**Descent Region Algorithm to Find Out Global Minimum Point**

Loc Nguyen

Vietnam Institute of Mathematics

**Abstract**

**1. Introduction to global optimization**

**2. Descent region algorithm**

Suppose *f* is the target function that we need to find out its global minimum point with regard that *f* is scalar-by-vector function.

Where is real number field and is real vector space.

The proposed method is iterative algorithm whose input is an initial point ***ω****0* and the output is global minimum point **z***\*\**. The ideology of this algorithm is that given a known local minimum point **z***\**, the better local minimum point is searched only in a so-called *descend region* under the known point **z**\*. If the target function has global minimum, the algorithm is terminated after the finite number of iterations; at that time the local minimum point **z**\* approaches the global minimum point **z***\*\**. In the algorithm, the descend region is begun by a so-called *descent point* which is defined as the point under the minimum point **z**\* and so the next better minimum point is searched under descent point. Suppose **z***i* is the descent point at the *ith* iteration and **z***i* is initialized by ***ω****0*. The algorithm has many iterations and each iteration includes two steps:

1. *Step 1*: Searching local minimum point step
2. *Step 2*: Determining descent region step

In general, following is the pseudo-code like C language for descent region algorithm. Note the input is an arbitrary point ***ω****0* and the output is global minimum point **z***\*\**.

*//Initialization*

**z***0* ***= ω****0*

**z\*\*** = **z***\** = +∞

*i* = *0*

Loop

**z***\** := *searching local minimum point with the input* **z***i* (step 1)

**z***i* :*= determining descent point with the input* **z***\** (step 2)

**z\*\*** = **z***\**

*i = i + 1*

While (*no descent point* **z***i found*)

**2.1. Searching local minimum point step**

With the descent point **z***i*, we apply descent gradient method to find out the local minimum point **z***\**. Note that **z***i* is initialized by ***ω****0*. We know that descent gradient method is also iterative method; suppose at iteration *j* in this method, the next minimum-candidate point is computed as following:

Where,

* The point is the previous minimum-candidate point and is initialized by the starting point **z***i*.
* The direction is the descent direction, which is the opposite of gradient of function *f*. We have where denotes the gradient of function *f*.
* The value is the length of the descent direction .

After *m* iterations, the point converges to the local minimum point **z***\**. If the terminated condition of descent gradient method is equal to ***0*** ( = ***0***), then **z***\** is likely local minimum point and we go to step 2. Otherwise, the terminated condition is that descent gradient method runs after *m* iterations, we compare the current minimum point **z***\** with the previous minimum point ***y****\**:

* If **z***\** is equal to ***y****\**, then our algorithm stops and **z***\** is the global minimum point, **z***\*\** = **z***\**.
* If **z***\** is not equal to ***y****\**, then go to step 2.

**2.2. Determining descent region step**

Given the local minimum point **z***\** found out in step 1 and let *ε* is very small pre-defined positive number, *ε* > *0*. Let *h*(***x***) be hyper-plane that goes through the level *f*(**z***\**)– *ε* and is parallel to the abscissa hyper-plane or domain plane of function *f*, we have:

Let = *f*(**z***\**) be the minimum value of target function *f* at **z***\**, we have:

The intersection between hyper-plane *h*(***x***) and the function *f*(***x***) is a contour at level + *ε* with note that *f*(***x***) is a hyper-surface. Following is the equation of such contour:

Or,

Note that this equation is called *intersection equation*. If the contour is not existent; in other words, if the hyper-plane *h*(***x***) does not intersect with the surface *f*(***x***), then the algorithm is stopped and **z***\** is the global minimum point and we have **z***\*\** = **z***\**. Otherwise, suppose **z***0* is a point that belongs to the contour; in other words, **z***0* is a solution of the equation *f*(***x***) = – *ε* with note that the method to find out **z***0* is proposed later, then we do two important tasks:

* Increasing *i* by *1*, we have *i* = *i*+*1*.
* The new descent point **z***i* is assigned by the solution **z***0* and we have **z***i* = **z***0* with attention that the index *i* was increased by *1*. After that, we go back step 1.

It is easy to recognize that the next better local minimum point is searched only in the region under the solution **z***0*. This region is *descent region* and so this algorithm is called descent region method and solution **z***0* is also called descent point. Although both **z***i* and **z***0* are called descent point, the point **z***i* is known as descent point at the *ith* iteration and **z***0* is solution of the equation *f*(***x***) = – *ε*. The pseudo-code for descent region algorithm is refined as below with note that the input is an arbitrary point ***ω****0* and the output is global minimum point **z***\*\**.

*//Descent region algorithm*

*//Initialization*

**z***0* ***= ω****0*

**z\*\*** = **z***\** = +∞

*ε* := *very small pre-defined number*

*i* = *0*

Loop

*//Step 1*: *Searching local minimum point*

**z***\** := *searching local minimum point with the input* **z***i*

If **z***\* not found* then

break

Else If **z***\* equal previous minimum point* then

**z***\*\** = **z***\**

break

End If

*//Step 2*: *Determining descent region*

**z***0* := *solution of intersection equation with the input*s***x****\* and ε*

**z***i =* **z***0*

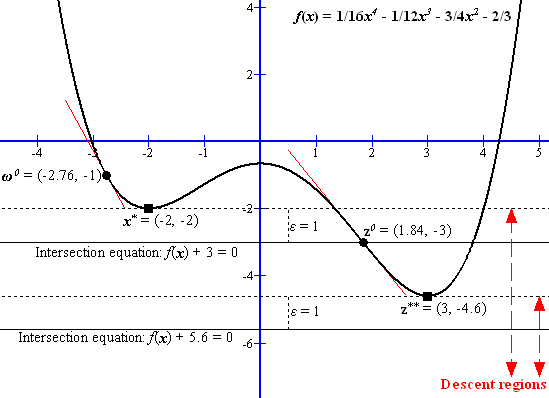
**z\*\*** = **z***\**

*i = i + 1*

While (*no descent point* **z***0 found*)

For example, given target one-variable function

Given initial point ***ω****0* = (–*2.76*, –*1*), it is easy to receive the first local minimum point ***x****\** = (–*2*, –*2*) by applying gradient method. Suppose the very small number *ε* is *1*, the descent point **z***0* which is solution of intersection equation *f*(*x*) – *f\** + *ε* = *f*(*x*) – (–*2*) + *1* = *f*(*x*) + *3* = *0* is solved by the method mentioned in next section; hence we get **z***0* = (*1.84*, –*3*). Starting from **z***0* = (*1.84*, –*3*), the next local minimum point is found out, ***x****\** = (*3*, –*4.6*) by applying gradient method again. The new intersection equation *f*(*x*) + *5.6* = *0* has no solution and it is concluded that the global minimum point is ***x****\*\** = (*3*, –*4.6*). Following figure depicts this example.



**Figure 1.** An example of descent region method

**3. Method to determine descent region**

As aforementioned, descent point**z***0* is a solution of the equation of the intersection between hyper-plane *h*(***x***) and target function *f*(***x***) and descent region is the region under the solution **z***0*. So determining descent region is equivalent to solving the *intersection equation* in order to find out its solution **z***0*.

Where **z***\** is a local minimum point. Note that – + *ε* is the scalar constant.

Let *y* = *f*(***x***) – + *ε*, we have:

The surface specified by *y* = *f*(***x***) – + *ε* is the same to the one specified by *g*(***x***) = *0* with attention that *y* is the scalar variable while ***x*** = is the vector variable. Let be the gradient vector of function *g*, we have:

Where is the partial derivative of *f* with regard to partial variable *xi*. There is convention that the gradient vector is row vector. The value of gradient vector at arbitrary point ***x****0* is:

Where is the value of partial derivative of *f* with regard to partial variable *xi* at ***x****0*. Please distinguish the arbitrary starting point ***x****0* from the initial point ***ω****0* of descent region algorithm mentioned in previous section.

The gradient vector is normal vector of tangent hyper-plane of *g* and so this tangent hyper-plane at point is specified by following equation:

We deduce that

So we have:

Where is variable, is an arbitrary point, **z***\** is a local minimum point and *ε* is very small pre-defined vector.

So the equation of tangent hyper-plane of *g* at point is

The intersection between tangent hyper-plane and the hyper-plane *y* = *0* is the solution of following equation:

Because and and are scalar values, we have:

Where,

The equation (2) is the hyper-line which represents the intersection between tangent hyper-plane and the hyper-plane *y* = *0* and so equation (2) is called intersection equation. Because equation (2) is intersection hyper-line, it has many solutions which are points belonging on it. Now we find out only one solution ***x****1* of equation (2) with regard that ***x****1* satisfies two following conditions:

1. Point ***x****1* is the projection of ***x****0* on the intersection hyper-line.
2. Point ***x****1* belongs to intersection hyper-line.

The first condition implies that the vector ***p****1* = ***x****0* – ***x****1* is parallel to orthogonal vector of intersection hyper-line. It is easy to infer from equation (2) that such orthogonal vector is . Therefore, the first condition is interpreted by following equation:

Where ***x*** = and *k* are unknowns.

The second condition implies that ***x****1* satisfies equation (2).

We set up the set of equations so as to determine ***x****1* as below:

Such set of equations is called projection set of equations or projection system. Note that the unknowns of projection system are , ,…, and *k*. Let ***A****0* and ***b****0* are matrix and vector such that:

The projection system is re-written as below:

Matrix ***A****0* is called projection matrix and vector ***b****0* is called projection vector. The determinant of matrix ***A****0* denoted |***A****0*|isnot equal to *0* if and only ifthe gradient vector is not equal to ***0*** because two any rows or columns *ith* and *jth* of matrix ***A****0* are linearly independent if and only if both respective partial derivatives and are not equal to *0*. Suppose |***A****0*| is not equal to *0*, it is easy to find out ***x****1* by Cramer’s method as following:

Let is the matrix constructed by replacing *jth* column in matrix by projection vector .

I proposed the iterative method which is the simulation of the Newton – Raphson method so as to solve equation (2) based on equation (3). Suppose we have an approximate solution ***x****k* at the *kth* iteration, we set up the projection system based on ***x****k* in order to find out the next better solution ***x****k+1*; hence ***x****k+1* is solution of following equation:

Where,

As aforementioned, the positive number *ε* is pre-defined very small number and is the local minimum value at **z***\**. Note that ***A****k* and ***b****k* are totally determined according to ***x****k* and ***x****k* is initialized by an arbitrary point ***x****0*. It is easy to infer that the solution ***x****k+1* is calculated as below:

Where is the matrix constructed by replacing *jth* column in projection matrix by projection vector .

It is easy to recognize that ***x****k+1* is calculated based on previous ***x****k* and so the proposed method is iterative process. If the determinant |***A****0*| is equal to *0* at the *kth* iteration, then the algorithm stops and ***x****k* is the approximate solution. If the descent point**z***0* which is precise solution of intersection equation (2) is existent, then the approximate point ***x****k* or ***x****k+1* will approach descent point**z***0* after the finite number of iterations. In other words, the descent region which supports us to search global minimum point is found out.

The convergence of the proposed method is dependent on the starting point ***x****0*. If ***x****0* is not in the volume where the method converges, no solution **z***0* is found although there is existence of solutions of equation (2). Moreover, the closer to descent point **z***0* the point ***x****0* is, the faster the convergence speed is. So the way to choose right ***x****0* is very important, which is mentioned in next section.

The terminated condition of determining descent point **z***0* method is very important. Although it is totally assured that equation (2) has no solution when the determinant |***A****k*| is equal to *0* at the *kth* iteration, the cost of solving out the descent point **z***0* by the proposed iterative method is significant. So, it is necessary to test whether equation (2) has solution or not before finding out the descent point. If there is no existence of solutions of equation (2), the descent algorithm is stopped and we concludes that the current local minimum point **z***\** is global minimum point **z***\*\**. In practice, we often apply descent method into finding out global minimum in a given volume [***a***, ***b***] which is an interval in , rectangle in or volume in .

Therefore, if equation (2) is a polynomial equation, *p*(***x***) = *0* where *p*(***x***) = *f*(***x***) – (*f*(**z***\**) + *ε*) is a polynomial and ***x*** is scalar unknown, there is some methods to determine interval of solutions, for example, given *p*(***x***) = *an****x****n* + *an-1****x****n-1* +…+ *a1****x****1* + *a0* and let *A* be the largest among absolute values of coefficients, *A* = *max* {|*a1*|, |*a2*|,…, |*an-1*|, |*an*|}, then the upper bound *u* of all real solutions is calculated as below:

Where *an* is the coefficient of ***x****n* with attention that the number *n* denotes the *nth* power when we often use the superscript number to denote the index. If *u* is smaller than lower bound *a* of given interval [*a*, *b*], then equation (2) has no solution. We can use another method, Sturm’s theorem, to determine the number of distinct real solutions located in given interval [*a*, *b*]. If equation (2) is arbitrary equation, we still use the terminated condition |***A****k*| = 0 and so how to construct optimal starting point ***x****0* is very important and is mentioned in next section.

There is a case that target function *f* has infinitely many local minimum points, which means that *f* has no global minimum point or the global minimum point is the negative infinite number. This case leads the descent region algorithm to run in infinite loop. So we add one more terminated condition that the descent region algorithm will stop after *M* iterations. The pseudo-code for descent region algorithm is refined as below with note that the input is an arbitrary point ***ω****0* and the output is global minimum point **z***\*\**.

*//Descent region algorithm*

*//Initialization*

**z***0* ***= ω****0*

**z\*\*** = **z***\** = +∞

*ε* := *very small pre-defined number*

*M* := *the* *maximum number of iterations*

*i* = *0*

Loop

*//Step 1*: *Searching local minimum point*

**z***\** := *searching local minimum point with input*:**z***i*

If **z***\* not found* then

break

Else If **z***\* equal previous minimum point* then

**z***\*\** = **z***\**

break

End If

*//Step 2*: *Determining descent region*

**z***0* = +∞

Loop

***x****0* := *constructing optimal starting point*

*no\_solution* = *false*

*k* = *0*

Loop *//Solving intersection equation*

*Constructing matrix* ***A****k and vector* ***b****k according to* ***x****k,* ***z****\* and ε*

If |***A****k*| = *0* then

*no\_solution* = *true*

break

End If

*Constructing matrix according to and*

*k* = *k* + *1*

**z***0* = ***x****k*

While (*true*)

*ε* = *ε* + *ε*

While (**z***0* **z***\** and *no\_solution* = *false*)

If **z***0* = +∞or**z***0* **z***\** then *// If no descent point is found*

**z\*\*** = **z***\**

break

Else

*i = i + 1*

**z***i* = **z***0*

End If

While (*i* < *M*) *//prevent infinite loop when there is no global maximum*

**4. Construct optimal starting point for solving intersection equation**

Given the intersection equation (2), the problem needs solved is to select the starting point so that such point is in the volume where the proposed method in section 2 converges. The volume is defined as the sub-space of sub-set, denoted *v*(***a***, ***b***) or [***a***, ***b***].

The *capacity* of volume is defined by following formula:

Volume can be an interval *v*(***a***, ***b***) = [*a*, *b*] in , a rectangle *v*(***a***, ***b***) = [*a1*, *b1*] x [*a2*, *b2*] in or a volume *v*(***a***, ***b***) = [*a1*, *b1*] x [*a2*, *b2*] x [*a3*, *b3*] in . Note that the volume is infinite volume if any of its bound is infinite, for example

If the volume is infinite volume, its capacity is positive infinite, *c*(***a***, ***b***) = +∞.

*Solution volume* is defined as the volume in which the solution of equation (2) is existent. It is easy to recognize that [***a***, ***b***] is the solution volume of function *f* if and only if there exist two values ***x*** and ***y*** belonging to [***a***, ***b***] such that *f*(***x***)*f*(***y***) *0*.

The aforementioned problem is solved by determining the solution volume [***a***, ***b***] such that [***a***, ***b***] is as small as possible; in other words, given pre-defined volume [***a***, ***b***], what we need to do is to find out the sub-volume [***a****i,* ***b****i*] so that it satisfies three conditions:

1. It is in [***a***, ***b***]; in other words, [***a****i,* ***b****i*] [***a***, ***b***].
2. It is also solution volume, satisfying criterion (5).
3. It is as small as possible. This condition helps the method to determine descent point described in previous section converges as fast as possible.

Such sub-volume is called *optimal volume*. Because it is impossible to determine optimal volume by calculating exhaustedly *f*(***x***) for all ***x*** in [***a****i,* ***b****i*], I propose a novel probability approach to find out optimal volume. Firstly, suppose ***x*** and ***y*** are points in [***a****i,* ***b****i*] such that *f*(***x***) > 0 and *f*(***y***) < 0, respectively. Hence, ***x*** and ***y*** are called positive point and negative point, respectively. Let and be the number of positive and negative points, respectively and let *pi* is the ratio of to the number of total points in optimal volume [***a****i,* ***b****i*], we have:

Where is the number of total points and so, *pi* is the probability of occurrence of negative points in optimal volume [***a****i,* ***b****i*]. The probability method to find out optimal volume is based on two heuristic assumptions:

* If [***a****i,* ***b****i*] is optimal volume, then the probability *0* < *pi* < *1*, in other words, both and .
* The nearer to ½ the probability *pi* is, the more likely it is that [***a****i,* ***b****i*] is optimal volume.

The proposed approach is iterative algorithm whose input is volume [***a***, ***b***] and output is optimal volume [***a****i*, ***b****i*]. Given volume [***a***, ***b***] = [*a1*, *b1*] x [*a2*, *b2*] x … x [*an*, *bn*] is divided into *n\*m* sub-volumes [***a****i*, ***b****i*] = [, ] x [, ] x … x [, ] where *i* = . The algorithm has finitely many iterations. We do two tasks at each iteration:

1. Creating many enough random points in each volume [***a****i*, ***b****i*].
2. Counting the number of positive and negative points, and and calculating the probability *pi* of each sub-volume [***a****i*, ***b****i*] based on and . Which sub-volume that has probability *pi* being larger than *0* and smaller than *1* and nearest to ½ is chosen to be the input for next iteration.

There are two stopped conditions:

1. The deviation between probability *pi* and ½ or the capacity *c*(***a****i*, ***b****i*) is smaller than a pre-defined small number *δ*.
2. Or, the algorithm reaches the maximum number of iterations.

When the optimal volume is determined, the optimal starting point ***x****0* is any point in such volume. Of course, if there is a random point ***c****i* [***a****i*, ***b****i*] such that *f*(***c****i*) = *0*, then we have ***x****0* = ***c****i*. In general, following is the pseudo-code like C language for this algorithm whose input is volume [***a***, ***b***] and output is optimal starting point ***x****0*.

*//Constructing optimal starting point*

*//Initialization*

*Creating many enough random points in* [***a***, ***b***]

*N*–:*= the number of positive points in* [***a***, ***b***]

*N+* :*= the number of negative points in* [***a***, ***b***]

If *N*– > 0 and *N*+ > 0 then

[***a\****, ***b\****] = [***a***, ***b***]

Else

[***a\****, ***b\****] = *Ø*

End If

*δ* := *very small pre-defined number*

*M* := *the* *maximum number of iterations*

*i* = *0*

Loop

*Partitioning* [***a***, ***b***] *into n\*m sub-volumes* [***a****i*, ***b****i*]

*min\_value* = +∞

[***a****min*, ***b****min*] = *Ø*

For each [***a****i*, ***b****i*] in [***a***, ***b***]

*Creating many enough random points in* [***a****i*, ***b****i*]

*Counting the number of positive and negative points*, *and*

If *0* < *pi* < *1* and *min\_value value* then

*min\_value = value*

[***a****min*, ***b****min*] = [***a****i*, ***b****i*]

End If

End For

If [***a****min*, ***b****min*] *Ø* then

[***a\****, ***b\****] = [***a****min*, ***b****min*]

[***a***, ***b***] = [***a****\**, ***b****\**]

Else

break

End If

*i = i + 1*

While (*i* < *M* and *min\_value δ*)

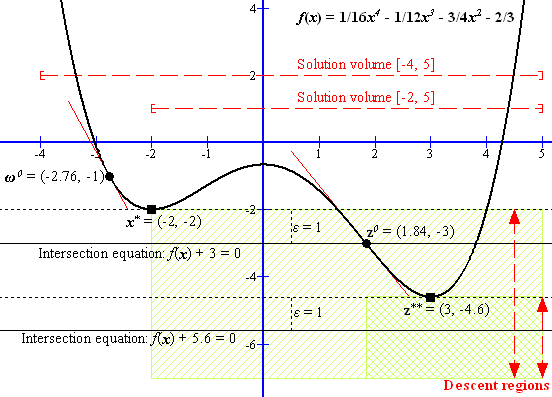
If [***a\****, ***b\****] *Ø* then

***x****0* := *any point in volume* [***a\****, ***b\****]

End If

The essence of probability algorithm is to narrow solution volume according to horizontal axis; similarly, the essence of descent region algorithm is to narrow the descent region according to vertical axis. If the descent region is also reduced according horizontal axis, the descent region algorithm will converge faster. Suppose in horizontal axis, the descent region algorithm begin seeking local minimum points in given initial volume [***a***, ***b***], such volume is called *searching volume*, which is reduced after each iteration. If the current local minimum point is ***x****\**, then the next searching volume is [***x****\**, ***b***]; in other words, the next descent point **z***0* is searched in [***x****\**, ***b***] according to horizontal axis and below ***x****\** according to vertical axis; of course, the solution volume of probability method to construct optimal starting point is [***x****\**, ***b***]. In general, the descent region is reduced according to both horizontal axis and vertical axis. Back the example in section 2.2, given target function,

Following figure depicts how to find out global minimum point of target function by the improved descent region algorithm.



**Figure 2.** Improved descent region method

Two descent regions are shaded areas. It is easy to recognize that the descent areas are narrowed according to two axes.

The pseudo-code for descent region algorithm is refined as below with note that the input is an arbitrary point ***ω****0* and searching volume [***a***, ***b***] and the output is global minimum point **z***\*\** with note that [***a***, ***b***] can be infinite volume.

*//Descent region algorithm*

*//Initialization*

**z***0* ***= ω****0* [***a***, ***b***]

**z\*\*** = **z***\** = +∞

*ε* := *very small pre-defined number*

*M* := *the* *maximum number of iterations*

*i* = *0*

Loop

*//Step 1*: *Searching local minimum point*

**z***\** := *searching local minimum point with input*:**z***i*

If **z***\* not found* then

break

Else If **z***\* equal previous minimum point* then

**z***\*\** = **z***\**

break

End If

*//Step 2*: *Determining descent region*

**z***0* = +∞

[***a***, ***b***] = [**z*\****, ***b***]

Loop

***x****0* := *constructing optimal starting point with input*: [***a***, ***b***]

**z***0* := *solution of intersection equation with input*:***x****\*, ε* and ***x****0*

*ε* = *ε* + *ε*

While (**z***0* **z***\** and *intersection equation has still solutions*)

If **z***0* = +∞or**z***0* **z***\** then *// If no descent point is found*

**z\*\*** = **z***\**

break

Else

*i = i + 1*

**z***i* = **z***0*

End If

While (*i* < *M*)

**5. Conclusion**