Optimization for Data Science

F. Rinaldi[1]

1 Università degli Studi di Padova | Dipartimento Matematica

Padova
2020

Outline

Optimization for Data Science

Unconstrained Problems and Descent Directions

Unconstrained Problems

We consider problems of the following form:

$$\begin{aligned} &\min\ f(x) \\ &x \in \mathbb{R}^n, \end{aligned} \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a continuous function in $\mathbb{R}^n$ (that is $f \in C(\mathbb{R}^n)$).

When trying to minimize a given objective function, an important role, both from a theoretical and a computational point of view, is played by the set of so-called *descent directions*:

Definition [Descent Directions]

We define *the set of descent directions* for f in $\bar{x}$ as follows:

$$D(\bar{x}) = \{d \in \mathbb{R}^n :\ \exists\, \delta > 0 \text{ such that } f(\bar{x} + \alpha d) < f(\bar{x}),\ \forall \alpha \in (0, \delta)\ \}.$$

Characterization of Minima

By means of descent directions we can give a first characterization of minima for an unconstrained programming problem.

Characterization of Minima

Let $f \in C(\mathbb{R}^n)$. If $x^* \in \mathbb{R}^n$ is a local (global) minimum for our Problem then

$$D(x^*) = \emptyset. \tag{2}$$

Proof

Let us assume, by contradiction, that condition (2) is not satisfied.

Then a vector $\bar{d} \in D(x^*)$ would exist and, keeping in mind definition of descent directions, we would have:

$$f(x^* + \alpha\bar{d}) < f(x^*)$$

for any $\alpha \in (0, \delta)$, with $\delta > 0$. Hence, x^* would not be minimum of f on $\mathbb{R}^n$.

First Order Descent Directions

We can now describe first order conditions that help to identify a descent direction.

Proposition [First Order Descent Directions]

Assume that $f : \mathbb{R}^n \to \mathbb{R}$ is continuously differentiable and let $d \in \mathbb{R}^n$ be a nonzero vector. If we have:

$$\nabla f(x)^\top d < 0, \tag{3}$$

then d is a descent direction for f in x. If we further have that f is a convex function and d is a descent direction for f in x, then condition (3) is satisfied.

Characterization of Minima (First-order Conditions)

First characterization of minima helps us to describe useful optimality conditions:

Theorem (Necessary Optimality Conditions)

Let $f \in C^1(\mathbb{R}^n)$. If $x^* \in \mathbb{R}^n$ is a local (global) minimum of problem (1) then

$$\nabla f(x^*) = 0. \tag{4}$$

Stationary Points

Algorithms can usually find points that satisfy the necessary condition described in the previous theorem.

This is the reason why we now characterize those points using the following definition:

Definition [Stationary Point]

A point $x^* \in \mathbb{R}^n$ is *stationary* for (1) if and only if

$$\nabla f(x^*) = 0. \tag{5}$$

Stationary Points II

Remark

- Previous theorem describes necessary optimality conditions.
- In general, x^* might be stationary, without being a minimum for f. We give an example below.

Example

Let us consider the following function:

$$f(x) = x^3.$$

This function admits derivatives of any order and $x^* = 0$ is such that

$$\frac{\partial f(x^*)}{\partial x} = 0,$$

hence it satisfies first order optimality conditions.

Sufficient Optimality Conditions

We now show that first order optimality conditions are also sufficient when we deal with a convex function.

Theorem [Conditions to identify global minima: convex case]

Let f be a convex continuously differentiable function on $\mathbb{R}^n$.

- Then a point $x^* \in \mathbb{R}^n$ is a global minimum if and only if $\nabla f(x^*) = 0$.
- Furthermore, if f is strictly convex over $\mathbb{R}^n$ and $\nabla f(x^*) = 0$, then x^* is the only global minimum for f.

Proof of Sufficient Optimality Conditions

Proof

Let x^* be a stationary point for f over $\mathbb{R}^n$, that is:

$$\nabla f(x^*) = 0. \tag{6}$$

For any point $x \in \mathbb{R}^n$, if function f is convex, from point (i) of Proposition related to first order convexity conditions, we can write:

$$f(x) - f(x^*) \overset{(i)}{\geq} \nabla f(x^*)^\top (x - x^*) \overset{(6)}{=} 0.$$

In case f is strictly convex from (ii) of the same proposition, we have:

$$f(x) - f(x^*) \overset{(ii)}{>} \nabla f(x^*)^\top (x - x^*) \overset{(6)}{=} 0.$$

Thus our result is proved.

Methods for Unconstrained Optimization

- We now describe methods for unconstrained optimization.
- We will first focus on some classic algorithms (e.g., gradient type methods).
- GOAL: Explaining the main features of every algorithm we present.
- Then we explain the main issues that huge scale data science problems pose.
- We analyze some variants of the classic methods aimed at guaranteeing good performance when dealing with those hard problems.

Remark

We mainly focus on machine learning problems related to huge dimensional datasets, and hence put more emphasis on practical algorithms with a machine learning flavor.

Basic Ideas

As we said before, we focus on problems of the form

$$\begin{array}{l} \min\ f(x) \\ x \in \mathbb{R}^n \end{array} \tag{7}$$

with $f : \mathbb{R}^n \to \mathbb{R}$.

- From now on we assume that f is continuously differentiable and admits minima.
- The methods we analyze here can find stationary points for a given objective function f, that is they find points belonging to the following set
$$\Omega = \{x \in \mathbb{R}^n : \nabla f(x) = 0\}.$$

A General Scheme

Algorithm 1 Optimization algorithm for unconstrained problems

```
Choose a point x_1 ∈ ℝ^n
For k = 1, ...
        If x_k ∈ Ω, then STOP
        Compute a descent direction d_k in x_k
        Compute a stepsize α_k > 0 by means of a line search
        Set x_{k+1} = x_k + α_k d_k
End for
```

Details

- We get an iterative scheme.
- Main idea: approximate, at each iteration, the original function with a nicer function (i.e., a function that is easier to handle).
- At Step 4, d_k obtained by minimizing a function $\eta(d)$ that approximates function $f(x_k + d)$.
- At Step 5, stepsize $\alpha_k > 0$ calculated by (approximately) minimizing the univariate function

$$\phi(\alpha) := f(x_k + \alpha d_k).$$

Questions

- How to choose the starting point?
- How to stop the algorithm?
- How to choose the search direction?
- How to calculate the stepsize?

How to choose the starting point?

- At Step 1, we need to choose a point $x_1 \in \mathbb{R}^n$.
- The starting point is usually given and depends on the specific function we want to minimize.
- Ideally, it should be the best (in terms of objective function value) available point.

How to stop the algorithm?

- At Step 3, if $x_k \in \Omega$, then STOP
- It is equivalent to check if
$$\nabla f(x_k) = 0.$$
- In practice, since we have a finite precision machine, we need a better criterion.
- A first option might be stopping the algorithm when
$$\|\nabla f(x_k)\| \le \epsilon,$$
with $\epsilon > 0$ sufficiently small.
- As we will see, this is not the best option in a huge scale framework.
- We need to find some other way to guarantee the current iterate is good enough.

How to choose the search direction?

- There exist different options for the choice of the search direction.
- Alternatives depend on the information available (e.g., first order second order derivatives of f).
- More specifically, we can have
  - **Zero-th order methods** (aka derivative free methods): evaluate the objective function along a suitable set of search directions (e.g., direct search);
  - **First order methods**: use the gradient when calculating the search directions (e.g., gradient methods, conjugate directions methods, Quasi-Newton methods);
  - **Second order methods**: further use Hessian of the objective function when calculating the search direction (e.g., Newton-like methods);

How to calculate the stepsize?

Calculation of α_k represents the so-called line search.

Here we report three different options

- Exact line search
- Armijo Rule
- Fixed Stepsize

Exact Line Search

Exact line search

α_k is the value obtained by exactly minimizing f along d_k, that is

$$\alpha_k = \arg\min_{\alpha} \phi(\alpha) = f(x_k + \alpha d_k)\ .$$

This is a viable option only when the function has some structure.

Armijo Rule

In this case, we only try to guarantee a decrease of the objective function with respect to some model.

Armijo Rule

We fix parameters δ and γ, with $\delta \in (0, 1)$ and $\gamma \in (0, 1/2)$, and we give a starting stepsize $\triangle_k$. We then try steps

$$\alpha = \delta^m \triangle_k,$$

with $m = 0, 1, 2, \ldots,$ until inequality

$$f(x_k + \alpha d_k) \leq f(x_k) + \gamma\alpha\nabla f(x_k)^\top d_k \tag{8}$$

is satisfied and set $\alpha_k = \alpha$.

Armijo Rule II

- Condition (8) is equivalent to say that the value

$$\phi(\alpha_k) = f(x_k + \alpha_k d_k)$$

is lower or equal than the line passing by the point $(0, \phi(0))$ and having a slope equal to $\gamma\dot{\phi}(0)$, that is:

$$\phi(\alpha_k) \leq y(\alpha_k) = \phi(0) + \gamma\dot{\phi}(0)\alpha_k.$$

- It is easy to see (using composition rule) that

$$\dot{\phi}(0) = \nabla f(x_k)^\top d_k.$$

- Inequality (8) ensures a *sufficient decrease* of f.

Remark

Might be expensive when dealing with data science problems (function evaluations might have a significant cost).

Armijo Line Search Example

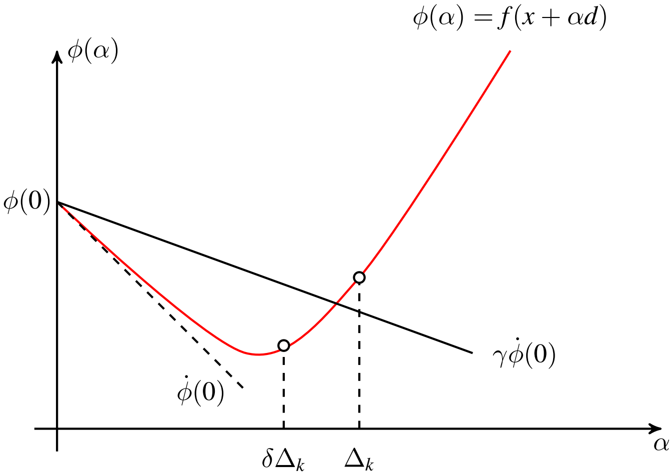

Figure: Example of Armijo line search (step $\alpha_k = \delta\Delta_k$ is chosen)

Fixed Stepsize

Fixed Stepsize

A fixed stepsize $s > 0$ is given at each iteration, that is

$$\alpha_k = s, \quad k = 0, 1, \ldots$$

Remark

- Considered only when the function satisfies some specific property.
- Choosing an appropriate stepsize might require some preliminary experimentation.