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

Dmitriy Prusskiy

J4132c

Accepted by

Dr Petr Chunaev

St. Petersburg

2021

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 $\alpha \in (0,1)$ and $\beta \in (0,1)$. Furthermore, generate the noisy data $\{x_k, y_k\}$, where $k = 0, ...,100$, according to the following rule:

$$y_k = \alpha x_k + \beta + \delta_k,\ x_k = \frac{k}{100},$$

where $\delta_k \sim N(0,1)$ are values of a random variable with standard normal distribution. Approximate the data by the following linear and rational functions:

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

by means of least squares through the numerical minimization (with precision $\varepsilon = 0.001$) of the following function:

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

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 is a first-order iterative optimization algorithm for finding a local minimum of a differentiable function. To find a local minimum of a function using gradient descent, one should take steps proportional to the negative of the gradient (or approximate gradient) of the function at the current point. Gradient descent is based on the observation that if the multi-variable function is defined and differentiable in a neighborhood of a point, then decreases fastest if one goes from in the direction of the negative gradient of F.

The idea of the **conjugate gradient method** is that instead of doing small steps in the direction of the negative gradient, we find a minimum of the given function in that direction, and then make a huge step to this point of minimum, and then repeat. This method is quite effective in solving unconstrained optimization problems such as energy minimization.

Newton's method in optimization is applied to the derivative *f ′* of a twice differentiable function *f* to find the roots of the derivative (solutions to *f ′(x) = 0*), also known as the stationary points of *f*. The geometric interpretation of Newton's method is that at each iteration, it amounts to the fitting of a paraboloid to the surface of *f(x)* at the trial value x_k having the same slopes and curvature as the surface at that point, and then proceeding to the maximum or minimum of that paraboloid (in higher dimensions, this may also be a saddle point). If *f* happens to be a quadratic function, then the exact extremum is found in one step.

The **Levenberg–Marquardt algorithm**, also known as the damped least-squares (DLS) method, is used to solve non-linear least squares problems. This method is a combination of two minimization methods: 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 and finding the minimum of the 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.

Results

The results were obtained using Python 3.8.5.

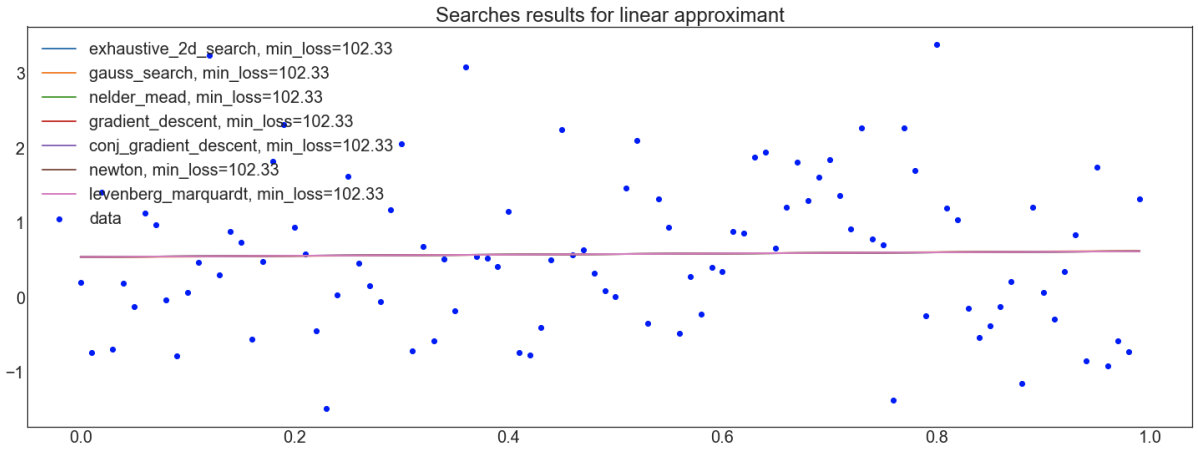
Gradient descent algorithm implementation used fixed learning rate 10^{-4}. Conjugate gradient method, Newton's method and Levenberg–Marquardt algorithm were implemented using scipy.optimize module. Gradients for linear and rational functions with respect to their parameters were calculated using algebraic form.

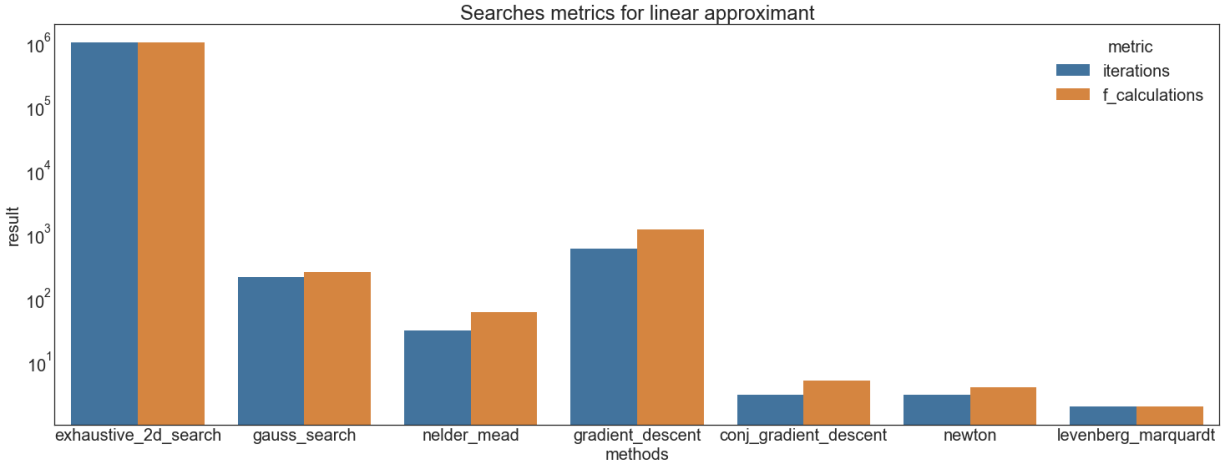
All described methods for both linear and rational approximations converge to the same parameters which coincide with results obtained by methods from Task 2.

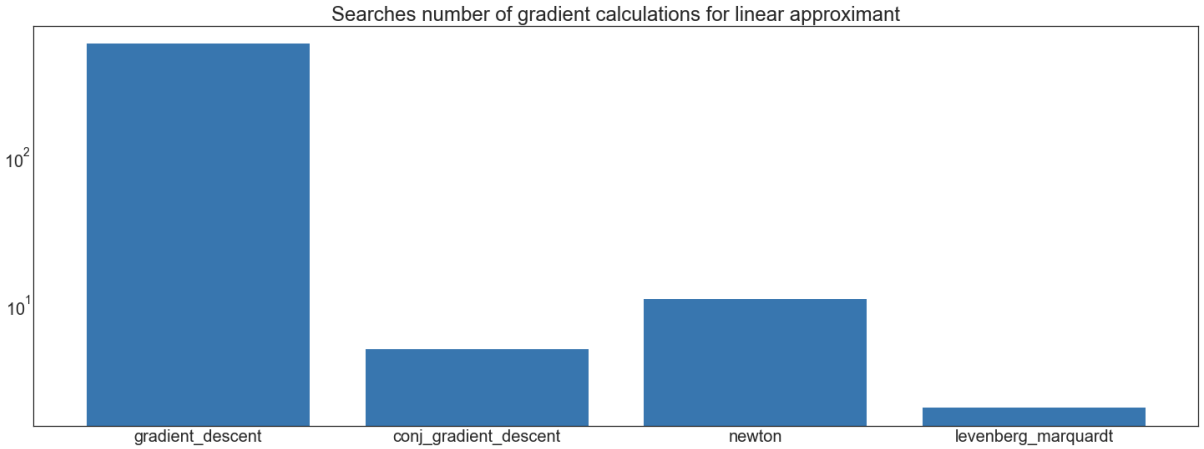
Gradient methods (except for simple gradient descent with fixed learning rate) perform less iterations, function and gradient evaluations. These methods perform less then 10 iterations, function and gradient evaluations for linear function and less then 30 - for rational function. In comparison the best method from Task 2 - Nelder-Mead method - performs about 100 function evaluations for both linear and rational functions.

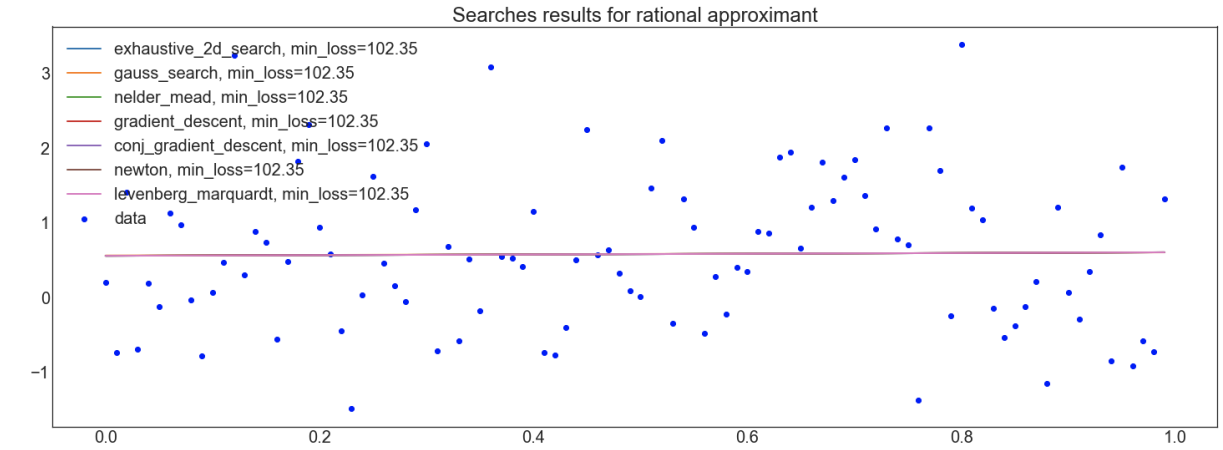
The best performance for linear approximation obtained by Levenberg–Marquardt algorithm, but for rational approximation Newton's method performed less iterations and function calculations.

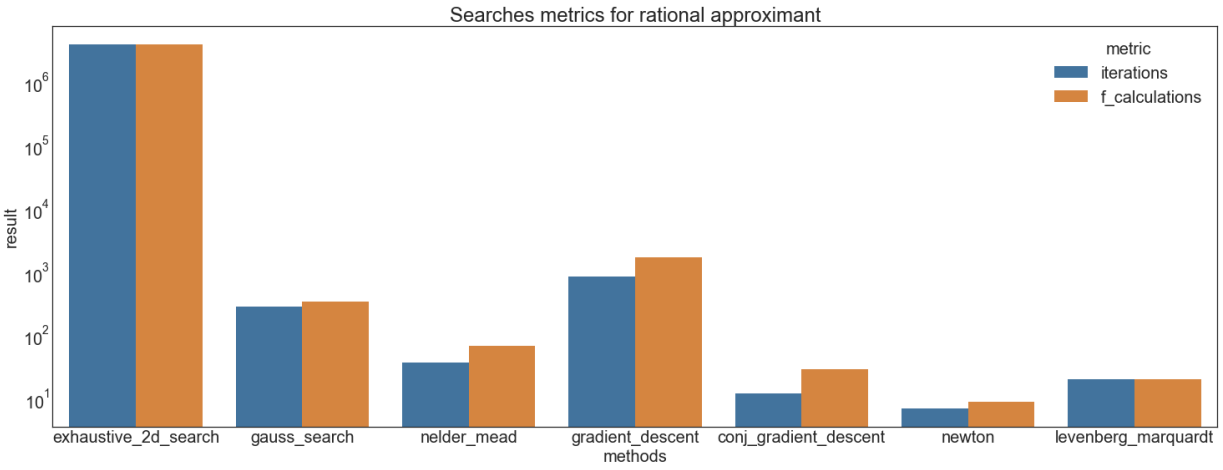
Below are graphs with the results of the study.

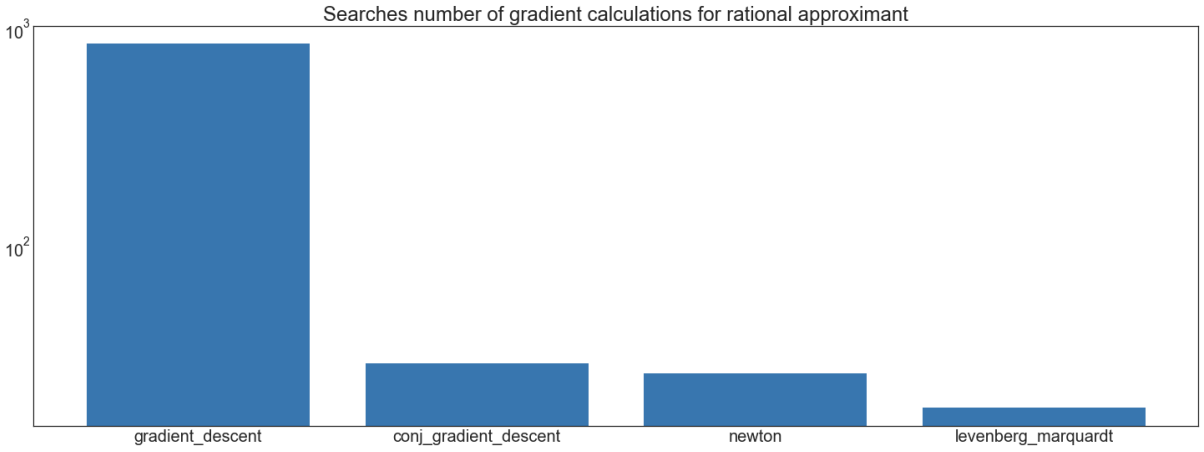

Searches results for linear approximant
exhaustive_2d_search, min_loss=102.33
gauss_search, min_loss=102.33
nelder_mead, min_loss=102.33
gradient_descent, min_loss=102.33
conj_gradient_descent, min_loss=102.33
newton, min_loss=102.33
levenberg_marquardt, min_loss=102.33
data
3
2
1
0
−1
0.0
0.2
0.4
0.6
0.8
1.0


Searches metrics for linear approximant
metric
iterations
f_calculations
result
10^6
10^5
10^4
10^3
10^2
10^1
exhaustive_2d_search
gauss_search
nelder_mead
gradient_descent
conj_gradient_descent
newton
levenberg_marquardt
methods


Searches number of gradient calculations for linear approximant
10^2
10^1
gradient_descent
conj_gradient_descent
newton
levenberg_marquardt







Conclusions

The goal of this study was to use gradient methods of optimization in the two-dimensional optimization tasks and compare obtained results with results from Task 2 for the same dataset. Analysis of the graphs presented in the Results section shows that all examined methods converge to the same values and gradient methods make it faster and take less computations then numerical methods from Task 2.

Appendix

Source code can be found at https://github.com/T1MAX/itmo_algorithms/tree/main/task_3