E0 270 Machine Learning **Jan 16, 2015**

Optimization Tutorial 1

Basic Gradient Descent

Lecture by Harikrishna Narasimhan

Note: This tutorial shall assume background in elementary calculus and linear algebra.

In many real-world applications of computer science, engineering, and economics, one often encounters numerical optimization problems of the following form, where the goal is to find the minimum of a real-valued function $f : \mathbb{R}^d \to \mathbb{R}$ over a (closed) constraint set $\mathcal{C} \subseteq \mathbb{R}^d$:

$$\min_{\mathbf{x} \in \mathcal{C}} f(\mathbf{x}).$$

In this three-part tutorial, we shall discuss theory and algorithms for solving such problems. We begin in this part with the basic gradient descent method for solving unconstrained optimization problems where $\mathcal{C} = \mathbb{R}^d$; in the next two parts of the tutorial, we shall focus on constrained optimization problems.[1]

One of the simplest examples of a unconstrained optimization problem is the following quadratic optimization problem, which we shall use as a running example in this document:

$$\min_{\mathbf{x} \in \mathbb{R}^d} f_{\text{quad}}(\mathbf{x}) = \frac{1}{2}\mathbf{x}^\top \mathbf{A}\mathbf{x} - \mathbf{b}^\top \mathbf{x}, \tag{OP1}$$

where $\mathbf{A} \in \mathbb{R}^{d \times d}$ and $\mathbf{b} \in \mathbb{R}^d$. Before we proceed to discuss methods for solving problems of the above form, we will find it useful to first provide some preliminaries on vector calculus and linear algebra.

1 Preliminaries

Throughout this document, we shall assume that the given function $f : \mathbb{R}^d \to \mathbb{R}$ that we wish to optimize is bounded below, i.e., $\exists\, B \in \mathbb{R}$ s.t. $f(\mathbf{x}) \geq B$, $\forall \mathbf{x} \in \mathbb{R}^d$.

Gradient and Hessian. The gradient of a multi-dimensional function $f : \mathbb{R}^d \to \mathbb{R}$ is a generalization of the first derivative of a one-dimensional function, and is given by a vector of first-order partial derivatives of f (if they exist):

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \frac{\partial f(\mathbf{x})}{\partial x_1} \\ \frac{\partial f(\mathbf{x})}{\partial x_2} \\ \vdots \\ \frac{\partial f(\mathbf{x})}{\partial x_d} \end{bmatrix}.$$

The Hessian of f is a generalization of the second derivative in one-dimension, and is a matrix of second-order partial derivatives of f (if they exist):

$$\nabla^2 f(\mathbf{x}) = \begin{bmatrix} \frac{\partial^2 f(\mathbf{x})}{\partial x_1^2} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_1 \partial x_d} \\ \vdots & \ddots & \vdots \\ \frac{\partial^2 f(\mathbf{x})}{\partial x_d \partial x_1} & \cdots & \frac{\partial^2 f(\mathbf{x})}{\partial x_d^2} \end{bmatrix}.$$

[1] While we focus on minimization problems in the tutorial, the theory and methods discussed apply to maximization problems as well.

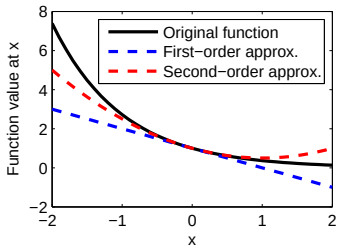

Figure 1: Plot of $f(x) = e^{-x}$ and its first-order and second-order Taylor approximations at $x_0 = 0$.

It can be verified that for the quadratic function in (OP1), $\nabla f(\mathbf{x}) = \mathbf{A}\mathbf{x} - \mathbf{b}$ and $\nabla^2 f(\mathbf{x}) = \mathbf{A}$. Also, let C^k denote the set of all real-valued functions whose k^{th}-order partial derivatives exist and are continuous.

Taylor series. We will often require to approximate a function in terms of its gradient and Hessian at a particular point $\mathbf{x}_0 \in \mathbb{R}^d$. For this, we shall resort to the Taylor expansion of the functions at $\mathbf{x}_0$. More specifically, the first-order Taylor expansion of a function $f : \mathbb{R}^d \to \mathbb{R}$ in C^1 is given by

$$f(\mathbf{x}) \;=\; f(\mathbf{x}_0) \,+\, \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0) \,+\, o(\|\mathbf{x} - \mathbf{x}_0\|_2), \tag{1}$$

while the second-order Taylor expansion of a function f in C^2 is given by

$$f(\mathbf{x}) \;=\; f(\mathbf{x}_0) \,+\, \nabla f(\mathbf{x}_0)^\top (\mathbf{x} - \mathbf{x}_0) \,+\, \frac{1}{2}(\mathbf{x} - \mathbf{x}_0)^\top \nabla^2 f(\mathbf{x}_0)(\mathbf{x} - \mathbf{x}_0) \,+\, o(\|\mathbf{x} - \mathbf{x}_0\|_2^2),$$

where $u(t) = o(v(t))$ shall denote $\lim_{t \to 0} \frac{u(t)}{v(t)} = 0$, i.e., $u(t)$ goes to 0 faster than $v(t)$ as $t \to 0$; this tells us that when $\|\mathbf{x} - \mathbf{x}_0\|_2^2$ is sufficiently small, the last term in the above expansions can be effectively ignored. For the one-dimensional function $f(x) = e^{-x}$, the first-order Taylor approximation at 0 is $1 - x$, while the second-order Taylor approximation at 0 is $1 - x + \frac{1}{2}x^2$; see Figure 1 for an illustration.

Positive (semi-)definiteness. Finally, a matrix $\mathbf{A} \in \mathbb{R}^{d \times d}$ is called positive semi-definite (p.s.d.) if $\forall \mathbf{x} \in \mathbb{R}^d, \mathbf{x} \neq 0$, $\mathbf{x}^\top \mathbf{A} \mathbf{x} \geq 0$ and positive definite (p.d.) if this inequality is strict; it can be verified that a p.s.d (p.d.) matrix has non-negative (positive) eigen values.

2 Necessary and Sufficient Conditions for Optimality

As a first step towards solving an unconstrained optimization problem, we provide necessary and sufficient conditions for the minimizer of a function in C^2. While our ideal goal is to find the global minimizer of f, i.e., a point $\mathbf{x}^* \in \mathbb{R}^d$ such that $f(\mathbf{x}^*) \leq f(\mathbf{x}), \forall \mathbf{x} \in \mathbb{R}^d$, due to practical considerations, often one needs to settle with a solution that is locally optimum:

Definition 1. A point $\mathbf{x}^* \in \mathbb{R}^d$ is said to be a local minimizer of a function $f : \mathbb{R}^d \to \mathbb{R}$ if there exists $\epsilon > 0$ such that $f(\mathbf{x}^*) \leq f(\mathbf{x})$ for all $\mathbf{x} \in \mathbb{R}^d$ within an ϵ-distance of $\mathbf{x}^*$, i.e., $\|\mathbf{x} - \mathbf{x}^*\|_2 \leq \epsilon$.

The following characterization of a local minimizer of f is well-known; see for example [3].

Proposition 1 (Necessary and sufficient conditions for local optimality). Let $f : \mathbb{R}^d \to \mathbb{R}$ be a function in C^2. If $\mathbf{x}^* \in \mathbb{R}^d$ is a local minimizer of f, then

1. $\nabla f(\mathbf{x}^*) = \mathbf{0}$
2. $\nabla^2 f(\mathbf{x}^*)$ is p.s.d.

Moreover, if for a given $\mathbf{x}^*$, $\nabla f(\mathbf{x}^*) = \mathbf{0}$ and $\nabla^2 f(\mathbf{x}^*)$ is p.d., then $\mathbf{x}^*$ is a local minimizer of f.

Let us apply the above characterization to the optimization problem in (OP1). Equating the gradient of f_{quad} in this problem to zero we have $\nabla f_{\text{quad}}(\mathbf{x}^*) = \mathbf{A}\mathbf{x}^* - \mathbf{b} = \mathbf{0}$. If $\mathbf{A}$ is invertible, then $\mathbf{x}^* = \mathbf{A}^{-1}\mathbf{b}$; further, if $\nabla^2 f_{\text{quad}}(\mathbf{x}^*) = \mathbf{A}$ is p.d., we have that $\mathbf{x}^*$ is a local minimizer of f_{quad}.

Clearly, in order to minimize a function f over $\mathbb{R}^d$, we will have to find a point with zero gradient; the Hessian at this point can then be used to determine if this point is indeed a (local) minimizer of f. In the next section, we describe a technique for finding a point in $\mathbb{R}^d$ where the gradient of f is zero.

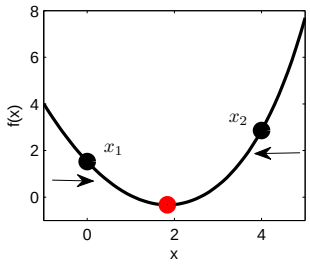

Figure 2: Plot of a one-dimensional function f. The minimizer of the function is indicated by a red mark. Note that the slope/gradient of f at a point x_1 to the left of the minimizer is positive, while that at point x_2 to the right of the minimizer is negative. The arrows indicate directions that yield lower values of f compared to the current point.

3 Basic Gradient Descent

We now describe an iterative method for optimizing a given function f; this method starts with an initial point and generates a sequence of points whose gradients converge to zero.

We begin by building some intuition using the example one-dimensional function shown in Figure 2. Observe that at a point x_1 lying to the left of the minimizer of this function (indicated with a red mark), the slope/gradient of this function is negative; this suggests that one can decrease the value of the function by moving to the right of x_1. Similarly, at a point x_2 to the right of the minimizer, the slope/gradient of the function is positive, suggesting that one can decrease the function value by moving to the left of x_2. In both cases, moving in a direction opposite to the sign of the function slope at a point produces a decrease in its value. In fact, we shall see now that for any function $f : \mathbb{R}^d \to \mathbb{R}$ in C^1, one can decrease the value of f at a point, by moving in a direction opposite to that of the gradient of f at that point .

In particular, let $\mathbf{x}_0 \in \mathbb{R}^d$ and $\mathbf{y} \in \mathbb{R}^d$ be obtained by subtracting from $\mathbf{x}_0$ the gradient of f scaled by a factor $\eta > 0$, i.e., $\mathbf{y} = \mathbf{x}_0 - \eta \nabla f(\mathbf{x}_0)$. From the first-order Taylor expansion of f at $\mathbf{x}_0$ (see Eq. (1)), we have

$$\begin{aligned} f(\mathbf{y}) &= f(\mathbf{x}_0) + \nabla f(\mathbf{x}_0)(\mathbf{y} - \mathbf{x}_0) + o(\|\mathbf{y} - \mathbf{x}_0\|_2) \\ &= f(\mathbf{x}_0) - \eta \|\nabla f(\mathbf{x}_0)\|^2 + o(\eta \|\nabla f(\mathbf{x}_0)\|_2). \end{aligned}$$

Clearly, if η is sufficiently small, the third term in the above expression becomes negligible compared to the first two terms, giving us $f(\mathbf{y}) < f(\mathbf{x}_0)$. Based on this observation, we now describe an iterative method for optimizing f, where each iteration generates a new point by moving in the direction of the negative gradient at the current point; this method is popularly known as the gradient descent method.

Gradient Descent Method

Input: $f : \mathbb{R}^d \to \mathbb{R}$
Initialize: $\mathbf{x}_0 \in \mathbb{R}^d$
Parameter: T
$t = 0$
for $t = 1$ **to** T
- Select step-size $\eta_t > 0$
- $\mathbf{x}_{t+1} = \mathbf{x}_t - \eta_t \nabla f(\mathbf{x}_t)$

Output: $\mathbf{x}_{T+1}$

As seen earlier, by choosing an appropriate step-size η_t in each iteration t, we will have $f(\mathbf{x}_{t+1}) < f(\mathbf{x}_t)$. It is however not immediately clearly how one should choose η_t in each iteration. While a small value of η_t will result in slow convergence, with a large value, we will not be able to guarantee a decrease in f after every iteration. Below, we describe two schemes for selecting η_t, both of which come with convergence guarantees.

Exact line search. A natural first-cut approach for step-size selection would be to choose a value that produces the maximum decrease in function value at the given iteration:

$$\eta_t \in \operatorname*{argmin}_{\eta > 0} f(\mathbf{x}_t - \eta \nabla f(\mathbf{x}_t)). \tag{2}$$

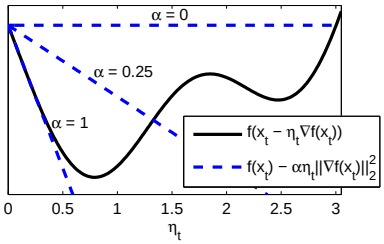

Figure 3: Illustration of backtracking line search for an example function $f : \mathbb{R}^d \to \mathbb{R}$. The solid black curve is a plot of $f(\mathbf{x}_{t+1}) = f(\mathbf{x}_t - \eta_t \nabla f(\mathbf{x}_t))$ as a function of step-size η_t. The dotted blue curves are plots of the linear function $f(\mathbf{x}_t) - \alpha\eta_t \|\nabla f(\mathbf{x}_t)\|_2^2$ as a function of η_t for different values of α. When $\alpha = 1$, this gives us a line that lies below the solid curve; when $\alpha < 1$, we get a line that has portions above the solid curve, with $\alpha = 0$ corresponding to a horizontal line. Given a value of $\alpha \in (0, 1)$, the backtracking line search scheme aims to find a value of η_t for which $f(\mathbf{x}_t - \eta_t \nabla f(\mathbf{x}_t)) \leq f(\mathbf{x}_t) - \alpha\eta_t \|\nabla f(\mathbf{x}_t)\|_2^2$; as seen from the plots, for any value of $\alpha \in (0, 1)$, such a η_t always exists.

Note that the above sub-problem is itself an optimization problem (involving a single variable). In some cases, this sub-problem can be solved analytically, resulting in a simple closed-form solution for η_t. For example, in the case of the quadratic problem in (OP1), the above exact line search scheme gives us:

$$\eta_t = \frac{\|\nabla f(\mathbf{x}_t)\|_2^2}{\nabla f(\mathbf{x}_t)^\top \mathbf{A} \nabla f(\mathbf{x}_t)}.$$

However, in many optimization problems of practical importance, it is often quite expensive to solve this inner line search problem exactly. In such cases, one resorts to an approximate approach for solving Eq. (2); we next describe one such scheme.

Inexact line search. An alternate to the above exact line search approach is a simple heuristic where one starts with a large value of η and multiplicatively decreases this value until $f(\mathbf{x}_t - \eta \nabla f(\mathbf{x}_t)) < f(\mathbf{x}_t)$. However, this scheme may not yield sufficient progress in each iteration to provide convergence guarantees for the gradient descent method. Instead, consider the following sophisticated version of this heuristic that does indeed guarantee a sufficient decrease in the function value after each iteration [1]:

Inexact (Backtracking) Line Search

Input: $f : \mathbb{R}^d \to \mathbb{R}$, $\mathbf{x}_t$
Parameter: $\alpha \in (0, 1)$, $\beta \in (0, 1)$
$\eta_t = 1$
while $f(\mathbf{x}_t - \eta_t \nabla f(\mathbf{x}_t)) > f(\mathbf{x}_t) - \alpha\eta_t \|\nabla f(\mathbf{x}_t)\|_2^2$
    $\eta_t = \beta\eta_t$
Output: η_t

Recall that the first order Taylor approximation of $f(\mathbf{x}_t - \eta_t \nabla f(\mathbf{x}_t))$ is $f(\mathbf{x}_t) - \eta_t \|\nabla f(\mathbf{x}_t)\|_2^2$. The above scheme aims to find a step-size η_t that is lesser than a scaled version of this linear approximation: $f(\mathbf{x}_t) - \alpha\eta_t \|\nabla f(\mathbf{x}_t)\|_2^2$. As seen from Figure 3, such a value of η_t always exists, ensuring that the above procedure terminates. Moreover, at termination, we will have $f(\mathbf{x}_{t+1}) = f(\mathbf{x}_t - \eta_t \nabla f(\mathbf{x}_t)) \leq f(\mathbf{x}_t) - \alpha\eta_t \|\nabla f(\mathbf{x}_t)\|_2^2 < f(\mathbf{x}_t)$, as desired. Note that the parameter α now gives us a handle on the amount decrease in function value after each iteration and will determine the speed of convergence of the gradient descent method.

Under (a slight variant of) the above backtracking line search scheme (and under a certain smoothness assumption on the function being optimized), the gradient descent method can be shown to generate a sequence of iterates whose gradients converge to zero; see for example [2] for a formal guarantee. Moreover, when the function being optimized satisfies certain additional properties, one can give explicit rates of convergence for the gradient descent method (under both the exact and backtracking line search schemes), i.e., bound the number of iterations required by the method to reach a solution that is within a certain factor of the optimal function value [1]. We shall discuss these guarantees in the second part of this tutorial.

4 Next Lecture

In the next lecture, we shall discuss the Newton's method for unconstrained optimization that has better convergence guarantees than the gradient descent method described in the previous section. Following this, we shall start with constrained optimization and look at the Karush-Kuhn-Tucker (KKT) conditions for optimality of a solution to a constrained optimization problem.

References

[1] S. Boyd and L. Vandenberghe. *Convex Optimization.* Cambridge University Press, 2004.

[2] R. Fletcher. *Practical methods of optimization.* John Wiley & Sons, 2nd edition, 1987.

[3] D. Luenberger and Y. Ye. *Linear and Nonlinear Programming.* Springer, 3nd edition, 2008.

Practice Exercise Questions

1. **Second-order conditions for optimality.** You have been asked to find the minimizer of a (bounded, continuous, twice differentiable) function $f : \mathbb{R}^2 \to \mathbb{R}$ and run the gradient descent method on this function with three different initial points. Suppose the three runs of the method give you three different final points $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3 \in \mathbb{R}^2$ with Hessian matrices:

$$\nabla^2 f(\mathbf{x}_1) \;=\; \begin{bmatrix} 2 & -1 \\ -1 & 2 \end{bmatrix}; \qquad \nabla^2 f(\mathbf{x}_2) \;=\; \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix}; \qquad \nabla^2 f(\mathbf{x}_3) \;=\; \begin{bmatrix} 4 & 2 \\ 2 & 1 \end{bmatrix}.$$

   Which among the above points will you declare as a (local) minimizer of f?

2. **Gradient descent with constant step size.** As a part of a course assignment, you are required to minimize the following quadratic function $g : \mathbb{R}^2 \to \mathbb{R}$ using the gradient descent method:

$$g(x_1, x_2) \;=\; 4x_1^2 \;-\; 2x_1x_2 \;+\; 4x_2^2 \;-\; 5x_1 \;+\; 3x_2 \;-\; 1.$$

   You choose to implement the exact line search strategy to select the step size in each iteration of the method, and obtain the minimizer of g. Surprisingly, your friend tells you that he obtained the same solution by running the steepest descent method on g with a constant step size of 0.1. Can you give an explanation for why your friend's approach would work for g? *Hint: Observe that the step size used by your friend is the reciprocal of the largest eigen value of the Hessian of g at any point.*