Optimzational Methods in Computational Physics

Trevor Taylor

December 2018

Abstract

Methods of optimzation are an vital part of physics. This paper will compare three different optimzation methods, Gradient Descent, Congugate Gradient, and BFGS Quasi Newton Methods

1 Background

Minimization is the attempt to find out where a function has the minimum value and is very important in the field of physics, since nature itself attempts to minimize many things. In fact, the majority of Langrangian dynamics works because it is the minimization of the diffrence between the potential and kinetic energies. In addition, all particles will decent to a a state of lowest potential in put in a field, which applies to the fields of thermodynamics, classical dynamics, astronomy, and many more.

In order to minimize a function we need to find where

$\frac{df(x)}{dx} = 0$

This may be difficult at times and computational methods can provide numerical answers when it is impossible to find an analytical results. Three methods will be discussed in this paper Gradient Descent, Conjugate Gradient, and BFGS Quasi-Newton

2 Methods

The gradient decent method is one of the simplest methods to find a minimum. It involves an initial guess and then the computation of the gradient at the intial point. Then the program moves the old point to a new position in the direction of the gradient and for some length. The equation for gradient decent becomes

$x_{n+1} = x_n + a\nabla f(x)$

The parameter a is called the step length and can be computed in many different ways. One of the most common methods is called a line search. A line search is a way to find how far to go in order to reach the minimum in a certain direction. It is an iteritive funciton of the form

$x_{n+1} = x_n + a_n p_n$

It will either find the exact step length to find the minimum or it will get close to the minimum in methods such as the conjugate gradient where exactness is not as necessary.

Another method to compute the step length is to use the Barzilai-Borwein method which depends on the current

position and gradient and the previous position and gradient to get a new step length at each iteartion

$a = \frac{(x_n - x_{n-1})^T(\nabla f(x_n) - \nabla f(x_{n-1})}{||\nabla f(x_n) - \nabla f(x_{n-1}||^2}$

The next method used is the Conjugate Gradient method. This makes use of conjugate directions to solve equations in fewer steps. The first step of this method is to find a residual. This is how far away the function is away from the minimum

$r = \frac{df(x_n}{dx}$

After computing this residual you move in the direction of the residual by a step length determined by a line search. The position and residual can now be updated.

Next a new direction of travel must be defined. This direction ends up being conjugate to the previous direction which improves the efficency of the program dramatically.

$p_{n+1} = r_{n+1} - \frac{r_{n+1}^T r_{n+1}}{r_n^T r_n} p_n$

The last method is the BFGS Quasi Newton Method. Newton method normally require a Hessian matrix which is a matrix of all of the all of the partial derivatives. If the function being minimized has a significant amount of dimentions then computing the Hessian matrix can be computationally expensive. To prevent this an approximate Hessian is used and whenever an approximate Hessian is used the method becomes a Quasi-Newton method.

The first step of any Quasi Newton method is to have a starting Hessian matrix as well as an intial starting point. Often times the initial Hessian Matrix is the identity matrix if no knowledge of the Hessian Matrix is known. The next step in the Quasi-Newton method is to compute a search direciton based on the inverse Hessian and the gradient.

$p_n = -B_n^{-1} \nabla f(x_n)$

Next a simple line search is performed to get the step size required to minimize the potential in that direction

$x_{n+1} = x_n + a_n p_n$

The next goal is to update the Hessian Matrix. This is the main part where different Quasi-Newton methods differ. If we update using the BFGS method we need

$s_n = x_{n+1} - x_n$

$y_n = \nabla f(x_n) - \nabla f(x_{n+1})$

To use in our updating of the Hessian Matrix

$B_{n+1}^{-1} = (I - \frac{s_n y_n^T}{s_n^T y_n}) B_n (I - \frac{y_n s_n^T}{y_n^T s_n}) + \frac{s_n s_n^T}{y_n^T y_n}$

Since only the inverse Hessian Matrix is needed to calculate a better aproimation of the minimum it is often updated instead of having the Hessian matrix updated and then finding its inverse.

3 Experiment

Each of the three methods will be tested with a variety of functions and their speeds, accuracy, and amount of iterations will be compared against each other. The Gradient Decent code was written by me and can be found on the my github page at https://github.com/Taylot6. In order to eliminate any inconsitancies in code the Conjugate Gradient method and the BFGS method will be gotten from the function scipy.optimize.minimize. The first thing to test is the overall accuracy of the three functions with a very simple function. The function used is $x^2 + \frac{y^2}{4}$ The results for each three method is as follows

From this we can see that the Gradient Descent method does not perform

Table 1: Results of Three methods

method	Minimum f	x	y	iterations
GradDes	-0.33	0.66	0.16	10
CG	2.51e-17	-5.01e-09	-7.11e-09	4
BFGS	6.17e-17	-1.11e-08	-5.71e-10	4

nearly as well as the other methods do. Therefore the rest of this paper will compare the diffrences in Conjugate Gradient and Quasi Newton Methods.

The next thing to test is to see how accurate each method is and how many iterations it takes for each to converge. The tolerance of each will be one to promote fast convergance and as it will show the ability of each function to converge quickly.

The first function used will be the Beale Function since it will be a good challenge for each of the methods. The results for a tolerance of one are.

Table 2: Results for tol=1 Beale Function

Method	Min f	x	y	it
BFGS	2.79e-07	2.9	0.49	17
CG	2.86e-08	3.0	0.50	12

There are more examples of data on my github as well

The next thing to test is the total speed of each algorhythm to converge based on a tolerance of one and find the average of each loop. The functions ran were the Baele and Himmelblau.

Table 3: Time values

Method	Function	time
BFGS	Himmel	.0011
BFGS	Beale	.0051
CG	Himmel	.0011
CG	Beale	.0047

4 Discussion

From all of these tests we see a few things. One important thing is that in functions have low amounts of dimentions and for functions that are not too complex that there is little diffrence between BFGS and Congugate Gradient methods. One thing to note is that Gradient Descent method is not nearly as efficent in the catagories where it was tested, however under different conditions it might have more of a purpose.

5 Refrences

œBroydenFletcherGoldfarbShannoAlgorithm. Wikipedia,WikimediaFoundation,25Nov. 2018,en.wikipedia.org/wiki/Broyden%E2%80%93Fletcher%E2%80%93Goldfarb%E2%80%93Shanno_algorithm.

œQuasi-NewtonMethod.Wikipedia, WikimediaFoundation,11Sept.2018, en.wikipedia.org/wiki/Quasi-Newton_method.

https://github.com/scipy/scipy/blob/master/scipy/optimize/optimize.py#L1010

œTheConceptofConjugateGradientDescentinPython. IlyaKuzovkin,ikuz.eu/2015/04/15/the-concept-of-conjugate-gradient-descent-in-python/.