Comparison of Optimization Methods

Paul Pluscht

May 2022

Abstract

The following paper compares the performance of various optimization functions. To ensure accuracy and reproducibility, the scipy implementaions of the algorithms will be used. The metric used to compare algorithms will primarily be function evaluations.

1 Brief Descriptions of the Algorithms Used

1.1 Newton's Method

Newton's method is the oldest algorithm discussed in this paper. If the set of standard assumptions are met, the method can guarantee quadratic convergence in all dimensions. However, the method requires the explicit calculation of the gradient and hessian, making this method very computationally expensive. The following defines the standard assumptions needed to guarantee quadratic convergence.

- f is twice differentiable and lipschitz

$$\exists \gamma \ \forall x, y \in X, \ ||\nabla^2 f(x) - \nabla^2 f(y)|| \leq \gamma |x - y|$$

- At the minimum point x^*,

$$\nabla f(x^*) = 0$$

- At the minimum point x^*, $\nabla^2 f(x^*)$ is positive definite.

The method works by solving the local quadratic model for the precise zero. If the this new point is below our tolerance, we can stop the iteration at this time step. Otherwise, the next time step is used to define the new local quadratic model. The local quadratic model is defined by the following equation.

$$m_n(x) = f(x) + \nabla f(x)^T (x - x_n) + \frac{1}{2}(x - x_n)^T \nabla^2 f(x_n)(x - x_n)$$

In practice, to solve this equation we solve

$$\nabla^2 f(x_n))s = -1 \times \nabla f(x_n)$$

. Then, our next point $x_{n+1} = x_n + s$. The figure below shows an single timestep for a one dimensional function.

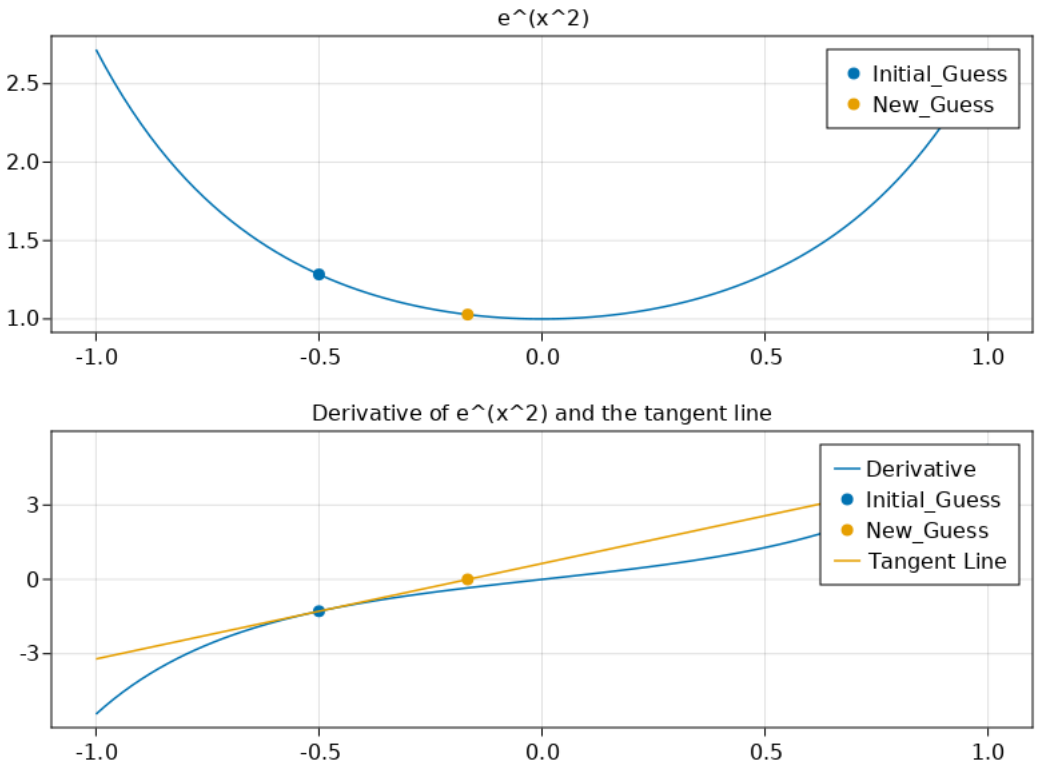

Figure 1: One Timestep of Newton's Method

1.2 BFGS

BFGS is a method with a similar structure to Newton's method, but we use the function values to approximate the hessian. In addition, variations of this method can use the function values to approximate the gradient; however, this does slow the algorithm slightly. If the standard assumptions from Newton's method are satisfied, then the BFGS method converges q-superlinearly.

1.3 Nelder-Mead

Nelder-Mead is a simplex algorithm that takes $N + 1$ points where N is the number of dimensions in the space. Each iteration, one of the vertices is replaced by a point with a lower function value. Given the set of points $[x_1, x_2, ...x_{N+1}]$

that satisfy $f(x_1) \leq f(x_2) \leq \ ... \leq f(x_{N+1})$ the Nelder-Mead algorithm is the following.

1. Calculate the mean of the first N points; $centroid = \frac{1}{N}\sum_{i=1}^{N} x_i$
2. Our first guess is the reflection of the worst point over the centroid; $x_{reflection} = centroid + \delta(centroid - x_{N+1})$ If delta is equal to one, this is a reflection, but it can be a different value. The function is evaluated at this point. If $f(x_1) \leq f(x_{reflection}) \leq f(x_N)$, replace x_{N+1} with $x_{reflection}$.
3. If $f(x_{reflection}) \leq f(x_1)$, then calculate $x_{expansion} = centroid + expansion(centroid - x_{reflection})$. This point is colinear with the reflected point, just extended out farther. If $f(x_{expansion}) \leq f(x_{reflection})$, replace x_{N+1} with $x_{expansion}$. Otherwise, replace x_{N+1} with $x_{reflection}$.
4. If $f(x_N) \leq f(x_{reflection}) \leq f(x_{N+1})$, compute the outer contraction point $x_{outer} = centroid - contraction(centroid - x_{reflection})$. If $f(x_{outer}) \leq f(x_{reflection})$, replace x_{N+1} with x_{outer}. Otherwise go to 6
5. If $f(x_{reflection}) \geq f(x_{N+1})$, compute the inner contraction point $x_{inner} = centroid - contraction(centroid - x_{N+1})$. If $f(x_{inner}) \leq f(x_{N+1})$, replace x_{N+1} with x_{inner}. Otherwise go to 6.
6. In this step, we restart the algorithm by shrinking the simplex. Replace all points except the closest point with $x_i = x_i - shrink(x_1 - x_i)$

The goal of Nelder-Mead is to find the minimum with the minimum number of function evalutaions. Each loop takes either one or two evaluations unless a shrink must be performed.

Unlike the other methods that have been discussed, there are no guarantees on Nelder-Mead's convergence for smooth functions. In situation where this occurs, the algorithm will stagnate at unoptimal points. However, if the gradient is lipshcitz, then we are guaranteed local convergence. In practice, Nelder-Mead will frequently converge to the optimal point. The following figures show the first four iterations of the algorithm. The function used has a minimum at $(0, 0)$.

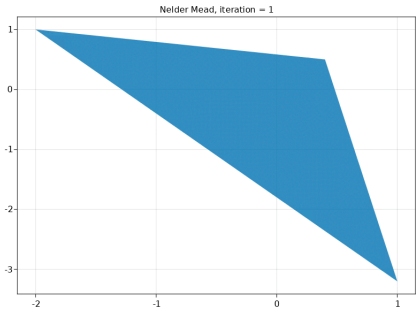

Figure 2: Intial Configuration for Nelder Mead

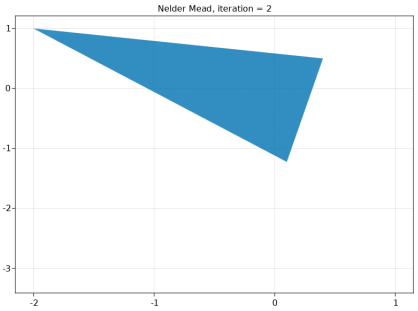

Figure 3: Second Time Step

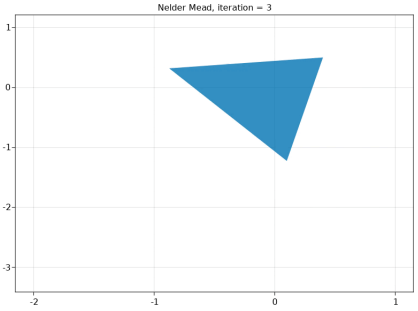

Figure 4: Third Time Step

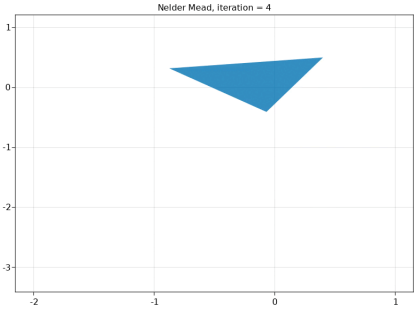

Figure 5: Fourth Time Step

1.4 Line Search

Line search is a generalized search method where $x_{k+1} = x_k + \alpha p_k$ where p_k is the search direction. If $p_k = -\nabla f(x)$, this method is called gradient descent as we simply move in the opposite direction of the gradient. If $p_k = -H^{-1}\nabla f(x)$, this is Newton's method. Scipy's line search method is implemented to satify the strong Wolfe conditions which are defined as following:

- $f(x_k + \alpha_k p_k) \leq +c_1 \alpha_k p_k^T(x_k)$
- $|p_k^T(x_k + \alpha_k p_k)| \leq c_2 |p_k^T \nabla f(x_k)|$

Line search satisfying the strong Wolfe conditions will generally show better global convergence than newton's method when the standard assumptions are not met.

2 Problem One: Simple Convex Function

For the first demonstration, we will use the simple convex function $f(x) = e^{x^T x}$. For this function, the gradient satisfies $\nabla f(x) = [2x_1 e^{x^T x}, ..., \ 2x_n e^{x^T x}]$. The terms on the diagonal of the hessian are $H_{ii} = (4x_i^2 + 2)e^{x^T x}$. The off-diagonal terms are $H_{ij} = 4x_i x_j e^{x^T x}$. In one dimension, this function is simply e^{x^2}, the derivative is $2xe^{x^2}$, and the second derivative is $4x^2 e^{x^2} + 2e^{x^2}$. For this example, we will compare BFGS with a gradient, BFGS with an approximate gradient, and Nelder-Mead.

2.1 BFGS with a defined gradient vs BFGS with a numerically estimated gradient

First, using our convex function, we will compare the the two methods after 10 iterations. To do so, I made 100 random samples and only kept the iterations where neither algorithm terminated before ten iterations (this yielded 68 observations). For each, a 3-vector was uniformly sampled from $(-2.5, 2.5)$ for each component. After running both for ten iterations, the distance from the minimum at $(0, 0)$ and the function value was recorded. The figure below shows the resulting dataframe.

	F_val	F_val_wout_grad	x_dist	x_dist_wout_grad	eval_wout_grad	eval_w_grad
0	1.025188	1.352354	0.157721	0.549406	24.0	52.0
1	1.134156	2.072584	0.354809	0.853696	24.0	56.0
2	5.368044	18.613654	1.296327	1.709940	32.0	64.0
3	1.040188	1.963441	0.198498	0.821400	24.0	52.0
4	1.005276	1.095002	0.072542	0.301258	24.0	52.0

Figure 6: Head of Dataframe

The next figure shows some summary statistics from the result

	F_val	F_val_wout_grad	x_dist	x_dist_wout_grad	eval_wout_grad	eval_w_grac
count	68.000000	68.000000	6.800000e+01	68.000000	68.000000	68.000000
mean	6.363699	27.152862	4.302272e-01	0.643978	25.558824	56.352941
std	24.997802	134.116108	5.625206e-01	0.687143	2.673152	11.030169
min	1.000000	1.000000	8.788446e-10	0.000139	24.000000	48.000000
25%	1.000013	1.000284	3.590497e-03	0.016859	24.000000	48.000000
50%	1.058189	1.270214	2.377139e-01	0.489058	24.000000	52.000000
75%	1.622127	2.904601	6.953543e-01	1.027600	26.000000	61.000000
max	167.140296	944.717702	2.262484e+00	2.617420	36.000000	96.000000

Figure 7: Summary of Dataframe

As can be seen, the gradient method outperformed the the non-gradient method in every metric. The total function error and distance from the minimum are lower on average in both cases. In fact, the approximate gradient method did not outperform the gradient method for any value of x.

2.2 BFGS without a defined gradient vs Nelder Mead

Using the same function, we will now compare BFGS and Nelder Mead with the same number of function evaluations. After sampling a random 3 vector, the BFGS method is run for 10 iterations. Then, nelder mead is run for the same number of function evaluations. The function value and distance from the optimal point is recorded. The following summary table was generated.

	F_val_BFGS	F_val_NM	x_dist_BFGS	x_dist_NM
count	100.000000	100.000000	1.000000e+02	100.000000
mean	18.783947	1.357748	4.379823e-01	0.441226
std	111.011009	0.431605	6.408041e-01	0.270261
min	1.000000	1.000004	7.976005e-09	0.001975
25%	1.000000	1.037403	1.253732e-04	0.191596
50%	1.000310	1.233946	1.758666e-02	0.458495
75%	1.841306	1.463807	7.811853e-01	0.617043
max	944.717702	3.221613	2.617420e+00	1.081611

Figure 8: Summary of Dataframe

It looks like BFGS is outperforming Nelder Mead, but the worst case scenarios are far worse for BFGS. This greatly influences the data as the function chosen is very steep and one poor run could ruin 99 good runs. The following plot shows the log function value at each run.

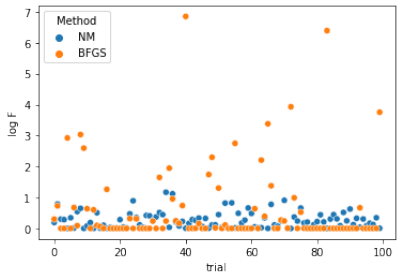

Figure 9: Log of function value vs Sample number with 10 iterations of BFGS

As it can be seen, there is a larger amount of variance in how BFGS performs. Now, we will repeat this comparison, but will iterate BFGS 20 times as opposed to 10.

	F_val_BFGS	F_val_NM	x_dist_BFGS	x_dist_NM
count	100.000000	100.000000	1.000000e+02	100.000000
mean	1.033249	1.117853	2.486635e-02	0.221595
std	0.238336	0.176396	1.384502e-01	0.228450
min	1.000000	1.000000	7.052519e-09	0.000040
25%	1.000000	1.000179	3.157768e-07	0.013249
50%	1.000000	1.011869	2.016529e-06	0.107515
75%	1.000000	1.209412	5.678977e-04	0.436025
max	3.057117	1.870666	1.057106e+00	0.791388

Figure 10: Summary of Runs

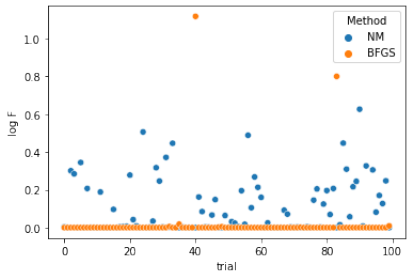

Figure 11: Log of function value vs Sample number with 20 iterations of BFGS

There are fewer outliers now, and the outliers are far closer to the Nelder Mead values.

3 Problem Two: Rosenbrock Function

For this demonstration, we will use the rosenbrock function $f(x) = (a - x^2) + b(y - x^2)^2$. This is a non-linear function with a minimum at (a, a^2), and a function evaluation of zero at that point. The traditional values of a = 1 and b = 100 will be used for the sake of this problem.

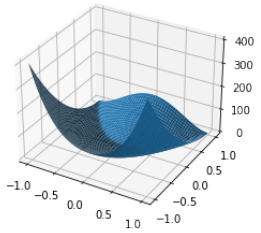

Figure 12: Rosenbrock Function

3.1 BFGS without a defined gradient vs Nelder Mead

Using the Rosenbrock function, we will now compare BFGS and Nelder Mead with the same number of function evaluations. After sampling a random 2 vector, the BFGS method is run for 50 iterations. Then, nelder mead is run for the same number of function evaluations. The function value and distance from the optimal point is recorded. The following summary table was generated.

	F_val_BFGS	F_val_NM	x_dist_BFGS	x_dist_NM
count	1.000000e+02	1.000000e+02	100.000000	100.000000
mean	1.856085e-03	2.171360e-02	1.396175	1.302812
std	9.997684e-03	5.323942e-02	0.078285	0.227258
min	4.000564e-13	1.054386e-10	0.948465	0.578726
25%	1.869540e-11	5.220811e-10	1.414204	1.346701
50%	2.005458e-11	3.380915e-08	1.414204	1.414185
75%	2.242717e-11	3.878825e-03	1.414205	1.414247
max	6.422597e-02	2.615587e-01	1.415914	1.701979

Figure 13: Summary of Dataframe

It looks like BFGS is outperforming Nelder Mead in most all situations. BFGS is superior in 83 of the 100 runs conducted with better worst case scenarios. The following plot shows the log function value at each run.

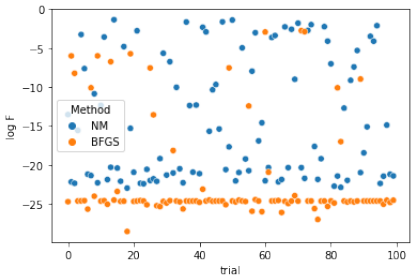

Figure 14: Log of function value vs Sample number with 50 iterations of BFGS

4 Citations

- Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu

Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E.A. Quintero, Charles R Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods, 17(3), 261-272.

- Iterative Methods For Optimization. C.T. Kelley