FEDERAL STATE AUTONOMOUS EDUCATIONAL INSTITUTION
OF HIGHER EDUCATION
ITMO UNIVERSITY

Report
on the practical task No. 3

Algorithms for unconstrained nonlinear optimization. First- and second order methods

Performed by
Dmitry Grigorev, Maximilian Golovach
J4133c

Accepted by
Dr Petr Chunaev

St. Petersburg
2022

1. Goal of the work

The goal of this work is compound and consists of the following points:

- to become familiar with first- and second order methods in the problem of unconstrained nonlinear optimization;
- to apply the methods on the practical problem and compare them with each other;
- to compare the obtained results with the results of Task 2.

2. Formulation of the problem

Let $\alpha,\ \beta \in (0,1)$ are two arbitrary values, $x_k = \frac{k}{100},\ k = 0, \ldots, 100$ and y_k are defined by i.i.d. $\delta_k \sim N(0,1)$ according to the following rule:

$$y_k = \alpha x_k + \beta + \delta_k.$$

Given such two parametrized families of functions as:

1. $F(x,a,b) = ax + b$,
2. $F(x,a,b) = \frac{a}{1+bx}$,

we need to find the optimal function for given data $(x_k, y_k)_{k=0}^{100}$ according to the least squares:

$$D(a,b) = \sum_{k=0}^{100} (F(x_k,a,b) - y_k)^2$$

using

- gradient descent,
- conjugate gradient descent,
- Newton's method,
- Levenberg-Marquardt method.

All functions which will be obtained have to be visualized with the given data and the line which generates these data. Furthermore, we have to compare the algorithms in terms of precision, number of iterations and number of function evaluations using the results.

3. Brief theoretical part

Suppose that we are given with a function $f: G \subset \mathbb{R}^n \to \mathbb{R}$ where G is connected open set in $\mathbb{R}^n$. We need to minimize this function on G.

3.1. Gradient descent method

Let the function f to be once continuously differentiable on G ($f \in C^1(G)$). The gradient descent method utilizes the fundamental fact that the gradient of function ∇f in a point is directed towards the fastest growth of the function among all other directions (**steepest ascent**) and, vice versa, $-\nabla f$ — the direction of the **steepest descent**. At each step of the algorithm next point is resulted by moving from the previous point in the direction of the antigradient at this point.

The gradient descent has such parameters as initial point x_0 and α which is called. **learning rate**.

The problems of this algorithm: — prone to criss-cross pattern of moves between two consequential points in the valleys of function what slows the rate of convergence. — near the points where $\nabla f(x) \approx 0$ the convergence is slow.

Algorithm 1 Gradient descent method algorithm

```
Require: $f \in C^1(G)$, $x_0 \in G$ — initial point, $\varepsilon > 0$ — precision.
  $k \leftarrow 0$
  while $\|x_k - x_{k-1}\| > \varepsilon$ do
      $k \leftarrow k + 1$
      $x_k \leftarrow x_{k-1} - \alpha \nabla f(x_{k-1})$
  end while
Ensure: $\widehat{x} = x_k$ — point of local minimum or critical point
```

3.2. Conjugate gradient descent

Originally, the conjugate optimization method was introduced for solving the linear system of equations problem with symmetric positive-definite matrix. The main idea of this method is that the twice differentiable function with non-degenerate second derivative at given point behaves like a quadratic paraboloid.

Let $f \in C^2(G)$, $H = \frac{d^2 f}{dx^2}$ and we are given with an initial point $x_0 \in G$ and precision $\varepsilon > 0$. At each step in point x_k the algorithm uses not only the direction of the steepest descent $-\nabla f(x_k)$ but the additional direction s_k which is conjugated to s_{k-1} in the sense of orthogonality with respect to $\mathbf{H}_{k-1} = H(x_{k-1})$:

$$s_k^T \mathbf{H}_{k-1} s_{k-1} = 0.$$

The conjugacy is obtained by means of the following equation which defines s_k:

$$s_k = -\nabla f(x_k) + \beta_k s_{k-1}.$$

Algorithm 2 Conjugate gradient descent algorithm

```
Require: $f \in C^2(G)$, $H = \frac{d^2 f}{dx^2}$, $x_0 \in G$ — initial point, $\varepsilon > 0$ — precision.
  $k \leftarrow 0$
  $g_0 \leftarrow -\nabla f(x_0)$
  $s_0 \leftarrow g_0$
  $\alpha_0 \leftarrow \arg\min_{\alpha>0} f(x_0 + \alpha s_0)$
  $x_1 \leftarrow x_0 + \alpha_0 s_0$
  while $\|x_k - x_{k-1}\| > \varepsilon$ do
      $k \leftarrow k + 1$
      $g_k \leftarrow -\nabla f(x_k)$
      Obtain $\beta_k$ from the known formula (see below)
      $s_k \leftarrow g_k + \beta_k s_{k-1}$
      $\alpha_k \leftarrow \arg\min_{\alpha>0} f(x_k + \alpha s_k)$
      $x_k \leftarrow x_k + \alpha_k s_k$
  end while
Ensure: $\widehat{x}$ — point of local minimum or critical point
```

There are several ways to calculate β_n. The classical one is **Fletcher-Reeves**:

$$\beta_k = \frac{g_k^T g_k}{g_{k-1}^T g_{k-1}}$$

The another one is **Polak–Ribière**:

$$\beta_k = \frac{g_k^T (g_k - g_{k-1})}{g_{k-1}^T g_{k-1}}$$

The advantage in using Polak–Ribiere formula is that it automatically reset its moves (i.e. $s_k \approx g_k$) when when little progress is made over the last iteration. This property can speed up the convergence near the solution point [1].

The conjugate gradient methods have such an advantage as no matrix operations are required in their algorithms but they are not robust. These methods fix the problem of gradient descent with behavior in function valleys.

3.3. Newton's method

Newton's method is the representative of **Trust region approach algorithms**. The trust region approach assumes that the objective function f is well approximated by a quadratic function $q_k(\delta)$ obtained by truncating the Taylor series for $f(x_k+\delta)$. In the case of Newton's approach q_k is as follows:

$$f(x_k+\delta) \approx q_k(\delta) = f(x_k) + g_k^T\delta + \frac{1}{2}\delta^T\mathbf{H}_k\delta,$$

where $g_k = \nabla f(x_k)$, $\mathbf{H}_k = H(x_k) = \frac{d^2f}{dx^2}(x_k)$. The next point x_{k+1} is resulted from the minimization of $q_k(\delta)$ which is equivalent to solving the system $\mathbf{H}_k\delta = -g_k$.

Algorithm 3 Newton's method algorithm

```
Require: $f \in C^2(G)$, $H = \frac{d^2f}{dx^2}$, $x_0 \in G$ — initial point, $\varepsilon > 0$ — precision.
  $k \leftarrow 0$
  while $\|x_k - x_{k-1}\| > \varepsilon$ do
      $g_k \leftarrow \nabla f(x_k)$
      $\mathbf{H}_k \leftarrow H(x_k)$
      $\delta_k \leftarrow -\mathbf{H}_k^{-1}g_k$
      $k \leftarrow k+1$
      $x_k \leftarrow x_{k-1} + \delta_{k-1}$
  end while
Ensure: $\widehat{x}$ — point of local minimum or critical point
```

Newton's method may fail to converge when $\mathbf{H}_k$ is not positive definite.

3.4. Levenberg-Marquardt method

Levenberg-Marquadt method is called to solve the problem of Newton's method in non-guaranteed positive definiteness of $\mathbf{H}_k$. Here δ_k is computed from the system:

$$(\mathbf{H}_k + \nu\mathbf{I})\delta_k = -g_k,$$

where $\nu \geq 0$ is chosen so that the matrix $\mathbf{H}_k + \nu\mathbf{I}$ is positive definite.

4. Results

5. Data structures and design techniques used in algorithms

In the implementation of the algorithms and for random variables generation the Python's package Numpy is used since it allows to vectorize calculations what accelerates the evaluations.

6. Conclusion

As the result of this work, we considered direct methods of optimization in the cases of one and two variables. The methods were compared by the number of iterations required to obtain the solution with given precision, the number of function evaluations and precision. All of the considered methods except Nelder-Mead method were implemented in Python programming language. Discussion for data structures and design techniques which were used in the implementations is provided. The work goals were achieved.

7. Appendix

Algorithms implementation code is provided in [2].

Bibliography

1. Cavazzuti M. Optimization Methods. — Springer Berlin, Heidelberg, 2013. — ISBN: 9783642311864.
2. Grigorev D., Golovach M. Code repository. — `https://github.com/dmitry-grigorev/AlgoAnalysisDevelopment`. — 2022.