Optimization: Méthode de Gradient Conjugué

Sherly Sherly

EIT Digital Masters, May 2019

1 Introduction

The goal of this project is to program, validate and experiment with the conjugate gradient algorithm. There will be two variants of the conjugate gradient that will be experimented in this project 1) Conjugate Gradient Algorithm 2) Non Linear Conjugate Gradient (NLCG), Fletcher-Reeves with constant step. Different cost functions will be implemented for comparison.

2 Background

2.1 Cost Functions

2.1.1 $J_1(v)$

The cost function $J_1(v)$ is defined as:

$$J_1(v) = \sum_{i=1}^{N}(v_i - 1)^2$$

The gradient $\bigtriangledown J_1(v)$ is given by:

$$\bigtriangledown J_1(v) = 2(v - 1)$$

where v is a vector.

In order to utilize the Conjugate Gradient Algorithm, we formulate the cost function to the form of:

$$J(v) = \frac{1}{2}(Av, v) - (f, v)$$

We propose $A = 2I$ where I is the identity matrix.

2.1.2 $J_2(v)$

The cost function $J_2(v)$ is defined as:

$$J_2(v) = \sum_{i=1}^{N}(v_i - i)^2$$

The gradient $\bigtriangledown J_2(v)$ is given by:

$$\bigtriangledown J_2(v) = 2(v_i - i)$$

where v_i is the i-th element in the vector v.

In order to utilize the Conjugate Gradient Algorithm, we formulate the cost function to the form of:

$$J(v) = \frac{1}{2}(Av, v) - (f, v)$$

We propose $A = 2I$ where I is the identity matrix.

2.1.3 $J_3(v)$

The cost function $J_3(v)$ is defined as:

$$J_3(v) = \frac{1}{2}(Av, v) - (f, v)$$

with $f = (1, .., 1)$ and $A = tridiag[-1, 2, -1]$.

The gradient $\bigtriangledown J_3(v)$ is given by:

$$\bigtriangledown J_3(v) = Av - f$$

2.1.4 $J_4(v)$

The cost function $J_4(v)$ is defined as:

$$J_4(v) = \frac{1}{2}(Bv, v) - (f, v)$$

with $f = (1, .., 1)$ and $B = tridiag[-1, -1, 4, -1, -1]$.

The gradient $\bigtriangledown J_4(v)$ is given by:

$$\bigtriangledown J_4(v) = Bv - f$$

2.1.5 $J_\epsilon(v)$

The cost function $J_\epsilon(v)$ is defined as:

$$J_\epsilon(v) = \sum_{i=1}^{N}(v_i)^2 + \frac{1}{\epsilon}\sum_{i=1}^{N}\{(v_i + v_{i+1} - \frac{N}{2})\}^2$$

The gradient $\bigtriangledown J_\epsilon(v)$ is given by:

For $i = 1$:

$$\bigtriangledown J_\epsilon(v) = \frac{1}{\epsilon}(2v_i + 2(v_i + 2v_{i+1} - N/2))$$

For $2 \leq i \leq N - 1$:

$$\bigtriangledown J_\epsilon(v) = \frac{1}{\epsilon}(6v_i + 2v_{i-1} + 2v_{i+1} - 2N)$$

For $i = N$:

$$\bigtriangledown J_\epsilon(v) = \frac{1}{\epsilon}(2v_i + 2(v_{i-1} + v_i - N/2))$$

From the first derivative above, we note that there is only subtractions for the factor with the variable N in it and it is a first order polynomial. In the second derivative, we expect that the value for the second derivative are all positive with respect to v_i, hence we get that it is positive definite and hence there exists a global optimum.

2.2 Gradient Algorithms

The methods described in this section are utilized to perform unconstrained optimization. We consider the minimization problem:

$$J(u) = inf_{v \in V} J(v), u \in V \tag{1}$$

The goal is to find $u_k \in V$ which minimizes the function. We start from an arbitrary initialization $u_0 \in V$ and build u_k where u_k is expected to converge to a solution using the gradient descent method.

2.2.1 Gradient Algorithm with Fixed Step

The fixed step algorithm is defined by:

$$u_{k+1} = u_k - \rho \bigtriangledown J(U_k) \tag{2}$$

where ρ is the fixed step

The algorithm is described below.

Algorithm 1 Gradient Algorithm with Fixed Step

```
function PASFIXE(u_0, ρ0, ϵ, k_max)
    for k in 1:k_max do
        [cost, gradient] ← costfunction(u_0)
        if ||gradient|| < ϵ then
            break;
        else
            u_1 ← u_0 − ρ ∗ gradient
            u_0 ← u_1
        end if
    end for
    return u_0
end function
```

2.2.2 Conjugate Gradient Method

The conjugate gradient algorithm iterations are defined by:

$$u_{k+1} = u_k - \rho_k d_k \tag{3}$$

where d_k is the direction of descent.

We initialize u_0 and let $d_0 = J'(U_0)$.

The second direction can be set to

$$d_1 = J'(u_1) + \beta_0 d_0 \tag{4}$$

In order to find β_0, we utilize the conjugate constraint, $< d_1, Ad_0 >= 0$.
Performing scalar product of Ad_0 on both sides of (4):

$$< d_1, Ad_0 >=< J'(u_1), Ad_0 > +\beta_0 < d_0, Ad_0 >= 0$$

$$- < J'(u_1), Ad_0 >= \beta_0 < d_0, Ad_0 >$$

$$\beta_0 = \frac{-<J'(u_1), Ad_0>}{<d_0, Ad_0>}$$

Similarly, at iteration k, we get:

$$d_{k+1} = J'(u_{k+1}) + \beta_k d_k \tag{5}$$

$$\beta_k = -\frac{< J'(u_{k+1}), Ad_k >}{< d_k, Ad_k >} \tag{6}$$

Derivative is equal to zero at the optimum =¿ $< J'(u_{k+1}), d_k >$.
Taking derivatives on both sides:

$$J'(u_{k+1}) = J'(u_k + \rho_k d_k)$$

$$J'(u_{k+1}) = J'(u_k) + A\rho_k d_k$$

Performing scalar product of d_k on both sides:

$$< J'(u_{k+1}), d_k >=< J'(u_k), d_k > + < A\rho_k d_k, d_k >= 0$$

$$\rho_k = -\frac{< J'(u_k), d_k >}{< Ad_k, d_k >} \tag{7}$$

The gradient is described below.

Algorithm 2 Conjugate Gradient Algorithm

```
function CONJUGUEE(u_0, ε, k_max)
    [J_0, G_0] ← costfunction(u_0)
    d_0 ← G_0 // initialization
    for k in 1:k_max do
        [J_k, G_k] ← costfunction(u_0)
        if ‖G_k‖ < ε then
            break;
        else
            ρ_k ← (G_k * d_0)/(Ad_0 * d_0)
            u_1 ← u_0 − ρ_k * d_0
            [J_1, G_1] ← costfunction(u_1)
            β_k ← −(G_1 * Ad_0)/(d_0 * Ad_0)
            d_1 ← G_1 + β_k * d_0
            u_0 ← u_1
            d_0 ← d_1
        end if
    end for
    return u_0
end function
```

2.2.3 Non Linear Conjugate Gradient Method (Fletcher-Reeves)

In this project, we are performing the conjugate gradient algorithm (Fletcher-Reeves) with a fixed step i.e. constant ρ.

The iterations are defined by:

$$u_{k+1} = u_k - \rho_k d_k \tag{8}$$

where d_k is the direction of descent.

We initialize u_0 and let $d_0 = J'(U_0)$.

The direction of descent is as per the conjugate gradient algorithm.

$$d_{k+1} = J'(u_{k+1}) + \beta_k d_k \tag{9}$$

In this variant of Fletcher-Reeves, $beta_k$ is defined as:

$$\beta_k = -\frac{< J'(u_{k+1}), J'(u_{k+1}) >}{< J'(u_k), J'(u_k) >} \tag{10}$$

The gradient is described below.

Algorithm 3 Non Linear Conjugate Gradient (Fletcher-Reeves) Algorithm

function CONJUGUEEFLETCHER(u_0, ϵ, ρ_0, k_{max})
  $[J_0, G_0] \leftarrow costfunction(u_0)$
  $d_0 \leftarrow G_0$ // initialization
  for k in *1:*k_{max} **do**
    $[J_k, G_k] \leftarrow costfunction(u_0)$
    if $\|G_k\| < \epsilon$ **then**
      break;
    else
      $u_1 \leftarrow u_0 - \rho_0 * d_0$
      $[J_1, G_1] \leftarrow costfunction(u_1)$
      $\beta_k \leftarrow -(G_1 * G_1)/(G_k * G_k)$
      $d_1 \leftarrow G_1 + \beta_k * d_0$
      $u_0 \leftarrow u_1$
      $d_0 \leftarrow d_1$
    end if
  end for
  return u_0
end function

3 Experiments

For the experiments reported in the subsections below, the threshold ϵ are set to 10^{-6} with max iterations of 1000. The matrix A in this experiment is given as $A = 2I$ where I is the identity matrix.

3.1 Conjugate Gradient Algorithm on J_1 and J_2

In this section, we will show the results of utilizing the Conjugate Gradient Algorithm on dimensions N = {10, 20, 40}.

Convergence for J_1 and J_2 for Conjugate Gradient			
N	Cost	Iter.	Convergence
10	J_1	1	Achieved convergence
20	J_1	1	Achieved convergence
40	J_1	1	Achieved convergence
10	J_2	1	Achieved convergence
20	J_2	1	Achieved convergence
40	J_2	1	Achieved convergence

The same experiment conducted with the algorithms of gradient descent with fixed step and optimal step has converged in one iteration as well. In this case, we are not able to compare the performance of the algorithms and instead we verified the implementation of the Conjugate Gradient Algorithm on the two cost functions.

3.2 Conjugate Gradient Algorithm on J_3 and J_4

In this section, we will show the results of utilizing the Conjugate Gradient Algorithm on dimensions N = {20, 40, 80, 100, 200}.

Convergence for J_3 and J_4 for Conjugate Gradient			
N	Cost	Iter.	Convergence
20	J_3	20	Achieved convergence with iter = N
40	J_3	40	Achieved convergence with iter = N
80	J_3	80	Achieved convergence with iter = N
100	J_3	100	Achieved convergence with iter = N
200	J_3	200	Achieved convergence with iter = N
20	J_4	17	Achieved convergence with iter < N
40	J_4	28	Achieved convergence with iter < N
80	J_4	51	Achieved convergence with iter < N
100	J_4	62	Achieved convergence with iter < N
200	J_4	118	Achieved convergence with iter < N

The conjugate gradient algorithm should converge to the optimal solutions in at most n steps. A and B is a matrix in $\mathbb{R}^{NxN}$. Let vectors $p_1, p_2, \ldots, p_n \in \mathbb{R}^N$ be pair wise conjugates and it forms the base of $\mathbb{R}^N$. Let $x_0 \in \mathbb{R}^N$ and decompose the error $e_0 = x^* x_0$ by the conjugate vectors $p_1, p_2, \ldots, p_n \in \mathbb{R}^N$

$$e_0 = x^* x_0 = \sigma_1 p_1 + \sigma_2 p_2 + \cdots + \sigma_n p_n$$

Evaluating the coefficients $\sigma_1, \sigma_2, \ldots, \sigma_n \in \mathbb{R}$ is equivalent to solve the problem $Ax = b$ as obtaining e_0 we are able to get $x^* = x_0 + e_0$. Therefore, we achieve convergence in at most N steps.
When the spectrum of the matrix A is good, e.g. clustered spectrum, the algorithm can converge in $<< N$ steps. Observing the convergence for the two cost functions with the matrix $A = A$ for J_3 and $A = B$ for J_4, we see that J_3 converges in N steps while J_4 converges in $<< N$ steps.

3.3 Conjugate and Fixed Step Gradient on J_3 and J_4

In this section, we will show the results of utilizing the Gradient Algorithm with Fixed and Conjugate Gradient on dimensions N = {20, 40, 80, 100, 200} and $\rho = 0.5$.

Convergence for J_3 and J_4 for Fixed Step and Conjugate Gradient					
Algo	J	N	ρ	Iter.	Convergence
GF	J_3	20	0.5	NA	Did not achieve convergence at max iteration
GF	J_3	40	0.5	NA	Did not achieve convergence at max iteration
GF	J_3	80	0.5	NA	Did not achieve convergence at max iteration
GF	J_3	100	0.5	NA	Did not achieve convergence at max iteration
GF	J_3	200	0.5	NA	Did not achieve convergence at max iteration
GF	J_4	20	0.5	6	Unable to converge, invalid cost/gradient
GF	J_4	40	0.5	6	Unable to converge, invalid cost/gradient
GF	J_4	80	0.5	6	Unable to converge, invalid cost/gradient
GF	J_4	100	0.5	6	Unable to converge, invalid cost/gradient
GF	J_4	200	0.5	6	Unable to converge, invalid cost/gradient
GC	J_3	20	NA	20	Achieved convergence, iter=N
GC	J_3	40	NA	40	Achieved convergence, iter=N
GC	J_3	80	NA	80	Achieved convergence, iter=N
GC	J_3	100	NA	100	Achieved convergence, iter=N
GC	J_3	200	NA	200	Achieved convergence, iter=N
GC	J_4	20	NA	17	Achieved convergence, iter<N
GC	J_4	40	NA	28	Achieved convergence, iter<N
GC	J_4	80	NA	51	Achieved convergence, iter<N
GC	J_4	100	NA	62	Achieved convergence, iter<N
GC	J_4	200	NA	118	Achieved convergence, iter<N

The gradient descent with fixed step does not achieve converge at max iteration for J_3 at max iteration of 1000. Increasing the max iteration limit, a convergence is observed at iteration = 1361. The same algorithm with fixed step does not converge for J_4.

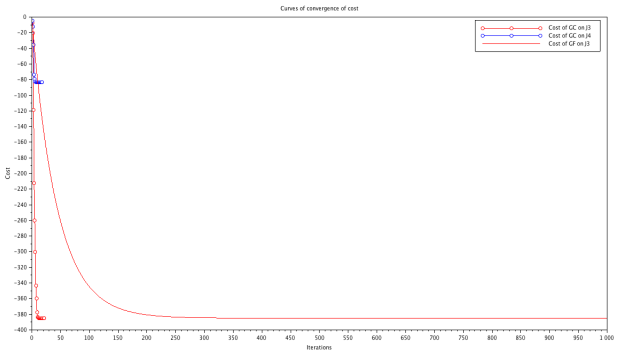

Figure 1: Curves of converge of cost for J_3 and J_4

We observe from the figure above that the conjugate gradient algorithm

converges at a faster rate than the algorithm with fixed step.

3.4 NLCG (Fletcher Reeves) on J_ϵ

In this section, we will show the results of utilizing the Non Linear Conjugate Gradient Algorithm on $\epsilon = 10^p$ where $p = 0, 1, 2, 3$ and for N = {10, 20, 40}.

Convergence for J_ϵ for NLCG				
ρ	N	p	Iter.	Convergence
0.5	10	0	NA	Unable to converge, Invalid cost or gradient
0.1	10	0	40	Achieved convergence
0.1	20	0	49	Achieved convergence
0.1	40	0	53	Achieved convergence
0.1	10	1	NA	Unable to converge, invalid cost/ gradient
0.01	10	1	30	Achieved convergence
0.01	20	1	31	Achieved convergence
0.01	40	1	32	Achieved convergence
0.1	10	2	NA	Unable to converge, invalid cost/gradient
0.01	10	2	NA	Unable to converge, invalid cost/gradient
0.001	10	2	33	Achieved convergence
0.001	10	2	34	Achieved convergence
0.001	10	2	36	Achieved convergence
0.1	10	3	NA	Unable to converge, invalid cost/gradient
0.01	10	3	NA	Unable to converge, invalid cost/gradient
0.001	10	3	NA	Unable to converge, invalid cost/gradient
0.0001	10	3	38	Achieved convergence
0.0001	20	3	38	Achieved convergence
0.0001	40	3	40	Achieved convergence

We are performing the NLCG with a constant step and variable ϵ in the cost function J_ϵ in this experiment. We can observe from the table above that the convergence is achieved with trade-off between the step size ρ and the ϵ. As we decrease the ϵ i.e. increasing p we also require a decrease in the step size ρ in order for the algorithm to converge.

We can also observe that the convergence is achieved in $n > N$ iterations. This is due to the fact that we are utilizing a fixed step for the ρ which is not optimal.

Question: Compare the calculated optimal solution u^* as well as its cost $J_\epsilon(u^*)$ and its gradient to those of the vector c defined by $c_i = N/4$.

$$u^* = \begin{pmatrix} 1.3821138 \\ 2.2357724 \\ 1.9105691 \\ 2.0325204 \\ 1.9918699 \\ 1.9918699 \\ 2.0325204 \\ 1.9105691 \\ 2.2357724 \\ 1.3821138 \end{pmatrix}, \; J_\epsilon(u^*) = \begin{pmatrix} 7.498 \times 10^{-8} \\ 0.0000002 \\ 0.0000003 \\ 0.0000003 \\ 0.0000002 \\ 0.0000002 \\ 0.0000003 \\ 0.0000003 \\ 0.0000002 \\ 5.880 \times 10^{-8} \end{pmatrix}$$

$$\bigtriangledown J_\epsilon(u^*) = 47.764228$$

$$c = \begin{pmatrix} 2.5 \\ 2.5 \\ 2.5 \\ 2.5 \\ 2.5 \\ 2.5 \\ 2.5 \\ 2.5 \\ 2.5 \\ 2.5 \end{pmatrix}, \; J_\epsilon(c) = \begin{pmatrix} 5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 5 \\ 5 \end{pmatrix}$$

$$\bigtriangledown J_\epsilon(c) = 62.5$$

We can observe that the cost of the function at point u^* is lower than the cost at point c and this is expected as we are minimizing the cost function. At the optimal point u^* it should give us a lower cost than an arbitrary point. At u^* the gradient is almost 0 which is also an expected behaviour for optimality.

In addition, with multiple runs of the algorithm, we observe that it converges to the same optimum, u^*. The cost function has a global optimum.

4 Conclusion

To conclude, we can observe that the method conjugate gradient algorithm converges in N steps. This is an improvement over the algorithms with optimal step or fixed step which does not converge in a finite number of iterations.