

## 1. 梯度下降

- 1.1 对于梯度下降的数学理解
- 1.2 单变量函数的梯度下降
- 1.3 双变量的梯度下降
- 1.4 学习率的选择

# 1. 梯度下降

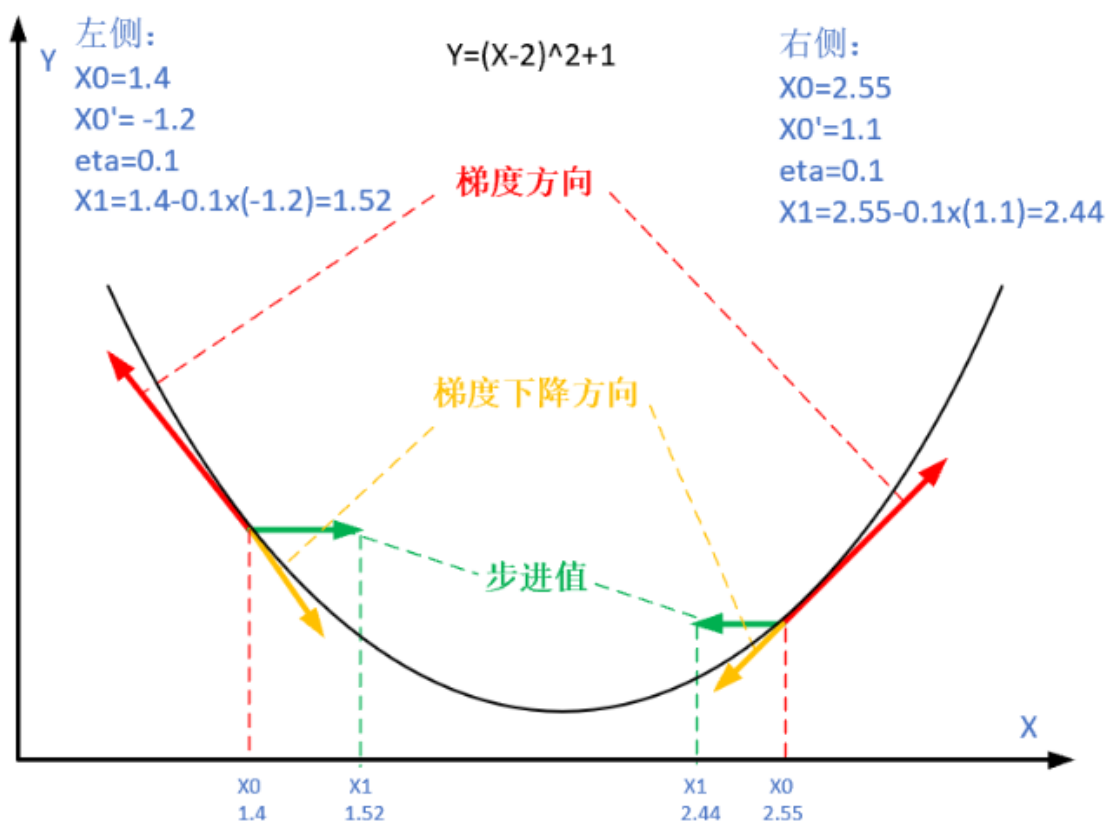
## 1.1 对于梯度下降的数学理解

梯度下降的数学公式：

$$\theta_{n+1} = \theta_n - \eta \cdot \nabla J(\theta) \quad (1)$$

其中：

- $\theta_{n+1}$ ：下一个值；
- $\theta_n$ ：当前值；
- $-$ ：减号，梯度的反向；
- $\eta$ ：学习率或步长，控制每一步走的距离，不要太快以免错过了最佳景点，不要太慢以免时间太长；
- $\nabla$ ：梯度，函数当前位置的最快上升点；
- $J(\theta)$ ：函数。

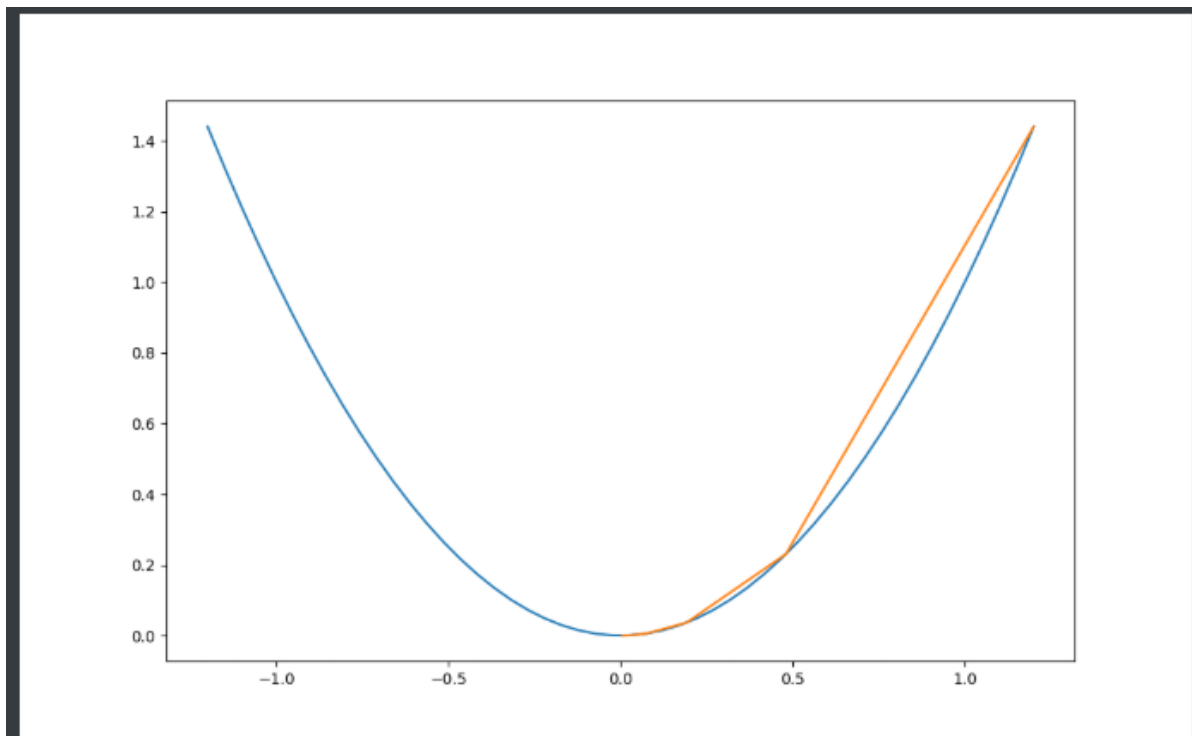


梯度下降的目的就是使得x值向极值点逼近。

## 1.2 单变量函数的梯度下降

假设一个单变量函数： $J(x) = x^2$

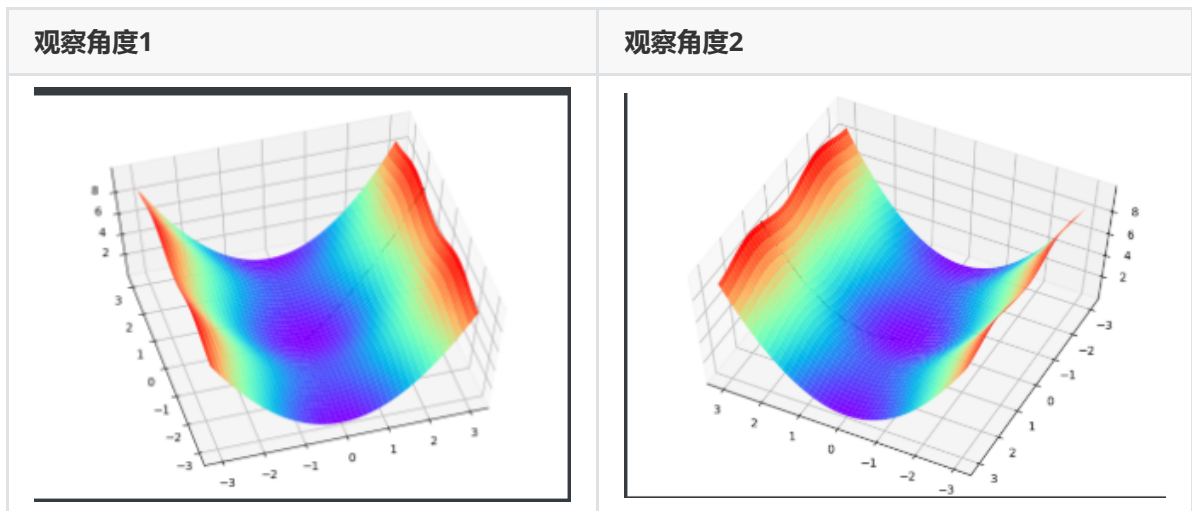
为求解该函数的最小值，得到如下图形：



### 1.3 双变量的梯度下降

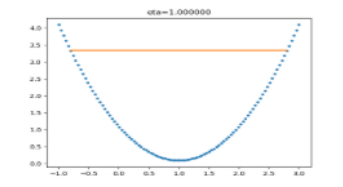
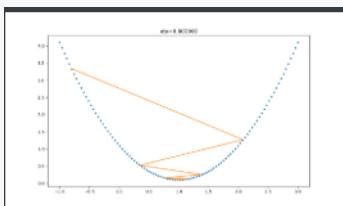
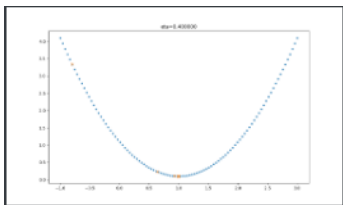
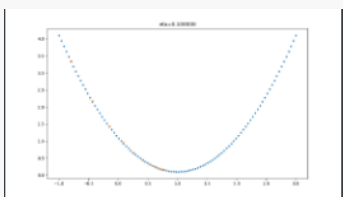
假设一个双变量函数： $J(x, y) = x^2 + \sin^2(y)$

注意看两张图中间那条隐隐的黑色线，表示梯度下降的过程



### 1.4 学习率的选择

学习率被表示为 $\eta$ 。在代码里，我们把学习率定义为 `learning_rate`，或者 `eta`

学 习 率	迭代路线图	说明
1.0		学习率太大，迭代的情况很糟糕，在一条水平线上跳来跳去，永远也不能下降。
0.8		学习率大，会有这种左右跳跃的情况发生，这不利于神经网络的训练。
0.4		学习率合适，损失值会从单侧下降，4步以后基本接近了理想值。
0.1		学习率较小，损失值会从单侧下降，但下降速度非常慢，10步了还没有到达理想状态。