Lecture 9

Multi-Dimensional Optimization: Gradient-Based Algorithms

9.1 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.

9.2 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:

$$\mathbf{x}^{(k+1)} = \mathbf{x}^{(k)} - \alpha_k \nabla f(\mathbf{x}^{(k)}) \tag{9.1}$$

   for some choice of $\alpha_k \geq 0$

3. Stop the iterations when the gradient is sufficiently close to zero, or when a predetermined 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:

$$\alpha_k = \arg\min_{\alpha \geq 0} \mathbf{x}^{(k)} - \alpha \nabla f(\mathbf{x}^{(k)}) \tag{9.2}$$

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 9.1.

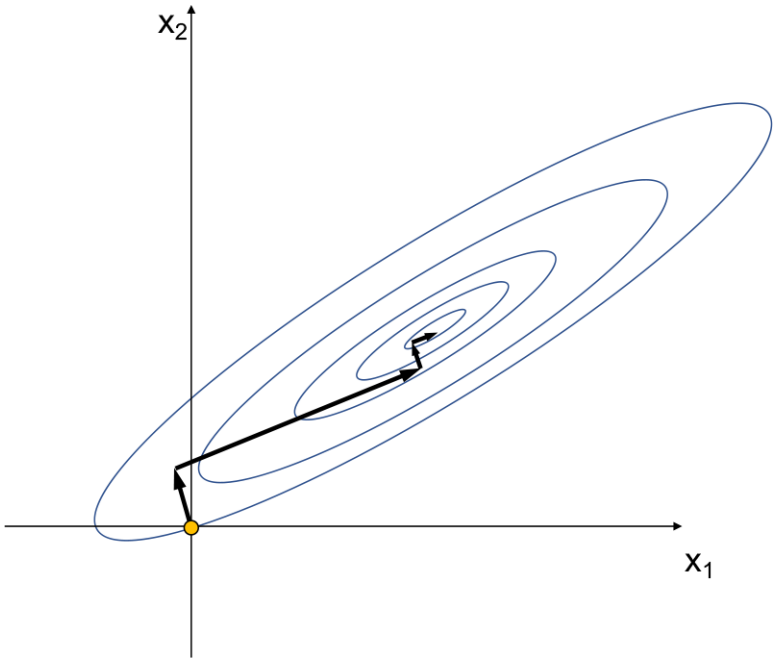

Figure 9.1: Pictorial depiction of the steepest descent algorithm, which leads to orthogonal consecutive directions and possibly substantial zig-zag (ie: requiring many iterations to approach a minimizer).

9.2.1 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: $\mathbf{d}_j^T \mathbf{H} \mathbf{d}_k$ for 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.

9.3 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.

- *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 $t \to \infty$. 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

[1] For 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

be within Ω as $t \to \infty$. 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".