Unconstrained Vector Optimization

Abstract

Purpose: The purpose of this assignment is to find stationary point of multivariable functions, using various optimization methods, and then to compare and evaluate the effectiveness of each

Methods: There we're three optimization methods performed, a fixed step iteration method, an Armijo backtracking method, and damped Newtons method. All three optimization methods were performed by modifying code provided by the course instructor. The accuracy to the true stationary points we're compared to the estimates found, and the iterations performed we're also evaluated.

Results: The fixed step size and the Armijo backtracking both converged on all three functions with the Armijo taking less steps on average. The Damped Newtons method with Armijo backtracking converged in less iterations on two functions but failed to converge on function one.

Conclusion: All three methods performed reasonably well, with 8/9 attempts successfully converging to a stationary point. The Damped newtons method with Armijo back tracking converged in significantly less steps than the other two methods, but failed to converge on the first function.

Introduction

The scientific question tested in this assignment is the effectiveness of three methods at finding the stationary points of a multivariable equations. A stationary point is a point of a graph we're the derivative is equal to zero. Stationary points can have three forms, concave down (or a maximizer) is when the stationary point is located at the top of a parabola, concave up (or a minimizer) is when the stationary point is located at the bottom of a parabola, and a saddle point where the stationary point is located at neither the top or bottom of a parabola. The three methods used we're a fixed step iteration method, an Arminio backtracking method, and damped Newtons method. Each method was tested on three equations, each equation having the same initial starting point [-1.2,1]. The equations are:

Function	**Expression**
$f_1(\vec{w})$	$((w_1 + 1.21) - 2(w_2 - 1))^4 + 64(w_1 + 1.21)(w_2 - 1)$
$f_2(\vec{w})$	$2w_2^3 - 6w_2^2 + 3w_1^2 w_2$
$f_3(\vec{w})$	$100(w_2 - w_1^2)^2 + (1 - w_1)^2$

The scientific question of how effective a fixed step iteration method, an Arminio backtracking method, and Damped Newtons method with Armijo backtracking was answered by performing the methods on three given functions, and comparing the accuracy and the number iterations taken to converge of each of the three methods.

Methods

This assignment consisted of two parts. The first part of this assignment consisted of creating Matlab code to find and classify stationary point of the three functions given. The second of which has three subtasks within it.

Classification of Stationary Points:

To complete this you must compute the partial derivatives of the multivariable function, and solve for when they equal zero, as this will be a stationary point because the first derivative represents the magnitude of the gradient, and when this is equal to zero the function must be flat at that point. This is done using the grad function in Matlab to find the first derivative, and the solve function to solve for when it is equal to zero. Once you have a list of the stationary point, you can create the hessian matrix by taking the partial derivative of the partial derivatives you just computed. Once the hessian matrix is found you can classify the stationary points of the function by analyzing the eigen values of the hessian matrix at the current stationary point. Since the eigen values represent the curvature of the graph, we can infer that if they are all positive the graph must be curving upwards, making the point a local minimum, and if they are all negative the graph must be curving downwards, meaning it must be a local maximum, and if neither are true then it must be a saddle point.

*I wasn't sure if this was to be included in the methods section, the assignment says not to provide any code but does not mention the report, so I assumed it was to be included.

The second part of this assignment was to modify the starter code provided by the course instructor to implement three optimization techniques.

Fixed Step Size Iteration:

The fixed-size step iteration method works by having a starting point and creating an estimate of the local minimum by adding the current value, with the current gradient multiplied by the fixed step size. In this assignment the fixed step size was 0.001. By inserting the current estimate of the local minimum into the derivative of the respective function we can get the current gradient value of that function. If the absolute value given back falls within our tolerance range, we can then use the current value as our estimate as the gradient is very close to zero, meaning our function has gotten to its local minimum. The tolerance range used was 10^{-6}.

Armijo Backtracking/ Damped Newtons Method with Armijo Backtracking:

Damped newtons method mixed with Armijo backtracking is an iterative optimization method that aims to minimize the number of iterations performed while finding stationary points. The function works very similarly to Armijo backtracking with a different method for finding the direction vector. The first step is to initialize a step size, a starting point, an initial estimate, and a direction vector. In our implementation the step size was (fill this in later), the starting point was [-1.2,1], and the initial estimate is found by evaluating the function at the starting point. In Armijo backtracking without newtons method the directional vector is the negative of the transpose of the function evaluation of the first partial derivatives at the initial estimate. With Newtons method the direction vector is found by solving a linear equation of the function evaluation of the hessian matrix at the initial estimate, and the transpose of the function evaluation of the first partial derivatives at the initial estimate. Once these are initialized, we enter a while loop that will only break once our current gradient is less than our tolerance, in this implementation we had a tolerance of 10e-6. When the gradient is less than our tolerance it is considered stationary because that point on the graph is considered flat. While in each iteration of the while we first perform the Armijo's steps. This consists of creating a current estimate by evaluating the function at the previous estimate plus the direction vector multiplied by the alpha value, our alpha value was 0.5 in this implementation. The Armijo inequality is:

$$f(t_k + \beta^\gamma s_0 dt_k) \leq f(t_k) + \alpha_k \beta^\gamma s_0 d_k$$

or in simpler terms given the current context of this report:

currentEstimate >= previousEstimate+α*currentGradient*stepSize*(direction vector)

This inequality is calculated after using the current estimate we just calculated, and if it fails it's an indication that the current estimate has overshot the minimizer and we need to adjust our step size to go the opposite direction, this is performed by entering a second while loop. Inside this interior while loop the step size is multiplied by beta, and the current estimate is recalculated using this new stepsize, in our implementation the beta value was 0.5. The inequality is then re-evaluated, and the iterations continue until the inequality returns false. Once the inequality returns false new estimates are created for the exterior while loop. The first estimate is the current estimate of the stationary point, this is done by adding the new stepsize (if while loop was entered) multiplied by the direction vector. The function and the hessian matrix is then evaluated at this point. These are then used to set up newtons method for finding the directional vector. Using the values returned from evaluating the hessian matrix at the current estimate, and the transpose of the values returned from evaluating the first partial

derivatives at the current estimate, we can set up and solve a linear equation which will give us our new direction vector, in Armijo backtracking without Damped Newtons method the direction vector is the negative of the transpose of the function evaluation of the first partial derrivatives at the current estimate. The function is then incremented and enters a new iteration if necessary.

Results

Table 1: Numerical values of stationary points, numerical values of Hessian matrices, signs of the eigenvalues, and the kinds of stationary points. Each stationary point is characterized as a local minimizer (min), a local maximizer (max), a saddle point (sdl), or as inconclusive using derivatives (inc). Values are empty for entries that do not have a distinct stationary point.

Stationary Info.	**Function**		
	f_1	f_2	f_3
$\vec{w}_1$	$\begin{bmatrix} -0.21 \\ 0.5 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 0 \end{bmatrix}$	$\begin{bmatrix} 1 \\ 1 \end{bmatrix}$
$\nabla^2 f(\vec{w}_1)$	$\begin{bmatrix} 48 & -32 \\ -32 & 192 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ 0 & -12 \end{bmatrix}$	$\begin{bmatrix} 802 & -400 \\ -400 & 200 \end{bmatrix}$
At $\vec{w}_1$, λ_1 & λ_2:	$\lambda_1 > 0,\ \lambda_2 > 0$	$\lambda_1 < 0,\ \lambda_2 = 0$	$\lambda_1 > 0,\ \lambda_2 > 0$
At $\vec{w}_1$, kind is:	min	inc	min
$\vec{w}_2$	$\begin{bmatrix} -1.21 \\ 1 \end{bmatrix}$	$\begin{bmatrix} 0 \\ 2 \end{bmatrix}$	$\begin{bmatrix} \sim \\ \sim \end{bmatrix}$
$\nabla^2 f(\vec{w}_2)$	$\begin{bmatrix} 0 & 64 \\ 64 & 0 \end{bmatrix}$	$\begin{bmatrix} 12 & 0 \\ 0 & 12 \end{bmatrix}$	$\begin{bmatrix} \sim & \sim \\ \sim & \sim \end{bmatrix}$
At $\vec{w}_2$, λ_1 & λ_2:	$\lambda_1 < 0,\ \lambda_2 > 0$	$\lambda_1 > 0,\ \lambda_2 > 0$	$\lambda_1 \,?\, 0,\ \lambda_2 \,?\, 0$
At $\vec{w}_2$, kind is:	sdl	min	~

$\vec{w}_3$	$\begin{bmatrix} -2.21 \\ 1.5 \end{bmatrix}$	$\begin{bmatrix} \sim \\ \sim \end{bmatrix}$	$\begin{bmatrix} \sim \\ \sim \end{bmatrix}$
$\nabla^2 f(\vec{w}_3)$	$\begin{bmatrix} 48 & -32 \\ -32 & 192 \end{bmatrix}$	$\begin{bmatrix} \sim & \sim \\ \sim & \sim \end{bmatrix}$	$\begin{bmatrix} \sim & \sim \\ \sim & \sim \end{bmatrix}$
At $\vec{w}_3$, λ_1 & λ_2:	$\lambda_1 > 0,\ \lambda_2 > 0$	$\lambda_1 ? 0,\ \lambda_2 ? 0$	$\lambda_1 ? 0,\ \lambda_2 ? 0$
At $\vec{w}_3$, kind is:	min	~	~

Table 2: Numerical results for unconstrained optimization

METHODS/FUNCTIONS	F_1		F_2		F_3	
	ITERS	$\vec{w}$	ITERS	$\vec{w}$	ITERS	$\vec{w}$
FIXED	482	[-0.21 1]	1486	[0 2]	32076	[1 1]
BACKTRACKING	36	[-0.21 1]	15	[0 2]	10704	[1 1]
DAMPED NEWTONS	200.	[-0.76 0.55]	8	[0 2]	20	[1 1]

8.3333e-04	-0.0017
63.9983	63.9967

Figure 1: Example output of ill-conditioned Hessian matrix for f1 using newtons method

Discussion

The fixed step size iterations was tied for the most accurate results in comparison to the true minimizers, while also performing the highest number of steps on average. This is expected as the

method is predicated on taking small steps down the gradient, meaning there's a small chance that you overshoot your minimizer (if step size is chosen correctly), but also it will take more iterations as the step size is small. The Armijo backtracking also had all three functions correctly converge, with on average a lower number of iterations in comparison to the fixed step sized iterations. The third function took significantly more steps to converge in comparison to the first two for both the fixed step size iterations, and the Armijo backtracking. From examining the graph of this function, we can see there is very steep and narrow value where our minimizer is located. This could have cause oscillating behavior in our Armijo backtracking, caused by the over correction in the backtracking step, as the gradient vector is used in the calculation of the Armijo inequality (on the less than side), causing the interior while loop to be exited quicker, leading to a larger step value and a overall larger backtracking estimate.

Reminder of the Armijo inequality:

currentEstimate >= previousEstimate+α*currentGradient*stepSize*(direction vector)

The larger number of steps taken by the fixed step sized iterations could be due to the actual distance (-1.21 -> 1) traveled in the x axis was larger than in the other functions. The Damped Newtons method with Armijo backtracking performed well on two functions, converging faster than the two previous methods, but did not converge on the first function. Increasing the maximum number of iterations did not seem to cause the function to converge, as remained at the same values as from lower. Stepping through the function we can see that there are many ill-conditioned Hessian matrices. In an ill-conditioned Hessian matrix, there is a significant difference in magnitude between the largest and smallest eigenvalues. This means that some directions in the optimization space are much steeper or more curved than others. The presence of very small eigenvalues implies that some directions have very shallow curvature, leading to small step sizes. As a result, the algorithm may take very small steps in these directions, causing slow convergence. It could also be cause by miss implementation of the optimization method, though the other two functions converged properly, and all three functions converged on the other two methods, meaning the only thing that could be causing this is mis-inputting the hessian matrix for the first function as it is only used in the damped newtons method. (I've double checked this, it probably is the case as usually when we are taught new techniques, they're more effective than previous ones, but I can't see the fault currently)

It should be taken into account that although there were less steps taken by the Armijo, and the damped newtons with Armijo backtracking the actual number of computation steps performed is skewed. When counting the number of iterations performed, we only keep track of the exterior while loop (if applicable) while Armijo's and damped newtons have interior while loops as well. While overall these methods tend to converge to a stationary point in less total computational steps, the results could be misleading in how much more efficient these methods are.

The initial stepsize chosen plays a large role in the performance of the optimization methods. In fixed step size if the initial step size is too large, you may overshoot your stationary point causing the while loops tolerance condition to never fail, and the point never to be found. In the Armijo, and the damped newtons with Armijo backtracking the initial step size also plays a large role. If chosen too small the maximum number of iterations may be surpassed and the stationary point may never be reached (this is also true for fixed step size). If chosen too large you could cause oscillation in your estimate, where the backtracking somewhat continually overcorrects itself causing the estimate to oscillate between both ends of the parabola, and if very large it will return incorrect estimates after performing the maximum

number of iterations. There's a very large advantage to these methods with built-in backtracking as it allows you to have a much larger initial step size. This is because the backtracking acts as a safety net to the method, where if you overshoot, you won't keep stepping the wrong direction, but rather be reversed back towards the stationary point. With a larger initial stepsize, you can generally converge faster as you're progressing through the gradient faster. Damped Newtons method allowed for a much larger initial step size than Armijo backtracking without Newtons method implemented. This is because Newton's method uses the second-order information (Hessian matrix) to make more informed decisions about the step size. When the Hessian is well-conditioned and provides useful curvature information, Newton's method can exploit this to take larger steps, potentially leading to faster convergence. By experimenting with the initial step size on Damped Newtons method the best choice in terms of minimizing iterations, seemed to lie around 4 or 5. But it's important to note that this is very sensitive with small changes quickly causing the method to return incorrect results.

Real world applications of optimization techniques come in many forms, many coming in the machine learning field but can very as many fields utilize optimization (structural engineering for example). Armijo backtracking/Damped Newtons with Armijo backtracking typically converges faster than fixed step and has a reduced risk of divergence, but fixed step is much simpler to implement and typically less computationally intensive. The importance of what optimization method you choose (or if you choose one) often depends on the task at hand. Certain models choose not to implement nonlinear optimization methods, whether that's due to computational concerns, assumptions of linearity, or other reasons, but within the context of this assignment, it's very rare that you'd use a linear optimization technique on three-dimensional (or higher) data in a real-world dataset as many are too complex. The learning rate or the gradient in our case also can play a role in the selection of whether to include backtracking, and if so which backtracking method to utilize.