the sequence is said to converge quadratically.Taking logs yields that $\log(\|\mathbf{x}_{n+1} - \mathbf{x}^*\|) \approx -p\log(\|\mathbf{x}_n - \mathbf{x}^*\|)$. For a linear sequence, the magnitude of the error declines exponentially, whereas the *order of magnitude of the error* declines exponentially for a superlinearly convergent sequence. For a quadratically converging sequence, each iteration tends to roughly double the number of digits of precision. For example, if the error in the first iterate is approximately 0.1, the next iterate will have error on the order of 10^{-2}, the next will have error on the order of 10^{-4}, and so on. An example of superlinear convergence is given by the sequence $x_n = 2^{-2^n}$.

An Extended Definition of Convergence Rates: The above appraoch to defining the rate of convergence can't handle every sequence however. For example, the sequence $1, 1, \frac{1}{2}, \frac{1}{2}, \frac{1}{4}, \frac{1}{4}, \ldots$ does not converge linearly in the sense of (1.2). To cover

these situations, a sequence is also said to converge linearly/sublinearly/superlinearly if there is an associated auxillary sequence ϵ_n such that $\|\mathbf{x}_{n+1} - \mathbf{x}^*\| \leq \epsilon_n$ for all $n \geq 0$, and the sequence ϵ_n converges linearly/sublinearly/superlinearly to zero.[3]

Linear Convergence and Iterated Mappings: Nearly all estimation algorithms used in statistics start with an initial estimate θ_0 and generate a sequence of estimates by $\theta_{n+1} = \mathbf{M}(\theta_n)$ for some mapping $\mathbf{M}(\cdot)$. The alogrithm is stopped when the generated sequence has converged within a tolerance of the limit θ^*. Examples include the Newton-Raphson Method, Fisher's Method of Scoring, Gradient Descent, the EM Algorithm, Block Relaxation, and many imputation methods. As shall be seen, a statistically motiviated fitting alogorithm will nearly always converge linearly unless it has been specifically engineered so that $\mathbf{M}'(\theta^*) = 0$.

Linear convergence is common for convergent sequences defined by repeatedly applying a function $\mathbf{f}$ so that $\mathbf{x}_{n+1} = \mathbf{f}(\mathbf{x}_n)$. To see this, perform a Taylor expansion about the limit point $\mathbf{x}^*$:

$$\begin{aligned}
\mathbf{f}(\mathbf{x}_n) &\approx \mathbf{f}(\mathbf{x}^*) + \mathbf{f}'(\mathbf{x}^*)(\mathbf{x}_n - \mathbf{x}^*) \\
\mathbf{f}(\mathbf{x}_n) &\approx \mathbf{x}^* + \mathbf{f}'(\mathbf{x}^*)(\mathbf{x}_n - \mathbf{x}^*) \\
\mathbf{f}(\mathbf{x}_n) - \mathbf{x}^* &\approx \mathbf{f}'(\mathbf{x}^*)(\mathbf{x}_n - \mathbf{x}^*) \\
\mathbf{x}_{n+1} - \mathbf{x}^* &\approx \mathbf{f}'(\mathbf{x}^*)(\mathbf{x}_n - \mathbf{x}^*)
\end{aligned}$$

Taking norms of both sides yields that:

$$\|\mathbf{x}_{n+1} - \mathbf{x}^*\| \lesssim \|\mathbf{f}'(\mathbf{x}^*)\| \|\mathbf{x}_{n+1} - \mathbf{x}^*\|$$

The situaion here is a little subtle because $\mathbf{f}$ is a multivariate function. The exact rate is of convergence is controlled by the norm of the Jacobian matrix $\mathbf{f}'(\mathbf{x})$ at $\mathbf{x}^*$. So long as there is a matrix norm such that $\|f'(\mathbf{x}^*)\| < 1$ the sequence will converge linearly at worst, though faster than linear convergence is potentially possible if 0 is an eigenvalue of $\mathbf{f}'(\mathbf{x}^*)$.[4] If $\mathbf{f}'(\mathbf{x}^*) = \mathbf{0}$, the convergence will be superlinear.

More sophisticated optimisation algorithms such as the Newton-Raphson method are specifically designed so that $\mathbf{f}'(\mathbf{x}^*) = 0$, so that the convergence is superlinear.

1.2 Gradient Descent and the Newton-Raphson Method

The simplest derivative-based optimisation algorithm is known as *Gradient Descent*:

[3]The simple definition presented here is known as *Q-Convergence*, and the extended definition is known as *R-Convergence* [24].

[4]Consider for example the multivariate sequence defined by $(x_{n+1}, y_{n+1}) = (x_n^2, y_n/2)$. This convergence towards zero is superlinear in the x direction, but only linear in the y direction. If $(x_0, y_0) = (0.5, 0)$, then the convergence will be superlinear, but the y component will generally drag the convergence rate down to linear convergence.

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \alpha \mathbf{g}_n$$

The fixed parameter $\alpha > 0$ controls how big a step the method will take on each iteration. Gradient descent has the property that the directions it generates will always point 'downhill' so that a small step will decrease the objective function:

$$\begin{aligned} f(\mathbf{x}_{n+1}) &= f(\mathbf{x}_n - \alpha \mathbf{g}_n) \\ &\approx f(\mathbf{x}_n) - \alpha \mathbf{g}_n^\top \mathbf{g}_n \\ &= f(\mathbf{x}_n) - \alpha \|\mathbf{g}_n\|^2 \end{aligned}$$

This means that $f(\mathbf{x}_{n+1}) < f(\mathbf{x}_n)$ so long as α isn't too big. Gradient descent is simple but it is slow, woefully slow sometimes[7, 24]. If the objective function isn't sufficiently well behaved, gradient descent can even converge sublinearly [38].

What consitutes an ill-behaved function might not be as obvious as it might seem. Consider the question of minimising the function $f(x) = x^4$ using gradient descent, starting at $x = 0.5$ with $\alpha = 0.2$. The minimum of $f(x)$ is of course at $x = 0$, so the absolute value of the iterates x_n is a measure of the error. Figure 1.1 plots the log errors for the first 20,000 iterations, It is readily apparent that the alogrithm is converging sublinearly and that the rate of convergence is absolutely and utterly woeful. It takes around 250 iterations before the error falls below 10^{-3}, around 1900 iterations before the error falls below 10^{-4}, and 13800 iterations before the error falls below 10^{-5}. The data suggests that the number of iterations needed to reduce the error to 10^{-n-1} is approximately 7.4 times the number of iterations needed to achieve an error of 10^{-n}. So it could take over 100,000 iterations to get an error of less than 10^{-6}.

Gradient descent can be thought of as the naive choice if one only has access to $f(\mathbf{x})$ and $\nabla f(\mathbf{x}$. Suppose if second derivatives were available as well. What would the naive choice be in this case? Perform a second-order Taylor expansion of the objective function around $\mathbf{x}_n$:

$$f(\mathbf{x}) \approx f_n + \mathbf{g}_n^\top (\mathbf{x} - \mathbf{x}_n) + \frac{1}{2}(\mathbf{x} - \mathbf{x}_n)^\top \mathbf{H}_n (\mathbf{x} - \mathbf{x}_n)$$

The expression on the lefthand side is minimised by $\mathbf{x} = \mathbf{x}_n - \mathbf{H}_n^{-1}\mathbf{g}_n$. Given the iterate $\mathbf{x}_n$, the *Newton-Raphson* method defines the next iterate by:

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mathbf{H}_n^{-1}\mathbf{g}_n$$

The Newton-Raphson Method generally converges quadratically so long as one is already near the optimal point.[24] Its biggest weaknesses are the cost of constantly computing the Hessians, the possibilty that $-\mathbf{H}_n^{-1}\mathbf{g}_n$ fails to be a descent direction, and the possibilty of poor performance or even divergence if one is far from the optimal value.

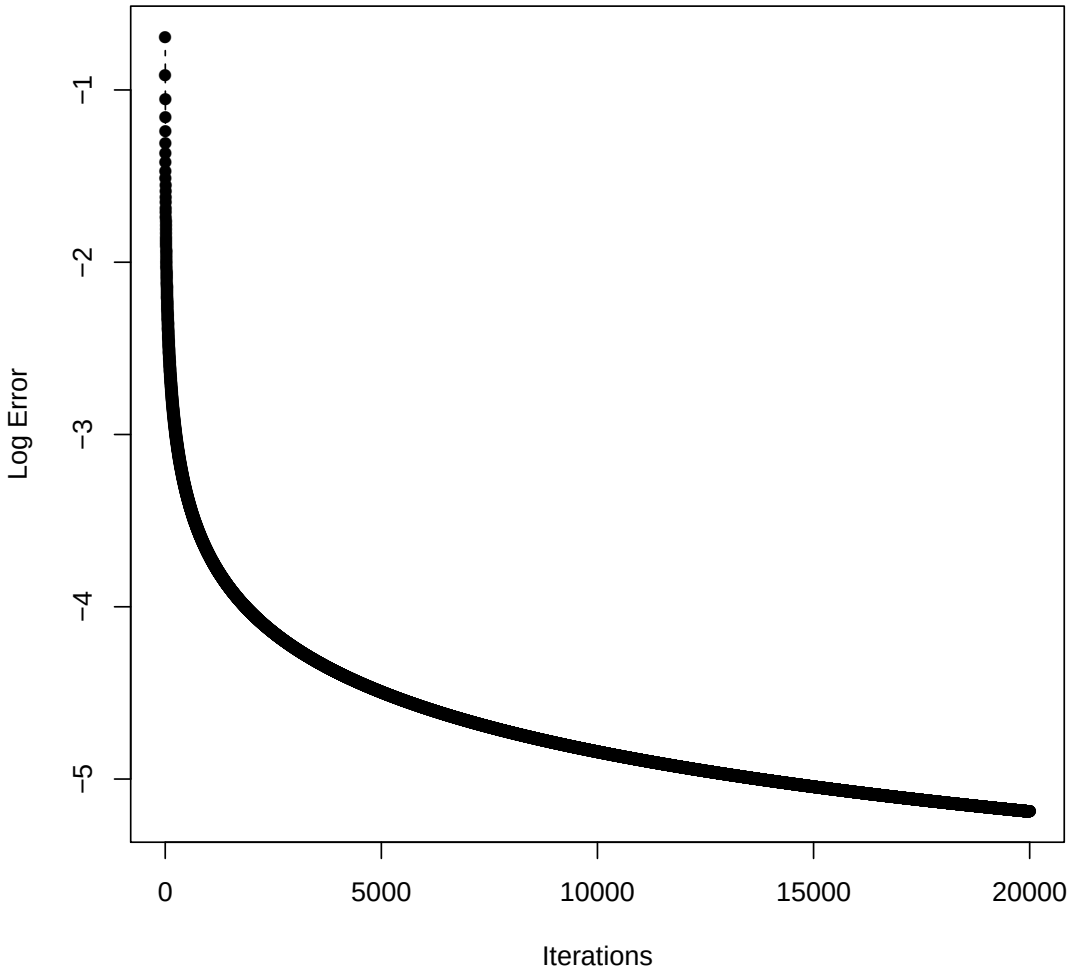

Figure 1.1: Plot of the log errors from applying gradient descent to the function $f(x) = x^4$.

1.3 Chord Methods

Chord Methods attempt to approximate the Hessian matrix by using a constant matrix $\mathbf{Q}$. The next iterate is defined by:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{Q}\mathbf{g}_n$$

The case $\mathbf{Q} = -\alpha\mathbf{I}$ with $\alpha > 0$ corresponds to gradient descent.

The Chord Method might look crude, but it is useful. In some cases for example, it is faster to compute the Hessian once at start and stick with it throughout the entire algorithm than to constantly recompute it [14]. In other words, set $\mathbf{Q} = \mathbf{H}_0$. . This would be epecially so if evaluating the gradient is cheap compared to the cost of computing the Hessian.

It is important that the matrix $-\mathbf{Q}$ be positive definite. The derivative of $f(x_n)$ in the direction $\mathbf{Q}\mathbf{g}_n$ is proportional to $\mathbf{g}_\mathbf{n}^\top\mathbf{Q}\mathbf{g}_\mathbf{n}$ since:

$$f(\mathbf{x}_n + \mathbf{Q}\mathbf{g}_n) \approx f(\mathbf{x}_n) + \mathbf{g}_n^\top\mathbf{Q}\mathbf{g}_n$$

Ensuring that $\mathbf{x}^\top\mathbf{Q}\mathbf{x} < 0, \forall\mathbf{x} \neq \mathbf{0}$ means that $\mathbf{Q}\mathbf{g}_n$ will always be a descent direction so that f can be reduced by taking a step in its direction.

It can be shown that the Chord Method converges linearly[14], a very informal sketch will be provided here.[5] Let $\mathbf{g}(\mathbf{x})$ denote the mapping $\mathbf{g}(\mathbf{x}) = \mathbf{x} + \mathbf{Q}\nabla f(\mathbf{x})$. Note that $\mathbf{x}_{n+1} = \mathbf{g}(\mathbf{x}_n)$. Suppose the sequence $\mathbf{x}_n$ converges, to determine how quickly the converge occurs, perform a Taylor expansion about the limit $\mathbf{x}^*$:

$$\begin{aligned}\mathbf{x}_{n+1} - \mathbf{x}^* &= \mathbf{g}'(\mathbf{x}^*)(\mathbf{x}_n - \mathbf{x}^*) \\ &= (\mathbf{I} + \mathbf{Q}\mathbf{H})(\mathbf{x}_n - \mathbf{x}^*) \\ &= \mathbf{K}(\mathbf{x}_n - \mathbf{x}^*)\end{aligned}$$

For brevity, we let $\mathbf{H}$ denote the Hessian of f at $\mathbf{x}^*$. The convergence of the Chord Method around $\mathbf{x}^*$ is governed by the matrix $\mathbf{K} = \mathbf{I} + \mathbf{Q}\mathbf{H}$. If $\mathbf{K} = 0$, then $\mathbf{Q} = \mathbf{H}^{-1}$ and the method converges superlinearly. It is very rarely the case that the Hessian at the limit point is available though. Usually the matrix $\mathbf{Q}$ is only an approximation to $\mathbf{H}^{-1}$. The better the approximation, the smaller the matrix $\mathbf{K}$ will be, and the faster the rate of convergence.

1.3.1 Using the Chord Method For Maximum Likelihood Estimation With The Poisson Distribution

For the problem of maximum likelihood estimation in one variable, the mapping associated with the Chord Method is given by by $g(\theta) = \theta + qS(\theta_n)$. In this case,

[5] This is a straightforward generalisation of results presented in [13] to the multivariate case.

$g'(\theta) = 1 + qqI(\theta)$ The Chord Method will converge to the MLE if $|1 + qI(\hat{\theta})| < 1$. The closer q is to $1/I(\hat{\theta})$, the faster the rate of convergence.

Consider the Poisson Distribution. The MLE[6] is given by $\hat{\lambda} = \bar{x}$. The usual Newton-Raphson iteration is given by:

$$\bar{\lambda}_{n+1} = \bar{\lambda}_n - \frac{S(\bar{\lambda}_n)}{I(\bar{\lambda}_n)},$$

A Chord Method is of the form:

$$\tilde{\lambda}_{n+1} = \tilde{\lambda}_n + qS(\tilde{\lambda}_n).$$

To find a value of q, rely on the fact that the variance of the MLE is asympotically equal to $1/I(\theta)$. Since the MLE is the sample mean, the Fisher Information can thus be approximated by estimating the variance of the sample mean, which is straightforwardly yields $m = \hat{\sigma}^2/n$,where $\hat{\sigma}^2$ is the sample variance.

Figure 1.2 compares the convergence of the Chord Method and the standard Newton-Raphson Method. The linear convergence of the Chord Method is readily apparent on the log plot, and the Chord Method performs reasonably well compared to the Newton-Raphson Method.

The asymptotic method used to justify the choice of q suggests that as the sample size gets bigger, the sample variance and observed Fisher Information might get closer and closer together, so that the Chord Method should converge more quickly. A simulation was conducted and the results are presented in Figure 1.3. Each curve plotted is the avearge over many error curves generated by generating samples from a poisson distribution and running the Chord Method on them. It can be seen that the method tends to converge more quickly as the sample size increases.

[6]The Score function is given by $S(\lambda) = n(\frac{\bar{x}}{\lambda} - 1)$ and the information is given by $I(\lambda) = -\frac{n\bar{x}}{\lambda^2}$.

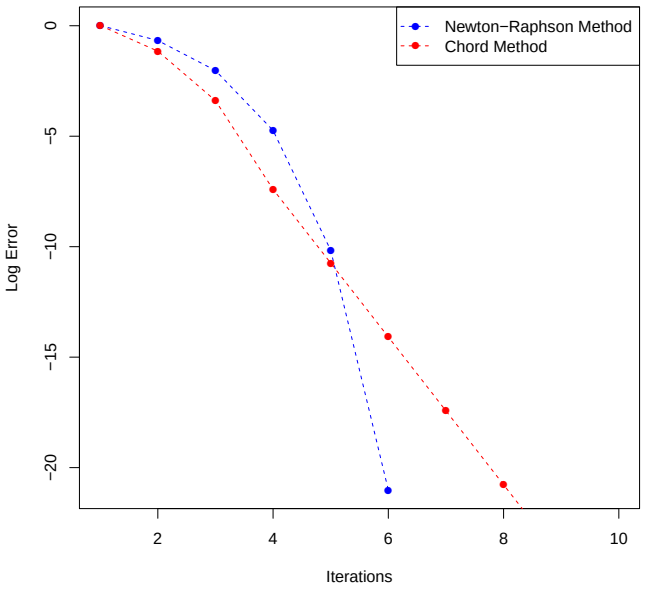

Figure 1.2: Plot comparing the convergence of the Chord Method and the Newton-Raphson Method for finding the MLE of a Poisson Distribution

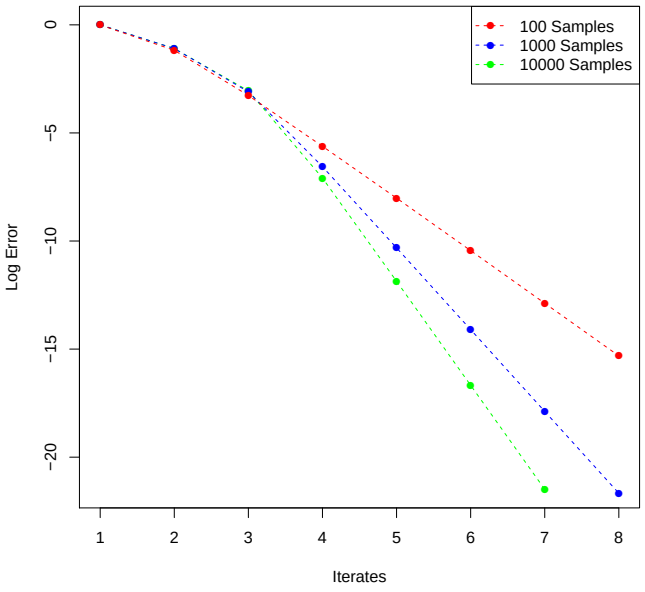

Figure 1.3: The Chord Method converges more quickly as the sample size increases.

1.4 Higher Order Methods and Quasi-Newton Methods

Instead of using a fixed matrix on each iteration as with the Chord Method, more advanced methods allow the matrix $\mathbf{Q}$ to vary on each iteration:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \mathbf{Q}_n \mathbf{g}_n$$

The choice $\mathbf{Q}_n = \mathbf{H}_n^{-1}$ corresponds to the Newton-Raphson Method. The discussion in Section 1.3 suggests that to ensure faster than linear convergence, it is necessary to ensure that $\mathbf{I} + \mathbf{Q}_n\mathbf{H}$ goes to $\mathbf{0}$ as n goes to infinity.[7] Not every method that changes $\mathbf{Q}_n$ on each interation has this property. Consider for example Fisher's Method of Scoring, which uses the expected information matrix $\mathcal{I}(\theta)$ to approximate the observed information $\mathbf{I}(\theta)$. It is not the case that $\mathcal{I}(\hat{\theta}) = \mathbf{I}(\hat{\theta})$, so one should not expect $\mathbf{I} + \mathbf{Q}_n\mathbf{H} \to \mathbf{0}$ as the algorithm converges to the MLE $\hat{\theta}$. As a result, Fisher's Method of Scoring will only converge linearly, though at a decent pace so long as it is a reasonable approximation to the observed Fisher Information.[8]

One obvious approach is to only recompute and invert the Hessian every m iterations, and otherwise recycle it. This is known in the literature as the *Shamanskii Method*[14]. This method can achieves superlinear convergence while reducing the need to to compute and solve Hessian matrices. With $m = 3$ for example, the Shamanskii Method takes the form:

$$\begin{aligned}
\bar{\mathbf{x}}_n &= \mathbf{x}_n - \mathbf{H}_n^{-1}\nabla f(\mathbf{x}_n) \\
\tilde{\mathbf{x}}_n &= \bar{\mathbf{x}}_n - \mathbf{H}_n^{-1}\nabla f(\bar{\mathbf{x}}_n) \\
\mathbf{x}_{n+1} &= \tilde{\mathbf{x}}_n - \mathbf{H}_n^{-1}\nabla f(\tilde{\mathbf{x}}_n)
\end{aligned}$$

Notice that the intermediate values are thrown away and only the final values $\mathbf{x}_{n+1}$ are retained.

The Shamanskii Method with $m = 2$ was applied to the Poisson estimation problem introduced in Section 1.3.1, the results are plotted in Figure 1.4. This method is faster than both the Newton-Raphson Method and the Chord Method. The line chart in the plot terminates prematurely because the Shamanskii's Method converges completely after five iterations so that the log error becomes $-\infty$.

Quasi-Newton Methods use the computed gradients to construct approximations to the true Hessians as the algorithm progesses.[24] These methods produce a sequence of psuedo-Hessians $\mathbf{B}_n$ that satisfy the Secant Condition:

[7] As noted in [24], it is actually only required that the $\mathbf{Q}_n$ approximates $\mathbf{H}_n$ along the directions which the alogorithm is searching.

[8] As the sample size grows larger, the expected Fisher Information gets increasingly good at approximating $\mathbf{I}(\hat{\theta})^{-1}$, so that Fisher's Method of Scoring tends to converge faster and faster as the sample size gets bigger in a similar fashion to Figure 1.3. But that doesn not mean that Fisher's Method of Scoring achieves superlinear convergence when applied to one specific sample.

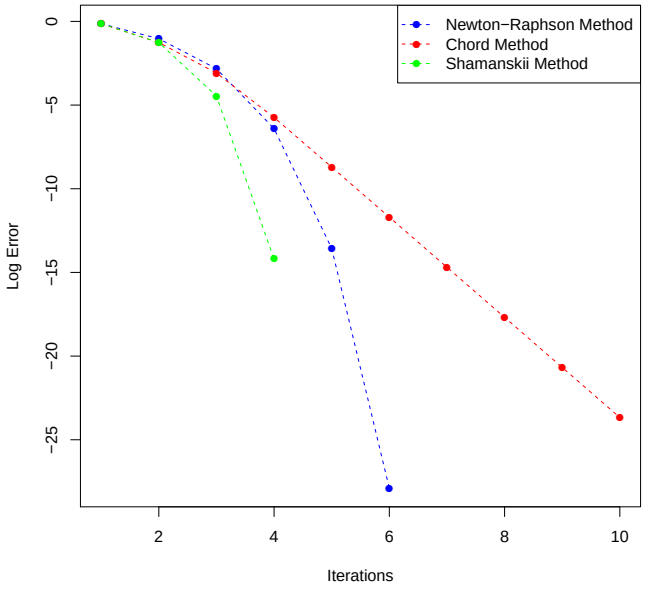

Figure 1.4: Applying the Shamanskii Method to the Poisson MLE problem

$$\mathbf{B}_n(\mathbf{x}_n - \mathbf{x}_{n-1}) = \mathbf{g}_{n+1} - \mathbf{g}_n$$

In one dimension, finding a B_n that satisfies the Secant Condition is equivalent to computing a finite difference approximation to the second deriviative:

$$\begin{aligned} B_n(x_n - x_{n-1}) &= f'(x_n) - f'(x_{n-1}) \\ B_n &= \frac{f'(x_n) - f'(x_{n-1})}{x_n - x_{n-1}} \end{aligned}$$

For multivariate problems, the second derivative is in the form of a matrix, so there is not enough information to construct a full approximation afresh on each iteration. Rather the approximate Hessian is partially updated using one of several approaches.

R's `optim` routine uses the BFGS method to compute the next approximate Hessian[27]. BFGS finds the symmetric matrix $\mathbf{B}_{n+1}$ satisfying the secant condition such that the inverse $\mathbf{B}_{n+1}^{-1}$ minimises a weighted Frobenius distance between itself and the previous inverse $\mathbf{B}_n^{-1}$. A low memory variant of BFGS known as L-BFGS is also available in R's standard libary[24, 27].

Quasi-Newton Methods converge superlinearly, but are not quadratically convergent like Newton's Method. They have the advantage of avoiding the cost of computing Hessian Matrices, so they can prove faster than Newton's Method in practice despite more iterations being needed.

1.5 Line Search Methods

So far, every method has entailed computing a direction $\mathbf{p}_n$ and letting $\mathbf{x}_{n+1} = \mathbf{x}_n + \alpha\mathbf{p}_n$, where it is often the case that $\alpha = 1$. The generated search directions $\mathbf{p}_n$ have been required to be a descent directions so that:

$$\frac{d}{d\alpha} f(\mathbf{x}_n + \alpha\mathbf{p}_n)\Big|_{\alpha=0} < 0$$

This means that if α is small enough, then a step in direction $\mathbf{p}_n$ will reduce the value of the objective function. At most, the optimisation method only knows the values of $f_n, \mathbf{g}_n$ and $\mathbf{H}_n$, so it can construct a quadratic or linear approxmition to to the objective function $\mathbf{x}_n$ and use that to find the next point.

As illustrated in Figure 1.5, the model can fail if one takes too big a step. For complex estimation problems, the objective funtion often has multiple peaks and troughs, so one must be careful that one has not wandered out of the range of validity of the locally constructed approximation.

Perhaps the default choice is to set α to some very small value. The first problem is that α might be unnecessarily small so that convergence is needlessly slow. The second problem is that the quality of a choice of α is governed by the higher order derivatives of the objective function, especially the higher order derivatives at the optimal point [9]. It will often be the case that such information is not available. Imagine the frustration if one had started an optimisation routine with a value of α that looked reasoanable for the initial values, and then came back 12 hours later to find the routine had failed as it approached the optimal point because it α was ultimately too high. In addition, a global choice of α might be far too low in some places.

A more sophisticated approach is to allow α to vary on each iteration:

$$\mathbf{x}_{n+1} = \mathbf{x}_n + \alpha_n\mathbf{p}_n$$

Alongside the problem of choosing the search direction $\mathbf{p}_n$, it is now necessary to decide how far in this direction to go. Conducting a Line Search entails probing the objective function along the ray $\{\mathbf{x}_n + \alpha\mathbf{p}_n | \alpha > 0\}$ to find the next point.

For convenience, let the function $\phi(\alpha) = f(\mathbf{x}_n + \alpha\mathbf{p}_n)$ denote the restriction of $f(\cdot)$ to the ray from $\mathbf{x_n}$ in the direction $\mathbf{p}_n$. Note that $\phi(0) = f(\mathbf{x}_n)$ and $\phi'(\alpha) = \mathbf{p}_n^\top \nabla f(\mathbf{x}_n + \alpha\mathbf{p}_n)$.

The goal of the line search is to find a point along the ray generated by $\mathbf{x}_n$ and $\mathbf{p}_n$ that is satisfactory. In principle one could of course attempt to find the value α^* that minimises $\phi(\alpha)$, but this is generally regarded as excessive. Usually one only iterates until sufficient improvement has been found and uses that point for the next iteration. [24]

[9] The literature very often works with Lipschitz continuity instead of differentiability. For simplicity in exposition, diffentiability will be used here.

Besides setting $\alpha_n = \alpha$ for all n, the line search method is known as *Backtracking Line Search*. In this case, α is replaced by $\rho\alpha$, where $0 < \rho < 1$ until a satisfactory point is found.

Algorithm 1 Sketch of Backtracking Line Search

Require: That $0 < \rho < 1$; that $\alpha_0 > 0$; and that $\mathbf{p}$ is a descent direction at $\mathbf{x}$.
  function BacktrackingLineSearch($\mathbf{x}, \mathbf{p}$)
    $\alpha \leftarrow \alpha_0$
    while $f(\mathbf{x} + \alpha\mathbf{p})$ is unsatisfactory **do**
      $\alpha \leftarrow \rho\alpha$
    end while
    return $(\mathbf{x} + \alpha\mathbf{p})$
  end function

In order to ensure the line search method converges,[10] the steps taken are required to satisfy the Wolfe Conditions.[11]

$$f(\mathbf{x}_n + \alpha_n\mathbf{p}_n) \leq f(\mathbf{x}_n) + c_1\alpha_n\mathbf{p}_n^\top\nabla f(\mathbf{x}_n) \quad \textbf{(W1)}$$
$$|\mathbf{p}_n^\top\nabla f(\mathbf{x}_n + \alpha_n\mathbf{p}_n)| \leq c_2|\mathbf{p}_n^\top\nabla f(\mathbf{x}_n)| \quad \textbf{(W2)}$$

Restated in terms of $\phi(\alpha)$, the Wolfe conditions are:

$$\phi(\alpha_n) \leq \phi(0) + c_1\alpha_n\phi'(0) \quad \textbf{(W1')}$$
$$|\phi'(\alpha_n)| \leq c_2|\phi'(0)| \quad \textbf{(W2')}$$

The first condition ensures sufficient decrease in the objective function, the second ensures a sufficent decrease in the gradient between steps. Intuitively, the bigger $|\phi'(0)|$ is, the bigger the decrease in the objective function demanded and the smaller the decrease in slope demanded, and vice versa.

1.5.1 More sophisticated Line Search Algorithms

More sophisticated line search alogrithms try to make better use of the values of $\phi(\alpha)$ and $\phi'(\alpha)$ computed in the course of the search. The first approach is to construct a quadratic interpolant through the points $\phi(0), \phi'(0)$, and $\phi(\bar{\alpha})$, where $\bar{\alpha}$ is the current value of α. The constructed quadratic $m(\alpha)$ has the properties that $m(0) = \phi(0), m'(0) = \phi'(0)$ and $m(\bar{\alpha}) = \phi(\bar{\alpha})$ The next value of α is taken to be the minimiser of $m(\alpha)$.

If the line search has been running for longer, so that values $\phi(\bar{\alpha})$ and $\phi(\tilde{\alpha})$ are available, then cubic interpolation is possible thorugh the values $\phi(0), \phi'(0), \phi(\bar{\alpha})$ and

[10] There are other conditions, but the Wolfe conditions are standard. Refer to [24] for more detail

[11] Strictly, these are the Strong Wolfe conditions. The usual Wolfe conditions only require that $-\mathbf{p}_n^\top\mathbf{g}_{n+1} \leq -c_2\mathbf{p}_n^\top\mathbf{g}_n$.

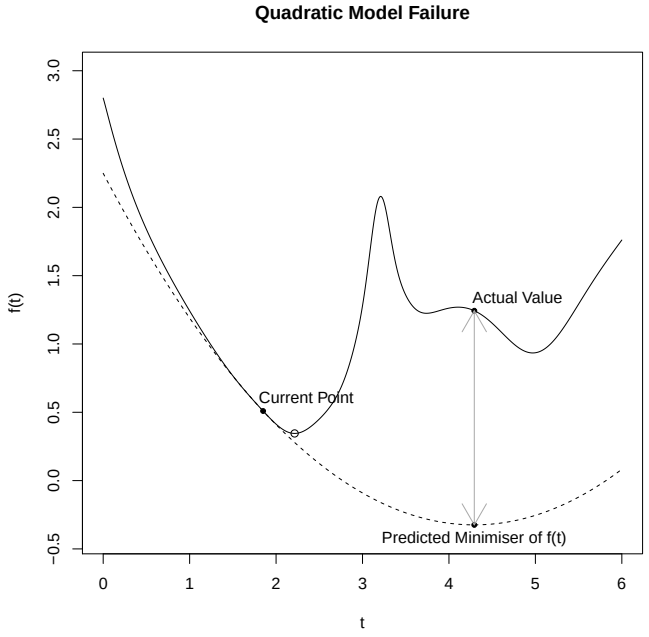

Figure 1.5: Extrapolating too far out can lead to disaster!

$\phi(\tilde{\alpha})$. The next value of α is taken as the minimum of the interpolating cubic, which entails solving a quadratic equation. While cubic methods can converge more rapidly than quadratic methods[12] they are also sometimes unstable and can have multiple critical points.

More exotic choices proposed in the literature include the possibility of using rational functions to approximate $\phi(\alpha)$ instead of polynomials, as discussed in [1]. An example of such an approximation is:

$$\phi(\alpha) \approx \frac{p\alpha^2 + q\alpha + r}{s\alpha + u} \tag{1.3}$$

Here, the values p, q, r, s and u are the parameters of the rational function to be determined.

[12]It is claimed in the literature that line searaches using cubic interpolants constructed using only values of $\phi(\alpha)$ and no derivatives do not converge quadratically.[34] It can be shown using similar arguments to those used in Section **??** that the error term for the cubic interpolant through $\phi(0), \phi'(0), \phi(\bar{\alpha})$ and $\phi(\tilde{\alpha})$ is of the form to $\frac{\phi^{(4)}(\epsilon)}{4!}\alpha^2(\alpha - \bar{\alpha})(\alpha - \tilde{\alpha})$. Barring the unlikely event that the cubic is minimised at $\alpha = 0$, the α^2 term will be bounded away from zero. In other words, holding one point fixed at $\alpha = 0$ will generally hurt the rate of convergence. Thus for a large number of iterations, the distance between the ith iterate α_i and the minimiser of $\phi(\alpha)$, which was already denoted α^*, might be something like $(\alpha_{i+1} - \alpha^*) \approx C(\alpha_i - \alpha^*)(\alpha_{i-1} - \alpha^*)$, which suggests that the cubic interpolation method does not converge quadratically.

Bibliography

[1] Jonathan Barzilai and Aharon Ben-Tal. Nonpolynomial and inverse interpolation for line search: synthesis and convergence rates. *SIAM Journal on Numerical Analysis*, 19(6):1263–1277, 1982.

[2] John P Boyd. *Chebyshev and Fourier spectral methods*. Courier Corporation, 2001.

[3] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[4] Richard P Brent. *Algorithms for minimization without derivatives*. Courier Corporation, 2013.

[5] Jiguo Cao and James O Ramsay. Parameter cascades and profiling in functional data analysis. *Computational Statistics*, 22(3):335–351, 2007.

[6] Kwun Chuen Gary Chan. Acceleration of expectation-maximization algorithm for length-biased right-censored data. *Lifetime data analysis*, 23(1):102–112, 2017.

[7] Edwin KP Chong and Stanislaw H Zak. *An introduction to optimization*, volume 76. John Wiley & Sons, 2013.

[8] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[9] Ronald Aylmer Fisher. The wave of advance of advantageous genes. *Annals of eugenics*, 7(4):355–369, 1937.

[10] Bengt Fornberg. Generation of finite difference formulas on arbitrarily spaced grids. *Mathematics of computation*, 51(184):699–706, 1988.

[11] PR Graves-Morris, DE Roberts, and A Salam. The epsilon algorithm and related topics. *Journal of Computational and Applied Mathematics*, 122(1-2):51–80, 2000.

[12] David R Hunter and Kenneth Lange. A tutorial on MM algorithms. *The American Statistician*, 58(1):30–37, 2004.

[13] Eugene Isaacson and Herbert Bishop Keller. *Analysis of numerical methods*. Courier Corporation, 2012.

[14] Carl T Kelley. *Iterative methods for linear and nonlinear equations*, volume 16. Siam, 1995.

[15] Carl T Kelley. *Implicit filtering*, volume 23. SIAM, 2011.

[16] C.T. Kelley. A brief introduction to implicit filtering. `https://projects.ncsu.edu/crsc/reports/ftp/pdf/crsc-tr02-28.pdf`, 2002. [Online; accessed 12-October-2019].

[17] Jack Kiefer. Sequential minimax search for a maximum. *Proceedings of the American mathematical society*, 4(3):502–506, 1953.

[18] Kenneth Lange. *Optimization*. Springer, 2004.

[19] Kenneth Lange. *Numerical analysis for statisticians*. Springer Science & Business Media, 2010.

[20] Kenneth Lange. The MM algorithm. `https://www.stat.berkeley.edu/~aldous/Colloq/lange-talk.pdf`, April 2007. [Online, accessed 18-September-2019].

[21] Randall J LeVeque. *Finite difference methods for ordinary and partial differential equations: steady-state and time-dependent problems*, volume 98. Siam, 2007.

[22] Steve McConnell. *Code complete*. Pearson Education, 2004.

[23] Geoffrey McLachlan and Thriyambakam Krishnan. *The EM algorithm and extensions*, volume 382. John Wiley & Sons, 2007.

[24] J Nocedal and SJ Wright. *Numerical Optimisation*. Springer verlag, 1999.

[25] Naoki Osada. *Acceleration methods for slowly convergent sequences and their applications*. PhD thesis, Nagoya University, 1993.

[26] Yudi Pawitan. *In all likelihood: statistical modelling and inference using likelihood*. Oxford University Press, 2001.

[27] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2013. ISBN 3-900051-07-0.

[28] James Ramsay. Functional data analysis. *Encyclopedia of Statistics in Behavioral Science*, 2005.

[29] Jim O Ramsay, Giles Hooker, David Campbell, and Jiguo Cao. Parameter estimation for differential equations: a generalized smoothing approach. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 69(5):741–796, 2007.

[30] Walter Rudin et al. *Principles of mathematical analysis*, volume 3. McGraw-hill New York, 1964.

[31] Larry Schumaker. *Spline functions: basic theory*. Cambridge University Press, 2007.

[32] Christopher G Small. A survey of multidimensional medians. *International Statistical Review/Revue Internationale de Statistique*, pages 263–277, 1990.

[33] Karline Soetaert, Thomas Petzoldt, and R. Woodrow Setzer. Solving differential equations in R: Package deSolve. *Journal of Statistical Software*, 33(9):1–25, 2010.

[34] Arie Tamir. Line search techniques based on interpolating polynomials using function values only. *Management Science*, 22(5):576–586, 1976.

[35] Gerald Teschl. *Ordinary differential equations and dynamical systems*, volume 140. American Mathematical Soc., 2012.

[36] Keller Vandebogart. Method of quadratic interpolation. `http://people.math.sc.edu/kellerlv/Quadratic_Interpolation.pdf`, September 2017. [Online; accessed 13-September-2019].

[37] Jet Wimp. *Sequence transformations and their applications*. Elsevier, 1981.

[38] Stephen Wright. Optimization for data analysis. In Michael W. Mahoney, John C. Duchi, and John C. Duchi, editors, *The Mathematics of Data*, chapter 2, pages 49–98. American Mathematical Society and IAS/Park City Mathematics Institute and Society for Industrial and Applied Mathematics, 2018.

[39] Tong Tong Wu, Kenneth Lange, et al. The MM alternative to EM. *Statistical Science*, 25(4):492–505, 2010.