Optimization: Méthodes de Gradient

Sherly Sherly

EIT Digital Masters, April 2019

1 Introduction

The goal of this project is to program, validate and experiment with the different gradient algorithms. Two gradient algorithms will be utilized: 1) gradient algorithm with fixed step and 2) gradient algorithm with optimal step. There is a third gradient algorithm, gradient algorithm with variable step (à pas variable) which will be defined but not compared in this project. Different cost functions will be implemented for comparison between the two algorithms.

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.

The second order derivative $\bigtriangledown^2 J_1(v)$ is given by:

$$\bigtriangledown^2 J_1(v) = 2$$

The function is convex and the optimal solution is $v = 1$ where v is a vector.

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.

The second order derivative $\bigtriangledown^2 J_2(v)$ is given by:

$$\bigtriangledown^2 J_2(v) = 2$$

The function is convex and the optimal solution is given by $\{v \mid v_i = i\}$.

2.1.3 $J_R(v)$

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

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

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

$$\bigtriangledown J_R(v) = 2(v_i - v_{i-1}^2) - 4v_i(v_{i+1} - v_i^2) + 2(v_i - 1)$$

From the definition of $J_R(v)$:

$$J_R(v) = \sum_{i=1}^{N-1}\{(v_{i+1} - v_i^2)^2 + (v_i - 1)^2\} \geq 0$$

To achieve at minimum at value $J_R(v) = 0$, $v_i = 1$. The global minimum of $J_R(v)$ is when $v = 1$, where v is a vector of N dimension.

The second order derivative $\bigtriangledown^2 J_R(v)$ is given by:

$$\bigtriangledown^2 J_R(v) = 4(3v_i^2 - v_{i+1} + 1)$$

The function is not strictly convex and thus there is no uniqueness of solution. There exists multiple local minimums.

2.1.4 $J_H(x, y)$

The cost function $J_H(x, y)$ is defined as:

$$J_H(x, y) = (x^2 + y - 2)^2 + (y^2 - 2x + 1)^2$$

The gradient $\bigtriangledown J_H(x, y)$ is given by:

$$\bigtriangledown J_H(x, y) = \begin{pmatrix} 4x^3 + 4xy - 4y^2 - 4 \\ 2x^2 + 4y^3 - 8xy + 6y - 4 \end{pmatrix}$$

The second order derivative $\bigtriangledown^2 J_H(x, y)$ is given by:

$$\bigtriangledown^2 J_H(x, y) = \begin{pmatrix} 12x^2 + 4y & 4x - 8y \\ 4x - 8y & 12y^2 - 8x + 6 \end{pmatrix}$$

The trace is given by:

$$trace(\bigtriangledown^2 J_H(x, y)) = 12x^2 + 4y + 12y^2 - 8x + 6$$

For the function to be convex, it requires that the $trace \geq 0$. In this case, the trace is only positive if $y \geq 2x - 1.5$.

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

```
1: function PASFIXE($u_0$, $\rho 0$, $\epsilon$, $k_{max}$)
2:     for $k$ in 1:$k_{max}$ do
3:         $[cost, gradient] \leftarrow costfunction(u_0)$
4:         if $\|gradient\| < \epsilon$ then
5:             break;
6:         else
7:             $u_1 \leftarrow u_0 - \rho * gradient$
8:             $u_0 \leftarrow u_1$
9:         end if
10:    end for
11:    return $u_0$
12: end function
```

2.2.2 Gradient Algorithm with Optimal Step

The optimal step algorithm is defined by:

$$u_{k+1} = u_k - \rho_k \bigtriangledown J(U_k) \tag{3}$$

$$\rho_k = argmin_{\rho > 0} J(U_k - \rho \bigtriangledown J(U_k)) \tag{4}$$

We find the optimal ρ_k at each iteration as the step for the gradient descent algorithm.

In this project, we define a parabolic approximation function $P(\rho)$ to minimize ρ_k.

$$f(\rho) = J(U_k - \rho \bigtriangledown J(U_k)) \approx P(\rho) \tag{5}$$

$$f'(\rho) =< \bigtriangledown J(U_k - \rho \bigtriangledown J(U_k), - \bigtriangledown J(U_k) > \tag{6}$$

When $\rho = 0$:

$$f(0) = J(U_k) \tag{7}$$

$$f'(0) =< \bigtriangledown J(U_k), - \bigtriangledown J(U_k) >= -\| \bigtriangledown J(U_k)\|^2 \tag{8}$$

When $\rho = \rho_{k-1}$:

$$f(\rho_{k-1}) = J(U_k - \rho_{k-1} \bigtriangledown J(U_k) \tag{9}$$

Parabolic approximation:

$$P(\rho) = a\rho^2 + b\rho + c \tag{10}$$

$$P'(\rho) = 2a\rho + b \tag{11}$$

At optimal point, $P'(\rho) = 0$:

$$\rho = \frac{-b}{2a} \tag{12}$$

Applying (7) for approximations:

$$P(0) = a * 0^2 + b * 0 + c \implies c = J(U_k) \tag{13}$$

Applying (8) for approximations:

$$P'(0) = 2a * 0 + b \implies b = -\| \bigtriangledown J(U_k)\|^2 \tag{14}$$

Applying (9) for approximations:

$$P(\rho_{k-1}) = a\rho_{k-1}^2 + b\rho_{k-1} + c = J(v)$$

$$\text{with } v = U_k - \rho_{k-1} \bigtriangledown J(U_k)$$

$$a = \frac{J(v) - b\rho_{k-1} - c}{\rho_{k-1}^2} \tag{15}$$

We will then apply a, b, c to $P(\rho)$ and minimize it to find the optimal value as defined by (12) for ρ at each step.

The algorithm is described below.

Algorithm 2 Gradient Algorithm with Optimal Step

```
function PASOPTIMAL($u_0$, $\rho 0$, $\epsilon$, $k_{max}$)
    $[J_k, G_k] \leftarrow costfunction(u_0)$ // initialization for the first step
    $u_k \leftarrow u_0 - \rho_0 * G_k$
    $[J_k, G_k] \leftarrow costfunction(u_k)$
    for $k$ in $1{:}k_{max}$ do
        $[cost, gradient] \leftarrow costfunction(u_0)$
        if $\|gradient\| < \epsilon$ then
            break;
        else
            // a, b, c based on the polynomial approximation
            // as described above
            $c \leftarrow j$
            $b \leftarrow -\|gradient\|^2$
            $u_k \leftarrow u_0 - \rho_0 * gradient$
            $[J_k, G_k] \leftarrow costfunction(u_k)$
            $a \leftarrow \frac{J_k - b\rho_0 - c}{\rho_0^2}$
            $\rho_0 \leftarrow \frac{-b}{2a}$
            $u_1 \leftarrow u_0 - \rho_0 * gradient$
            $u_0 \leftarrow u_1$
        end if
    end for
    return $u_0$
end function
```

2.2.3 Gradient Algorithm with Variable Step

The fixed step algorithm is defined by:

$$u_{k+1} = u_k - \rho_k \bigtriangledown J(U_k) \tag{16}$$

where ρ_k is a function in terms of ρ i.e. $f(\rho)$.

The algorithm is described below.

Algorithm 3 Gradient Algorithm with Variable Step

```
1: function PASVARIABLE($u_0$, $\rho0$, $\epsilon$, $k_{max}$)
2:     for $k$ in $1{:}k_{max}$ do
3:         $[cost, gradient] \leftarrow costfunction(u_0)$
4:         if $\|gradient\| < \epsilon$ then
5:             break;
6:         else
7:             $u_1 \leftarrow u_0 - \rho * gradient$
8:             $\rho \leftarrow rhofunction(\rho)$
9:             $u_0 \leftarrow u_1$
10:        end if
11:    end for
12:    return $u_0$
13: 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.

3.1 Gradient with Fixed Step on J_1 and J_2

In this section, we will show the results of utilizing the Gradient Algorithm with Fixed Step on dimensions N = {10, 20, 40} and ρ = {1, 0.5}.

Convergence for J_1 and J_2 for Fixed Step

N	ρ	Cost	Iter.	Convergence
10	1	J_1	NA	Did not achieve convergence at max iteration
20	1	J_1	NA	Did not achieve convergence at max iteration
40	1	J_1	NA	Did not achieve convergence at max iteration
10	0.5	J_1	1	Achieved convergence with $u = (1)_N$
20	0.5	J_1	1	Achieved convergence with $u = (1)_N$
40	0.5	J_1	1	Achieved convergence with $u = (1)_N$
10	1	J_2	NA	Did not achieve convergence at max iteration
20	1	J_2	NA	Did not achieve convergence at max iteration
40	1	J_2	NA	Did not achieve convergence at max iteration
10	0.5	J_2	1	Achieved convergence with $u = (1)_N$
20	0.5	J_2	1	Achieved convergence with $u = (1)_N$
40	0.5	J_2	1	Achieved convergence with $u = (1)_N$

At $\rho = 1$ the algorithm does not converge for $J_1(v)$ and $J_2(v)$.

Proof
For fixed step:

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

Initializing u_0 and taking $J_1(v)$ as the cost function:

$$\bigtriangledown J(u_k) = 2(u_k - 1)$$

$$u_1 = u_0 - \bigtriangledown J(u_0) = u_0 - 2(u_0 - 1) = 2 - u_0$$

$$u_2 = u_1 - \bigtriangledown J(u_1) = u_1 - 2(u_1 - 1) = 2 - u_1 = 2 - (2 - u_0) = u_0$$

We observe from the equations above that there is a recursive value between u_0 and u_1 that leads to the non-convergence of the algorithm. In the case of $J_2(v)$ the same behaviour is observed. As such, we can observe that no convergence is reached at max iteration for both $J_1(v)$ and $J_2(v)$.

3.2 Gradient with Optimal Step on J_1 and J_2

In this section, we will show the results of utilizing the Gradient Algorithm with Optimal Step on dimensions N = {10, 20, 40} and ρ = {1, 0.5}.

Convergence for J_1 and J_2 for Optimal Step				
N	ρ	Cost	Iter.	Convergence
10	1	J_1	1	Achieved convergence with $u = (1)_N$
20	1	J_1	1	Achieved convergence with $u = (1)_N$
40	1	J_1	1	Achieved convergence with $u = (1)_N$
10	0.5	J_1	1	Achieved convergence with $u = (1)_N$
20	0.5	J_1	1	Achieved convergence with $u = (1)_N$
40	0.5	J_1	1	Achieved convergence with $u = (1)_N$
10	1	J_2	1	Achieved convergence with $u = (1)_N$
20	1	J_2	1	Achieved convergence with $u = (1)_N$
40	1	J_2	1	Achieved convergence with $u = (1)_N$
10	0.5	J_2	1	Achieved convergence with $u = (1)_N$
20	0.5	J_2	1	Achieved convergence with $u = (1)_N$
40	0.5	J_2	1	Achieved convergence with $u = (1)_N$

Comparing the results against the results for the algorithm with fixed step, we observe that the algorithm converges for $\rho = 1$. This is because the value of ρ is not fixed at 1 at each iteration and hence the optimal algorithm does not meet the recursive behaviour in the fixed step algorithm.

3.3 Gradient with Fixed and Optimal Step on J_R

In this section, we will show the results of utilizing the Gradient Algorithm with Fixed and Optimal Step on dimensions N = 10 and ρ = 1, 0.5, 0.2, 0.1, 0.05, 0.01.

Convergence for J_R for Fixed Step and Optimal Step			
Algorithm	ρ	Iter.	Convergence
Fixed	1	6	Unable to converge, invalid cost/gradient
Fixed	0.5	6	Unable to converge, invalid cost/gradient
Fixed	0.2	13	Unable to converge, invalid cost/gradient
Fixed	0.1	49	Achieved convergence
Fixed	0.05	74	Achieved convergence
Fixed	0.01	404	Achieved convergence
Optimal	1	31	Achieved convergence
Optimal	0.5	29	Achieved convergence
Optimal	0.2	26	Achieved convergence
Optimal	0.1	28	Achieved convergence
Optimal	0.05	25	Achieved convergence
Optimal	0.01	30	Achieved convergence

We observe from the experimental results above that convergence is reached for gradient algorithm with fixed only if ρ is small enough. ρ can be alternatively called as the learning rate of the algorithm. In the event where the step is too large, it may cause a divergence or it may cause it to skip over the optimum and hence not reaching the optimum. In this case, the value of the cost has diverged and thus there is a divergence.

Another notable observation is that they do not converge to the same optimum, we can observe multiple local minimums being reached from the random initialization of the optimization algorithm. This function is not strictly convex.

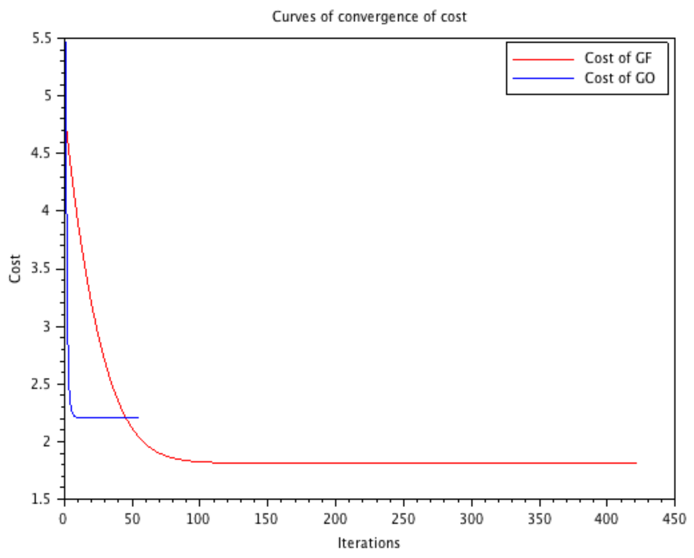

Figure 1: Curves of converge of cost for J_R, $\rho = 0.01$

We observe from the figure above that the gradient algorithm with optimal step converges at a faster rate than the algorithm with fixed step.

3.4 Gradient with Fixed and Optimal Step on J_H

In this section, we will show the results of utilizing the Gradient Algorithm with Fixed and Optimal Step on $u_0 = \{(0,0),(1.5,-1.5)\}$ and $\rho = \{1,\ 0.5,\ 0.2,\ 0.1,\ 0.05,\ 0.01\}$.

Convergence for J_H for Fixed Step and Optimal Step				
Algorithm	u_0	ρ	Iter.	Convergence
Fixed	(0, 0)	1	NA	Does not achieve convergence at max iter
Fixed	(0, 0)	0.5	NA	Does not achieve convergence at max iter
Fixed	(0, 0)	0.2	NA	Does not achieve convergence at max iter
Fixed	(0, 0)	0.1	NA	Does not achieve convergence at max iter
Fixed	(0, 0)	0.05	NA	Does not achieve convergence at max iter
Fixed	(0, 0)	0.01	NA	Does not achieve convergence at max iter
Optimal	(0, 0)	1	20	Achieved convergence
Optimal	(0, 0)	0.5	9	Achieved convergence
Optimal	(0, 0)	0.2	18	Achieved convergence
Optimal	(0, 0)	0.1	12	Achieved convergence
Optimal	(0, 0)	0.05	18	Achieved convergence
Optimal	(0, 0)	0.01	10	Achieved convergence
Fixed	(1.5, -1.5)	1	NA	Does not achieve convergence at max iter
Fixed	(1.5, -1.5)	0.5	NA	Does not achieve convergence at max iter
Fixed	(1.5, -1.5)	0.2	NA	Does not achieve convergence at max iter
Fixed	(1.5, -1.5)	0.1	NA	Does not achieve convergence at max iter
Fixed	(1.5, -1.5)	0.05	NA	Does not achieve convergence at max iter
Fixed	(1.5, -1.5)	0.01	NA	Does not achieve convergence at max iter
Fixed	(1.5, -1.5)	0.005	NA	Does not achieve convergence at max iter
Optimal	(1.5, -1.5)	1	11	Achieved convergence
Optimal	(1.5, -1.5)	0.5	13	Achieved convergence
Optimal	(1.5, -1.5)	0.2	47	Achieved convergence
Optimal	(1.5, -1.5)	0.1	14	Achieved convergence
Optimal	(1.5, -1.5)	0.05	36	Achieved convergence
Optimal	(1.5, -1.5)	0.01	12	Achieved convergence

For each initialization of u_0 they converge to the same minimum regardless of the value of ρ. $u_0 = (0, 0)$ converges to the local minimum $(1, 1)$ and $u_0 = (1.5, -1.5)$ converges to the local minimum $(1.9196395, -1.685016)$. There exists multiple local minimum and hence we can be sure that the function is not strictly convex. We also observe that the gradient algorithm with fixed step does not converge at all.

The gradient $\bigtriangledown J_H(x, y)$ is given by:

$$\bigtriangledown J_H(x, y) = \begin{pmatrix} 4x^3 + 4xy - 4y^2 - 4 \\ 2x^2 + 4y^3 - 8xy + 6y - 4 \end{pmatrix}$$

According to the theorem of convergence, if a function f is Lipschitz and $\bar{f} = minf(x) > -\infty$, then the gradient descent algorithm with fixed step size satisfying $\rho < \frac{2}{L}$ where L is the Lipschitz boundary, will converge to a stationary point. A plot for $J_H(v)$ is attempted and it has shown that the function is not bounded. The function $J_H(v)$ is not bounded and hence it is not Lipschitzian. Hence, we might not observe convergence for the algorithm with fixed step.

4 Conclusion

In conclusion, $J_1(v)$ and $J_2(v)$ are convex functions and we observe good convergence behaviour for both the algorithm with fixed steps and optimal steps besides the recursive behaviour explained for $\rho = 1$. In the case for the cost functions $J_R(v)$ and $J_H(v)$ we observe multiple local minimums, we could observe different behaviours for the different non-convex functions. In the case where it is not strictly convex i.e. there exists more than 1 local minimum, we could implement other algorithms on top of this in order to search for a better minimum. Non-convex optimization are NP-Hard and a suggestion such as Tabu Search [Appendix] might improve the minimum but not necessarily converge to the global minimum and it will increase computation time.

A Appendix

Tabu Search is an algorithm which searches for the best neighbour of the current solution in each step and in this algorithm, the local optimum can lead to a worse or better step. With the Tabu list, this algorithm prevents it from falling into only the local minimum and thus possibly yielding better solution. This enables the exploration of new solution space with the goal of preventing the fall into the local optimum and eventually finding the global optimal solution.

Tabu list is a list that stores the previously visited candidate solution and it is used to prevent local search steps to immediately return to the previously visited candidate solution and avoid cycling.