Lecture 4: CS395T Numerical Optimization for Graphics and AI — Fundamentals of Unconstrained Optimization

Qixing Huang
The University of Texas at Austin
huangqx@cs.utexas.edu

1 Disclaimer

This note is adapted from Section 2 of

- *Numerical Optimization* by Jorge Nocedal and Stephen J. Wright. Springer series in operations research and financial engineering. Springer, New York, NY, 2. ed. edition, (2006)

2 Classifying Optimization Problems

2.1 Unconstrained versus Constrained Optimization

Optimization can be divided into unconstrained problems and constrained problems. Unconstrained problems are typically formulated as

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \quad f(\boldsymbol{x}),$$

where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a continuous function (usually we also assume f is smooth, and please refer to the discussion later).

Constrained optimization problems are formulated as

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \quad & f(\boldsymbol{x}) \\ \text{subject to} \quad & \boldsymbol{x} \in \mathcal{D} \end{aligned}$$

$\mathcal{D}$, which is called the feasible region in most cases, are specified by various constraints. The types of constraints include linear/non-linear constraints as well as equality/inequality constraints.

Note that unconstrained and constrained problems are not mutually exclusive. For example, when solving a linear program

$$\begin{aligned} \underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \quad & \boldsymbol{c}^T\boldsymbol{x} \\ \text{subject to } & A\boldsymbol{x} \geq \boldsymbol{b} \end{aligned} \tag{1}$$

We can convert (1) into a unconstrained problem using the so-called log-barrier:

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \quad \boldsymbol{c}^T\boldsymbol{x} - \lambda^T \log(A\boldsymbol{x} - \boldsymbol{b})$$

As another example, in non-rigid deformation we typically solve the following optimization problem

$$\underset{\boldsymbol{p}_i,\boldsymbol{c}_i,1\leq i\leq n}{\text{minimize}} \quad \mu\sum_{i\in\mathcal{H}}\|\boldsymbol{h}_i-\boldsymbol{p}_i\|^2+\sum_{i=1}^{n}\|\exp(\boldsymbol{c}_i\times)(\boldsymbol{p}_i^{\text{rest}}-\boldsymbol{p}_j^{\text{rest}})-(\boldsymbol{p}_i-\boldsymbol{p}_j)\|^2, \tag{2}$$

where $R_i=\exp(\boldsymbol{c}_i\times)$ gives the parameterization of the rotation at each vertex using the exponential map. It is clear that

$$\begin{aligned}\underset{\boldsymbol{p}_i,R_i,1\leq i\leq n}{\text{minimize}} \quad & \mu\sum_{i\in\mathcal{H}}\|\boldsymbol{h}_i-\boldsymbol{p}_i\|^2+\sum_{i=1}^{n}\|R_i(\boldsymbol{p}_i^{\text{rest}}-\boldsymbol{p}_j^{\text{rest}})-(\boldsymbol{p}_i-\boldsymbol{p}_j)\|^2\\ \text{subject to} \quad & R_i\in SO(3),\quad 1\leq i\leq n\end{aligned} \tag{3}$$

(3) is called constrained optimization. However, (3) turns out to be easier to optimize, e.g., via alternating minimization. When R_i is fixed, $\boldsymbol{p}_i$ can be optimized by solving a linear system. When $\boldsymbol{p}_i$ are fixed, R_i can be optimized independently, by solving a registration with known correspondence problem.

2.2 Smooth versus Non-smooth

Optimization problems can be classified into smooth optimization problems (e.g., f is smooth) and non-smooth optimization problems (e.g., the second order derivatives of f do not exist at some points). Non-smooth problems are hard to optimize, since we cannot predict what is going on when crossing singular points. What people typically do is to convert non-smooth problems into smooth problems. For example,

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \quad \|A\boldsymbol{x}-\boldsymbol{b}\|_1 \tag{4}$$

can be first converted into solving a linear program

$$\begin{aligned}\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \quad & \boldsymbol{1}^T\boldsymbol{s}\\ \text{subject to} \quad & A\boldsymbol{x}-\boldsymbol{b}\leq\boldsymbol{s}\\ & A\boldsymbol{x}-\boldsymbol{b}\geq-\boldsymbol{s}\end{aligned}$$

which can then be converted into solving a smooth optimization via the log-barrier:

$$\underset{\boldsymbol{x}\in\mathbb{R}^n}{\text{minimize}} \quad \boldsymbol{1}^T\boldsymbol{s}-\lambda\boldsymbol{1}^T(\log(\boldsymbol{s}-A\boldsymbol{x}+\boldsymbol{b})-\lambda\boldsymbol{1}^T\log(A\boldsymbol{x}-\boldsymbol{b}+\boldsymbol{s}) \tag{5}$$

Note that the value of λ is gradually decreasing so that the objective functions of (4) and (5) are close enough.

2.3 Convex versus Non-convex

Convex optimization problems stand for the problems where both the objective function and the constraints are convex functions (we will talk about this later). The remaining problems are generally called non-convex problems. Convex problems are generally easier to solve than non-convex problems. On the other hand, when solving non-convex problems, we typically solve local convex proxies, e.g., trust region methods. Convex problems have been the major focus in the research community. Recent interests shift from convex optimization to non-convex optimization. In particular, training neural networks amounts to solve non-convex optimization problems.

3 What is a Solution?

Definition 3.1. *A point $\boldsymbol{x}^\star$ is a global minimizer if $f(\boldsymbol{x}^\star)\leq f(\boldsymbol{x})$ for all $\boldsymbol{x}\in\mathbb{R}^n$.*

Global minimizers are hard to find, and most algorithms can only find local minimizers. Formally speaking,

Definition 3.2. *A point $\boldsymbol{x}^\star$ is a local minimizer if there is a neighborhood $\mathcal{N}$ of $\boldsymbol{x}^\star$ such that $f(\boldsymbol{x}^\star) \leq f(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathcal{N}$.*

Recall what we have learned in calculus that the neighborhood of $\boldsymbol{x}^\star$ is simply an open set that contains $\boldsymbol{x}^\star$. A point that satisfies this definition is sometimes called a weak local minimizer. This terminology distinguishes it from a strict local minimizer, which is the outright winner in its neighborhood. Formally,

Definition 3.3. *A point $\boldsymbol{x}^\star$ is a strict local minimizer (also called a strong local minimizer) if there is a neighborhood of $\mathcal{N}$ of $\boldsymbol{x}^\star$ such that $f(\boldsymbol{x}^\star) < f(\boldsymbol{x}$ for all $\boldsymbol{x} \in \mathcal{N}$ with $\boldsymbol{x} \neq \boldsymbol{x}^\star)$.*

A different and sometimes more useful definition is as follows:

Definition 3.4. *A point $\boldsymbol{x}^\star$ is an isolated local minimizer if there is a neighborhood $\mathcal{N}$ of $\boldsymbol{x}^\star$ such that $\boldsymbol{x}^\star$ is the only local minimizer in $\mathcal{N}$.*

The book gives one example where strict local minimizers are not isolated:

$$f(x) = x^4 \cos(1/x) + 2x^4, \quad f(0) = 0.$$

f is twice continuously differentiable and has a strict local minimizer at $x^\star = 0$.

3.1 Recognizing a Local Minimum

If f is twice continuously differentiable, we may be able to tell that $\boldsymbol{x}^\star$ is a local minimizer (and possibly a strict local minimizer) by examining just the gradient $\nabla f(\boldsymbol{x}^\star)$ and the Hessian $\nabla^2 f(\boldsymbol{x}^\star)$. The mathematical tool used to study minimizers of smooth functions is Taylor's thereom. The proof can be found in any Calculus textbook.

Theorem 3.1. *(Taylor's Theorem) Suppose that $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and that $\boldsymbol{p} \in \mathbb{R}^n$. Then we have that*

$$f(\boldsymbol{x} + \boldsymbol{p}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x} + t\boldsymbol{p})^T \boldsymbol{p}, \tag{6}$$

for some $t \in (0, 1)$. Moreover, if f is twice continuously differentiable, we have that

$$\nabla f(\boldsymbol{x} + \boldsymbol{p}) = \nabla f(\boldsymbol{x}) + \int_0^1 \nabla^2 f(\boldsymbol{x} + t\boldsymbol{p})\boldsymbol{p} dt, \tag{7}$$

and that

$$f(\boldsymbol{x} + \boldsymbol{p}) = f(\boldsymbol{x}) + \nabla f(\boldsymbol{x})^T \boldsymbol{p} + \frac{1}{2}\boldsymbol{p}^T \nabla^2 f(\boldsymbol{x} + t\boldsymbol{p})\boldsymbol{p}, \tag{8}$$

for some $t \in (0, 1)$.

Theorem 3.2. *(First-Order Necessary Conditions) If $\boldsymbol{x}^\star$ is a local minimizer and f is continuously differentiable in an open neighborhood of $\boldsymbol{x}^\star$, then $\nabla f(\boldsymbol{x}^\star) = 0$.*

Suppose $\nabla f(\boldsymbol{x}^\star) \neq 0$. The proof looks at the value of f in the neighborhood of $\boldsymbol{x}^\star$ in the direction defined by $-\frac{\nabla f(\boldsymbol{x}^\star)}{\|\nabla f(\boldsymbol{x}^\star)\|}$.

Theorem 3.3. *(Second-Order Necessary Conditions). If $\boldsymbol{x}^\star$ is a local minimizer of f and $\nabla^2 f$ is continuous in an open neighborhood of $\boldsymbol{x}^\star$, then $\nabla f(\boldsymbol{x}^\star) = 0$ and $\nabla^2 f(\boldsymbol{x}^\star)$ is positive semidefinite.*

The proof applies the continuity of $\nabla^2 f(\boldsymbol{x})$ in the neighborhood of $\boldsymbol{x}^\star$ as well the second-order Taylor expansion,

$$f(\boldsymbol{x}^{star} + \bar{t}\boldsymbol{p}) = f(\boldsymbol{x}^\star) + \bar{t}\boldsymbol{p}^T \nabla f(\boldsymbol{x}^\star) + \frac{1}{2}\bar{t}^2 \boldsymbol{p}^T \nabla^2 f(\boldsymbol{x}^\star + t\boldsymbol{p})\boldsymbol{p}.$$

Theorem 3.4. *(Second-Order Sufficient Conditions). Suppose that $\nabla^2 f$ is continuous in an open neighborhood of $\boldsymbol{x}^\star$ and that $\nabla f(\boldsymbol{x}^\star) = 0$ and $\nabla^2 f(\boldsymbol{x}^\star)$ is positive definite. Then $\boldsymbol{x}^\star$ is a strict local minimizer of f.*

The following theorem is related to convex functions.

Theorem 3.5. *When f is convex, any local minimizer $\boldsymbol{x}^\star$ is a global minimizier of f. If in addition f is differentiable, then any stationary point $\boldsymbol{x}^\star$ is a global minimizer of f.*

The proof uses the convexity property, i.e.,

$$\boldsymbol{x} = \lambda \boldsymbol{z} + (1-\lambda)\boldsymbol{x}^\star, \qquad \text{for some} \lambda \in (0,1],$$

then we have

$$f(\boldsymbol{x}) \le \lambda f(\boldsymbol{z}) + (1-\lambda) f(\boldsymbol{x}^\star).$$

4 Overview of Algorithms

There are generally two strategies: Line search and Trust region. In this class, we will also talk about alternating minimization and stochastic methods, which are variants of this basic method.

4.1 Line Search

In the line search strategy, the algorithm chooses a direction $\boldsymbol{p}_k$ and searches along this direction from the current iterate $\boldsymbol{x}_k$ for a new iterate with a lower function value. The distance to move along $\boldsymbol{p}_k$ can be found by approximately solving the following one-dimensional minimization problem to find a step length α:

$$\underset{\alpha}{\text{minimize}}\ f(\boldsymbol{x}_k + \alpha \boldsymbol{p}_k) \tag{9}$$

By solving (9) exactly, we would derive the maximum benefit from the direction $\boldsymbol{p}_k$, but an exact minimization is expensive and unnecessary. Instead, the line search algorithm generates a limited number of trial step lengths until it finds one that loosely approximates the minimum of (9). At the new point a new search direction and step length are computed, and the process is repeated.

The search direction is crucial in line search algorithms.

- Steepest descent: $\boldsymbol{p} = -\nabla f_k / \|\nabla f_k\|$.
- Newton method: $\boldsymbol{p}_k^N = -(\nabla^2 f_k)^{-1} \nabla f_k$.

4.2 Trust Region

In the second algorithmic strategy, known as trust region, the information gathered about f is used to construct a model function m_k whose behavior near the current point $\boldsymbol{x}_k$ is similar to that of the actual objective function f. Because the model m_k may not be a good approximation of f when $\boldsymbol{x}$ is far from $\boldsymbol{x}_k$, we restrict the search for a minimizer of m_k to some region around $\boldsymbol{x}_k$. In other words, we find the candidate step $\boldsymbol{p}$ by approximately solving the following sub-problem:

$$\min\ m_k(\boldsymbol{x}_k + \boldsymbol{p}), \qquad \text{where } \boldsymbol{x}_k + \boldsymbol{p} \text{ lies inside the trust region.} \tag{10}$$

If the candidate solution does not produce a sufficient decrease in f, we conclude that the trust region is too large, and we shrink it and re-solve (10). Usually, the trust region is a ball defined by $\|\boldsymbol{p}\| \le \delta$, where the scalar $\delta > 0$ is called the trust-region radius. For some problems, box-shaped trust regions may also be used. The model m_k in (10) is usually defined to be a quadratic function of the form

$$m_k(\boldsymbol{x}_k + \boldsymbol{p}) = f_k + \boldsymbol{p}^T \nabla f_k + \frac{1}{2}\boldsymbol{p}^T B_k \boldsymbol{p}, \tag{11}$$

where f_k , ∇f_k , and B_k are a scalar, vector, and matrix, respectively. As the notation indicates, f_k and ∇f_k are chosen to be the function and gradient values at the point $\boldsymbol{x}_k$, so that m_k and f are in agreement to first order at the current iterate $\boldsymbol{x}_k$. The matrix B_k is either the Hessian $\nabla^2 f_k$ or some approximation to it.