Solving quadratic unconstrained optimization problems using Eigenvalues and Eigenvectors

Anjali Chourdia
New York University

April 4, 2022

Abstract

This paper discusses three existing methods to solve unconstrained optimization problems, namely steepest descent, Newton method and conjugate-gradient method. Further, I explain my approach to solve unconstrained optimization problems by utilizing eigenvectors and eigenvalues with the line-search method.

1 Introduction

Unconstrained optimization methods refers to the task of finding a local minimum point in the absence of any constraint conditions. The unconstrained optimization problems can be generally represented as:

$$minf(x), x \in R^n \tag{1}$$

where R^n is an n dimensional Euclidean space and f is a continuously differentiable real valued function. Vanilla line search methods for solving (1) are of the form:

$$x_{k+1} = x_k + \alpha p \tag{2}$$

where x_{k+1} is the k+1 th iterate generated by taking a positive step α in the descent direction p. The line search direction p is usually required to satisfy the descent condition:

$$g^T p < 0 \tag{3}$$

to ensure that p is a descent direction for the function at x_k.
In this paper, we will focus on the unconstrained optimization of quadratic functions:

$$f(x) = \frac{1}{2} x^T H x + c^T x \tag{4}$$

where Hessian H is the Hessian of the quadratic and c is an n dimensional vector.
For function (4), the value of f(x+α p) is given by:

$$f(x_k + \alpha p) = f(x_k) + \alpha g_k^T p + \frac{1}{2}\alpha^2 p^T H p \tag{5}$$

where g_k is the gradient of f at x=x_k, given by:

$$g_k = Hx + c \tag{6}$$

From (5), we can see that to reduce the value of function as we move from k to k+1 th iterate, we need to solve this minimization problem:

$$min \alpha g_k^T p + \frac{1}{2}\alpha^2 p^T H p, \alpha \in R \tag{7}$$

Solving (7), by setting the derivative of the equation to be zero, we get:

$$\alpha = \frac{-g_k^T p}{p^T H p} \tag{8}$$

Iterative methods of minimizing f in which this choice of α (8) is used are known as the exact-line search methods.
In the subsequent sections, we are going to explore the search techniques used by Steepest Descent Method, Newton's method and Conjugate Gradient method, following which I will discuss my approach to solve the unconstrained optimisation problems using exact-line search and eigenvectors.
As a side note, local quadratic models have also found to be useful to minimize other non-linear functions, therefore it's worthwhile to spend sometime to analyze the existing unconstrained optimization techniques for quadratic functions.

2 Unconstrained optimization: Existing methods

2.1 Steepest Descent

Steepest Descent method[1] is the simplest gradient method for unconstrained optimization (1) and was proposed by Cauchy in 1847. The basic idea of the steepest descent method is to move in the best possible descent direction according to the local available information, in an effort to minimize f as much as possible.
The update at each step is of the form:

$$x_{k+1} = x_k + \alpha_k(-g_k) \tag{9}$$

where $g_k = g(x_k) = \bigtriangledown f(x_k)$ and α_k can be obtained by using some line search condition such as the Wolfe condition, Goldstein conditions, or exact-line search condition(8).

If exact-line search is used, then step size is given by:

$$\alpha = -\frac{(g^T)(-g)}{(-g^T)H(-g)} = \frac{(g^T)(g)}{(g^T)H(g)} \tag{10}$$

Although the steepest descent method always converges, they are practically inefficient and they have a poor performance for ill-conditioned matrices[2]. As k -> ∞, the steepest descent iterates satisfy:

$$\frac{||x_{k+1} - x^*||}{||x_k - x^*||} \leq \frac{cond(H) - 1}{cond(H) + 1} \tag{11}$$

2.2 Newton Method

Newton's method[9] utilizes the Taylor Series local quadratic model of the function to be minimized at every iterate x=x_k:

$$F(x) = f(x_k) + g_k^T(x - x_k) + \frac{1}{2}(x - x_k)^T H(x_k)(x - x_k) \tag{12}$$

where $g_k = \bigtriangledown f(x_k)$ is the gradient of f and H is the hessian of f and $|x - x_k| < \delta$ for sufficiently small δ.
Solving for $\bigtriangledown F(x) = 0$, we get:

$$x - \bar{x} = p = -H(x_k)^{-1} g_k \tag{13}$$

(13) defines the Newton step at every iterate. As can be seen from the expression of Newton's search direction expression, Newton's method assumes that H is a non-singular matrix

In our case, where the function itself is quadratic, (12) is accurate for every point x on the function and Newton search direction expression for our quadratic function looks like:

$$p = -H^{-1} g_k \tag{14}$$

where H is the hessian of quadratic function and $g_k = Hx_k + c$ (6)

The step size to be taken in the above obtained descent direction p can be chosen to be some constant(1 for pure Newton search) or can be obtained by using Armijo sufficient decrease and backtracking conditions.

Newton's method continues to run until it hits a stationary point($g_k = 0$), as can be seen from its choice of search direction and it can be proved that it always converges. Newton's method can be divided into two phases[8] - damped Newton phase ($||g_k||_2 \geq \eta$), where most iterations require backtracking steps and the quadratically convergent phase ($||g_k||_2 < \eta$) where all iterations use step size 1 and gradient converges to zero quadratically. Even though Newton's method might take a few backtracking steps, it can be proved that there is a finite bound

on the number of iterations which it might take such backtracking steps in the damped phase.

Generally speaking, Newton's method performs better than Steepest Gradient Descent, and converges quadratically.

2.3 Conjugate-Gradient Method

The conjugate-gradient method is a powerful line-search method which obtains the search direction by conjugating the negative gradients of the function. The search directions of the conjugate-gradient thus have the form:

$p = -g_k + \beta_k p_{k-1}$, if k$\geq$1 and $p = -g_k$ if k=0

where $\beta_k \in R$. The best known choices for β are given by the Fletcher-Reeves[3] (FR), Polak-Ribire(PR), and Hestenes Stiefel(HS) formulas.

$$\beta_k^{FR} = \frac{||g_k||^2}{||g_{k-1}||^2} \tag{15}$$

$$\beta_k^{PR} = \frac{< g_k, g_{k-1} >}{||g_{k-1}||^2} \tag{16}$$

$$\beta_k^{HS} = \frac{< g_k, g_{k-1} >}{< p_{k-1}, g_{k-1} >} \tag{17}$$

The final update at each iteration is of the form:

$$x_{k+1} = x_k + \alpha * p \tag{18}$$

where stepsize α can be obtained by one dimensional line search:

$$\alpha = \frac{-g_k^T p}{p^T H p} \tag{19}$$

The numerical performance of the Fletcher-Reeves method is sometimes as efficient as the Polak-Ribire and Hestenes Stiefel methods, but it is often much slower[5].

The classical conjugate-gradient method converges to the minimizer linearly. In fact, the exact number of iterations required for the convergence of CG-method can be derived by Krylov subspace.

2.4 Expanding Subspace Method

Expanding subspace method is another linearly converging unconstrained optimisation algorithm which takes advantage of the fundamental property that any n dimensional vector

$\in R^n$, including the minimizer can be expressed uniquely as a linear combination of the basis vectors. At each iteration k, x_{k+1} is calculated by a linear combination of x_k, k linearly basis vectors(each added one after the other in preceeding iterations) and a new linearly independent vector added from the basis set to the set S in this iteration.

3 My approach

My approach to solve the unconstrained optimisation is inspired by the expanding subspace method and the fundamental properties of eigenvector and eigenvalues. Any set of N eigenvectors corresponding to N distinct eigenvalues of a matrix for a basis set. Thus, any vector x, including the minimizer x^* can be represented by a linear combination of N linearly independent eigenvectors. I propose to take a step in the direction of one of the eigenvectors of the Hessian(H) of the quadratic in every iteration. At every iteration, $g^T p$ is computed by taking p as one of the eigenvectors. If $g^T p > 0$, then a step is taken along the negative of the eigenvector e with eigenvalue λ is taken to ensure that chosen p is a descent direction. Either way, we have:

$$Hp = \lambda p \tag{20}$$

The stepsize is calculated using the exact-line search method:

$$\begin{aligned} \alpha &= \frac{-g_k^T p}{p^T H p} \\ &= \frac{-g_k^T p}{\lambda p^T p} \end{aligned}$$

x_{k+1} at any iteration k is thus equal to:

$$x_{k+1} = x_k + \frac{-g_k^T p}{\lambda p^T p} p \tag{21}$$

The gradient update at each step is of the form:

$$g_{k+1} = g_k + \lambda p \tag{22}$$

3.1 Experiment and Results

I used this eigenvalue approach on 15 quadratic functions, each with a randomly generated Hessian(4x4 or 5x5) with non-zero eigenvalues in the range(-15,150),c initialized as an n-dimensional vector of all 1s and x initialized as an n dimensional vector with all zeros. The program was allowed to run for a maximum of 50 iterations or until gradient value becomes lower than 10^-7, whichever happens before. I also ran conjugate-gradient method on all these quadratic functions as a baseline to gauge the performance of my algorithm. The results of the experiment can be found in the table below.

The norm value of the gradient consistently decreased at every iterate for every problem using the eigenvalue approach. This is expected by the property of line-search step size taken along a descent direction $g^T p < 0$. But what was surprising was that the method converged in n iterations for some quadratics, including on the benchmark matrix identified by Schnabel and Eskow[7] that gives SE90 difficulties, with a large drop in the norm of the gradient value (3 to 8 powers of 10) in the last iteration. For other quadratics, the method performs fairly well as opposed to the conjugate gradient method in which the gradient exploded in some of the cases due to the extremely small value of stepsize. This can be justified by Powell's argument[6] that, under some circumstances, the Fletcher-Reeves method with exact line searches produces produce very small displacements, and normally does not recover unless a restart along the gradient direction is performed.

This problem can also be partially observed in the eigenvalue method as the method is seen to converge faster when the chosen step sizes are in the range$(10^-1 - 10^-2)$ but the method converges slowly when step sizes are of lower order. Irregardless when I ran the program for long enough(300-500 iterations), the method converged to theminimizer x^*.

-	**Eigenvalue method**			**Conjugate Gradient method**		
Eigenvalues	iter	$\|\|g\|\|$	$\|\|\alpha\|\|$	iter	$\|\|g\|\|$	$\|\|\alpha\|\|$
-1.3,0.5,2,4,5	50	$8.75 * 10^-2$	10^-1-10^-2	50	3.34	10^-1-10^-15
0.18,0.71,3.11,14.99	4	$3.3 * 10^-16$	10^-1-10^-2	$9.3 * 10^-14$	4	10^0-10^-2
-0.3,-0.4,-104.6,824.2	4	$5.4 * 10^-12$	10^0-10^-5	5	$1.9 * 10^-9$	10^0-10^-4
-1,2,31,40	43	$3.4 * 10^-8$	47	NaN	10^-1-10^-17	10^-2-10^-17
70,46,23,65,79	50	$2.1 * 10^-2$	10^-2-10^-4	13	$7.5 * 10^-8$	10^-2
-13,27,70,96,79	50	$5.1 * 10^-2$	10^-2	21	NaN	10^-16
23,24,98,41,71	50	$1.4 * 10^-1$	10^-2-10^-4	13	$6.3 * 10^-8$	10^-2
-3,7,46,60,46	50	$4.8 * 10^-1$	10^-2-10^-3	50	1.8	10^-2-10^-4
82,14,38,35,29	50	$3.9 * 10^-1$	10^-3-10^-1	16	$7.1 * 10^-8$	10^-2
1,1,1,1,1	50	$4.74 * 10^-4$	10^-1-10^-4	1	$3.5 * 10^-16$	1
28,1,51,75,35	50	$2.5 * 10^-2$	10^-2-10^-4	50	$9.6 * 10^-1$	10^-1-10^-16
-1,3,5,9,15	50	$1.6 * 10^-1$	10^-1-10^-4	42	NaN	10^-1-10^-17
-10,3,10,21,37	50	$1.4 * 10^-1$	10^-1-10^-4	37	NaN	10^-1-10^-17
-1,1,10,100,150	50	$4.1 * 10^-4$	10^-1-10^-4	18	NaN	10^-2-10^-15
25,76,14,79,97	50	$9.4 * 10^-2$	10^-1-10^-4	22	$6.2 * 10^-8$	10^-2

4 Summary

It is interesting to see that eigenvectors can be leveraged to solve the unconstrained optimisation problems. They also seem to provide a computational benefit as the computations required for each update are reduced due to the special properties of eigenvectors. The next steps would be to find a bound on the number of iterations this method would take at most before it converging to the minimizer x^*.

References

(1) A. Cauchy. Methodes generales pour la resolution des systemes dequations simultanees,. C.R. Acad. Sci. Par., 25:536538, 1847.

(2) J. C. Meza, Steepest descent, in: Wiley Interdisciplinay Revies: Computational Statistics, Vol. 2, 2010, pp. 719722.

(3) R. FLETCHER AND C. M. REEVES, Function minimization by conjugate gradients, Comput. J., 7 (1964), pp. 149-154.

(4) J. J. MoPI, B. S. GAIBOW, AND g. E. HILLSTROM, Testing unconstrained optimization software, ACM Trans. Math. Software, 7 (1981), pp. 17-41.

(5) Jean Charles Gilbert and Jorge Nocedal, Global Convergence Properties of Conjugate Gradient Methods for Optimization, SIAM Journal on Optimization 2 (1992), no. 1, 2142.

(6) Restart procedures for the conjugate gradient method, Math. Programming, 12 (1977), pp. 241-254. (7) Schnabel, R.B., Eskow, E.: A revised modified Cholesky factorization algorithm. SIAM J. Optim. 9(4), 11351148 (1999) (8) Ryan Tibshirani, "Newton's Method", http://www.stat.cmu.edu/ ryantibs/convexopt-S15/lectures/14-newton.pdf (9) D. P. Bertsekas, Constrained Optimization and Lagrange Multiplier Methods. Academic Press, 1982.