

Gradient descent

Chenyang Song

The gradient descent is an algorithm to perform optimization. Given a benchmark function, the method starts with an initial location and takes steps in the negative direction of the function gradient to solve the minimum of a function. In mathematical notation, we have the form:

$$\mathbf{x}_{i+1} = \mathbf{x}_i - \gamma_i (\nabla f(\mathbf{x}_i))^T \quad (1)$$

Where \mathbf{x}_i is the previous location, and γ_i is the step-size. Starting at the initial location \mathbf{x}_0 , we interactively apply (1) to obtain the local minimum value. Therefore, in the begin, we have to choose an appropriate initial location \mathbf{x}_0 and step-size γ_i . In this project, we will apply our method to three different functions, including the sphere function, the Rosenbrock function, Three-hump camel function. In generally, the formula of sphere function is:

$$f(\mathbf{x}) = \sum_{i=1}^n x_i^2 \quad (2)$$

The formula of Rosenbrock function is :

$$f(x) = \sum_{i=1}^{n-1} [100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2] \quad (3)$$

However, for these two functions, we make both $n=2$, because this way enables us to visualize the functions, which can help us to intuitively understand how gradient descent work. For this reason, the sphere function used in the project is:

$$f(x, y) = x^2 + y^2 \quad (4)$$

The Rosenbrock function is

$$f(x, y) = 100(y - x^2)^2 + (1 - x)^2 \quad (5)$$

And the formula of Three-hump camel function is

$$f(x, y) = 2x^2 - 1.05x^4 + \frac{x^6}{6} + xy + y^2 \quad (6)$$

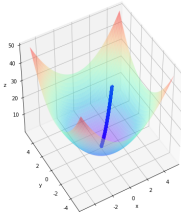
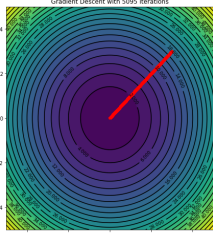
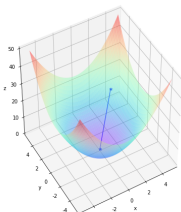
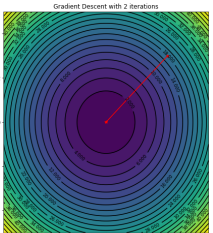
After the defining the benchmark functions, we can identify gradient of each function, which a

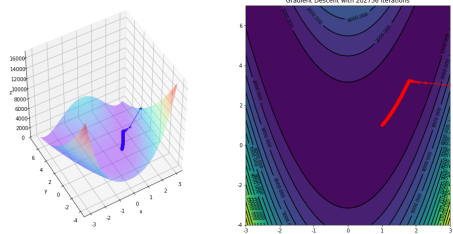
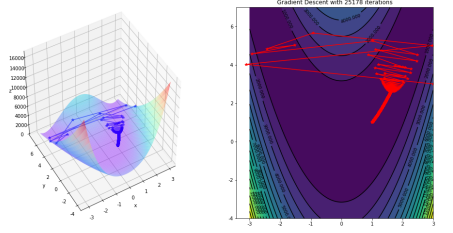
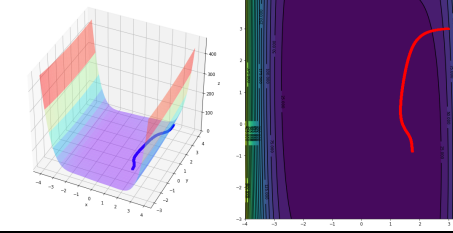
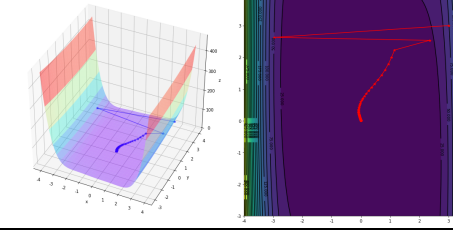
$\Delta f(x,y) = \left[\frac{\partial f(x,y)}{\partial x}, \frac{\partial f(x,y)}{\partial y} \right]$	$\frac{\partial f(x,y)}{\partial x}$	$\frac{\partial f(x,y)}{\partial y}$
Sphere function	$\frac{\partial f(x,y)}{\partial x} = 2x$	$\frac{\partial f(x,y)}{\partial y} = 2y$
Rosenbrock function	$\frac{\partial f(x,y)}{\partial x} = -400x(y - x^2) - 2(1 - x)$	$\frac{\partial f(x,y)}{\partial y} = 200 * (y - x^2)$
Three-hump camel function	$\frac{\partial f(x,y)}{\partial x} = 4x - 4.2x^3 + x^5 + y$	$\frac{\partial f(x,y)}{\partial y} = x + 2y$

Then we just need to set the initial position \mathbf{x}_0 and size-step γ_i for each function, and we have all parameters that are required in gradient descent. However, this step is not an easy works, and there are many challenges that I meet during the application. So, in the next section, we will then discuss challenges during training.

Challenges - step-size

It is difficult to choose a proper learning rate. Too small step-size leads to slow convergence, while too large learning rate may cause our method overshoot, fail to converge, or diverge. If we random choose a fixed initial location, $x=3$ and $y=3$, we set the different step-size and obtain the different convergence paths, which is show below:

	Initial	Step-size	Final Results	Graph	
Sphere	$x = 3$ $y = 3$	$\gamma_i = 0.001$	$x = 0.00011$ $y = 0.00011$ $f(x,y) = 0.00000$ $steps\ num = 5094$		
		$\gamma_i = 0.5$	$x = 0.00000$ $y = 0.00000$ $f(x,y) = 0.00000$ $steps\ num = 1$		

		$\gamma_i = 1.2$	OverflowError: (34, 'Result too large')	
Rosenbrock	$x = 3$ $y = 3$	$\gamma_i = 0.0001$	$x = 1.00112$ $y = 1.00224$ $f(x,y) = 0.00000$ $steps\ num = 202755$	
		$\gamma_i = 0.00085$	$x = 1.00038$ $y = 1.00077$ $f(x,y) = 0.00000$ $steps\ num = 25177$	
		$\gamma_i = 0.001$	OverflowError: (34, 'Result too large')	
Three-hump camel function	$x = 3$ $y = 3$	$\gamma_i = 0.001$	$x = 1.74754$ $y = -0.87325$ $f(x,y) = 0.29864$ $steps\ num = 5392$	
		$\gamma_i = 0.04125$	$x = -0.00001$ $y = 0.00003$ $f(x,y) = 0.00000$ $steps\ num = 161$	
		$\gamma_i = 0.05$	OverflowError: (34, 'Result too large')	

Therefore, we can see that, the selected γ_i has a significant impact on the final result. If γ_i is too large, we will obtain the error from our model, because the results too large. Take the sphere function as the simple example. when γ_i equal to 1.2, the first 5 integrations, our value of $f(x,y)$ will be [50, 98.0, 192.08, 376.48, 737.89, 1446.27]. We discover that the value tends to diverge rather

than converge. So, when the number of integrations increase, the value of $f(x, y)$ also rise. Finally, the value will be over the limitation of python and we will get the error. For too small γ_i , we discover that the efficiency of method is low. For instance, the number of iteration equal to 5095 for $\gamma_i = 0.001$, and 2 for $\gamma_i = 0.5$ in sphere function. And for Rosenbrock function, the number of iteration equal to 25178 for $\gamma_i = 0.0001$ and 25178 for $\gamma_i = 0.00085$. In addition, we also find an interesting situation that three-hump camel function cannot converge to the 0, which is minimum of function, when γ_i is too small. The key reason is that the gradient descent is to find the local minimum, so when γ_i is too small, the point may cannot jump from the certain local area. For this reason, the final result may be not the optimal results that we want. Therefore, it is important for us to find a proper step-size for function. The next section we will try to resolve Challenges.

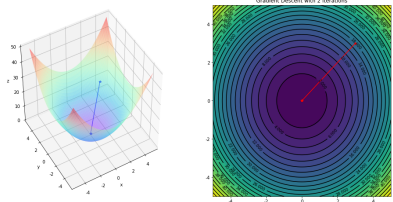
3 Algorithms

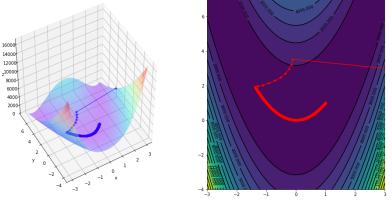
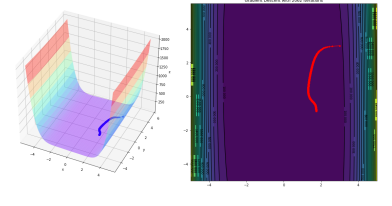
3.1 linear search

The linear search can help us to find the optimal γ_i to obtain the minimal $f(x)$ alone with the inverse direction of the gradient. In mathematical notion, we have the form:

$$Choose \gamma_i = \min_t f(\mathbf{x}_i - t(\nabla f(\mathbf{x}_i))^T) \quad (7)$$

However, if we adopt this process for every iteration, the method will be not much more efficient. For this reason, we just use this approach to identify initial γ_i . The strategy is to pick an upper board for γ_i and use binary search. In our project, the selection of the upper board is based on the OverflowError. We start from the a large initial γ_i , which will cause the OverflowError, and then we manually gradually make this γ_i smaller until the error disappears. Based on the this way, the result we have:

Function	Initial	γ_i	Final Results	Graph
Sphere	$x = 3$ $y = 3$	$\gamma_i = 0.5$	$x = 0.00000$ $y = 0.00000$ $f(x, y) = 0.00000$ <i>steps num = 1</i>	

Rosenbrock		$\gamma_i = 0.00043$	$x = 0.99946$ $y = 0.99892$ $f(x, y) = 0.00000$ $steps\ num = 39906$	
Three-hump camel function		$\gamma_i = 0.00275$	$x = 1.74754$ $y = -0.87368$ $f(x, y) = 0.29864$ $steps\ num = 2061$	

Through the binary search, although we can find the first-step minimal value within a range of the γ_i , but it is not mean that we can find the optimal and efficient solution of gradient descent. For example, the step-size of Rosenbrock is 0.00043 by using binary search, but we need 39906 steps to find the local minimum. On the other hand, we discover that if we set the step-size at 0.00085, we just need 25177 steps. The key reason is that it just helps us to find the minimal at the first step. In addition, somethings we cannot jump from the certain area, so we just can find a local minimum value rather function minimum that we really want. The three-hump camel function is good example for this situation. So in order to improve our method, we can adopt other two algorithms, which are Momentum and Nesterov accelerate.

3.2 Momentum

The momentum is a method that can assists accelerate gradient descent in the relevant direction and dampens oscillations. In mathematical notion, we have the form :

$$v_t = \alpha v_{t-1} + \gamma_i (\nabla f(x_i))^T \quad (8)$$

$$x_i = x_i - v_t$$

When we use this momentum, it is similar that we push a ball down a hill. The ball accumulates momentum as it rolls downhill, and it will become faster and faster. For this reason, it can faster to arrive the bottom of the hill. In the gradient descent, it momentum term can help accelerate the point

go down in the right directions, causing faster convergence. In our model, we just use this way on the Rosenbrock function and Three-hump camel function. And in order to compare Momentum approach and classical approach, the initial step-size is obtained from the classical approach by using binary search. The results is shown in the below table.

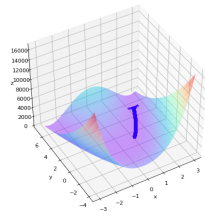
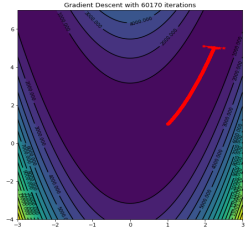
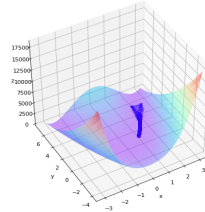
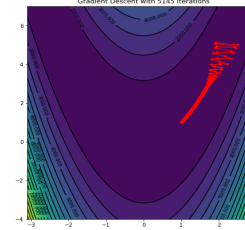
3.3 Nesterov accelerated gradient

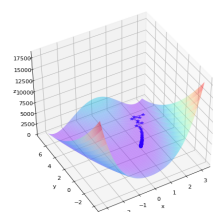
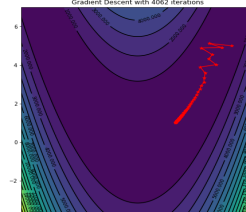
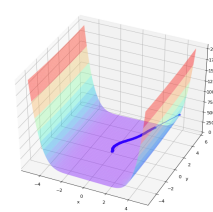
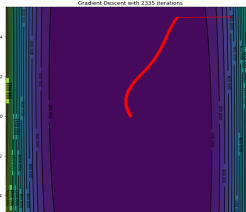
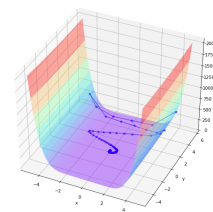
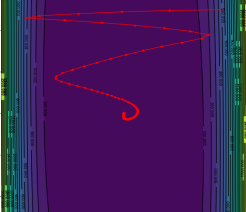
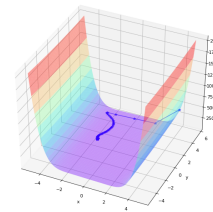
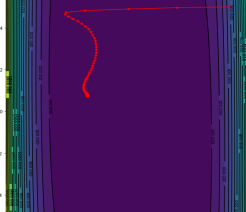
But, from the results, we discover that the point blindly follows the slope. It is means that when it arrives the bottom of hill, it rises again, For this reason, we want make it slow down before it raise again. This situation can be solved by the the Nesterov accelerated gradient (NAG). This way can predict the next location that the point will go, then the point can go in advance follow the direction of gradient of the future location. In mathematical notion, we have the form :

$$v_t = \alpha v_{t-1} + \gamma_i (\nabla f(x_i - \alpha v_{t-1}))^T \quad (9)$$

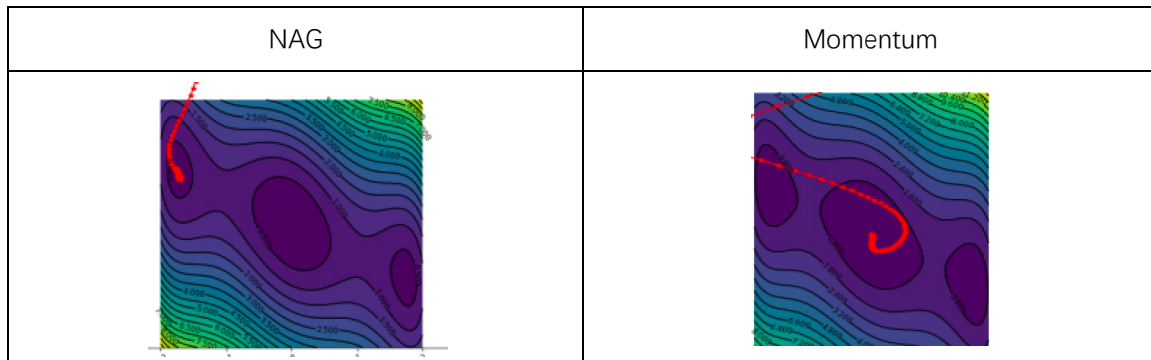
$$x_i = x_i - v_t$$

Therefore, this way can improve the efficient of the gradient descent. Especially, if our original gradient decent has a Zigzag path, this approach may work more well. In our project, we also adopt this way in our Rosenbrock function and Three-hump camel function.

	Initial parameters	Method	Final results	Graph
Rosenbrock	$x^* = 2.5$ $y^* = 5$ $\gamma_i = 0.00043$	Classical	$x = 1.00054$ $y = 1.00108$ $f(x, y) = 0.00000$ <i>steps num = 60169</i>	 
		Momentum	$x = 1.00016$ $y = 1.00032$ $f(x, y) = 0.00000$ <i>steps num = 5144</i>	 

		Neserov Accelerated Gradient	$x = 1.00005$ $y = 1.00010$ $f(x,y) = 0.00000$ $steps\ num = 4061$	 
Three-hump camel function	$x^*= 4$ $y^*=5$ $\gamma_i = 0.00275$	Momentum	$x = - 0.00005$ $y = 0.00011$ $f(x,y) = 0.00000$ $steps\ num = 2334$	 
		Momentum	$x = 0.00001$ $y = - 0.00001$ $f(x,y) = 0.00000$ $steps\ num = 250$	 
		Neserov Accelerated Gradient	$x = -1.74754$ $y = 0.87363$ $f(x,y) = 0.29864$ $steps\ num = 184$	 
<i>The selection of initial location is arbitrary, but the main purpose is to show how different way work</i>				

Through the above table, for the Rosenbrock function, we can see that the steps number of original methods is 60169, but we just need the steps number 4061 by using the Momentum, and 2334 by using NAG. On the other hand, for three-hump camel function, both momentum and NAG also improve the efficiency of the method, but we discover that the points of NAG are converge a different location compare with other tow method. The key reason is that the gradient descent give us the local minium rather than minimum of function. If the step-size cannot provide enough prower to help the jump from the such local area, the points will limit to this certain local area.



The above figure can give us more intuitive prospects. There are 3 dark area in the figure, where local minimum locates. The function minimum should be in the mid dark area. But the NAG gradually slow down its velocity compare with Momentum approach, So the point of momentum even go through first dark area, and it still can jump from this area based on its more strong velocity. Therefore, the NAG cannot always work well than Momentum.

4. In Conclusion

In this article, we use three different functions to study how the gradient descent work. However Through above analysis, we discover that the gradient descent is not a perfect way to solve the optimization problems. Although we can through same Algorithms to overcome some problems about selection of step-sizes, it is still hard for us to adjust the step-sizes to find an optimal path during the process.

In addition, the gradient descent just can help us to find the local minimum, so if we use different initial location, different step-sizes, or different algorithms, the final results may be different. Specially, for the non-convex function, we have challenge to avoid trapped in the numerous local minima. In addition, the saddle point also is a big challenge. It is hard for gradient descent to escape from this point, because the gradient is close to zero in all dimensions in this point. Therefore, we cannot arbitrarily use the gradient descent to solve every minimum problem. We need to make decision to use which method based on their characteristic.