10 Continuous optimization algorithms

In this session, we will take brief look at the continuous optimization algorithms. I will mainly focus on the unconstrained optimizations, but also point out to some aspects of the constrained ones. In the context of continuous optimization, the term algorithm often refers to a specific type of algorithms. In this type of algorithms, there exists a state $\mathbf{x}_n$, which is often the best solution so far. The algorithm is in fact a map $\mathbf{x}_{n+1} = \Phi(\mathbf{x}_n)$ which generates the next best solution from the current solution. You may have noticed that n shows the number iterations that we have applied the map. The algorithm starts from an **initial point** $\mathbf{x}_0$. The initial point can be selected in different ways. It can be purely random or it can be rule-based. Sometimes the output of another approximate optimization algorithm can be used as an initial point. This is called a **warm start**.

The most important issue in convex optimization is **convergence**. In practice, the algorithm stops after a finite (and often bounded) number of iterations, but in theory, the algorithm can iterate forever to generate an infinite sequence of solutions $\{\mathbf{x}_n\}_{n=0}^{\infty}$. The aim of the optimization algorithm is to insure that the limit

$$\bar{\mathbf{x}} = \lim_{n\to\infty} \mathbf{x}_n \tag{92}$$

is an optimal solution. Even this aim is not always possible to achieve. As seen, the sequence and consequently the limit depend on the initial point. However, if an algorithm is exact (converges to an optimal solution) and the optimal solution is unique, then $\bar{\mathbf{x}} = \mathbf{x}^*$ is the same for any choice of the initial point. An important concept for the algorithms is the **convergence speed**. For any $\epsilon > 0$, the convergence seed is defined by the smallest number $N(\epsilon)$, where for all $n > N(\epsilon)$ the relation $\|\mathbf{x}_n - \bar{\mathbf{x}}\| < \epsilon$ holds. The smaller the function $N(\epsilon)$ is, the faster the algorithm converges. If there exists a constant $\mu > 0$ such that

$$\lim_{n\to\infty} \frac{\|\mathbf{x}_n - \bar{\mathbf{x}}\|_2}{\|\mathbf{x}_{n+1} - \bar{\mathbf{x}}\|_2} = \mu \tag{93}$$

then it is said that the algorithm converges linearly and μ is called the **convergence rate**. In this case $\|\mathbf{x}_n - \bar{\mathbf{x}}\|_2$ decreases exponentially and the function $N(\epsilon)$ is logarithmic.

Consider the optimization

$$\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \tag{94}$$

Many algorithms provide convergence by ensuring that at each iteration $f(\mathbf{x}_n) \leq f(\mathbf{x}_{n+1})$. However, some algorithms may not.

Example 25. Line search with iterative grid refinement: If $\mathbf{x} = x \in [a\ b] \subset \mathbb{R}$ then the problem is called line search. The interval $[a\ b]$ is called the uncertainty interval. The line search can be written as

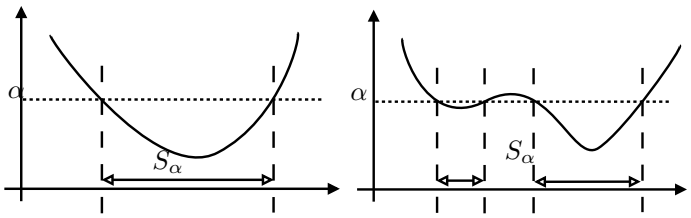
$$\min_{x\in[a\ b]} f(x) \tag{95}$$

The line search can be exactly solved for a group of functions, called **quasiconvex or unimodal**[3]: Define the **level set** S_α for a function $f(x)$ and a constant α as

$$S_\alpha = \{x \mid f(x) \leq \alpha\} \tag{96}$$

A function f is quasiconvex if for any α the level set S_α is an interval. Figure 9 shows an example of a unimodal (quasiconvex) function and an example of a function which is not unimodal (multimodal). A quasiconvex (unimodal) function, which is not constant in any interval is called strictly quasiconvex. Let us now give an algorithm to perform line search for unimodal functions. First, we define **uniform** $k-$**grid** on an interval $[a\ b]$ as the finite set $G_k([a\ b]) = \{a, a+\delta, a+2\delta, \ldots, a+(k-1)\delta\}$, where $\delta = \frac{b-a}{k}$. Then, we can perform the following steps:

[3]The term quasiconvex is also used for multi-variable functions, but unimodality is only defined for a real function.



(a) Unimodal (b) Multimodal.

Figure 9: The concept of unimodality.

1. Start from the uncertainty interval $I_0 = [a\ b]$. Take the grid $G^0 = G_k([a\ b])$ and define $\delta_0 = \frac{b-a}{k}$ take x_0 as the point in G^0 with the minimum objective value f. Set n=1.
2. Take the refined uncertainty interval $I_n = [x_{n-1} - \delta\ x_{n-1} + \delta]$, the refined grid $G^n = G_k([x_{n-1} - \delta\ x_{n-1} + \delta])$ and $\delta_n = 2\delta_{n-1}/k$. Take the minimum x_n over the grid.
3. Update n to $n + 1$ and goto step 2.

Notice that $x_n \in I_n$. Hence $|x_n - x_{n-1}| \leq \delta_n$. On the other hand, $\delta_n = \delta_0(2/k)^n$. This shows that the sequence x_n is absolutely convergent for $k > 2$. Denote the limit by $\bar{\mathbf{x}}$. Notice that x_n is the minimum over G^n. If k is odd then $x_{n-1} \in G_n$, which shows that $f(x_n) \leq f(x_{n-1})$. However, if k is even this might not be the case.

Now, suppose that f is strictly quasiconvex, we show that the optimal point $x^* \in I_n$ for all n. So, $x^* \in I_0 \cap I_1 \cap \ldots \cap I_n \cap \ldots = \{\bar{\mathbf{x}}\}$, and hence $\bar{\mathbf{x}} = \mathbf{x}^*$. The proof is by induction: $x^* \in I_0$ is trivial. Now, suppose that $x^* \in I_n$ and x_n is the minimum point over G^n. Without loss of generality, I assume that the x_n is a middle point in G^n (the other cases are similarly proved). Then, $f(x_n) \leq f(x_n + \delta)$ and $f(x_n) \leq f(x_n - \delta)$. It is simple to see that if the minimum point x^* is not in the interval $I_{n+1} = [x_n - \delta\ x_n + \delta]$, the function cannot be strictly quasiconvex as the level set $S_{\alpha=f(x_n)}$ will have two disjoint parts. Hence, $x^* \in I_{n+1}$, which proves the result.

Many continuous algorithms are based on **local search**. This means that the map $\mathbf{x}_{n+1} = \Phi(\mathbf{x}_n)$ is such that $\|\mathbf{x}_{n+1} - \mathbf{x}_n\|_2$ is small. One way to provide such a map is to solve the following restricted optimization

$$\begin{aligned} \mathbf{x}_{n+1} &= \arg\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \\ &\text{subject to} \\ \|\mathbf{x} - \mathbf{x}_n\|_2 &\leq \mu_n \end{aligned} \tag{97}$$

The parameter μ_n is called the step size. Clearly, in this case $f(\mathbf{x}_{n+1}) \leq f(\mathbf{x}_n)$ and the optimization often converges a point $\bar{\mathbf{x}}$. If we select μ_n such that it does not converge to 0, then the point $\bar{\mathbf{x}}$ is the minimum point in its small neighborhood i.e., there is an $\epsilon > 0$, such that

$$\begin{aligned} \bar{\mathbf{x}} &= \arg\min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \\ &\text{subject to} \\ \|\mathbf{x} - \bar{\mathbf{x}}\|_2 &\leq \epsilon \end{aligned} \tag{98}$$

Such a point is called a **local optimal solution**. In this context the optimal solution is referred to as a **global optimal solution**. A global optimal solution is also a local optimal solution, but there might be many local optimal solutions that are not globally optimal. There is an exception to this rule: For convex functions, any local optimal solution is a global optimal solution.

The restricted optimization in (97) might still not be easy to solve. However, since it is local we might be able to approximate it. The next example shows a very important method.

Example 26. The gradient descent method: If the step size is small and the function f is differentiable, we can use Taylor expansion to approximate the cost function in (97):

$$f(\mathbf{x}) \approx f(\mathbf{x}_n) + \nabla^T f(\mathbf{x}_n)(\mathbf{x} - \mathbf{x}_n) \tag{99}$$

Replacing this result in (97) leads to

$$\begin{aligned} \mathbf{x}_{n+1} = \arg\min_{\mathbf{x}\in\mathbb{R}^n} & f(\mathbf{x}_n) + \nabla^T f(\mathbf{x}_n)(\mathbf{x} - \mathbf{x}_n) \\ & \text{subject to} \\ & \|\mathbf{x} - \mathbf{x}_n\|_2 \leq \mu_n \end{aligned} \tag{100}$$

This optimization can be solved to obtain

$$\mathbf{x}_{n+1} = \mathbf{x}_n - \mu_n \nabla f(\mathbf{x}_n) \tag{101}$$

This is called gradient descent algorithm. Unlike the exact restricted optimization, here the step size is important. There are two methods to find the step size. first, to predefine a sequence $\{\mu_n\}_{n=0}^{\infty}$ of step sizes. Second, to choose it at each iteration. The following is a simple way to select the step size

$$\mu_n = \arg\min_{\mu>0} f(\mathbf{x}_n - \mu \nabla f(\mathbf{x}_n)) \tag{102}$$

In some cases the above rule can reduce the speed of convergence, but in many cases it is successful. The important observation is that the above optimization is a line search and can be exactly solved in some cases.