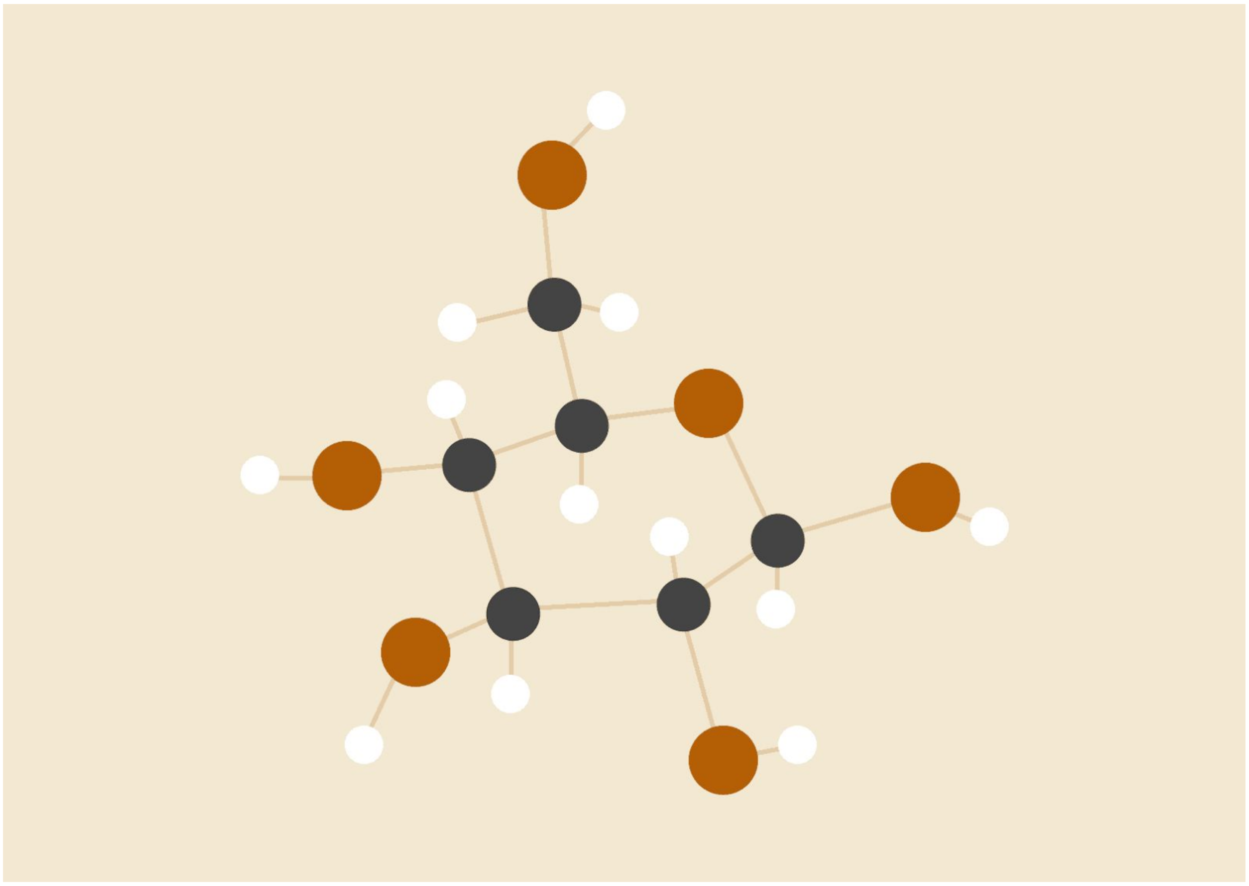
Quantitative Foundations

Project 2 - Unconstrained Optimization



Ayush Kumar Shah (as1211@rit.edu)

Murtaza Tamjeed (mt1256@rit.edu)

26.10.2020

Overview

In this project, we experimented with three optimization methods on three optimization functions and analyzed the performance. For each method, we tested the impacts of different line search parameters and compared the behavior and time of the three different methods.

1. Function one (fun1)

The function one given in the project is a quadratic function. We initialized x as ones(100, 1) * 50, and used maximum iterations as 1000, and delta as 1e-4 for the stopping condition. The global minimum was provided as $f(x)_{min} = 0$.

Optimization solutions of fun1

Gradient's Descent Method
f_min = 0.000062; total time=0.022105 sec; time per iteration=0.00024 sec; steps=92;

Newton Method
f_min = 0.000000; total time=0.00141 sec; time per iteration=0.00141 sec; steps=1;

Quasi-Newton Method
f_min = 0.000000; total time=0.048935 sec; time per iteration=0.000418 sec; steps=117;

Impact of parameters on the performance of fun1

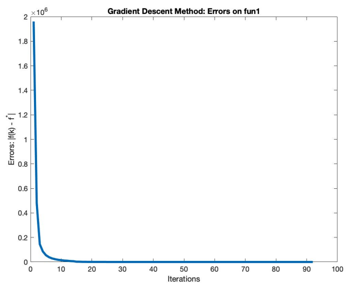
To understand the impact of line search parameters in the optimization for Gradient Descent, we used a grid search on 180 different combinations with varying alpha_i, c, and rho to find the best parameters. The best values for the parameters obtained were **(alpha_i = 1.0, c=0.58, rho=0.9).** For Quasi-Newton's method, we used a grid search with 36 different combinations varying c and rho but fixed alpha_i as 1. The best values for the parameters obtained were **(alpha_i = 1.0, c=0.1, rho=0.9).** We did not perform a grid search for Newton's method since it converges in a single step with the default parameters **(alpha_i = 1.0, c=0.1, rho=0.5).**

In general, using a lower value of alpha requires more steps to converge and the value of 1 is generally better to get better results. Likewise, the optimization results are generally good when c is set to a low value like 0.1 and the results get worse for larger values of c since the larger value of c may not find satisfactory step size (alpha).

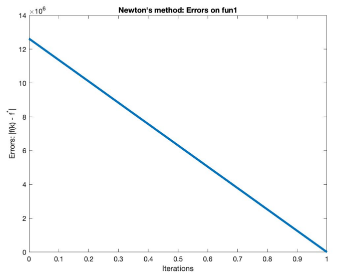
Further analysis and comparison of the algorithms on fun1

The Quasi-Newton's method took the longest time to converge while Newton's method the least. However, time per iteration was the least for gradient descent and highest for Newton's method due to the computation and storage of Hessian along with solving a linear system in Newton's method.

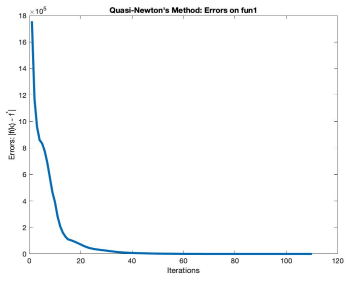
The gradient descent method was able to obtain functional value very close to the global minimum but not exactly equal while Newton's method and Quasi-Newton's method were able to converge to the global minimum. Moreover, Newton's method reached the global minimum in a single iteration since the input function is a quadratic function, as assumed by the method. Thus, Newton's method outperforms the other two methods for function one due to the function's behavior. The following figures present the convergence curve of the algorithms for function one.



Gradient descent



Newton



Quasi-Newton

2. Function two (fun2)

Function two in the project is a large dimensional function with logarithms. Since the logarithms of negative values are not real, the domain for the input is restricted by $b_i - a_i^T x >= 0$. We also updated the backtracking line search algorithm to ensure that the updated step-size (alpha) does not allow the input (x) to move out of the domain i.e. it satisfies the above constraint.

We initialized x as ones(100, 1) * 0.1, which is within the valid domain, and used maximum iterations as 300, and delta as 1e-2 for the stopping conditions since it involves larger computations due to the large dimension of the coefficients.

Optimization solutions of fun2

Gradient's Descent Method
f_min = -2440.219455; total time=301.782423 sec; time per iteration=1.005941 sec; steps=300;

Newton Method
f_min = -2443.244762; total time=6.212515 sec; time per iteration=0.414168 sec; steps=15;

Quasi-Newton Method
f_min = -2443.207609; total time=43.474292 sec; time per iteration=0.762707 sec; steps=57;

Impact of parameters on the performance of fun2

Since the function is very large dimensional, it took a long computation time (about 5 mins for gradient descent) to run. Hence, a grid search was not feasible for this function. So, we used the default parameters **(alpha_i = 1.0, c=0.1, rho=0.5).**

Further analysis and comparison of the algorithms on fun2

The gradient descent method could not result in a functional value that satisfies the stopping condition and hence stopped at the maximum iterations (300) in about 5 minutes. This might be due to the complex nature and large dimension of function 2. So, the gradient descent having a slow convergence (linear) could not converge to a value in time.

However, due to the faster convergence rate, Newton's method could converge very quickly in 15 steps, and Quasi-Newton in 57 steps. So, for a complex and large dimensional function like function2, Gradient Descent is not reliable, whereas the other two methods can converge although taking some time. Hence, the Newton method performs better than the other two functions on function two. However, the functional values obtained by all three algorithms are very close to each other.

3. Function three (fun3)

Function three in the project is a polynomial function of order 4. We initialized x as [50;50] and used maximum iterations as 1000, and delta as 1e-4 for the stopping conditions. The global minimum $f(x)_{min} = 0$, is clear from the function.

Optimization solutions of fun3

Gradient's Descent Method
x_min = [6.9814 ; 48.7434]; f_min = 35.778311; total time=0.004341 sec; time per iteration=0.000197 sec; steps=22;

Newton Method
x_min = [1.0001; 1.0001]; f_min = 0.000000; total time=0.0171 sec; time per iteration=0.000132 sec; steps=130;

Quasi-Newton Method
x_min = [0.9995;0.9991]; f_min = 0.000000; total time=0.022102 sec; time per iteration=0.000104 sec; steps=202;

Impact of parameters on the performance of fun3

Similar to function 1, we used a grid search on 180 different combinations with varying alpha_i, c, and rho to find the best parameters for Gradient Descent. The best values for the parameters obtained were **(alpha_i = 0.1, c=0.1, rho=0.1).** For Quasi-Newton's method, we used a grid search with 36 different combinations varying c and rho but fixed alpha_i as 1. The best values for the parameters obtained were **(alpha_i = 1.0, c=0.1, rho=0.74).** We did not perform a grid search for Newton's method.

In general, using a lower value of alpha requires more steps to converge and the value of 0.1 is generally better to get better results. Likewise, the optimization results are generally good when c is set to a low value like 0.1 and the results deviate from the global minimum for larger values of c since the larger value of c may not find satisfactory step size (alpha).

Further analysis and comparison of the algorithms on fun3

Although the gradient descent method converges fastest to a minimum value, it deviates very far from the global minimum (0). The stopping condition is satisfied as the difference is less than 1e-4 while the number of steps is just 22, which suggests that the method gets stuck in a local minimum. On the other hand, both Newton's method and Quasi_newton's method were successful to converge very close to a global optimum although it took a longer time than gradient descent. Also, in this case, the time per iteration was the least for Quasi_newton's method and highest for the Gradient Descent method.

Discussion and Conclusion

Overall, Newton's method performs well since it generally takes less time to converge. However, the gradient descent method, although does not require Hessian and involves fewer computations at each iteration, it takes a very long time to converge, especially for complex and high dimensional functions, and may get stuck at local minima. The time is dependent on the rate of convergence, which is linear, quadratic, and superlinear for Gradient's descent, Newton's method, and Quasi-Newton respectively.

Appendix

Experiment Results

Team Member Contributions:

We did most of the work together during our scheduled meetings. We implemented most of the algorithms during the breakout room sessions in the past two weeks as part of our class activities. We also met right after the classes trying to improve our code. More specifically for the project, Ayush took care of writing the functions, gradients, and Hessians for functions 1 and 2, improving our code related to functions 1 and 2 of the project, with the implementation of domain restriction for function 2 and grid search for selecting the best parameters; and Murtaza took care of writing the function, gradient and Hessian of function 3 and improving the code related to function 3. Also, Murtaza wrote a first draft of the project report and the presentation. Then, both Ayush and Murtaza worked together to improve them during multiple meetings.