CISC 371 Class 8

First-Order Optimization – Steepest Descent

Texts: [1] pp. 71–78; [2] pp. 49–57

Main Concepts:
- *Descent directions*
- *Convergence criteria*
- *Gradient descent with constant stepsize*
- *Stepsize selection by backtracking*

Sample Problem, Data Analytics: If we only have a local view of physical terrain, what direction do we use to find a river in a valley?

We now turn to the second of the three problems that we posed at the start of this course, which is how to perform unconstrained optimization of an objective function that has a vector argument. For us, optimization is the process of finding a local minimizer, which we will write mathematically as

$$\vec{w}^* = \operatorname*{argmin}_{\vec{w}\in\mathbb{R}^n} f(\vec{w}) \tag{8.1}$$

In optimization, we usually have a relatively local "view" of the function we are trying to minimize. An example is in navigating in unknown terrain without GPS or any other aid. Suppose that we left our vehicle beside a small river, climbed up, and now we are lost. We know that we are located on the side of one of the hills depicted in Figure 5.3. One way out of this terrain is to walk in the direction that is steepest downward. For the terrain that is shown, this will take us to either the valley or the flat fields that surround the hills.

We will pose this type of optimization as searching in a vector space $\mathbb{R}^n$ for a minimizer $\vec{w}^*$. We will do this search iteratively and in each step we will need to find:

- a direction vector $\vec{d} \neq \vec{0}$, and
- a stepsize $s > 0$

In Class 4, we introduced the concept of a direction, d, in searching for a scalar minimizer t^* of an objective $f(t)$. The scalar d was a unit value, so we had $|d| = 1$. The direction was used to guide the local search for the minimizer and we required that the direction locally reduced the value of the objective; mathematically, at a value w_0, we required that $f'(w_0)d < 0$.

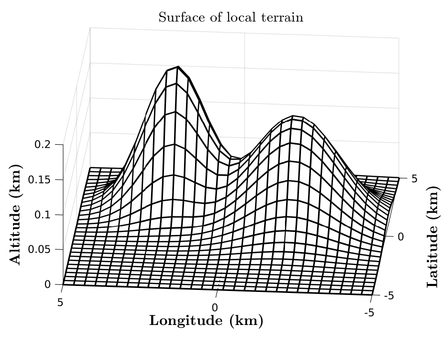


Figure 8.1: A function that represents local terrain. From any point, we might try to descend in the steepest direction to arrive at a river in a valley.

8.1 Descent Directions and the Gradient

To search for a minimizer $\vec{w}^*$ of a function $f(\vec{w})$, we will use a *direction* vector, which often – but not always – is a vector that has a unit length. The fundamental concept in unconstrained optimization is based on the idea of ***steepest descent***, which we motivated by an imaginary trip into hilly terrain.

Definition: descent direction

For any continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, for any vector $\vec{w}_0 \in \mathbb{R}^n$ a *descent direction $\vec{d}$ at $\vec{w}_0$* is defined as

$$D_{\vec{d}}(\vec{w}_0) < 0 \tag{8.2}$$

Unlike most textbooks – which use an intuitive understanding of steepest descent – we will define this term as the unit direction in which the directional derivative is minimal.

Definition: direction of steepest descent

For any continuously differentiable function $f : \mathbb{R}^n \to \mathbb{R}$, for any vector $\vec{w}_0 \in \mathbb{R}^n$ such that $\vec{w}_0$ is not a stationary point of f, the *direction of steepest descent* is defined as

$$\vec{d} = \underset{\vec{v}\in\mathbb{R}^n : \|\vec{v}\|=1}{\operatorname{argmin}} D_{\vec{v}}(\vec{w}_0) \tag{8.3}$$

Observations: For this course, in Definition 8.3 we will use the Euclidean norm. Another norm can be used, e.g., the L_∞ norm produces a *coordinate descent* algorithm that is useful in some problems. From the definition of the directional derivative, in a Euclidean search space with a Euclidean norm $\|\cdot\|_2$, Definition 8.3 is equivalent to

$$\vec{d} = \operatorname*{argmin}_{\vec{v}\in\mathbb{R}^n:\|\vec{v}\|_2=1} \underline{\nabla} f(\vec{w}_0)\vec{v} \tag{8.4}$$

Observation: For the Euclidean norm, Equation 8.4 can be solved using linear algebra. The solution to Equation 8.5, provided in the extra notes for this class and not scaling to a unit vector, is simply

$$\vec{d} = -[\underline{\nabla} f(\vec{w}_0)]^T \tag{8.5}$$

Equation 8.5 provides a direction vector for use in an iteration. We must also select a stepsize.

8.2 Fixed Stepsize and Backtracking

In most practical optimization problems, a stepsize must be selected by the user. There is no deterministic way to select a stepsize and the user must provide this, perhaps from an understanding of the application that needs the optimization.

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 pseudocode in Algorithm 8.1.

Algorithm 8.1 Steepest Descent, fixed stepsize

```
Require: kmax  $> 0$
Require: gmag  $> 0$
  w ← w0
  fcurr ← f(t)
  g ← $\underline{\nabla}$f(t)
  d ← $-[g]^T$
  k ← $0$
  while ¬ (converged) do
     w ← w + s*d                        ▷ fixed stepsize
     fcurr ← f(t)
     g ← $\underline{\nabla}$f(t)
     d ← $-[g]^T$
     k ← k+1
  end while
```

The convergence criteria in Algorithm 8.1 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.

Because a "good" fixed stepsize s_0 is seldom easy to find in practice, a common alternative is to use backtracking to select a stepsize. As we explored in Class 4, a backtracking algorithm will eventually find a stepsize that meets the Armijo condition. For searching in a vector space, we can replace the scalar terms in Equation 4.10 with vector terms, using $\underline{\alpha_k} = \underline{\nabla} f(\vec{w}_k)/2$. Our version of the Armijo backtracking condition is to loop until

$$f(\vec{w}_k + \beta^\gamma s_0 \vec{d}_k) \leq f(\vec{w}_k) + \underline{\alpha} \beta^\gamma s_0 \vec{d}_k \tag{8.6}$$

We can incorporate Equation 8.6 into Algorithm 8.1 to develop a method for steepest descent that uses backtracking to locally estimate the stepsize. This method is presented in Algorithm 8.2.

Algorithm 8.2 Steepest Descent, Armijo backtracking

```
Require: kmax  $> 0$
Require: gmag  $> 0$
  w ← w0
  fcurr ← f(w)
  g ← $\underline{\nabla}$f(w)
  d ← $-[g]^T$
  $\alpha$ ← g/2
  k ← 0
  while ¬(converged) do
     s ← s0
     fest ← f(w+s*d)
     while fest>(fcurr+$\alpha$*s*d) do
        s ← $\beta$*s                    ▷ Armijo backtracking
        fest ← f(w+s*d)
     end while
     w ← w + s*d
     fcurr ← f(w)
     g ← $\underline{\nabla}$f(w)
     d ← $-[g]^T$
     $\alpha$ ← g/2
     k ← k+1
  end while
```

Example: quadratic function for which the origin is a minimizer. Let us try using the function

$$f_1(\vec{w}) \quad = \quad \vec{w}^T K \vec{w} \quad \text{where} \quad K = \begin{bmatrix} 1/4 & 0 \\ 0 & 1 \end{bmatrix} \tag{8.7}$$

We can implement Algorithm 8.1 in `MATLAB`, using a starting point $\vec{w}_0 = \begin{bmatrix} -3.0 \\ 3.2 \end{bmatrix}$

A surface plot of $f_1(\vec{w})$, and contours with the point $\vec{w}_0$, are shown in Figure 8.2.

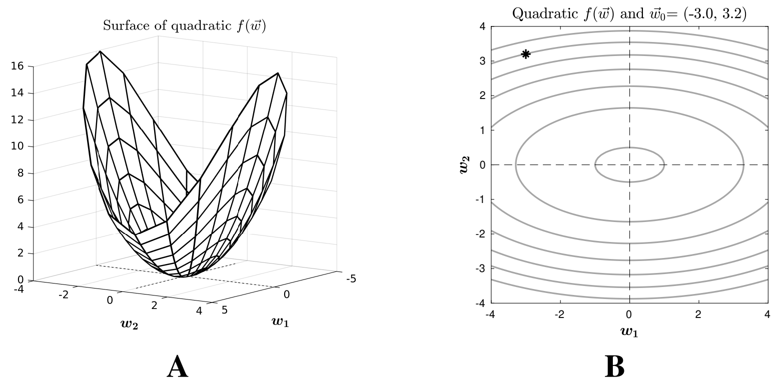


Figure 8.2: The function $f_1(\vec{w})$ in Equation 8.7. (A) Surface plot. (B) Contours and a point $\vec{w}_0$.

Two commonly used descent directions are the steepest descent of Equation 8.5, and coordinate descent in the direction of the coordinate of the largest absolute value of $\underline{\nabla} f(\vec{w})$. The directions, and slices along these directions, are shown in Figure 8.3. As expected, the slices are quadratic functions of a scalar argument; here, we use t to parameterize steps along the slice.

To numerically optimize $f_1(\vec{w})$, we can try two fixed stepsizes: $s = 0.85$ and $s = 0.25$. These are larger and smaller than is optimal, which were situations that we encountered when we studied functions with a scalar argument. The paths for four iterations of Algorithm 8.1 are shown in Figure 8.4. If we use a local quadratic approximation to compute the optimal stepsize, the algorithm of steepest descent will converge rapidly to the minimizer. When we use backtracking with the Armijo conditions, setting $s = 1.5$ and $\beta = 0.75$, we find that it begins with excellent behavior and produces a smoother search path as it approaches the minimizer.

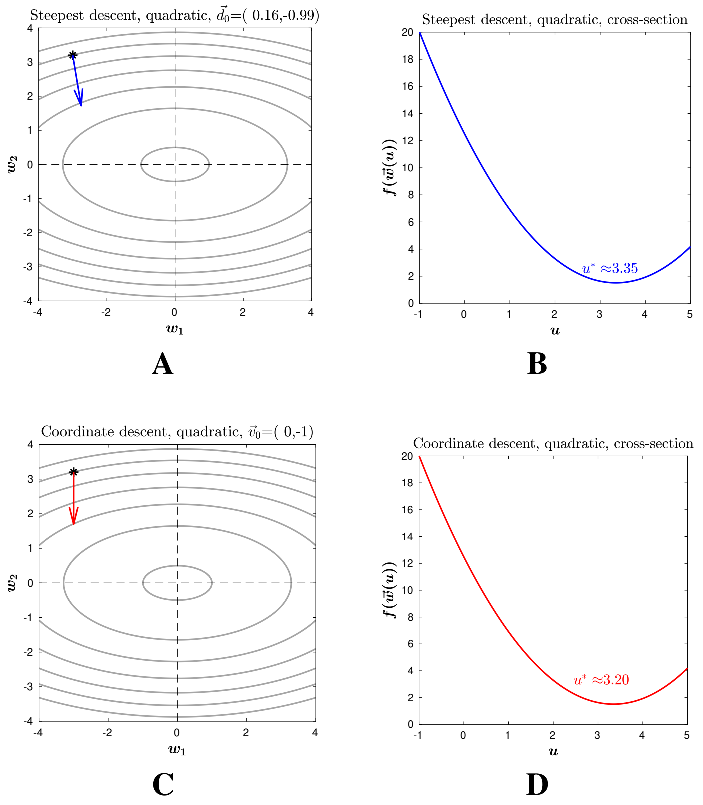


Figure 8.3: Descent directions from $\vec{w}_0$ for the quadratic function in Equation 8.7. (A) Contours are shown in gray and the direction of steepest descent is shown in blue. (B) A slice of $f_1(\vec{w})$ in the direction of steepest descent; the optimal step is approximately 3.35 units from $\vec{w}_0$. (C) Contours are shown in gray and the direction of coordinate descent is shown in magenta. (D) A slice of $f_1(\vec{w})$ in the direction of coordinate descent; the optimal step is approximately 3.20 units from $\vec{w}_0$.

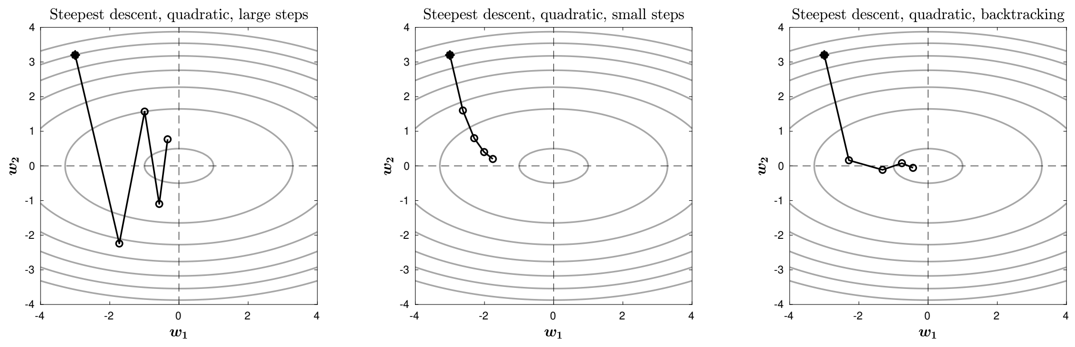
Example: exponential function for which the origin is a minimizer. Let us try using the function

$$f_2(\vec{w}) \quad = \quad -e^{\vec{w}^T K \vec{w}} \quad \text{where} \quad K = \begin{bmatrix} 1/4 & 0 \\ 0 & 1 \end{bmatrix} \tag{8.8}$$

Our starting point will be

$$\vec{w}_0 = \begin{bmatrix} -0.6 \\ 0.9 \end{bmatrix}$$

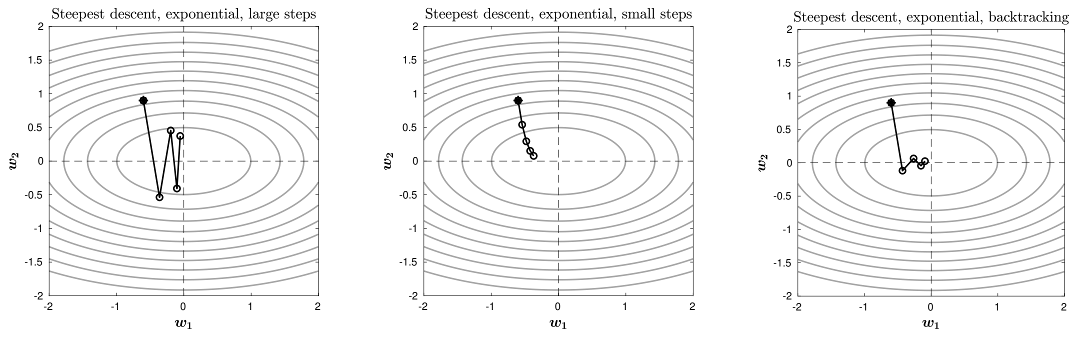
Our two fixed stepsizes will be $s = 4$ and $s = 1$, which have been selected to produce similar behaviour to the searches for the quadratic function $f_1(\vec{w})$. In Figure 8.5, we observe that locally quadratic approximation produces a path that is "jagged", with nearly orthogonal new search directions during the iteration. When we use backtracking with the Armijo conditions, setting



A **B**
C

Figure 8.4: Steepest descent of the function defined in Equation 8.7 using fixed and optimal stepsizes. (A) Stepsize of $s = 0.85$ produces an oscillating path in the search space. (B) Stepsize of $s = 0.25$ converges slowly towards the minimizer. (C) Backtracking converges slower than optimal stepsizes and with a less "jagged" search path.

$s = 10$ and $\beta = 0.75$, we find that it begins with excellent behavior and slows substantially as it approaches the minimizer. None of these methods is ideal for the function $f_2(\vec{w})$.



A **B**
C

Figure 8.5: Steepest descent of the exponential function defined in Equation 8.8 using fixed and variable stepsizes. (A) Stepsize of $s = 4$ produces an oscillating path in the search space. (B) Stepsize of $s = 1$ converges slowly towards the minimizer. (C) Backtracking with $s = 10$ and $\beta = 0.75$ converges slower than quadratically selected stepsizes and slows substantially near the minimizer.

The function $f_2(\vec{w})$ presents us with a conundrum. We do not know why convergence is relatively slow for backtracking, even with a relatively large initial stepsize. Some possible and interacting causes could be the nature of the function, the search direction, and the stepsize. We will explore some of these possibilities in the next class.

8.3 Extra Notes on Steepest Descent

We can use linear algebra to derive Equation 8.5. We will first define "steepest descent" for the Euclidean vector norm $\|\cdot\|_2$.

Definition: Euclidean direction of steepest descent

> For any continuously differentiable function $f:\mathbb{R}^n \to \mathbb{R}$, for any vector $\vec{w}_0 \in \mathbb{R}^n$ such that $\vec{w}_0$ is not a stationary point of f, the *Euclidean direction of steepest descent* is defined as
> $$\vec{d} = \underset{\vec{v}\in\mathbb{R}^n:\|\vec{v}\|_2=1}{\operatorname{argmin}} D_{\vec{v}}(\vec{w}_0) \tag{8.9}$$

Observation: When using the Euclidean norm, we can substitute the transpose of a vector in place of the 1-form in the directional derivative. As discussed in Class 6, we can represent Equation 5.5 in various ways:

$$\begin{aligned} D_{\vec{v}}f &= \sum_{j=1}^{n} \frac{\partial f}{\partial w_j} v_j \\ &= [\underline{\nabla} f]^T \cdot \vec{v} \end{aligned} \tag{8.10}$$

A fundamental theorem for vector norms is the Cauchy-Schwartz inequality. This inequality, for the dot product in a Euclidean vector space, is

$$|\vec{u}\cdot\vec{v}| \leq \|\vec{u}\|\|\vec{v}\| \tag{8.11}$$

The inequality of Equation 8.11 can be used to prove a basic lemma.

Lemma: minimum dot product

> *For any $\vec{u} \in \mathbb{R}^n \neq \vec{0}$, the unit vector $\vec{v} \in \mathbb{R}^n : \|\vec{v}\|_2 = 1$ that minimizes the Euclidean dot product $\vec{u}\cdot\vec{v}$ is*
> $$\frac{-\vec{u}}{\|\vec{u}\|_2} = \underset{\vec{v}\in\mathbb{R}^n:\|\vec{v}\|_2=1}{\operatorname{argmin}} \vec{u}\cdot\vec{v} \tag{8.12}$$

Proof: Let $\vec{u} \in \mathbb{R}^n$ and $\vec{v} \in \mathbb{R}^n : \|\vec{v}\|_2 = 1$. From the definition of the Euclidean norm,

$$\begin{aligned} \vec{u} \cdot \vec{u} &= \|\vec{u}\|_2^2 \\ \rightarrow \quad \vec{u} \cdot \frac{\vec{u}}{\|\vec{u}\|_2} &= \|\vec{u}\|_2 \\ \rightarrow \quad \vec{u} \cdot \frac{-\vec{u}}{\|\vec{u}\|_2} &= \|\vec{u}\|_2(-1) \end{aligned}$$

This equality, plus the Cauchy-Schwartz inequality of Equation 8.11, implies that $\dfrac{-\vec{u}}{\|\vec{u}\|_2}$ is a lower bound for $\vec{u} \cdot \vec{v}/\|\vec{v}\|_2$, so

$$\frac{-\vec{u}}{\|\vec{u}\|_2} = \underset{\vec{v} \in \mathbb{R}^n : \|\vec{v}\|_2 = 1}{\operatorname{argmin}} \; \vec{u} \cdot \vec{v}$$

This leads directly to the theorem for steepest descent.

Theorem: direction of steepest descent

For any continuous differentiable $f : \mathbb{R}^n \rightarrow \mathbb{R}$, for any $\vec{w}_0 \in \mathbb{R}^n$ such that $\vec{w}_0$ is not a stationary point of f, the direction of steepest descent is

$$\frac{-[\underline{\nabla} f]^T}{\|[\underline{\nabla} f]^T\|_2} = \underset{\vec{v} \in \mathbb{R}^n : \|\vec{v}\|_2 = 1}{\operatorname{argmin}} \; D_{\vec{v}} f(\vec{w}_0) \tag{8.13}$$

Proof: $\vec{w}_0$ is not a stationary point of f, so $\underline{\nabla} f(\vec{w}_0) \neq \underline{0}$ and $[\underline{\nabla} f(\vec{w}_0)]^T \neq \vec{0}$. Expanding the directional derivative, and using Lemma 8.12, is

$$\begin{aligned} \underset{\vec{v} \in \mathbb{R}^n : \|\vec{v}\|_2 = 1}{\operatorname{argmin}} \; D_{\vec{v}} f(\vec{w}_0) &= \underset{\vec{v} \in \mathbb{R}^n : \|\vec{v}\|_2 = 1}{\operatorname{argmin}} \; [\underline{\nabla} f(\vec{w}_0)]^T \cdot \vec{v} \\ &= \frac{-[\underline{\nabla} f(\vec{w}_0)]^T}{\|[\underline{\nabla} f(\vec{w}_0)]^T\|_2} \end{aligned}$$

End of Extra Notes

References

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

[2] Beck A: Introduction to Nonlinear Optimization: Theory, Algorithms, and Applications with MATLAB. Siam Press, 2014