第一次作业

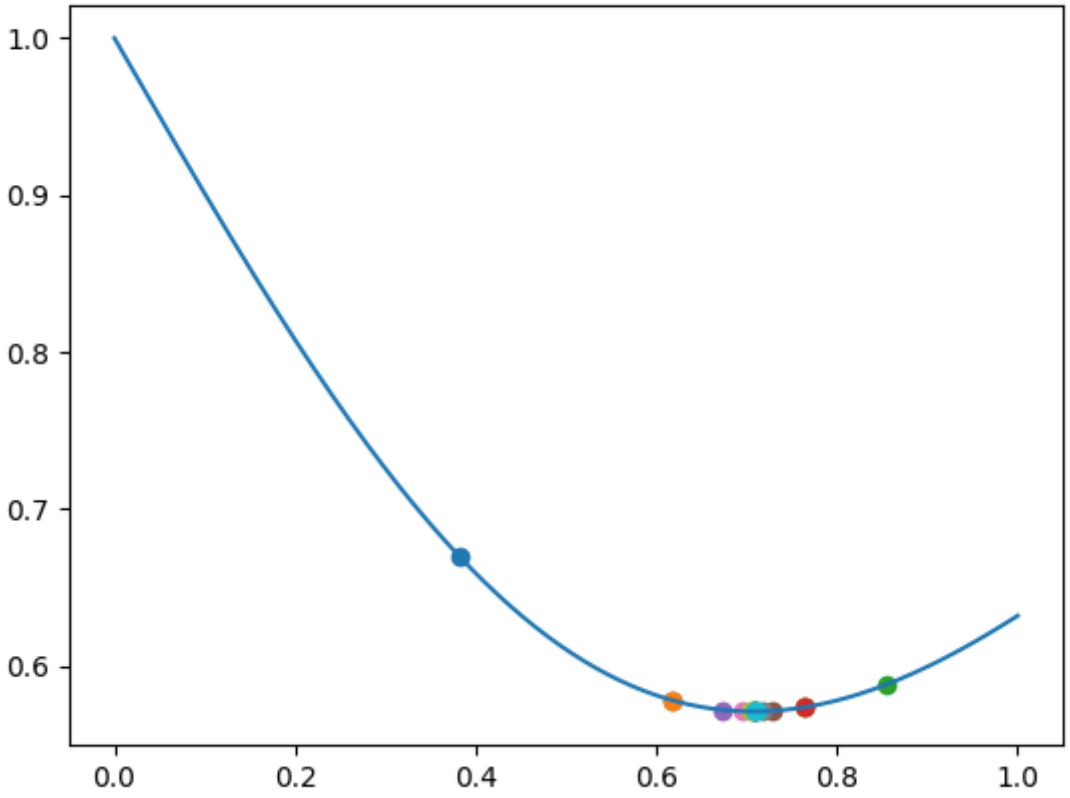
黄金分割法求函数 $\phi(x)=1-\alpha e^{-\alpha^2}$ 的极小值，区间为[0,1]，$\epsilon=0.01$。

```
import matplotlib.pyplot as plt
import numpy as np
# define the function
def f(x):
    return 1-x*pow(2.718, -x*x)
# parameters define
left = 0
right = 1
gold_parameter = 0.618
# make graph
x_numbers = np.linspace(left, right, 1000)
y_numbers = [f(x) for x in x_numbers]
plt.plot(x_numbers, y_numbers)
iter = 0
# iteration
while right - left > 0.01:
    length = right - left
    x1 = left + (1-gold_parameter)*length
    x2 = left + gold_parameter*length
    x = [x1, x2]
    y = [f(x1), f(x2)]
    plt.scatter(x, y)
    # judge
    if f(x1) > f(x2):
        left = x1
    else:
        right = x2
    iter = iter + 1
print("共迭代{0}次，最终结果是{1}".format(iter, f(x1)))
plt.show()
```

运行结果

```
共迭代10次，最终结果是0.5710968311798179
```

函数图像



使用Armijio搜索算法计算第k次的搜索步长α_k,目标函数

$$100(x_2 - x_1^2)^2 + (1 - x_1)^2$$

```
import matplotlib.pyplot as plt
import numpy as np
# calculate the result
from mpl_toolkits.mplot3d import axes3d
from matplotlib import cm


def f(x):
    x1 = x[0]
    x2 = x[1]
    return 100*(x2-x1*x1)*(x2-x1*x1)+(1-x1)*(1-x1)

# 3d-plot
def plot_function(f):
    fig = plt.figure()
    x1 = np.outer(np.linspace(-2, 2, 30), np.ones(30))
    x2 = x1.copy().T
    ax = fig.add_subplot(111, projection='3d')
    z1 = 100*(x2-x1*x1)*(x2-x1*x1)+(1-x1)*(1-x1)
    ax.plot_surface(x1, x2, z1, cmap='viridis', edgecolor='none')
    plt.show()
# calculate the gradient

def cal_gra(f, x):
    x1 = x[0]
    x2 = x[1]
    delta = 0.0001
    dx1 = (f([x1+delta, x2])-f([x1, x2])) / delta
    dx2 = (f([x1, x2+delta])-f([x1, x2]))/delta
    return np.array([dx1, dx2])
def Armijo(x,direction):
    # define parameters
    rou = 0.6
    dis = 1  # define the moving distance
    beta = 0.5  # define the constant moving distance
    iter = 1 # define iter times
    iter_max = 20
    print("原位置为({0},{1}),函数值为{2}".format(x[0],x[1],f(x)))
    if f(x + direction * dis) <= f(x) + rou * dis * cal_gra(f, x).dot( direction):
        dis = 1
    else:
        dis = beta
        while iter < iter_max:

            if f(x + direction * dis) <= f(x) + rou * dis * cal_gra(f, x).dot( direction):
                print("梯度为({0},{1})".format(cal_gra(f,x)[0],cal_gra(f,x)[1]))
                break
            dis = dis * rou   #! update formula
            iter = iter + 1
    x = x + dis * direction
    print("现位置为({0},{1}),函数值为{2}".format(x[0],x[1],f(x)))
    return dis
if __name__ == "__main__":
    x = np.array([1, -1])  # position
```

```
    direction = np.array([0, 1])
    print("搜索步长为{}".format(Armijo(x,direction)))
    plot_function(f)
```

运行结果

```
原位置为(1,-1),函数值为400
现位置为(1,0),函数值为100
搜索步长为1
```

函数图像