

Lecture 27

Multi-Dimensional Optimization: Gradient-Based Algorithms

27.1 Lecture Objectives

- Understand the basic steps of gradient-based methods in general, and the steepest descent method in particular.
- Understand the connection of the steepest descent method to the first-order optimality conditions (zero gradient) studied in previous lectures.
- Understand the limitations and alternatives to the steepest descent method.

27.2 Steepest Descent

How would you climb down to the bottom of a large canyon in a heavy fog and without the help of technology? This challenge resembles the optimization problems we consider in this course. One simple approach would be to look at the ground beneath your feet (which you can see), figure out the direction of steepest descent, and walk in that direction in a straight line until you are no longer descending. Then, you would stare at the ground again to find the new direction of steepest descent, and repeat the procedure until there is no descent direction (ie: you are at the bottom of the canyon, or at least at a “local” bottom). This is, in essence, the steepest descent algorithm.

Note that, for differentiable functions, the direction of the gradient $\nabla f(\mathbf{x})$ is the direction of steepest ascent (most rapid increase) of the function. Conversely, the direction opposite to the gradient, ie: $-\nabla f(\mathbf{x})$ is the direction of steepest descent (most rapid decrease) of the function. This choice of direction can be updated iteratively, along with a step size $\alpha_k \geq 0$ (see below), leading to the steepest descent algorithm. This algorithm can be described as follows:

1. Initialize the algorithm with some initial guess $\mathbf{x}^{(0)}$

2. For each iteration k , proceed to the next estimate as follows:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \nabla f(\mathbf{x}^{(k)}) \quad (27.1)$$

for some choice of $\alpha_k \geq 0$

3. Stop the iterations when the gradient is sufficiently close to zero, or when a pre-determined number of iterations is reached.

Now, the question is how to determine α_k at each iteration. In the method of steepest descent, α_k is determined via a one-dimensional line search (see previous lecture) that solves the problem:

$$\alpha_k = \arg \min_{\alpha \geq 0} f(\mathbf{x}^{(k)} - \alpha \nabla f(\mathbf{x}^{(k)})) \quad (27.2)$$

An interesting property of the steepest descent algorithm is that consecutive steps will have directions that are orthogonal to each other (see, eg: Chong and Zak, An Introduction to Optimization). Sometimes, this results in optimization trajectories that include a lot of zig-zagging as they approach the optimizer. This effect is illustrated in figure [27.1](#).

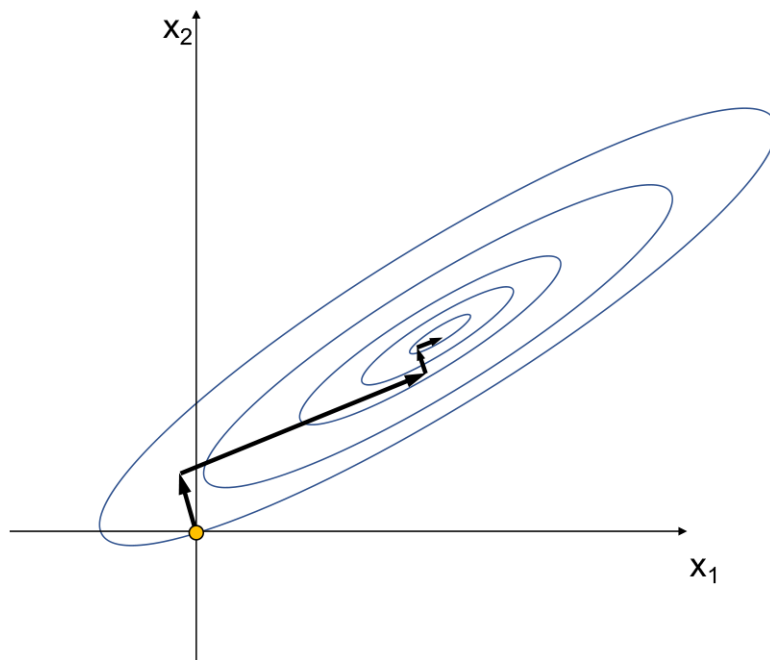


Figure 27.1: Pictorial depiction of the steepest descent algorithm, which leads to orthogonal consecutive directions and possibly substantial zig-zag (ie: requiring many iterations to approach a minimizer).

27.2.1 A simple example

Consider the following optimization problem:

$$\min_{\mathbf{x}} \frac{1}{2} \|\mathbf{Ax} - \mathbf{d}\|_2^2$$

where $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, $\mathbf{A} = \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix}$, and $\mathbf{d} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

Now, let us solve this problem by using the steepest descent (SD) algorithm, where we start with initial guess $\mathbf{x}^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$. Using calculations “by hand,” we will describe the next two steps of the (SD) algorithm, ie: determine $\mathbf{x}^{(1)}$ and subsequently (if the algorithm is not done), determine $\mathbf{x}^{(2)}$. Note that in this case, the line search part of SD (to determine the step size α) can be easily solved by hand (analytically, without need for iterative algorithms) because the function is quadratic.

This example is illustrated in figure [27.2](#).

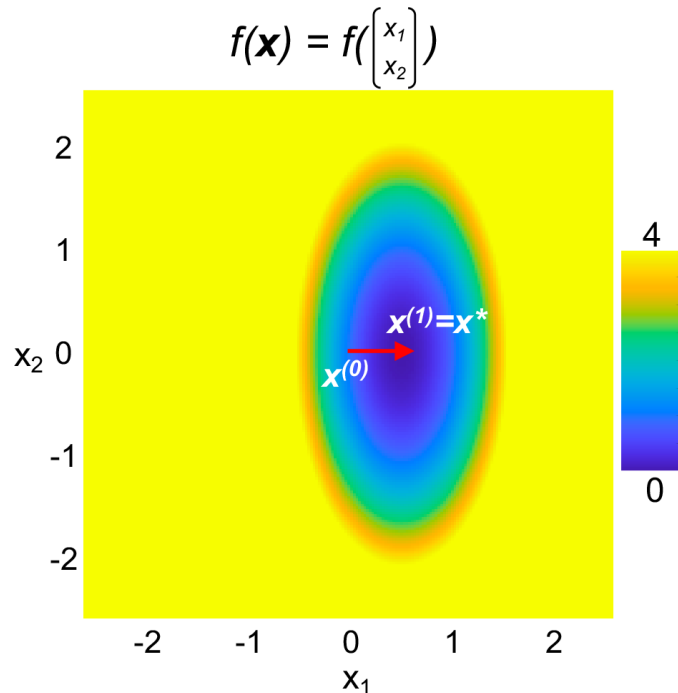


Figure 27.2: Pictorial depiction of the Steepest Descent method in a case where a single iteration leads to the solution.

First iteration

For each iteration, we need to do two things: 1) calculate the descent direction (ie: the negative gradient), and 2) determine the step size (via a 1D line search). Let’s start with

the search (descent) direction, which will require the gradient. Our gradient is expressed as:

$$\begin{aligned}\nabla f(\mathbf{x}^{(0)}) &= \mathbf{A}^T \mathbf{A} \mathbf{x}^{(0)} - \mathbf{A}^T \mathbf{d} \\ &= \begin{pmatrix} 4 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix} - \begin{pmatrix} 2 \\ 0 \end{pmatrix} \\ &= -\begin{pmatrix} 2 \\ 0 \end{pmatrix}\end{aligned}$$

And so our search direction for the first step will be $-\nabla f(\mathbf{x}^{(0)}) = \begin{pmatrix} 2 \\ 0 \end{pmatrix}$.

For the second part, ie: finding the step size α^* , we need to solve the following 1D minimization problem to minimize $f(\mathbf{x}^{(0)} - \alpha \nabla f(\mathbf{x}^{(0)}))$:

$$\begin{aligned}\alpha^* &= \arg \min_{\alpha} \frac{1}{2} \|\mathbf{A}(\mathbf{x}^{(0)} - \alpha \nabla f(\mathbf{x}^{(0)})) - \mathbf{d}\|_2^2 \\ &= \arg \min_{\alpha} \frac{1}{2} \left\| \alpha \begin{pmatrix} 2 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 2 \\ 0 \end{pmatrix} - \begin{pmatrix} 1 \\ 0 \end{pmatrix} \right\|_2^2 \\ &= \arg \min_{\alpha} \frac{1}{2} (4\alpha - 1)^2\end{aligned}$$

which can be observed by inspection to have its minimum at $\alpha^* = \frac{1}{4}$. Alternatively, one could differentiate with respect to α and set the derivative to zero, ie: $(4\alpha^* - 1) = 0$ which again leads to the same solution $\alpha^* = \frac{1}{4}$.

Now that we know the search direction and the step size, we can write our next ‘guess’ $\mathbf{x}^{(1)}$ as:

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} - \alpha^* \nabla f(\mathbf{x}^{(0)}) = \begin{pmatrix} 0.5 \\ 0 \end{pmatrix}$$

Second iteration

For the second iteration, we would repeat the two components of the first iteration, but based on our updated guess $\mathbf{x}^{(1)}$:

$$\begin{aligned}\nabla f(\mathbf{x}^{(1)}) &= \mathbf{A}^T \mathbf{A} \mathbf{x}^{(1)} - \mathbf{A}^T \mathbf{d} \\ &= \begin{pmatrix} 4 & 0 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 0.5 \\ 0 \end{pmatrix} - \begin{pmatrix} 2 \\ 0 \end{pmatrix} \\ &= \begin{pmatrix} 0 \\ 0 \end{pmatrix}\end{aligned}$$

But as you can see the gradient is now zero, so we cannot move further. The reason for this is that, for this problem, a single iteration happened to solve the problem. So after the first iteration and checking the gradient, we are done with our SD algorithm and our solution is $\mathbf{x}^* = \begin{pmatrix} 0.5 \\ 0 \end{pmatrix}$

27.2.2 Another example

Consider the following optimization problem:

$$\min_{\mathbf{x}} \frac{1}{2} \|\mathbf{Ax} - \mathbf{d}\|_2^2$$

where $\mathbf{x} = \begin{pmatrix} x_1 \\ x_2 \end{pmatrix}$, $\mathbf{A} = \begin{pmatrix} 1 & 1 \\ 0 & 1 \end{pmatrix}$, and $\mathbf{d} = \begin{pmatrix} 1 \\ -1 \end{pmatrix}$.

In this case, we will write our cost function as a standard quadratic form, as follows:

$$\min_{\mathbf{x}} \frac{1}{2} \mathbf{x}^T \mathbf{Q} \mathbf{x} - \mathbf{x}^T \mathbf{b} + c$$

where $\mathbf{Q} = \mathbf{A}^T \mathbf{A} = \begin{pmatrix} 1 & 1 \\ 1 & 2 \end{pmatrix}$, $\mathbf{b} = \mathbf{A}^T \mathbf{d} = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$, and $c = 2$ (although c is just a constant, so it does not really affect our algorithm). Expressing our cost function in this form does not change anything, but the quadratic form is common in optimization (including in the homework within this course), so it is helpful to be familiar with this way of writing a quadratic cost function.

Just like in the previous example, let us solve this problem by using the SD algorithm, where we start with initial guess $\mathbf{x}^{(0)} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$.

This example is illustrated in figure [27.3](#).

First iteration

Let's start with the search (descent) direction, which will require the gradient. Our gradient is expressed as:

$$\nabla f(\mathbf{x}^{(0)}) = \mathbf{Q}\mathbf{x}^{(0)} - \mathbf{b} = \begin{pmatrix} -1 \\ 0 \end{pmatrix}$$

And so our search direction for the first step will be $-\nabla f(\mathbf{x}^{(0)}) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$.

For the second part, ie: finding the step size α^* , we need to solve the following 1D minimization problem to minimize $f(\mathbf{x}^{(0)} - \alpha \nabla f(\mathbf{x}^{(0)}))$:

$$\begin{aligned} \alpha^* &= \arg \min_{\alpha} \frac{1}{2} (\mathbf{x}^{(0)} - \alpha \nabla f(\mathbf{x}^{(0)}))^T \mathbf{Q} (\mathbf{x}^{(0)} - \alpha \nabla f(\mathbf{x}^{(0)})) - (\mathbf{x}^{(0)} - \alpha \nabla f(\mathbf{x}^{(0)}))^T \mathbf{b} + c \\ &= \arg \min_{\alpha} \frac{1}{2} \alpha^2 - \alpha + 2 \end{aligned}$$

which can be observed by setting the derivative $(\alpha - 1)$ to zero, to have its minimum at $\alpha^* = 1$.

Now that we know the search direction and the step size, we can write our next 'guess' $\mathbf{x}^{(1)}$ as:

$$\mathbf{x}^{(1)} = \mathbf{x}^{(0)} - \alpha^* \nabla f(\mathbf{x}^{(0)}) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}$$

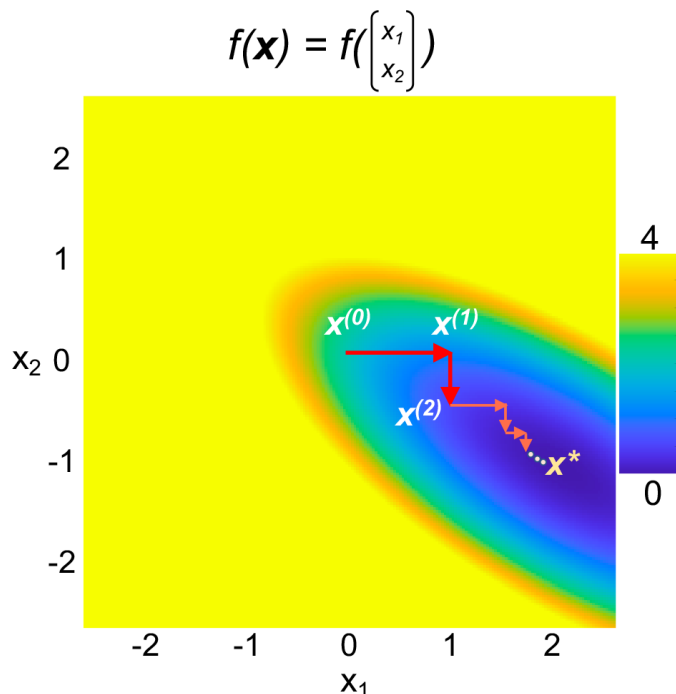


Figure 27.3: Pictorial depiction of the Steepest Descent method in a case where multiple iterations are needed to approach the solution, leading to a zig-zag trajectory.

Second iteration

Once again, let's calculate the gradient at the current location $\mathbf{x}^{(1)}$:

$$\nabla f(\mathbf{x}^{(1)}) = \mathbf{Q}\mathbf{x}^{(1)} - \mathbf{b} = \begin{pmatrix} 0 \\ 1 \end{pmatrix}$$

And so our search direction for the first step will be $-\nabla f(\mathbf{x}^{(0)}) = \begin{pmatrix} 0 \\ -1 \end{pmatrix}$.

To find the step size α^* , we need to solve the following 1D minimization problem to minimize $f(\mathbf{x}^{(1)} - \alpha \nabla f(\mathbf{x}^{(1)}))$:

$$\begin{aligned} \alpha^* &= \arg \min_{\alpha} \frac{1}{2} (\mathbf{x}^{(1)} - \alpha \nabla f(\mathbf{x}^{(1)}))^T \mathbf{Q} (\mathbf{x}^{(1)} - \alpha \nabla f(\mathbf{x}^{(1)})) - (\mathbf{x}^{(1)} - \alpha \nabla f(\mathbf{x}^{(1)}))^T \mathbf{b} + c \\ &= \arg \min_{\alpha} 2\alpha^2 - 2\alpha + 1.5 \end{aligned}$$

which can be observed by setting the derivative $(4\alpha - 2)$ to zero, to have its minimum at $\alpha^* = 1/2$.

Now that we know the search direction and the step size, we can write our next 'guess' $\mathbf{x}^{(2)}$ as:

$$\mathbf{x}^{(2)} = \mathbf{x}^{(1)} - \alpha^* \nabla f(\mathbf{x}^{(1)}) = \begin{pmatrix} 1 \\ -0.5 \end{pmatrix}$$

More iterations

In this case, we are not yet done, as the gradient is not yet zero and we have not reached the minimum of the cost function. For example, the gradient at the current location is

$$\nabla f(\mathbf{x}^{(2)}) = \begin{pmatrix} -0.5 \\ 0 \end{pmatrix}$$

In the steepest descent method, successive descent directions are perpendicular to each other. In this example (see figure 27.3), the direction for the first iteration happens to be “horizontal” along x_1 , so the direction for the second iteration is “vertical” along x_2 , the direction for the third iteration is again horizontal, and so on. This leads to the limitation that steepest descent does “zig-zagging” and requires many iterations to approach the solution in cases where the “shape” of the cost function is not favorable.

27.2.3 Overcoming the limitations of the steepest descent algorithm

This relatively slow convergence can potentially be alleviated through a variety of approaches, including:

1. Taking shorter steps, ie: re-evaluating the gradient before the step size (α_k determined by a line search algorithm). In the hiking analogy given above, this would be equivalent to considering changing direction before the current direction is ‘exhausted’ (ie: before reaching the minimizer of each step’s line search).
2. Using different directions. For instance, instead of using directions (gradients) that are orthogonal to the previous direction, a successful algorithm is based on using directions that are ‘conjugate’ to each other (ie: $\mathbf{d}_j^T \mathbf{H} \mathbf{d}_k = 0$, for any pair of directions $\mathbf{d}_j$ and $\mathbf{d}_k$ used in different iterations, and for a matrix $\mathbf{H}$ that represents the Hessian of the function $f(\mathbf{x})$). This is termed the conjugate directions (or conjugate gradients)¹ method and will be explored further in the homework set.
3. Using higher-order derivatives, as in Newton’s method. This approach will be described in the following lecture.

27.3 Dealing with Constraints

The gradient-based algorithms described above are focused on unconstrained optimization problems. However, constrained optimization problems (including equality and/or inequality constraints) are common in imaging. Here we review common strategies for applying gradient-based algorithms to solve constrained optimization problems.

¹For further details into the CG algorithm, there are several references and texts, including the uniquely titled “An Introduction to the Conjugate Gradient Method Without the Agonizing Pain” by Jonathan Shewchuk, available at <https://www.cs.cmu.edu/quake-papers/painless-conjugate-gradient.pdf>

- *Indirect methods:* Instead of directly solving the constrained optimization problem, we can approximate the constraints (which determine a feasible or constraint set Ω , and impose that $\mathbf{x} \in \Omega$) through additional penalty functions that are added to the cost function: $\lambda\phi_\Omega(\mathbf{x})$. These additional penalty functions penalize values of $\mathbf{x}$ that are outside Ω . By creating a sequence of penalty functions $\lambda_t\phi_{t,\Omega}(\mathbf{x})$ such that in the limit they approximate the original ‘hard’ constraint (ie: infinite penalty for solutions outside of Ω), we can solve the original constrained problem through a sequence of unconstrained problems:

$$\min_{\mathbf{x}} f(\mathbf{x}) + \lambda_t\phi_{t,\Omega}(\mathbf{x})$$

as $t \rightarrow \infty$. Note that the solution to the intermediate problems (for low values of t) may lie outside the constraint set Ω , however, we expect that the solution will be within Ω as $t \rightarrow \infty$. Note that this indirect approach to solving constrained optimization problems is not specific to gradient-based methods, and indeed can be used in combination with a wide variety of optimization algorithms.

- *Direct methods:* These methods attempt to directly solve the original constrained optimization problem by starting out at a feasible point, and then descending along feasible directions (ie: directions that do not leave the constraint set Ω at least for small values of the step size α_k). For the steepest descent method, an approach that allows descent in constrained optimization problems is the ‘projected gradient’ approach, where the steepest descent direction $-\nabla f(\mathbf{x}^{(k)})$ (which itself may not be feasible if $\mathbf{x}^{(k)}$ is at the boundary of the constraint set Ω) is projected onto the set of feasible directions.

In this course, we will not study these algorithms in depth due to time limitations. Further depth is available in various texts and references, including Bertsekas’ “Nonlinear Programming” and Chong and Zak’s “An Introduction to Optimization”.