CISC 271 Class 26

Elementary Numerical Optimization

Text Correspondence: Beck [1] pp. 1–10

Main Concepts:

- *Stationary point, minimizer*
- *Derivative and direction of descent*
- *Function with a vector argument*
- *Steepest descent with constant stepsize*

Sample Problem, Signal Processing: For function with a scalar argument, how can we numerically estimate a stationary point?

For us, optimization is the process of selecting a "best" member of a set according to a criterion. We will briefly explore optimization of a function, which in this course is a mapping from a set of points to the real numbers. In particular, when the domain is the real numbers $\mathbb{R}$ then we write

$$f : \mathbb{R} \to \mathbb{R} \qquad \text{or} \qquad f(t)$$

we will refer to such a function as an *objective function*. The term "objective" will be used as an abbreviation for the longer term. A point, from the domain of f, is the *argument* of the function.

A maximum or minimum value of an objective happens at a *stationary point*, which occurs where the derivative of the zero.

Definition: stationary point of $f(t)$

For any $t \in \mathbb{R}$ and any $f : \mathbb{R} \to \mathbb{R}$, a scalar $t^* \in \mathbb{R}$ is a *stationary point* of f is defined as

$$f'(t^*) = 0 \tag{26.1}$$

An example of a function and its two stationary points is shown in Figure 26.1. In this case, the function $f_1(t) = -t/(t^2 + 1)$ can be differentiated and its stationary points can be found as the zeros of a rational function.

Even though a function might be "nice" – smooth and infinitely differentiable – it could be easy to write the objective function and difficult to find a mathematical expression for its stationary point(s).

In optimization, the concept of being "concave-up" is called *convexity*. In simple words, a convex function is where, if we pick any two points, function is "below" the line that "connects" the points.

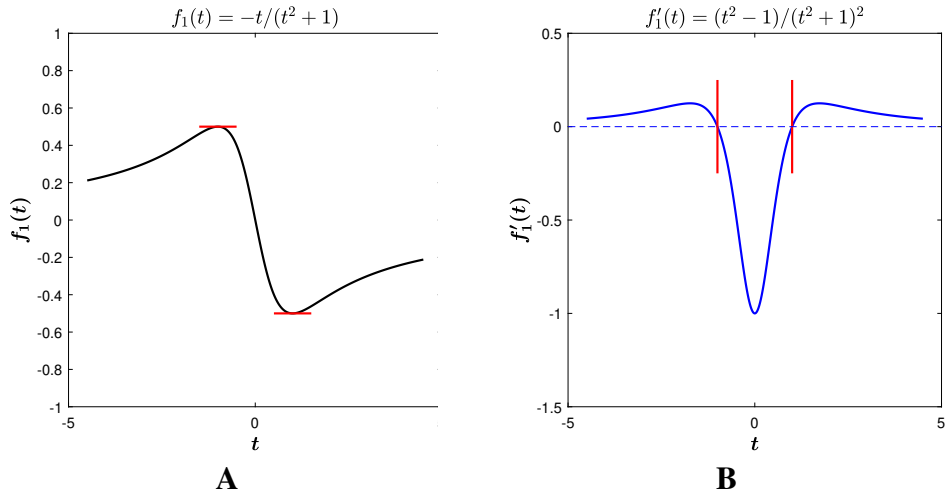

Figure 26.1: The function $f_1(t) = -t/(t^2+1)$ has two stationary points. (A) The function is shown in black and stationary points are shown in red. The stationary point at $t = +1$ is a local minimizer and the stationary point at $t = -1$ is a local maximizer. (B) The derivative of f_1 is shown in blue and its zeros are shown in red. Zeros of the derivative are stationary points of the function.

26.1 Scalar Argument – Fixed-Stepsize Search

Consider a scalar-valued function that is "nice"; then we can find the derivative at t_0 as $f'(t_0)$. If the derivative $f'(t_0)$ is positive then there might be a local minimum to the "left" of t_0; likewise, if f the derivative $f'(t_0)$ is negative then there might be a local minimum to the "right" of t_0. We can suppose that a local minimum is "pointed to" by $-f'(t_0)$. An example is illustrated in Figure 26.2.

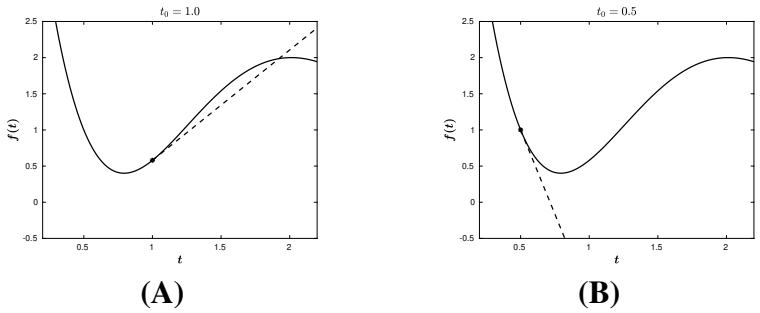

Figure 26.2: Function having a scalar argument, where the minimizer is in the direction of the negative of the derivative at a given point. (A) At $t_0 = 1$, the derivative is positive and the minimizer is "left" of t_0. (B) At $t_0 = 0.5$, the derivative is negative and the minimizer is "right" of t_0.

The process of finding a "good" estimate of a stationary point of an objective function having

a scalar argument is a *search*. Consider an iterative search at step k, where we want a "better" estimate of a stationary point at step $k+1$. Typically, at step k, we know:

- $f(t)$: 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
- $d_k = \pm 1$: direction of the search

The most elementary form of search is to take a prescribed step in the prescribed unit direction. In machine learning, the size of the step is often called the *learning rate* and is written as the Greek symbol eta η.

The user supplies the search method with a constant η and the search computes the new estimate of the true minimizer t^* as

$$\begin{aligned} d_k &= -\text{sign}(f'(t_k)) \\ t_{k+1} &= t_k + \eta d_k \end{aligned} \tag{26.2}$$

Equation 26.2 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

$$\begin{aligned} d_k &= -f'(t_k) \\ t_{k+1} &= t_k + \eta d_k \\ &= t_k - \eta f'(t_k) \end{aligned} \tag{26.3}$$

This method, using a fixed stepsize, is described as pseudo-code in Algorithm 26.1.

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 Figure 26.3. Because the stepsize is too large, the first application of Equation 26.3 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 26.4. Because the stepsize is too small, repeated applications of Equation 26.3 result the estimate of t^* approaching the original estimate t_1 at a slow rate.

Algorithm 26.1 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 + eta*d                          ▷ fixed stepsize
     d ← -f'(t)
     fcurr ← f(t)
     k ← k+1
  end while
```

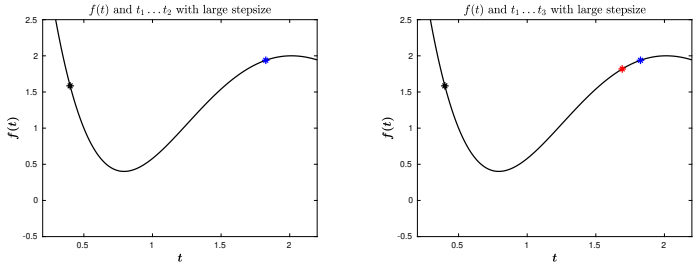

Figure 26.3: Fixed-stepsize search, with a stepsize that is too large, fails to approximate the true local minimum. The objective function is plotted as a solid black curve. (A) Initial estimate t_1 shown in black; a fixed stepsize produces an estimate, shown in blue, that is on the opposite side of 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.

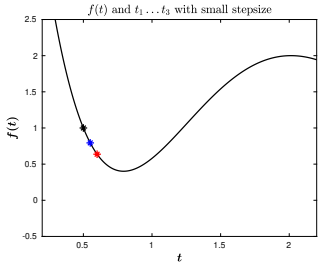

Figure 26.4: Fixed-stepsize search, with a stepsize that is too small, fails to approximate the true local minimum. The objective function is plotted as a solid black curve. The initial estimate t_1 is shown in black. Successive updates with a fixed stepsize, shown in blue and red, slowly approach the true minimizer.

26.2 Vector Argument – Fixed-Stepsize Search

In multi-variable calculus, a function is expressed as having two or more real-valued arguments. A function of two variables, $f(tw1, w_2)$, does not seem to have a derivative as defined for a function with a scalar argument. Instead, such as function has two *partial* derivatives that are defined as

$$\begin{aligned} \frac{\partial f}{\partial w_1}(w_1, w_2) &\stackrel{\text{def}}{=} \lim_{h \to 0} \frac{f(w_1 + h, w_2) - f(w_1)}{h} \\ \frac{\partial f}{\partial w_2}(w_1, w_2) &\stackrel{\text{def}}{=} \lim_{h \to 0} \frac{f(w_1, w_2 + h) - f(w_1)}{h} \end{aligned} \tag{26.4}$$

Instead, we will use an objective function with a vector argument that we will write as $f(\vec{w})$. To extend Equation 26.4 to vectors, we can recall from linear algebra the idea of an *elementary vector*. The j^{th} entry of $\vec{e}_j$ is 1 and every other entry is 0. Using this idea, we can express the partial derivatives of Equation 26.4 as

$$\frac{\partial f}{\partial w_j}(\vec{w}) \stackrel{\text{def}}{=} \lim_{h \to 0} \frac{f(\vec{w} + h\,\vec{e}_j) - f(\vec{w})}{h} \tag{26.5}$$

Equation 26.5 is the same as the definition from multi-variable calculus, using vectors instead of multiple scalar arguments. The usual rules for finding a partial derivative are the same, such as holding other variables as constant when performing the differentiation.

Previously in this course, a mathematical object that looks like a row matrix was called *one-form* or a *1-form*. The idea of a 1-form leads us to a fundamental definition for numerical optimization.

Definition: gradient operator of $f(\vec{w})$

> For $\vec{w} \in \mathbb{R}^n$ and any continuously differentiable $f : \mathbb{R}^n \to \mathbb{R}$, the *gradient operator* of a function $f(\vec{w})$ is defined as the 1-form
>
> $$\underline{\nabla} f(\vec{w}) \stackrel{\text{def}}{=} \left[\frac{\partial f}{\partial w_1}(\vec{w}) \quad \frac{\partial f}{\partial w_2}(\vec{w}) \quad \dots \quad \frac{\partial f}{\partial w_n}(\vec{w}) \right] \tag{26.6}$$

The gradient is a 1-form but we want to search in the vector space $\mathbb{V}$ that a weight vector $\vec{w}$ "lives" in. At any point $\vec{w}_0$, what direction is the function $f(\vec{w})$ steepest in the "downward" sense?

The answer is a straightforward extension of the idea we used for a function with a scalar argument. The direction of steepest descent, at a point $\vec{w}_0$, is the negative transpose of the gradient at that point:

$$\vec{d} = -[\underline{\nabla} f(\vec{w}_0)]^T \tag{26.7}$$

The fundamental algorithm in unconstrained optimization is the *steepest descent algorithm*. This method, using a fixed stepsize and an unscaled gradient 1-form, is described as pseudo-code in Algorithm 26.2.

Algorithm 26.2 Steepest Descent, fixed stepsize

```
Require: kmax  > 0
Require: gmag  > 0
  w ← w0
  fcurr ← f(w)
  g ← ∇f(w)
  d ← −[g]^T
  k ← 0
  while ¬ (converged) do
     w ← w + eta*d                          ▷ fixed stepsize
     fcurr ← f(w)
     g ← ∇f(w)
     d ← −[g]^T
     k ← k+1
  end while
```

The convergence criteria in Algorithm 26.2 may include a combination of limiting the number of iterations to a value `kmax`, and requiring the magnitude of the gradient to be below some positive number `gmag`.

End of Extra Notes