CISC 371 Class 4

Minimizing by Inexact Line Search, or Backtracking

Texts: [1] pp. 106–114; [2] pp. 31–41

Main Concepts:
- *Computational effort of function and derivative evaluations*
- *Bracketed case: approximation*
- *Forward-backward search for a better objective value*

Sample Problem, Signal Processing: For the step response of a simple dynamical system that oscillates, where is the first minimum of the oscillation?

The methods of interval bracketing and approximation are sometimes called "exact" methods, which means that the purpose of the method is to find an estimate $\hat{t}$ of the true minimizer t^* of the objective function.

When minimizing an objective function that has a vector argument, the minimization of a function of a scalar argument is often a sub-process. Viewed from this larger perspective, it is often acceptable to compute a "good" estimate $\hat{t}$ that is not necessarily very close to the true minimizer t^*.

A method for finding such a "good" estimate $\hat{t}$ is called an *inexact* method. We will explore a limited range of the many results for inexact methods in the context of a specific inexact method, which is *backtracking search*.

4.1 Line Search

The process of finding a "good", but less than optimal, argument of an objective function of a scalar is sometimes called a *line search*. Later in this course, we will see how such a line arises in the optimization of a function with a vector argument. Here, we will state the usual assumptions of line search.

Because line search is a sub-process of a larger optimization method, the user supplies the line-search method with information. One crucial piece of information is the *direction* to be searched. A line search typically has a single current estimate, and the user specifies the direction in which the line search is to be conducted. For a scalar argument, the direction d is either $d = +1$ when the expected minimizer is greater than the current estimate, or $d = -1$ when the expected minimizer is less than the current estimate.

A method for line search is also supplied, by the user, with the objective function and a current estimate t_k of the true minimizer t^*. In addition, because the user has already calculated these values for reasons having to do with the overall optimization, the line search is supplied with the objective evaluated at the estimate, as f_k, and the first derivative at the estimate, as f'_k. A line search therefore is supplied with:

- $f(t)$: objective function that can be evaluated
- t_k: current estimate of the true minimizer t^*
- f_k: objective function evaluated at t_k
- f'_k: first derivative of the objective function evaluated at t_k
- $d = \pm 1$: direction of the line search

4.2 Fixed-Stepsize Search

The most elementary form of line search is to take a prescribed step in the prescribed unit direction. The user supplies the search method with a constant s_0 and the line search computes the new estimate of the true minimizer t^* as

$$\begin{aligned} d_k &= -\text{sign}(f'(t_k)) \\ t_{k+1} &= t_k + s_0 d_k \end{aligned} \tag{4.1}$$

Equation 4.1 has a serious problem: as we approach a local minimizer, the sequence will usually "oscillate" around the true value. Instead, at iteration number k, we scale the direction d_k by the magnitude of the derivative $f'(t_k)$. This new estimate is, depending on whether we want to use the direction d_k or simply use the derivative, is

$$\begin{aligned} d_k &= -f'(t_k) \\ t_{k+1} &= t_k + s_0 d_k \\ &= t_k - s_0 f'(t_k) \end{aligned} \tag{4.2}$$

Equation 4.2 is simple to implement and, for optimization in a vector space, a perhaps surprisingly common method. For example, with the artificial neuron called the Perceptron, the basic learning rule is a fixed stepsize that guarantees convergence for certain types of data. This method, using a fixed stepsize and an unscaled derivative, is described as pseudocode in Algorithm 4.1.

In practice, selecting the stepsize can be difficult. One common problem that arises is that the stepsize is too large; the search can variously become worse, "oscillate" around a local minimum, or produce an erratic sequence of estimates. An example of using a fixed stepsize is shown in

Algorithm 4.1 Scalar minimization, fixed stepsize

```
Require: kmax  $> 0$
Require: dmag  $> 0$
  t ← t0
  fcurr ← f(t)
  d ← -f′(t)
  k ← 0
  while ¬ (converged) do
    t ← t + s0⋆d                    ▷ fixed stepsize
    d ← -f′(t)
    fcurr ← f(t)
    k ← k+1
  end while
```

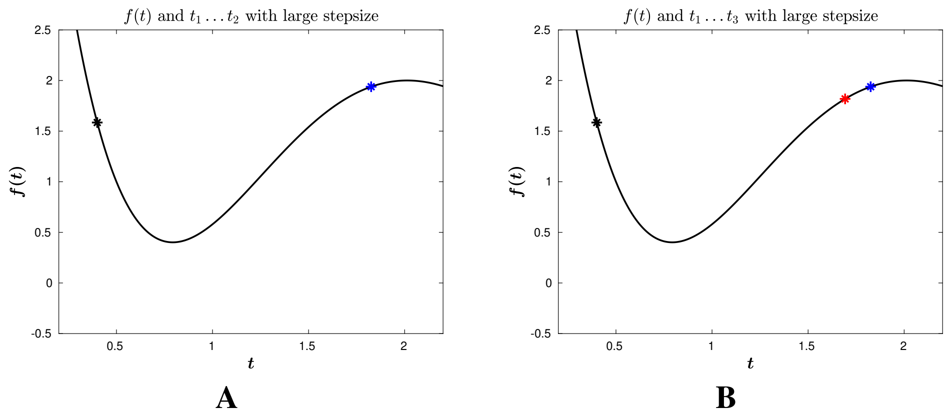

Figure 4.1: Line search, with a fixed stepsize that is too large, fails to approximate the true local minimum. The objective function is plotted as a solid black curve. (A) Initial estimate t_1 shown in black; a fixed stepsize produces an estimate, shown in blue, that is on the opposite side of the nearest minimizer and has a larger function value than that of the initial estimate. (B) Another application of the fixed stepsize update, shown in red, begins to approach the minimizer.

Figure 4.1. Because the stepsize is too large, the first application of Equation 4.2 results in an the estimate of t^* that is worse than the current estimate t_k.

Another example of using a fixed stepsize is shown in Figure 4.2. Because the stepsize is too small, repeated applications of Equation 4.2 result the estimate of t^* approaching the original estimate t_1 at a slow rate.

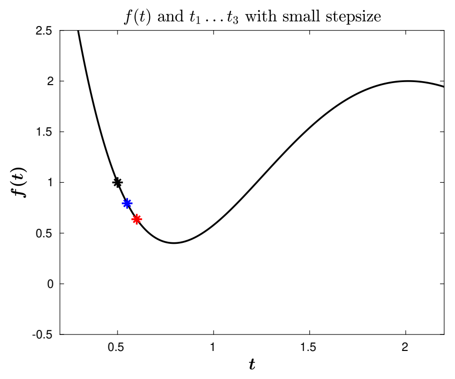

Figure 4.2: Line search, with a fixed stepsize that is too small, fails to approximate the true local minimum. The objective function is plotted as a solid black curve. The initial estimate t_1 is shown in black. Successive updates with a fixed stepsize, shown in blue and red, slowly approach the true minimizer.

4.3 Inexact, or Backtracking, Line Search

In practice, optimization that is conducted with an objective function that has a vector argument may be relatively tolerant to error in the line-search estimate $\hat{t}$ of the true minimizer t^*. From the large corpus of theoretical and applied results is available, in this course we will use only a small subset of results.

We will make an important assumption about a user invoking a line-search method: the stepsize s_0 that the user provides may be *too large* but will not be too small. This assumption is justified in practice because a too-small stepsize is detectable as a long computation to convergence; but, a too-large stepsize might produce either a divergent computation or – perhaps worse – a convergence to an undesirable local minimum of the overall objective function.

An implication of this assumption about the user is that we will never try to increase the stepsize from the provided number s_0. Instead, we will concentrate on decreasing the stepsize by as small a quantity as we can. This process of decreasing the stepsize from the provided number s_0 is called *backtracking*. We will explore backtracking by considering a fundamental result from real analysis.

The foundation for many methods of inexact line search is an approximation of the objective function using a Taylor series. We will introduce a free variable u to parameterize the objective function; this frees us from over-using the variable name "t".

Expanding a Taylor series around the initial estimate t_k, and

$$f(t_k + u) = f(t_k) + f'(t_k)u + \cdots \tag{4.3}$$

One nuance is that we are searching for a new estimate $\hat{t}$ in the direction d. We will modify Equation 4.3, then truncate second-order and higher-order terms, to get the approximation

$$\begin{array}{rcl} f(t_k+u) & \approx & f(t_k)+uf'(t_k) \\ \text{or } f(t_k+sd) & \approx & f(t_k)+s|f'(t_k)|d \end{array} \tag{4.4}$$

A geometric observation of Equation 4.4 is that an objective function with a local minimum near t_k will be convex. This implies that, locally, the objective function is "above" the tangent line. An algebraic implication of the geometry is that, for a user-provided number s_0, locally we will always have $f(t_k+s) > f(t_k)+s|f'(t_k)|d$. From an earlier class in this course, Figure 3.7 shows a locally convex function and the tangent line at a given estimate t_k.

To begin our development of a method for backtracking, we can observe that we could exponentially "back off" from the user-provided fixed stepsize that gives us t_k+s_0d. This could be an exponential sequence such as

$$\begin{array}{rcl} s & = & s_0, \dfrac{1}{2}s_0, \dfrac{1}{4}s_0, \ldots \\ & = & \dfrac{1}{2^\gamma}s_0 \text{ for } \gamma=0,1,2,\ldots \\ & = & \beta^\gamma s_0 \text{ for } \gamma=0,1,2,\ldots \\ & & \text{with } \beta=1/2 \end{array} \tag{4.5}$$

Any backtracking hyper-parameter $0<\beta<1$, when used in Equation 4.5, will produce an exponential sequence that converges to zero. We might use such a sequence to select a stepsize. One criterion that we could use is that the new estimate of the minimizer must produce a function value that is less than the current estimate. This could be a simple comparison, such as

$$f(t_k+\beta^\gamma s_0d) < f(t_k) \tag{4.6}$$

One limitation of the back-off test in Equation 4.6 is that it will select the largest stepsize that is less than or equal to the user-provided s_0 and that provides an improvement in the estimate of the minimizer. This is illustrated graphically in Figure 4.3, where the dashed line is the current level of the function for the estimate t_k.

There are many ways to *backtrack* from the fixed stepsize estimate of $t_{k+1}=t_k+s_0d$. One way is to back-off the stepsize, in a sequence such as that of Equation 4.5, and simultaneously to change the comparison from the simple one in Equation 4.6 to a more informed comparison.

The idea in backtracking is that, as we exponentially back-off from the user-supplied stepsize s_0, we can also exponentially reduce one side of an inequality constraint. We know, from the

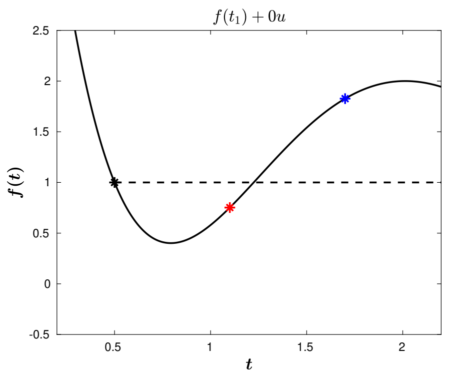

Figure 4.3: An objective function with a local minimum is plotted as a solid black curve. The current level of the function, here at t_1, is shown as a dashed line. A user-provided step of s_0 is shown in blue and the first exponential back-off value is shown in red. The largest back-off step that improves the estimate will be selected.

gradient inequality, that for any value t that is near an estimate t_k of a minimizer, we have

$$\begin{aligned} f(t) &\not\leq f(t_k) + s_0 f'(t_k) \\ \rightarrow f(t_k + \beta^\gamma s_0 d) &\not\leq f(t_k) + s_0 f'(t_k) \end{aligned} \tag{4.7}$$

In a 1966 article, Larry Armijo [3] presented a proof for any objective function that has a strict bound on its first derivative. He showed that there is some non-negative integer γ such that

$$f(t_k - \beta^\gamma s_0 f'(t_k)) - f(t_k) \leq -\frac{1}{2^\gamma} s_0 |f'(t_k)|^2 \tag{4.8}$$

His proof generalizes from a power of $1/2$ to a power of any $0 < \beta < 1$. Let us introduce the shorthand

$$\alpha_k \overset{\text{def}}{=} f'(t_k)/2 \tag{4.9}$$

In an Armijo backtracking search, we can combine a generalization of Equation 4.8 with the shorthand of Equation 4.9 to produce a concise constraint. We increment γ until we have

$$f(t_k + \beta^\gamma s_0 d_k) \leq f(t_k) + \alpha_k \beta^\gamma s_0 d_k \tag{4.10}$$

Armijo backtracking locally estimates the stepsize in minimization. We can code an iteration or recursion that uses the user-supplied values of s_0 and β to backtrack until Equation 4.10 is satisfied. This method is presented in Algorithm 4.2. Backtracking with a parameter $\beta = 1/2$ is illustrated in Figure 4.4, in which a stepsize of $s/4$ is needed to satisfy the Armijo condition.

Algorithm 4.2 Scalar minimization, Armijo backtracking

```
Require: kmax  $> 0$
Require: dmag  $> 0$
  t ← t1
  fcurr ← f(t)
  g ← f'(t)
  d ← -g
  $\alpha$ ← g/2
  k ← $0$
  while $\neg$(converged) do
    s ← s0
    fest ← f(t+s*d)
    while fest>(fcurr+$\alpha$*s*d) do
      s ← $\beta$*s                    ▷ Armijo backtracking
      fest ← f(t+s*d)
    end while
    t ← t + s*d
    fcurr ← f(t)
    g ← f'(t)
    d ← -g
    $\alpha$ ← g/2
    k ← k+1
  end while
```

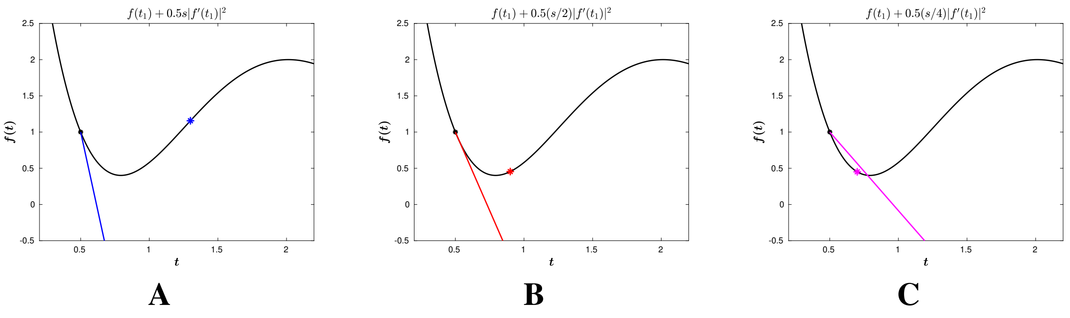

Figure 4.4: Line search by backtracking with a divisor of $\beta = 1/2$. The objective function is plotted as a solid black curve. (A) The first backtracking stepsize s is too large; the scaled line and objective at $t_k + s_0 d$ are shown in blue. (B) The back-tracked stepsize $s/2$ is still too large; the scaled line and objective at $t_k + (s/2)d$ are shown in red. (C) The back-tracked stepsize $s/4$ satisfies the Armijo condition; the scaled line and objective at $t_k + (s/4)d$ are shown in magenta.

References

[1] Antoniou A, Lu WS: Practical Optimization: Algorithms and Engineering Applications. Springer Science & Business Media, 2007

[2] Nocedal J, Wright S: Numerical Optimization. Springer Science & Business Media, 2006

[3] Armijo L: Minimization of functions having Lipschitz continuous first partial derivatives. *Pac J Math* 16(1):1–3, 1966