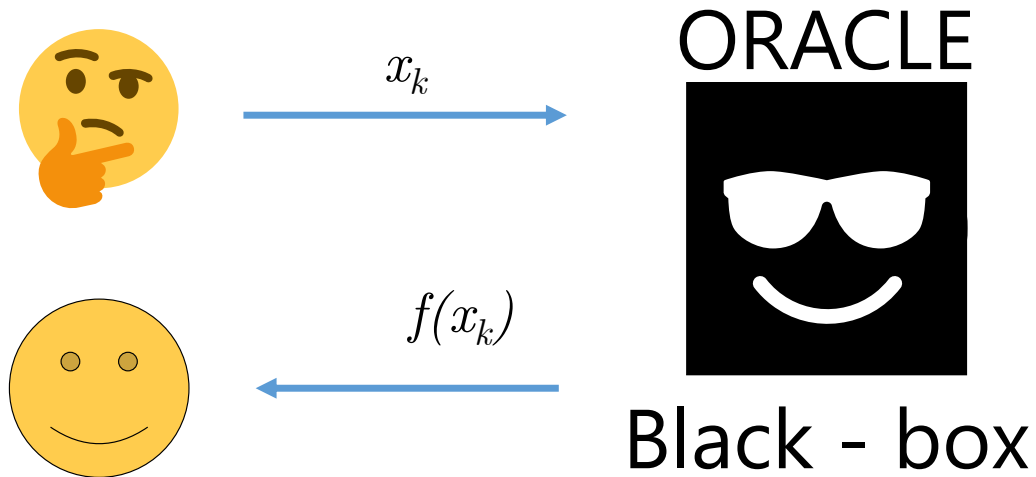


Zero order methods



Now we have only zero order information from the oracle. Typical speed of convergence of these methods is sublinear. A lot of methods are referred both to zero order methods and global optimization.

Code

- Global optimization illustration - [Open in Colab](#)
- Nevergrad library - [Open in Colab](#)

Simulated annealing

Problem

We need to optimize the global optimum of a given function on some space using only the values of the function in some points on the space.

$$\min_{x \in X} F(x) = F(x^*)$$

Simulated Annealing is a probabilistic technique for approximating the global optimum of a given function.

Algorithm

The name and inspiration come from annealing in metallurgy, a technique involving heating and controlled cooling of a material to increase the size of its crystals and reduce their defects. Both are attributes of the material that depend on its thermodynamic free energy. Heating and cooling the material affects both the temperature and the thermodynamic free energy. The simulation of annealing can be used to find an approximation of a global minimum for a function with many variables.

Steps of the Algorithm

Step 1 Let $k = 0$ - current iteration, $T = T_k$ - initial temperature.

Step 2 Let $x_k \in X$ - some random point from our space

Step 3 Let decrease the temperature by following rule $T_{k+1} = \alpha T_k$ where $0 < \alpha < 1$ - some constant that often is closer to 1

Step 4 Let $x_{k+1} = g(x_k)$ - the next point which was obtained from previous one by some random rule. It is usually assumed that this rule works so that each subsequent approximation should not differ very much.

Step 5 Calculate $\Delta E = E(x_{k+1}) - E(x_k)$, where $E(x)$ - the function that determines the energy of the system at this point. It is supposed that energy has the minimum in desired value x^* .

Step 6 If $\Delta E < 0$ then the approximation found is better than it was. So accept x_{k+1} as new started point at the next step and go to the step **Step 3**

Step 7 If $\Delta E \geq 0$, then we accept x_{k+1} with the probability of $P(\Delta E) = \exp^{-\Delta E/T_k}$. If we don't accept x_{k+1} , then we let $k = k + 1$. Go to the step **Step 3**

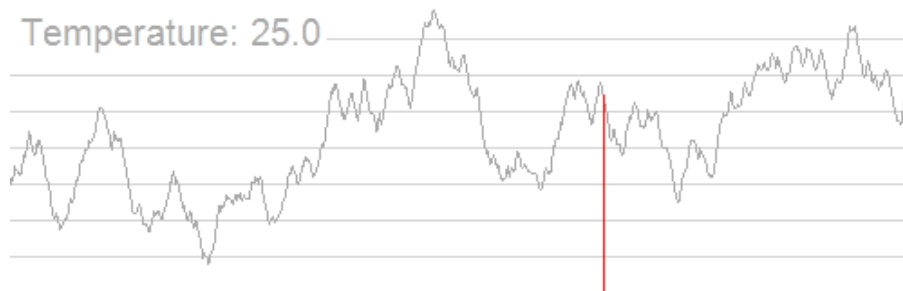
The algorithm can stop working according to various criteria, for example, achieving an optimal state or lowering the temperature below a predetermined level T_{min} .

Convergence

As it mentioned in [Simulated annealing: a proof of convergence](#) the algorithm converges almost surely to a global maximum.









Illustration

A gif from [Wikipedia](#):



Example

In our example we solve the N queens puzzle - the problem of placing N chess queens on an N×N chessboard so that no two queens threaten each other.

The Problem

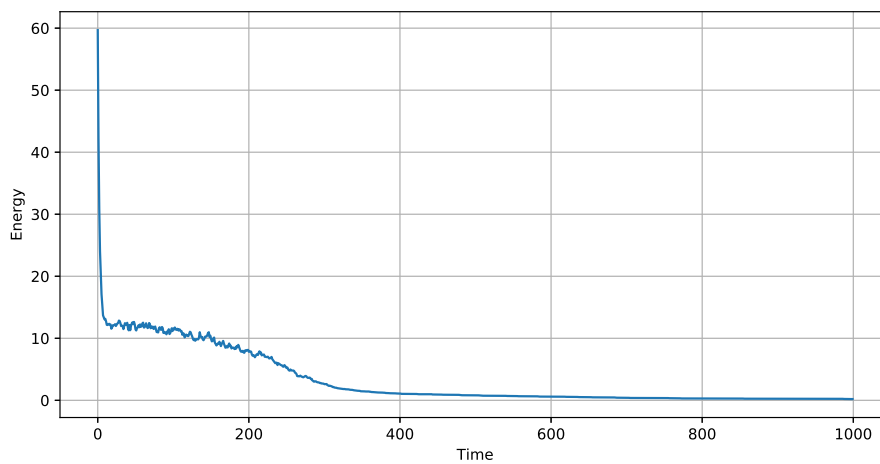
Let $E(x)$ - the number of intersections, where x - the array of placement queens at the field (the number in array means the column, the index of the number means the row).

The problem is to find x^* where $E(x^*) = \min_{x \in X} E(x)$ - the global minimum, that is predefined and equals to 0 (no two queens threaten each other).

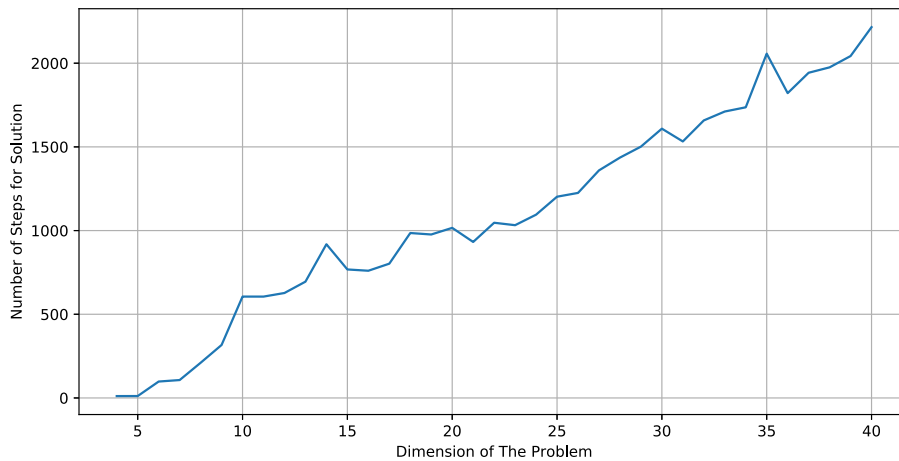
In this code $x_0 = [0, 1, 2, \dots, N]$ that means all queens are placed at the board's diagonal. So at the beginning $E = N(N - 1)$, because every queen intersects others.

Results

Results of applying this algorithm with $\alpha = 0.95$ to the N queens puzzle for $N = 10$ averaged by 100 runs are below:



Results of running the code for N from 4 to 40 and measuring the time it takes to find the solution averaged by 100 runs are below:

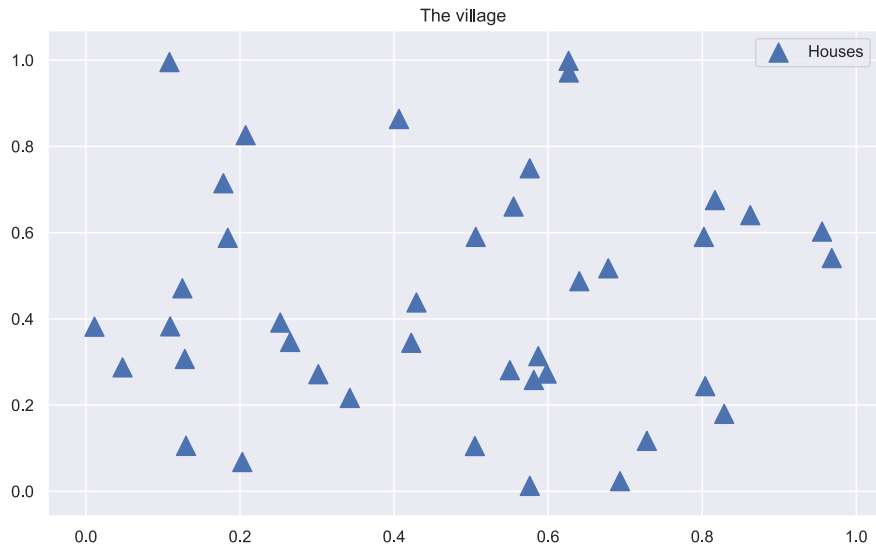


[Open in Colab](#)

Genetic algorithm

Problem

Suppose, we have N points in \mathbb{R}^d Euclidian space (for simplicity we'll consider and plot case with $d = 2$). Let's imagine, that these points are nothing else but houses in some 2d village. Salesman should find the shortest way to go through the all houses only once.



That is, very simple formulation, however, implies NP - hard problem with the factorial growth of possible combinations. The goal is to minimize the following cumulative distance:

$$d = \sum_{i=1}^{N-1} \|x_{y(i+1)} - x_{y(i)}\|_2 \rightarrow \min_y,$$

where x_k is the k -th point from N and y stands for the N - dimensional vector of indices, which describes the order of path. Actually, the problem could be [formulated](#) as an LP problem, which is easier to solve.

Genetic (evolution) algorithm

Our approach is based on the famous global optimization algorithm, known as evolution algorithm.

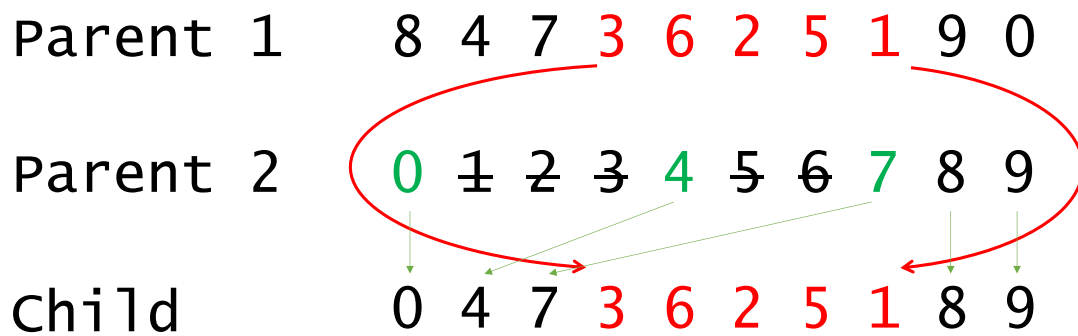
Population and individuals

Firstly we need to generate the set of random solutions as an initialization. We will call a set of solutions $\{y_k\}_{k=1}^n$ as *population*, while each solution is called *individual* (or creature).

Each creature contains integer numbers $1, \dots, N$, which indicates the order of bypassing all the houses. The creature, that reflects the shortest path length among the others will be used as an output of an algorithm at the current iteration (generation).

Crossing procedure

Each iteration of the algorithm starts with the crossing (breed) procedure. Formally speaking, we should formulate the mapping, that takes two creature vectors as an input and returns its offspring, which inherits parents properties, while remaining consistent. We will use [ordered crossover](#) as such procedure.

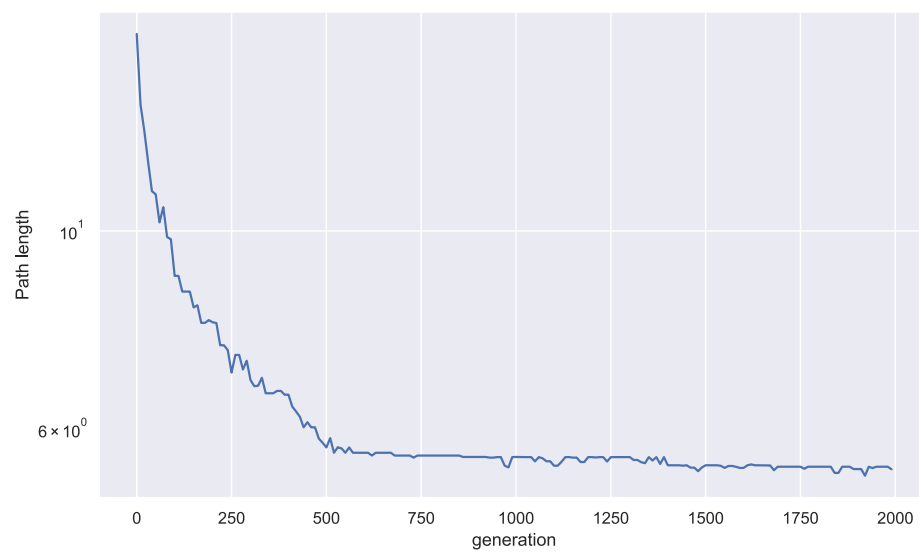
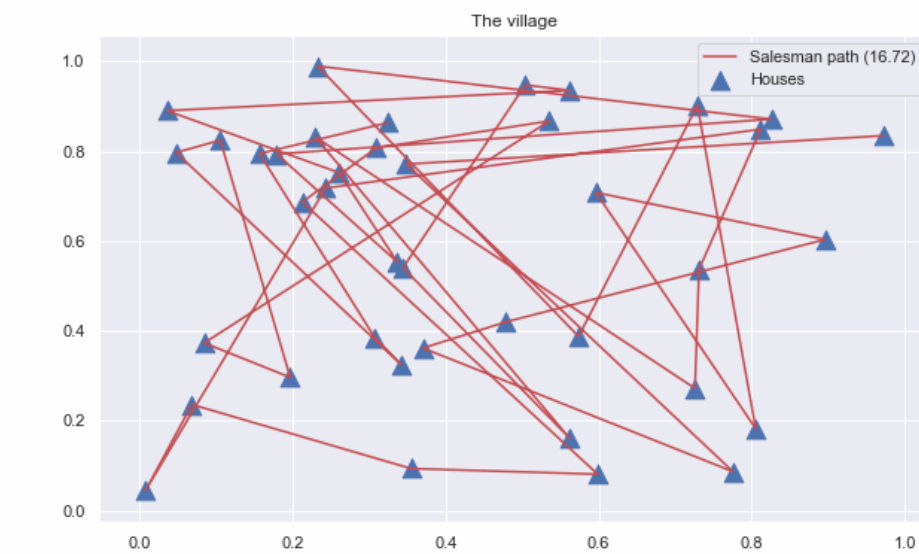


Mutation

In order to give our algorithm some ability to escape local minima we provide it with mutation procedure. We simply swap some houses in an individual vector. To be more accurate, we define mutation rate (say, 0.05). On the one hand, the higher the rate, the less stable the population is, on the other, the smaller the rate, the more often algorithm gets stuck in the local minima. We choose $\text{mutation rate} \cdot n$ individuals and in each case swap random $\text{mutation rate} \cdot N$ digits.

Selection

At the end of the iteration we have increased population (due to crossing results), than we just calculate total path distance to each individual and select top n of them.



In general, for any $c > 0$, where d is the number of dimensions in the Euclidean space, there is a polynomial-time algorithm that finds a tour of length at most $(1 + \frac{1}{c})$ times the optimal for geometric instances of TSP in

$$\mathcal{O}\left(N(\log N)^{\mathcal{O}(c\sqrt{d})^{d-1}}\right)$$

Code

 [Open in Colab](#)

References

- [General information about genetic algorithms](#)
- [Wiki](#)

Gradient descent

Summary

A classical problem of function minimization is considered.

$$x_{k+1} = x_k - \eta_k \nabla f(x_k) \quad (\text{GD})$$

- The bottleneck (for almost all gradient methods) is choosing step-size, which can lead to the dramatic difference in method's behavior.
- One of the theoretical suggestions: choosing stepsize inversly proportional to the gradient Lipschitz constant $\eta_k = \frac{1}{L}$.
- In huge-scale applications the cost of iteration is usually defined by the cost of gradient calculation (at least $\mathcal{O}(p)$).
- If function has Lipschitz-continous gradient, then method could be rewritten as follows:

$$\begin{aligned} x_{k+1} &= x_k - \frac{1}{L} \nabla f(x_k) = \\ &= \arg \min_{x \in \mathbb{R}^n} \left\{ f(x_k) + \langle \nabla f(x_k), x - x_k \rangle + \frac{L}{2} \|x - x_k\|_2^2 \right\} \end{aligned}$$

Intuition

Direction of local steepest descent

Let's consider a linear approximation of the differentiable function f along some direction h , $\|h\|_2 = 1$:

$$f(x + \eta h) = f(x) + \eta \langle f'(x), h \rangle + o(\eta)$$

We want h to be a decreasing direction:

$$f(x + \eta h) < f(x)$$

$$f(x) + \eta \langle f'(x), h \rangle + o(\eta) < f(x)$$

and going to the limit at $\eta \rightarrow 0$:

$$\langle f'(x), h \rangle \leq 0$$

Also from Cauchy–Bunyakovsky–Schwarz inequality:

$$|\langle f'(x), h \rangle| \leq \|f'(x)\|_2 \|h\|_2 \quad \rightarrow \quad \langle f'(x), h \rangle \geq -\|f'(x)\|_2 \|h\|_2 = -\|f'(x)\|_2$$

Thus, the direction of the antigradient

$$h = -\frac{f'(x)}{\|f'(x)\|_2}$$

gives the direction of the **steepest local** decreasing of the function f .

The result of this method is

$$x_{k+1} = x_k - \eta f'(x_k)$$

Gradient flow ODE

Let's consider the following ODE, which is referred as Gradient Flow equation.

$$\frac{dx}{dt} = -f'(x(t))$$

and discretize it on a uniform grid with η step:

$$\frac{x_{k+1} - x_k}{\eta} = -f'(x_k),$$

where $x_k \equiv x(t_k)$ and $\eta = t_{k+1} - t_k$ - is the grid step.

From here we get the expression for x_{k+1}

$$x_{k+1} = x_k - \eta f'(x_k),$$

which is exactly gradient descent.

Necessary local minimum condition

$$\begin{aligned} f'(x) &= 0 \\ -\eta f'(x) &= 0 \\ x - \eta f'(x) &= x \\ x_k - \eta f'(x_k) &= x_{k+1} \end{aligned}$$

This is, surely, not a proof at all, but some kind of intuitive explanation.

Minimizer of Lipschitz parabola

Some general highlights about Lipschitz properties are needed for explanation. If a function $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is continuously differentiable and its gradient satisfies Lipschitz conditions with constant L , then $\forall x, y \in \mathbb{R}^n$:

$$|f(y) - f(x) - \langle \nabla f(x), y - x \rangle| \leq \frac{L}{2} \|y - x\|^2,$$

which geometrically means, that if we'll fix some point $x_0 \in \mathbb{R}^n$ and define two parabolas:

$$\phi_1(x) = f(x_0) + \langle \nabla f(x_0), x - x_0 \rangle - \frac{L}{2} \|x - x_0\|^2,$$

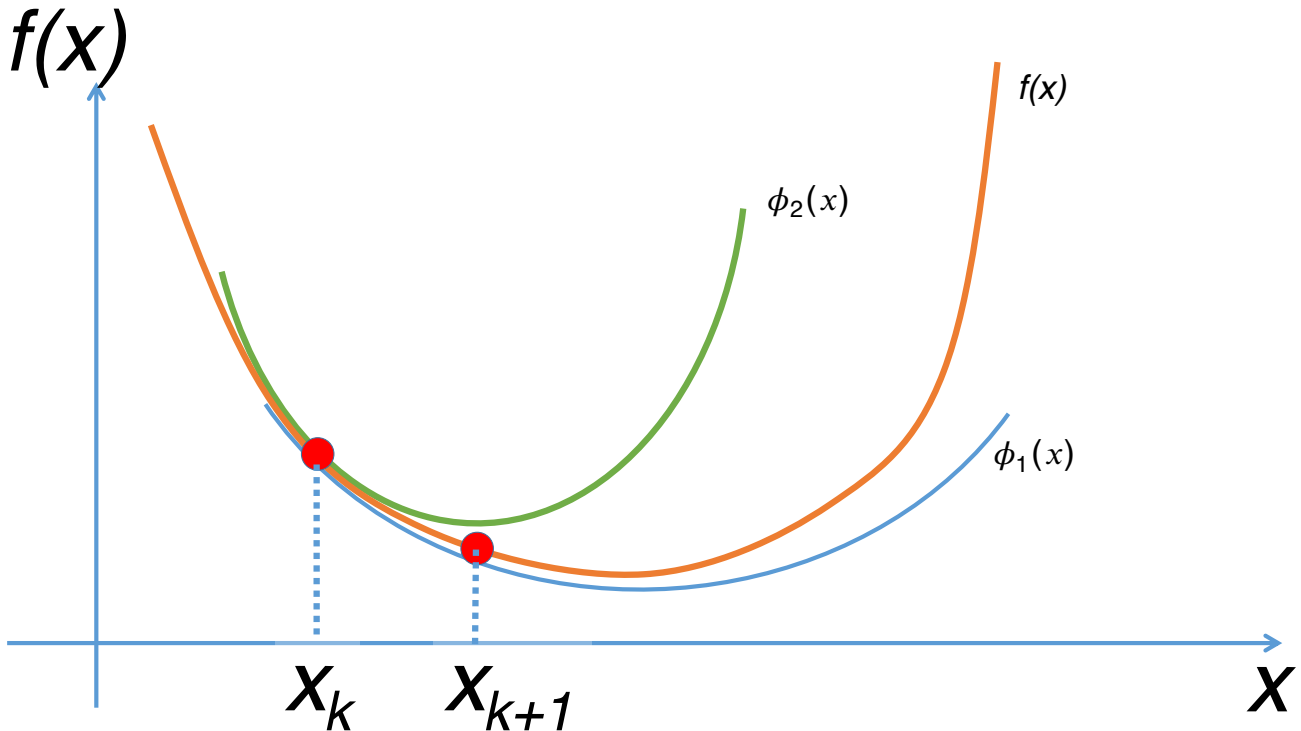
$$\phi_2(x) = f(x_0) + \langle \nabla f(x_0), x - x_0 \rangle + \frac{L}{2} \|x - x_0\|^2.$$

Then

$$\phi_1(x) \leq f(x) \leq \phi_2(x) \quad \forall x \in \mathbb{R}^n.$$

Now, if we have global upper bound on the function, in a form of parabola, we can try to go directly to its minimum.

$$\begin{aligned} \nabla \phi_2(x) &= 0 \\ \nabla f(x_0) + L(x^* - x_0) &= 0 \\ x^* &= x_0 - \frac{1}{L} \nabla f(x_0) \\ x_{k+1} &= x_k - \frac{1}{L} \nabla f(x_k) \end{aligned}$$



This way leads to the $\frac{1}{L}$ stepsize choosing. However, often the L constant is not known.

But if the function is twice continuously differentiable and its gradient has Lipschitz constant L , we can derive a way to estimate this constant $\forall x \in \mathbb{R}^n$:

$$\|\nabla^2 f(x)\| \leq L$$

or

$$-LI_n \preceq \nabla^2 f(x) \preceq LI_n$$

Stepsize choosing strategies

Stepsize choosing strategy η_k significantly affects convergence. General [line search algorithms](#) might help in choosing scalar parameter.

Constant stepsize

For $f \in C_L^{1,1}$:

$$\eta_k = \eta$$

$$f(x_k) - f(x_{k+1}) \geq \eta \left(1 - \frac{1}{2}L\eta\right) \|\nabla f(x_k)\|^2$$

With choosing $\eta = \frac{1}{L}$, we have:

$$f(x_k) - f(x_{k+1}) \geq \frac{1}{2L} \|\nabla f(x_k)\|^2$$

Fixed sequence

$$\eta_k = \frac{1}{\sqrt{k+1}}$$

The latter 2 strategies are the simplest in terms of implementation and analytical analysis. It is clear that this approach does not often work very well in practice (the function geometry is not known in advance).

Exact line search aka steepest descent

$$\eta_k = \arg \min_{\eta \in \mathbb{R}^+} f(x_{k+1}) = \arg \min_{\eta \in \mathbb{R}^+} f(x_k - \eta \nabla f(x_k))$$

More theoretical than practical approach. It also allows you to analyze the convergence, but often exact line search can be difficult if the function calculation takes too long or costs a lot.

Interesting theoretical property of this method is that each following iteration is orthogonal to the previous one:

$$\eta_k = \arg \min_{\eta \in \mathbb{R}^+} f(x_k - \eta \nabla f(x_k))$$

Optimality conditions:

$$\nabla f(x_{k+1})^\top \nabla f(x_k) = 0$$

Goldstein-Armijo

Convergence analysis

Convex case

Lipschitz continuity of the gradient

Assume that $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is convex and differentiable, and additionally

$$\|\nabla f(x) - \nabla f(y)\| \leq L\|x - y\| \quad \forall x, y \in \mathbb{R}^n$$

i.e., ∇f is Lipschitz continuous with constant $L > 0$.

Since ∇f Lipschitz with constant L , which means $\nabla^2 f \preceq LI$, we have $\forall x, y, z$:

$$(x - y)^\top (\nabla^2 f(z) - LI)(x - y) \leq 0$$

$$(x - y)^\top \nabla^2 f(z)(x - y) \leq L\|x - y\|^2$$

Now we'll consider second order Taylor approximation of $f(y)$ and Taylor's Remainder Theorem (we assume, that the function f is continuously differentiable), we have $\forall x, y, \exists z \in [x, y]$:

$$\begin{aligned}
f(y) &= f(x) + \nabla f(x)^\top (y - x) + \frac{1}{2}(y - x)^\top \nabla^2 f(x)(y - x) \\
&\leq f(x) + \nabla f(x)^\top (y - x) + \frac{L}{2}\|y - x\|^2
\end{aligned}$$

For the gradient descent we have $x = x_k, y = x_{k+1}, x_{k+1} = x_k - \eta_k \nabla f(x_k)$:

$$\begin{aligned}
f(x_{k+1}) &\leq f(x_k) + \nabla f(x_k)^\top (-\eta_k \nabla f(x_k)) + \frac{L}{2}(\eta_k \nabla f(x_k))^2 \\
&\leq f(x_k) - \left(1 - \frac{L\eta}{2}\right)\eta \|\nabla f(x_k)\|^2
\end{aligned}$$

Optimal constant stepsize

Now, if we'll consider constant stepsize strategy and will maximize $\left(1 - \frac{L\eta}{2}\right)\eta \rightarrow \max_{\eta}$, we'll get $\eta = \frac{1}{L}$.

$$f(x_{k+1}) \leq f(x_k) - \frac{1}{2L} \|\nabla f(x_k)\|^2$$

Convexity

$$f(x_k) \leq f(x^*) + \nabla f(x_k)^\top (x_k - x^*)$$

That's why we have:

$$\begin{aligned}
f(x_{k+1}) &\leq f(x^*) + \nabla f(x_k)^\top (x_k - x^*) - \frac{1}{2L} \|\nabla f(x_k)\|^2 \\
&= f(x^*) + \frac{L}{2} \left(\|x_k - x^*\|^2 - \left\|x_k - x^* - \frac{1}{L} \nabla f(x_k)\right\|^2 \right) \\
&= f(x^*) + \frac{L}{2} (\|x_k - x^*\|^2 - \|x_{k+1} - x^*\|^2)
\end{aligned}$$

Thus, summing over all iterations, we have:

$$\begin{aligned}
\sum_{i=1}^k (f(x_i) - f(x^*)) &\leq \frac{L}{2} (\|x_0 - x^*\|^2 - \|x_k - x^*\|^2) \\
&\leq \frac{L}{2} \|x_0 - x^*\|^2 = \frac{LR^2}{2},
\end{aligned}$$

where $R = \|x_0 - x^*\|$. And due to convexity:

$$f(x_k) - f(x^*) \leq \frac{1}{k} \sum_{i=1}^k (f(x_i) - f(x^*)) \leq \frac{LR^2}{2k} = \frac{R^2}{2\eta k}$$

Strongly convex case

If the function is strongly convex:

$$f(y) \geq f(x) + \nabla f(x)^\top (y - x) + \frac{\mu}{2} \|y - x\|^2 \quad \forall x, y \in \mathbb{R}^n$$

...

$$\|x_{k+1} - x^*\|^2 \leq (1 - \eta\mu)\|x_k - x^*\|^2$$

Bounds

Conditions	$\ f(x_k) - f(x^*)\ \leq$	Type of convergence	$\ x_k - x^*\ \leq$
Convex Lipschitz-continuous function(G)	$\mathcal{O}\left(\frac{1}{k}\right) \frac{GR}{k}$	Sublinear	
Convex Lipschitz-continuous gradient (L)	$\mathcal{O}\left(\frac{1}{k}\right) \frac{LR^2}{k}$	Sublinear	
μ -Strongly convex Lipschitz-continuous gradient(L)		Linear	$(1 - \eta\mu)^k R^2$
μ -Strongly convex Lipschitz-continuous hessian(M)		Locally linear $R < \bar{R}$	$\frac{\bar{R}R}{\bar{R} - R} \left(1 - \frac{2\mu}{L + 3\mu}\right)$

- $R = \|x_0 - x^*\|$ - initial distance
- $\bar{R} = \frac{2\mu}{M}$

Materials

- [The zen of gradient descent. Moritz Hardt](#)
- [Great visualization](#)
- [Cheatsheet on the different convergence theorems proofs](#)

Inexact line search

This strategy of inexact line search works well in practice, as well as it has the following geometric interpretation:

Sufficient decrease

Let's consider the following scalar function while being at a specific point of x_k :

$$\phi(\alpha) = f(x_k - \alpha \nabla f(x_k)), \alpha \geq 0$$

consider first order approximation of $\phi(\alpha)$:

$$\phi(\alpha) \approx f(x_k) - \alpha \nabla f(x_k)^\top \nabla f(x_k)$$

A popular inexact line search condition stipulates that α should first of all give sufficient decrease in the objective function f , as measured by the following inequality:

$$f(x_k - \alpha \nabla f(x_k)) \leq f(x_k) - c_1 \cdot \alpha \nabla f(x_k)^\top \nabla f(x_k)$$

for some constant $c_1 \in (0, 1)$. (Note, that $c_1 = 1$ stands for the first order Taylor approximation of $\phi(\alpha)$). This is also called Armijo condition. The problem of this condition is, that it could accept arbitrary small values α , which may slow down solution of the problem. In practice, c_1 is chosen to be quite small, say $c_1 \approx 10^{-4}$.

Curvature condition

To rule out unacceptably short steps one can introduce a second requirement:

$$-\nabla f(x_k - \alpha \nabla f(x_k))^\top \nabla f(x_k) \geq c_2 \nabla f(x_k)^\top (-\nabla f(x_k))$$

for some constant $c_2 \in (c_1, 1)$, where c_1 is a constant from Armijo condition. Note that the left-handside is simply the derivative $\nabla_\alpha \phi(\alpha)$, so the curvature condition ensures that the slope of $\phi(\alpha)$ at the target point is greater than c_2 times the initial slope $\nabla_\alpha \phi(\alpha)(0)$. Typical values of $c_2 \approx 0.9$ for Newton or quasi-Newton method. The sufficient decrease and curvature conditions are known collectively as the Wolfe conditions.

Goldstein conditions

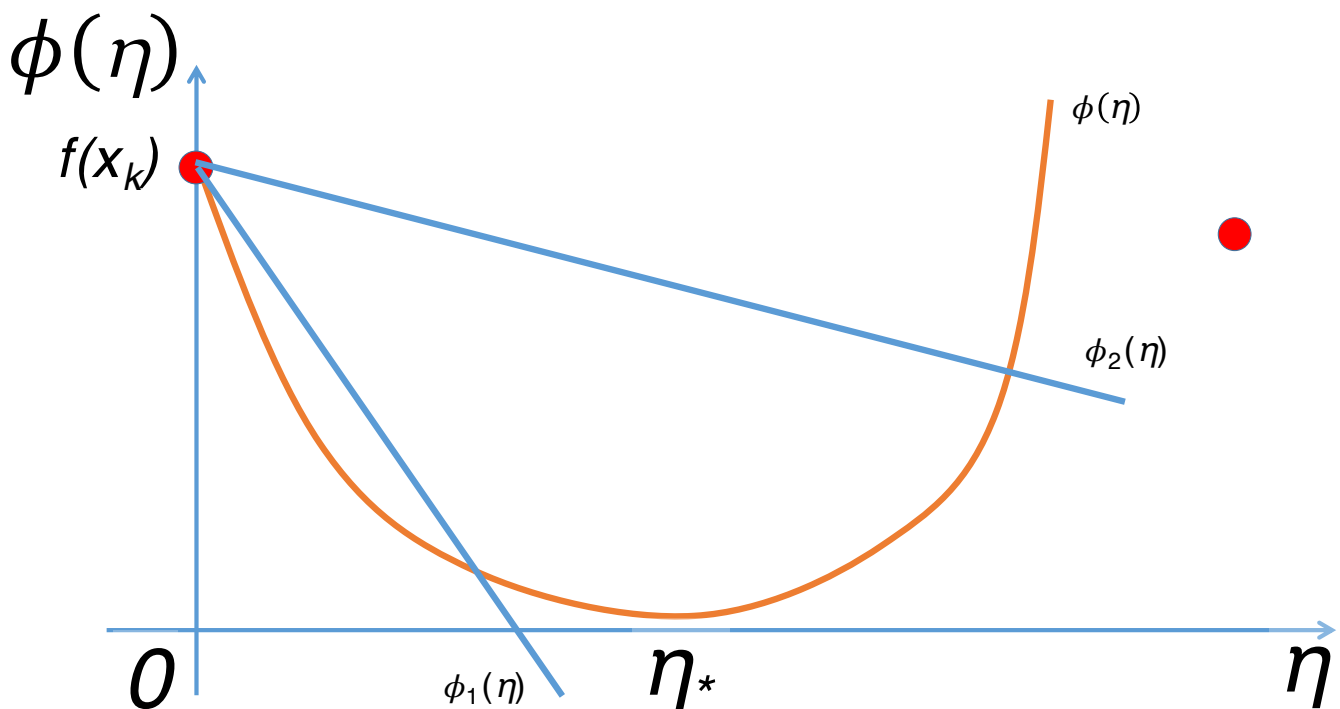
Let's consider also 2 linear scalar functions $\phi_1(\alpha), \phi_2(\alpha)$:

$$\phi_1(\alpha) = f(x_k) - \alpha \alpha \|\nabla f(x_k)\|^2$$

and

$$\phi_2(\alpha) = f(x_k) - \beta \alpha \|\nabla f(x_k)\|^2$$

Note, that Goldstein-Armijo conditions determine the location of the function $\phi(\alpha)$ between $\phi_1(\alpha)$ and $\phi_2(\alpha)$. Typically, we choose $\alpha = \rho$ and $\beta = 1 - \rho$, while $\rho \in (0.5, 1)$.



References

- Numerical Optimization by J.Nocedal and S.J.Wright.