

Unconstrained optimization

Saturday, September 2, 2017 12:56

For the glory of God

Introduction

- There is no universally good optimization algorithm. It really depends on particular types of problem.
- Most optimization algorithms require initial value and the **algorithms** may follow these general steps :
 - Begin at a starting point
 - They focus on the search method of Unconstrained optimization
 - Find a search direction
e.g. SNL main gate → Mountain
 - Determine a step size
 - Move to a new point (better than the previous point)
 - Iterate this procedure until a solution is converged.

Then, you may be wondering the follow-up questions as following :

- How does the algorithm select the starting point?
- What technique is used to find a search direction?
- How do we define the step size?
- When do we stop the algorithm?

Overview of Unconstrained optimization algorithm



Now, let's dive into each item.

Direct Search Method

- This method does not require gradient calculations and easy to implement to code.
- However, it may take more function calls to converge.

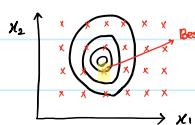
a) Grid search



- Discretize the space

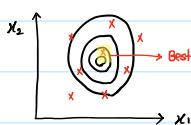
Number 17: long time more function calls to converge.

a) Grid search



- Discretize the space
- Evaluate each point
- pick the best one

b) Random search



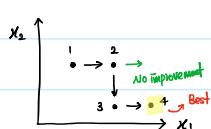
- (Like D.O.E)
- Generate a set of points by sampling from a probability distribution
- Evaluate f at the set of points
- pick the best one

c) Random walk



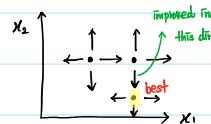
- Pick initial point and evaluate f
- Search in random direction with a fixed step size
- Evaluate and see if it is improved (If not, take another random direction)
- pick the best one

d) Compass search



- Pick initial point
 - Search in one univariate direction at a time with a fixed step size
 - Evaluate and see if it is improved (If not, take another univariate direction)
 - pick the best one
- what if no improvement? \Rightarrow change the step size

e) Coordinate pattern search

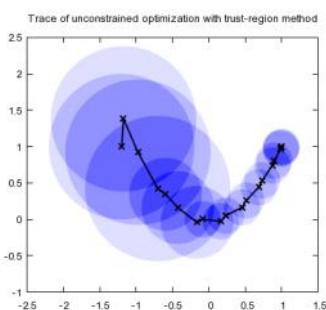


- Pick initial point
- Search in all univariate directions at the same time with a fixed step size
- Evaluate all points and choose the best one
- Iterate f until we pick the best

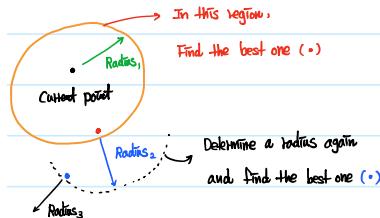
Trust region method

- This works in a way that first define a region around the current point solution and find the best solution.
 - Line search says we are going to pick a direction and then move to optimal length along the line in design space.
- ↳ However, the trust region method is to say that I am going to pick a radius and then

Swipe along the direction



For instance,



Line search method

- A line search method involves two general steps
 - of. assume that we know the initial point

Select a search direction (s_k)

Select the distance to move along the direction (α^*)

a) Select a search direction (s_k)

① Univariate direction

- $s_k = (1, 0)$ or $(0, 1)$; one of the coordinate variable direction
- The simplest approach to selecting a search direction



② Steepest descent direction

$$s_k = \frac{-\nabla f(x_{k-1})}{\|\nabla f(x_{k-1})\|}; \text{ it is like } \frac{\text{Direction}}{\text{Magnitude}} = \text{Unit direction}$$

or $s_k = -\nabla f(x_{k-1})$; This is because the magnitude doesn't matter when we speak of direction.

$$\Delta \mathbf{R} = \frac{|\nabla f(\mathbf{x}_{k+1})|}{\text{Magnitude}}$$

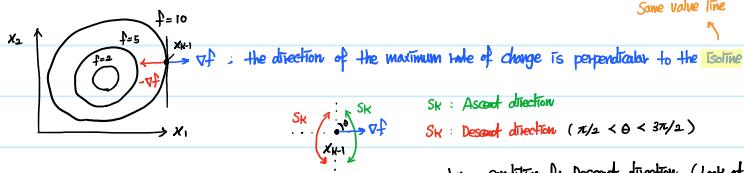
or $\mathbf{s}_k = -\nabla f(\mathbf{x}_{k+1})$; This is because the magnitude doesn't matter when we speak of direction.



Consider yourself sitting at a point and looking at the value of function in all directions around you.

\Rightarrow The direction with the maximum rate of decreasing is along $-\nabla f(\mathbf{x})$

Let's say



e.g. $f=10$

Some value line



s_k : Ascend direction

s_k : Descent direction ($\pi/2 < \theta < 3\pi/2$)

; where condition for Descent direction (Look at the contour)

$$\text{for } d=1, f(\mathbf{x}_{k+1} + \mathbf{s}_k) = f(\mathbf{x}_{k+1}) + \mathbf{s}_k^T \nabla f(\mathbf{x}_{k+1}) + \frac{1}{2} \mathbf{s}_k^T H(\mathbf{x}_{k+1}) \mathbf{s}_k \quad \mathbf{s}_k \cdot \nabla f < 0 \Leftrightarrow |\mathbf{s}_k| |\nabla f| \cos \theta < 0$$

$$\frac{d[f(\mathbf{x}_{k+1} + \mathbf{s}_k)]}{ds_k} = 0 \Leftrightarrow \nabla f(\mathbf{x}_{k+1}) + \mathbf{s}_k^T H(\mathbf{x}_{k+1}) = 0 \quad \therefore \mathbf{s}_k = -H(\mathbf{x}_{k+1})^{-1} \nabla f(\mathbf{x}_{k+1})$$

$$\therefore \theta = \pi/2 \text{ and } 3\pi/2$$



\hookrightarrow It will provide the direction towards the minimum.

③ Newton direction (Descent)

A point at which a given mathematical object is not defined

$$\mathbf{s}_k = -[H(\mathbf{x}_{k+1})]^{-1} \nabla f(\mathbf{x}_{k+1}) ; \text{ It can be derived from 2nd Taylor Expansion}$$

e.g. $x=0$ is a singularity

Here, the Hessian matrix $H(\mathbf{x}_{k+1})$ has to be positive definite.

What if H isn't? for instance, Semi-definite $\rightarrow \lambda=0 \rightarrow \text{Determinant}=0 \rightarrow$ Inverse matrix can't be existed.

\rightarrow It will be a singularity and the algorithm will break.



In general, Newton direction (2nd order) is more useful (or accurate) than 1st order methods such as steepest method.

However, there is no free lunch, e.g. the Hessian matrix is expensive to calculate.

④ Quasi-Newton direction

$$\mathbf{s}_k = -\beta_k^{-1} \nabla f(\mathbf{x}_{k+1})$$

$$\text{; where } \beta_k = \frac{\nabla f(\mathbf{x}_{k+1})^T \nabla f(\mathbf{x}_{k+1})}{\nabla f(\mathbf{x}_{k+1})^T \nabla f(\mathbf{x}_{k+1})}$$

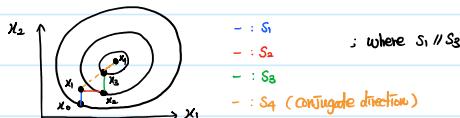
It was introduced to make up the drawback of Newton direction. (1st order method, though)

⑤ Conjugate direction

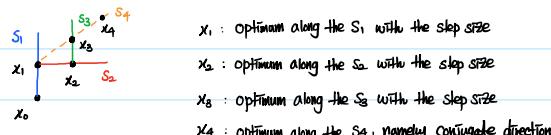
with respect to each minimum

The direction $(x_3^* - x_1^*)$ is called as Conjugate direction.

Let's dive into it graphically. Let's say:



To be more specific,



x_1 : optimum along the s_1 with the step size

x_2 : optimum along the s_2 with the step size

x_3 : optimum along the s_3 with the step size

x_4 : optimum along the s_4 , namely conjugate direction

Note that $s_1 \parallel s_3$

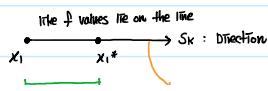
Rao (1996) states the theorem that:

If $f(x)$ is minimized sequentially and the conjugate direction is used, the minimum of the $f(x)$ will be found at or before the nth step, irrespective of the starting point.

b) Select the distance to move along the direction (d^*)

① Polynomial approximations

The idea is that we will approximate the function along the line defined by s_k with a polynomial in d .

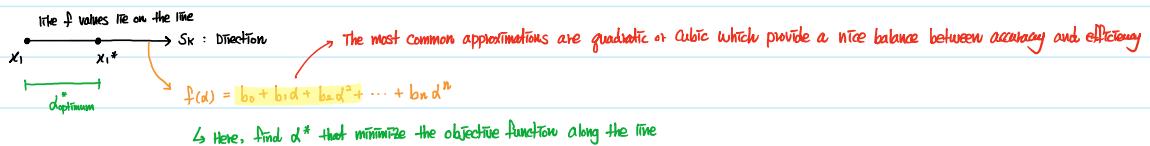


The most common approximations are quadratic or cubic which provide a nice balance between accuracy and efficiency

e.g. If we have $x_1, x_2, f \rightarrow$ quadratic function

Then $n=2 \rightarrow$ Solution should be found $n=2$

Step if we already know the conjugated direction.



↳ Here, find d^* that minimize the objective function along the line

This approach would be not good for multi-modal but good for unimodal.

For instance, two points and one for derivative information are given : $f(0) = 2$ / $f'(0) = -1$ / $f(0.5) = 2.2183$

$$\text{Real function} : f(x) = 1 - 3x + e^{2x}$$

$$\text{Polynomial approximation} : f(x) = b_0 + b_1 x + b_2 x^2$$

$$\text{where } f(0) = b_0 = 2$$

$$f'(0) = b_1 = -1$$

$$f(0.5) = 2 - 0.5 + 0.25 b_2 = 2.2183 \quad \therefore b_2 = 2.8782$$

$$\text{Hence, } f(x) = 2 - x + 2.8782 x^2$$

Then, in terms of the minimum,

$$\begin{cases} \text{Actual Value} : 1.89 \\ \text{Polynomial} : f(0.174) = 1.91 \end{cases}$$



what if we have the function value at three points?

i.e. $(a_1, f_1), (a_2, f_2), (a_3, f_3)$

→ we do same way as if we did it presented above. Then, how would we say it is enough?

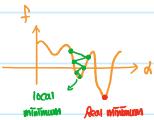
: The answer would be to check $f_{\frac{1}{2}} < \min(f_1, f_3)$ for quadratic polynomial approximation.

↳ This is called as "Bracketing the minimum"

② Golden section method (GSM) : Basically, it is easy to code.

· Unlike polynomial approximations, GSM works for multi-modal functions.

↳ We can't guarantee that it is a global solution such as

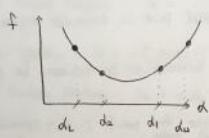


· GSM is an approach that successively refines the bracketing of the minimum with same fraction, namely golden ratio,

in order to converge to an estimate of the minimum. (It's better than Bi-section in terms of computational times)

· Let's figure it out graphically. (From hand-written note for preparation of mid-term) : GSM for unimodal function

$$\text{Let } f(d) = a + b d + c d^2$$



$$\Rightarrow d = \text{Golden ratio} \times (d_u - d_L)$$

$$\therefore d_1 = d_L + d / d_2 = d_u - d$$

$$\therefore \text{If } f(d_1) > f(d_u),$$

$$d_1' = d_L / d_u = d_u$$

$$d = \text{golden ratio} \times (d_u - d_1')$$

$$d_1 = d_1' + d / d_2 = d_u - d$$

$$\text{If } f(d_1) < f(d_u)$$

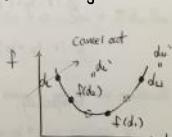
$$\text{advantage: } d_u = d_1 / d_1' = d_L$$

we can even know

the convergence rate

↑

$$5) \text{ Convergence: } \varepsilon = \frac{\Delta x}{x_{u,0} - x_{l,0}} ; \text{ where } \Delta x = x_u - x_l \text{ at each iteration}$$



According to the class material, $z = 0.38197$

Based on the Matlab code, d_1 was defined as :

$$d_1 = d_L + d_u ; d = 0.38197(d_u - d_L)$$

$$\Leftrightarrow d_1 = d_L + 0.38197 d_u - 0.38197 d_L$$

$$\therefore d_1 = (1-z) d_L + z d_u ; \text{ where } z = 0.38197$$

Then why? where does the equation come from?



Based on the definition of golden ratio, we have

$$\frac{\text{Long}}{\text{Short}} = \frac{d_u - d_1}{d_1 - d_L} = 1.618 \text{ with respect to } d_1$$

$$\Leftrightarrow d_u - d_1 = 1.618(d_1 - d_L)$$

$$\Leftrightarrow d_u - d_1 = 1.618 d_1 - 1.618 d_L$$

$$\Leftrightarrow d_1(1.618 + 1) = d_u + 1.618 d_L$$

$$\therefore d_1 = \frac{1}{2.618} d_u + \frac{1.618}{2.618} d_L$$

$$= 0.38197 d_u + 0.618 d_L$$

$$= z d_u + (1-z) d_L$$

c) The Wolfe condition

Let's say that we have obtained both s_k and d^*

; then, we may be wondering " what value of d^* is good enough ? "

- The answer will be given by the Wolfe condition. (It could be applied in each line search.)

These are from AE6310 hand-written notes :

- Okay, let's talk about Wolfe conditions in detail.

This is a condition that we are able to say α^* is good enough in each line search. In other words, the Wolfe conditions give quantitative meaning to good enough. (e.g. $\epsilon < 10^{-6}$ is not enough)

- There are two Wolfe conditions :

Condition 1. Armijo Rule

$$f(x_k) \leftarrow \text{new point} \quad f(x_{k-1} + \alpha s_k) \leq f(x_{k-1}) + C_1 \alpha s_k^T \nabla f(x_{k-1})$$

old point

This term is about slope.
It could be $\frac{df}{d\alpha}$

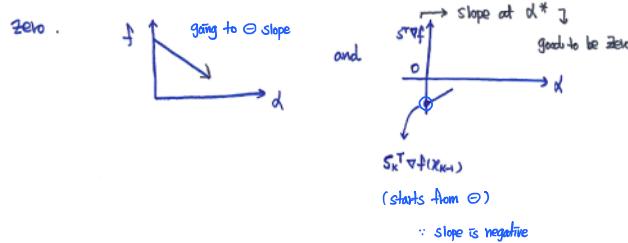
Condition 2. Curvature condition

$$s_k^T \nabla^2 f(x_{k-1} + \alpha s_k) \geq C_2 s_k^T \nabla f(x_{k-1})$$

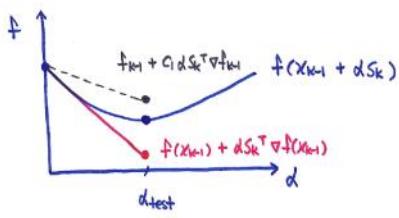
Something in terms of change

with $0 < C_1 < C_2 < 1$, typically $C_1 \sim 10^{-4}$ and $C_2 \sim 0.9$

- Please note that the derivative starts off negative in the search direction, and we are trying to make it go to zero.



Let's see a couple of cases.

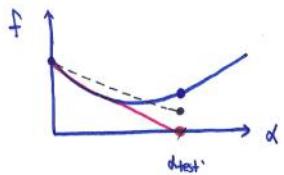


affect global vs. local minimum

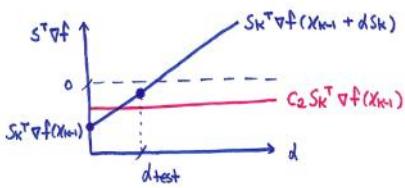
which will be determined by

x_k , S_k , and d^*

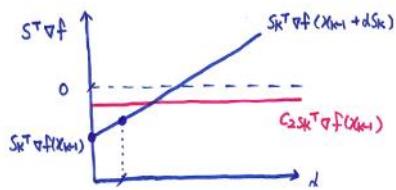
: d_{test} satisfies the
1st condition.



: d_{test}' doesn't satisfy
the first condition.

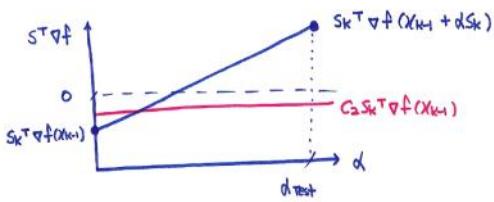


: d_{test} satisfies the
2nd condition.



: d_{test}' doesn't satisfy
the 2nd condition.

- What about this case?



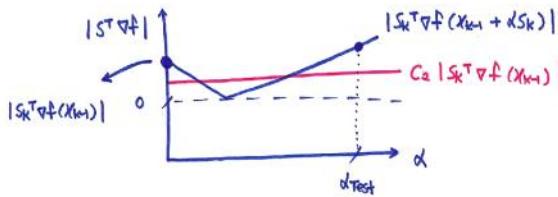
- Even though d_{test} satisfies the second Wolfe condition, it could be argued that it's not a very good approximation of the d^* because we could think that slope is approximately equal to zero at the d^* optimum. However, in this case, very large slope > 0 .
- This issue motivates the development of the Strong Wolfe conditions.

Only the curvature condition (2nd) is changed compared to the original Wolfe conditions. It is specifically modified to prevent values of d that increase the magnitude of the derivative.

$$|s_k^T \nabla f(x_{k+1} + d_{Sk})| \leq C_2 |s_k^T \nabla f(x_{k+1})|$$

$$\text{with } 0 < C_1 < C_2 < 1$$

- Hence, the previous α test is no longer to satisfy the Strong Wolfe Conditions, in particular, for 2nd condition.



- Now, let's take an example from Homework #1 to have better understanding of Wolfe conditions.

Question) 6 of 15

Perform an approximate line search to minimize

$$f(x_1, x_2) = x_1^2 - 2x_1x_2 + 4x_2^2 + x_1 - 2x_2 \text{ by using the}$$

Strong Wolfe conditions with $C_1 = 0.00001$ and $C_2 = 0.1$.

Start at $[-3, -2]$ and use a search direction of $[1, 1]$.

Give an α value that satisfies both Wolfe conditions. You will need to normalize the search direction to have magnitude = 1 first.

Answer) $\alpha = 3$. Let's go to next page. → Next page

To begin with,

$$\cdot S = [1, 1] \rightarrow \text{Normalization} \rightarrow S = \left[\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}} \right]$$

$$\cdot \nabla f(x_1, x_2) = \begin{bmatrix} 2x_1 - 2x_2 + 1 \\ -2x_1 + 8x_2 - 2 \end{bmatrix}$$

$$\Leftrightarrow \nabla f(-3, -2) = \begin{bmatrix} -1 \\ -12 \end{bmatrix}$$

Next, Let's check the Strong Wolfe conditions.

$$\cdot f(x_{k-1} + \alpha S_k) \leq f(x_{k-1}) + C_1 \alpha S_k^T \nabla f(x_{k-1})$$

$$\cdot |S_k^T \nabla f(x_{k-1} + \alpha S_k)| \leq C_2 |S_k^T \nabla f(x_{k-1})|$$

If we just plug what we know in to the inequality equation,

we have :



such as

$$1^{\text{st}} \text{ condition : } 0 \leq \alpha \leq 6.12765$$

$$\cdot f(x_{k-1}) = f(-3, -2)$$

$$2^{\text{nd}} \text{ condition : } \alpha \geq 2.7577$$

$$\cdot \nabla f(x_{k-1} + \alpha S_k)$$

$$\text{Hence, } \alpha^*(\text{minimum}) \approx 3$$

$$= \begin{bmatrix} -3 + \alpha \frac{1}{\sqrt{2}} \\ ... \end{bmatrix}$$

$$\cdot S_k^T = \left[\frac{1}{\sqrt{2}} \quad \frac{1}{\sqrt{2}} \right]$$

Convergence criteria

- Convergence criteria are metrics that we compute and check during optimization in order to determine when to stop the search
 - Maximum number of iterations
 - Absolute change in the objective function
 - Relative change in the objective function

Normalization

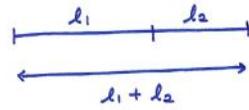
- It is typically considered to be good practice to normalize the design variables in some way.

(Note that it is just a good idea but it is not a magic)

- It achieves more consistent convergence \rightarrow It would be less messy up
- It reduces issues associated with numerical round-off error

Q: where does the number of Golden ratio come from?

- In general, the golden ratio defines as the ratio of the whole line to the larger segment is equal to the ratio of the larger segment to the small segment.



Then we have, $\frac{l_1 + l_2}{l_1} = \frac{l_1}{l_2}$

$$\Leftrightarrow 1 + \frac{l_2}{l_1} = \frac{l_1}{l_2}$$

If we set ϕ (golden ratio) = $\frac{l_1}{l_2}$, we have

$$\phi^2 - \phi - 1 = 0 \quad \therefore \phi = \frac{1 \pm \sqrt{5}}{2}$$

⚡ Ratio is
non-sense!
 ↑
 { -0.68...
 1.61803... (✓)

- This ratio indicates the ratio in which each parts are well-balanced as well as beautiful.

- For example,

