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

Ekaterina Deviatiarova,

Valeria Sakovskaya

J4134c

Accepted by

Dr Petr Chunaev

St. Petersburg

2022

**Goal**

The use of first- and second-order methods (Gradient Descent, Non-linear Conjugate Gradient Descent, Newton’s method and Levenberg-Marquardt algorithm) in the tasks of unconstrained nonlinear optimization.

**Formulation of the problem**

Generate random numbers and . Furthermore, generate the noisy data , where , according to the following rule:

, ,

where are values of a random variable with standard normal distribution. Approximate the data by the following linear and rational functions:

1. (linear approximant).
2. (rational approximant).

by means of least squares through the numerical minimization (with precision ) of the following function:

To solve the minimization problem, use the methods of Gradient Descent, Conjugate Gradient Descent, Newton’s method and Levenberg-Marquardt algorithm. If necessary, set the initial approximations and other parameters of the methods. Visualize the data and the approximants obtained in a plot separately for each type of approximant so that one can compare the results for the numerical methods used. Analyze the results obtained (in terms of number of iterations, precision, number of function evaluations, etc.) and compare them with those from Task 2 for the same dataset.

**Brief theoretical part**

**Gradient descent**

Gradient descent is an optimization algorithm used to minimize some function by iteratively moving in the direction of steepest descent as defined by the negative of the gradient. In machine learning, we use gradient descent to update the parameters of our model. Parameters refer to coefficients in Linear Regression and weights in neural networks.

You may recall the following formula for the slope of a line, which is , where represents the slope and is the intercept on the y-axis.

You may also recall plotting a scatterplot in statistics and finding the line of best fit, which required calculating the error between the actual output and the predicted output (y-hat) using the mean squared error formula. The gradient descent algorithm behaves similarly, but it is based on a convex function, such as the one represented in Figure 1.

The starting point is just an arbitrary point for us to evaluate the performance. From that starting point, we will find the derivative (or slope), and from there, we can use a tangent line to observe the steepness of the slope. The slope will inform the updates to the parameters—i.e. the weights and bias. The slope at the starting point will be steeper, but as new parameters are generated, the steepness should gradually reduce until it reaches the lowest point on the curve, known as the point of convergence.

Similar to finding the line of best fit in linear regression, the goal of gradient descent is to minimize the cost function, or the error between predicted and actual y. In order to do this, it requires two data points—a direction and a learning rate. These factors determine the partial derivative calculations of future iterations, allowing it to gradually arrive at the local or global minimum (i.e. point of convergence).

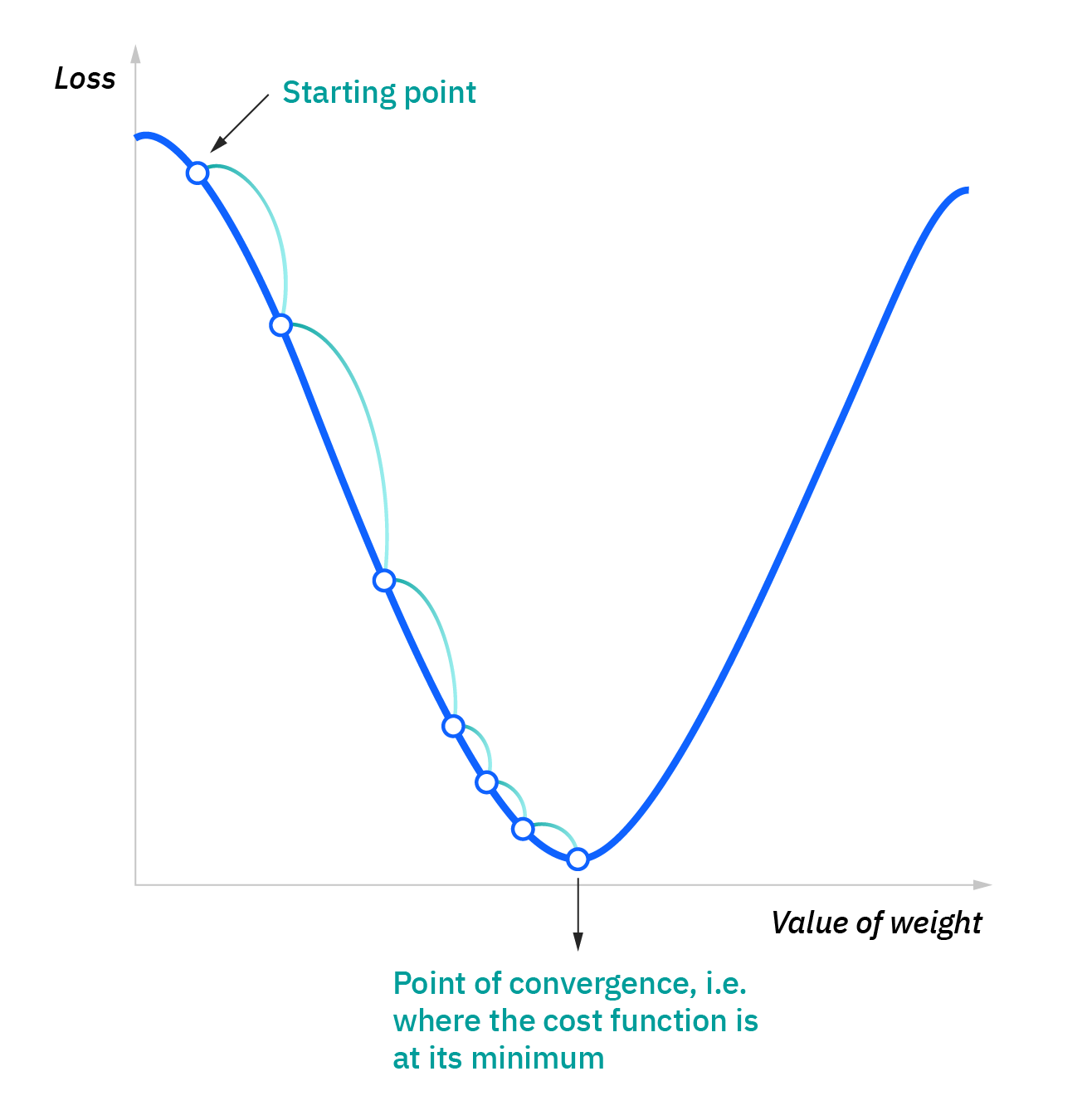


Figure 1 – Gradient Descent

**Conjugate gradient method**

The conjugate gradient method is a conjugate direction method in which selected successive direction vectors are treated as a conjugate version of the successive gradients obtained while the method progresses. The conjugate directions are not specified beforehand but rather are determined sequentially at each step of the iteration. If the conjugate vectors are carefully chosen, then not all the conjugate vectors may be needed to obtain the solution. Therefore, the conjugate gradient method is regarded as an iterative method. This also allows approximate solutions to systems where is so large that the direct method requires too much time.

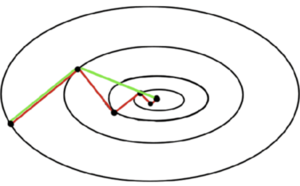


Figure 2 – A comparison of the convergence of gradient descent (in red)  
and conjugate vector (in green) for minimizing a quadratic function

**Newton’s method**

Newton's method, also known as the Newton–Raphson method, is a root-finding algorithm which produces successively better approximations to the roots (or zeroes) of a real-valued function.

Let be a differentiable function. Select a point based on a first approximation to the root, arbitrarily close to the function's root. To approximate the root, you then recursively calculate using:

As you recursively calculate, the 's often become increasingly better approximations of the function's root.

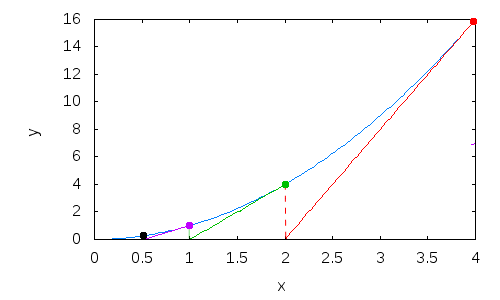


Figure 3 – A few iterations of Newton's method applied to starting with . The blue curve is . The other solid lines are the tangents at the various iteration points

**Levenberg-Marquardt algorithm**

The Levenberg-Marquardt algorithm combines two numerical minimization algorithms: the gradient descent method and the Gauss-Newton method. In the gradient descent method, the sum of the squared errors is reduced by updating the parameters in the steepest-descent direction. In the Gauss-Newton method, the sum of the squared errors is reduced by assuming the least squares function is locally quadratic in the parameters, and finding the minimum of this quadratic. The Levenberg-Marquardt method acts more like a gradient-descent method when the parameters are far from their optimal value, and acts more like the Gauss-Newton method when the parameters are close to their optimal value.

The Levenberg-Marquardt algorithm adaptively varies the parameter updates between the gradient descent update and the Gauss-Newton update,

,

where small values of the damping parameter result in a Gauss-Newton update and large values of result in a gradient descent update. The damping parameter is initialized to be large so that first updates are small steps in the steepest-descent direction. If any iteration happens to result in a worse approximation , then is increased. Otherwise, as the solution improves, is decreased, the Levenberg-Marquardt method approaches the Gauss-Newton method, and the solution typically accelerates to the local minimum.

**Results**

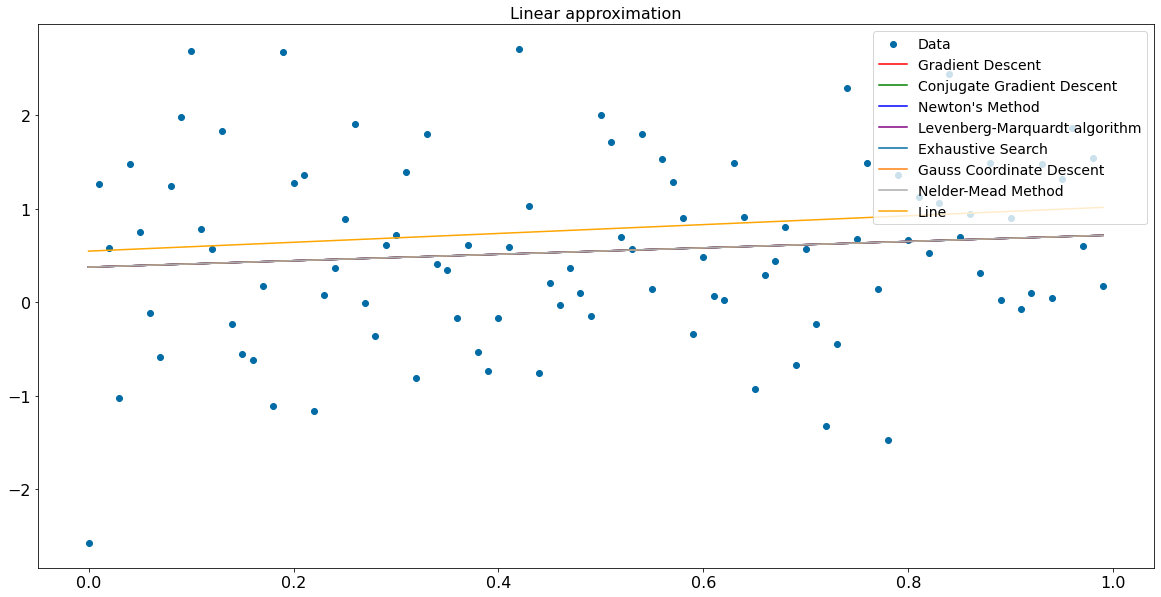


Figure 4 – Results of using gradient descent,   
the conjugate gradient method, the Newton method and the Levenberg-Marquardt algorithm to solve the linear approximation problem

It is well known that the specified optimization problem associated with linear approximation has a single solution, and therefore it is expected that these methods will give very similar optimal values for a and b, regardless of the choice of initial approximations.

In the case of rational approximation, significant nonlinearities arise, and therefore the choice of the initial approximation can significantly affect the result. Figure 5 shows an example when the methods of iteration, Gauss and Nelder-Mead still lead to fairly close solutions to the problem of rational approximation in the sense of least squares of deviations.

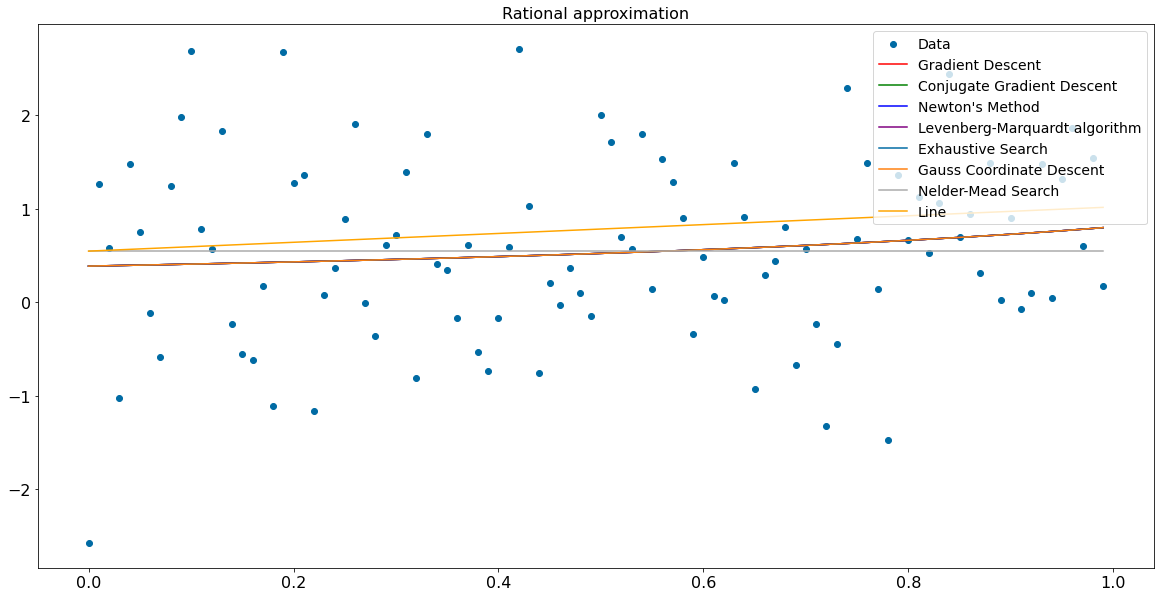


Figure 5 – Results of using gradient descent,   
the conjugate gradient method, the Newton method and the Levenberg-Marquardt algorithm to solve the rational approximation problem

In the case of rational approximation, significant nonlinearities arise, and even the choice of the initial approximation can significantly affect the result. Figure 5 shows an example when Gradient Descent, the Conjugate Gradient, Newton’s method and Levenberg-Marquardt algorithm still lead to fairly close solutions to the problem under consideration.

Now let’s analyze the results obtained (in terms of number of iterations, precision, number of function evaluations, etc.) and compare them with those from Task 2 for the same dataset.

Figure 6 shows the number of iterations performed in each algorithm. As we can see, Gradient descent method demonstrated the best result in terms of number of iterations (2 iterations), whereas the Nelder-Mead method gave the worst outcome (72 iterations).



Figure 6 – The numerical comparison between methods for linear approximation problem

As you can notice in Figure 7, the Exhaustive Search algorithm demonstrated the worst result (more than 45 iterations) in the task of rational approximation, although it performed 20 iterations in the task of linear approximation. Newton’s method gave the best outcome (less than 10 iterations).



Figure 7 – The numerical comparison between algorithms for rational approximation problem

The efficiency of an optimization algorithm refers to the computational effort required to obtain a solution. In mathematical programming, there are two primary measures of efficiency, the number of fundamental evaluations and the running time.

As we can see in the table below, the Conjugate gradient descent method finds a solution faster than any other algorithm on the same dataset. It’s running time is only 1.6 ms!

**Table 1.** Execution time of algorithms for linear approximation problem

|  |  |
| --- | --- |
| **Method** | **Execution Time, s** |
| Conjugate gradient descent method | 0.0016 |
| Gauss (coordinate descent) | 0.0026 |
| Gradient descent method | 0.0031 |
| Levenberg-Marquardt algorithm | 0.0058 |
| Nelder-Mead | 0.0132 |
| Newton’s method | 0.4112 |
| Exhaustive Search | 28.500 |

It can be seen from Table 2 that the time efficiency of the Levenberg-Marquardt algorithm is much higher than that of other algorithms because it is the hybrid technique that uses both Gauss–Newton and steepest descent approaches to converge to an optimal solution.

**Table 2.** Execution time of algorithms for rational approximation problem

|  |  |
| --- | --- |
| **Method** | **Execution Time, s** |
| Levenberg-Marquardt algorithm | 0.0022 |
| Gradient descent method | 0.0044 |
| Nelder-Mead | 0.0058 |
| Conjugate gradient descent method | 0.0064 |
| Gauss (coordinate descent) | 0.0074 |
| Newton’s method | 0.3894 |
| Exhaustive Search | 32.242 |

In both cases, the brute-force approach or exhaustive search is inefficient. This algorithm computes the function’s value at each point of a multidimensional grid of points, to find the global minimum of the function. And the number of grid points increases exponentially. That is why exhaustive search gave the worst outcome (28,5 and 32,2 seconds).

**Conclusions**

Minimization problem was solved using the methods of Gradient Descent, Conjugate Gradient Descent, Newton’s method and Levenberg-Marquardt algorithm. This report provides experimental results of using mentioned methods and algorithms in the task of unconstrained nonlinear optimization. Each method was used to solve the optimization problem.

Although, almost all methods found fairly close solutions, some of them were inefficient in terms of number of iterations and running time. E.g., Nelder-Mead method and exhaustive search gave the worst outcome for linear and rational approximation, correspondingly. As for the running time, Conjugate gradient descent method and the Levenberg-Marquardt algorithm showed best results (less than 3 ms).

**Appendix**

Source code can be found in GitHub repository: github.com/ellkrauze/algorithms2022.