Course Project Report: Advanced Math Analysis with Matlab

KeZheng Xiong
22920202204622

December 21, 2021

Abstract

The report for the end-of-term project of Advanced Math Analysis with MATLAB fall 2021 course. All the source code is open-sourced on the Github repository `https://github.com/SmartPolarBear/matlab-math-analysis-csxmu-2021` under **GPLv3** license

Contents

1 Problem 1 **2**
1.1 Problem Description 2
1.2 Solution 2
1.2.1 The gradient of the function 2
1.2.2 Find the extreme values 2
1.3 Analysis and Conclusion 3

2 Acknowledgment **3**

1 Problem 1

1.1 Problem Description

Given function $F(x, y) = 0.2x^2 + 0.1y^2 + sin(x + y)$, please work out its gradient. Based on the gradient, please find out the local extreme of function F(x,y) when both x and y are in the range of $[-2 * \pi, 2 * \pi]$. The 2D and 3D views of the function is given in Fig. 1.

1.2 Solution

1.2.1 The gradient of the function

I get the gradient of the function using the following code

```
1  syms x y;
2  f=0.2*x^2+0.1*y^2+sin(x+y);
3  diff(f,x)
4  diff(f,y)
```

Listing 1: Gradient Calculation

Based on the result, the gradient is

$$\nabla \cdot f(x, y) = (\frac{2 * x}{5} + cos(x + y), \frac{y}{5} + cos(x + y)) \tag{1}$$

1.2.2 Find the extreme values

To find the extreme values of $F(x, y)$ with gradient decent method, we walk little steps towards the direction of the gradient. To formalize this idea, the algorithm is shown as follows.

Algorithm 1 Gradient Descent

Input: Initial point x_0, a constant α, $k = 0$

while termiation condition does not hold **do**

$k = k + 1$

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

Various problems occurs if this brute-force algorithm is implemented directly. The speed of convergence is annoyingly slow if parameters are not chosen right. In fact, I never succeeded finding a set of parameters that works. A well-known optimization is called Stochastic gradient descent, or SGD, improve it significantly.

The given parameter α in the brute-force algorithm, which is referred as learning rate, will change each round according to the situation. To be more exact, SGD tries to find a learning rate m, so that it minimize the function

$$h(x, y, m) = \mathbf{x_0} + \nabla \cdot f(x, y) \tag{2}$$

so the following equation is solved each round in the while-loop

$$\frac{\partial h}{\partial m} = 0 \tag{3}$$

The implementation is shown in Code 2

```
1
2  function [endp,num,hist,list]=gradient_descent(f,x0,eps)
3      syms x y m;
4      d=-[diff(f,x);diff(f,y)];
```

```
5
6           nd=subs(d,x,x0(1));
7           nd=subs(nd,y,x0(2));
8           nrm=double(norm(nd));
9
10          list=[];
11
12          k=0;
13          while(nrm>=eps)
14          nx0=x0+m*nd;
15          nf=subs(f,x,nx0(1));
16          nf=subs(nf,y,nx0(2));
17          h=diff(nf,m);
18          nm=solve(h);
19
20          x0=x0+nm*nd;
21          k=k+1;
22
23          hist(k,:)=[double(x0(1));double(x0(2))];
24          vf=subs(f,x,x0(1));
25          vf=subs(vf,y,x0(2));
26          list = [list double(eval(vf))];
27
28          nd=subs(d,x,x0(1));
29          nd=subs(nd,y,x0(2));
30          nrm=double(norm(nd));
31          end
32
33          num=k;
34          endp=x0;
35      end
```

Listing 2: SGD Gradient Descent

To test the implementation, Each point of the steps taken in the procedure of gradient descent algorithm is plotted on Figure 1, in red scattered lines. The initial point chosen are$(-3,-4)$, $(0,-1)$, $(-2,-2)$. The choice of initial points are significant. It can **influence the speed of convergence dramatically**. Given a bad initial point, it may not reach the point of convergence at all, or take unbearable long time. Note that the second choice of $(0,-1)$ is especially difficult. It takes may attempts to find this point which has a high speed of convergence.

1.3 Analysis and Conclusion

Gradient descent is a widely-used way of finding the extreme values of any function. There are many optimizations for this method, which will improve the speed of convergence. SGD is taken in this project, which yields satisfying results.

SGD gradient descent has a major flaw that it is known for "easy to be trapped in a local minimum", which may account for the difficulties in finding a good initial point for the second extreme value. More sophisticated optimization such as Momentum gradient descent and AdaGrad gradient descent can be applied if the speed of convergence is too slow.

2 Acknowledgment

Thanks to (TODO)

Figure 1: The visualization of the steps of the SGD gradient descent algorithm