梯度下降

从最小二乘法到梯度下降

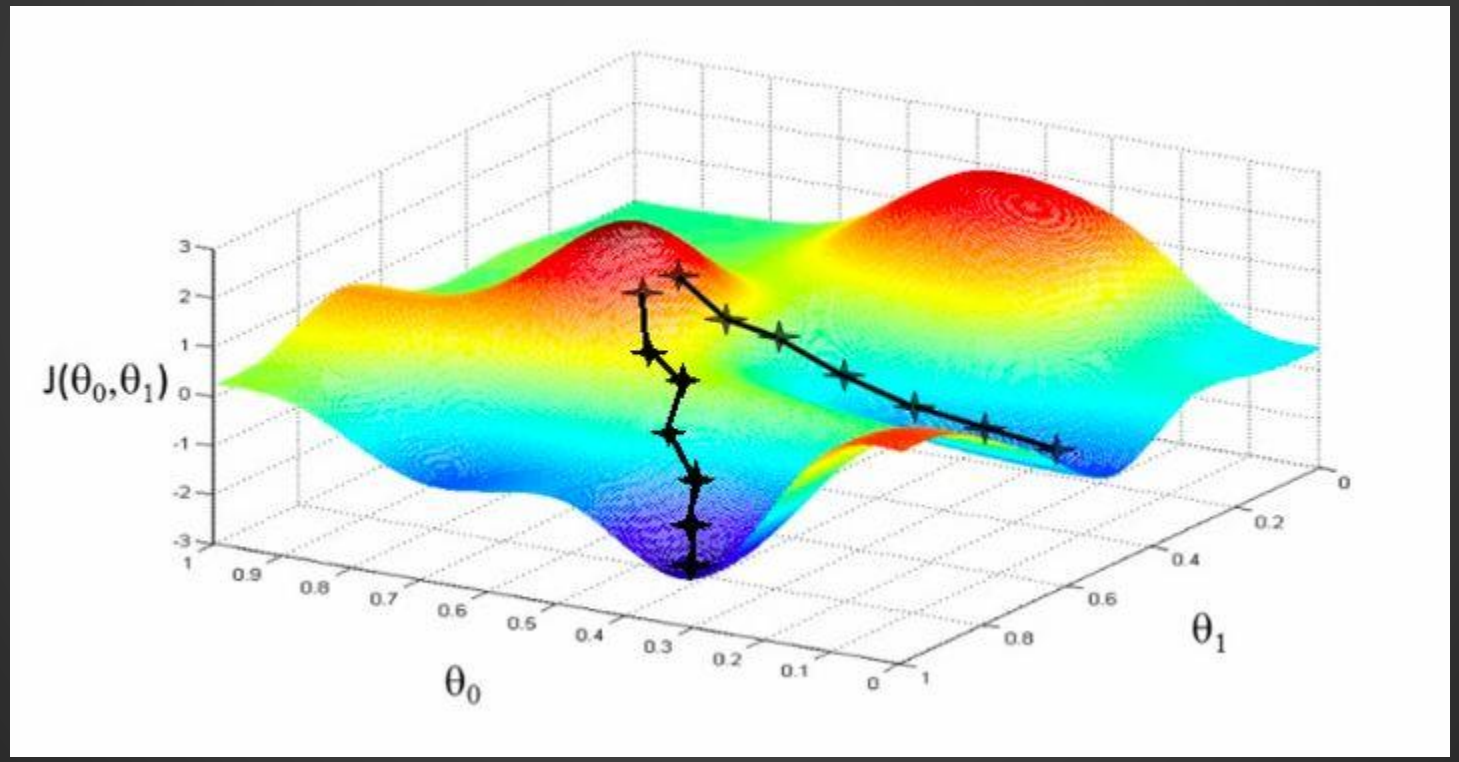
- 回顾前面的线性回归，我们采用最小二乘法，通过对参数求偏导数，列出方程组，解方程组，得到参数的值。这种通过方程直接计算出来的解被称为解析解。

- 在数据量不多，且样本的特征不多的情况下，可以直接用最小二乘法求最小值的解析解。但对于很多实际问题来说，数据量大、样本特征多，用最小二乘法求解，计算量会非常大，只能通过迭代的方式去求一个接近于实际最小值的点。

- 而梯度下降就是这种迭代式求最小值的方法中应用最为广泛的一种方法。

梯度下降法的基本思想

- 梯度下降法的基本思想可以类比为一个下山的过程。



下山类比梯度下降

下山	梯度下降
山	一个函数的图像
山上的某个位置	函数上的一个点
某个位置最陡峭的方向（山的海拔增加最快）	函数的梯度（函数值增加最快）
朝着陡峭的方向下降的方向	梯度的反方向，梯度下降的方向
山的最低点	函数的最小值

什么是梯度?

设函数z=f(x,y)在区域D上具有连续的一阶偏导数，则有梯度的计算公式：

$$\mathbf{grad}f(x,y)=\frac{\partial f}{\partial x}i+\frac{\partial f}{\partial y}j \qquad 或 \qquad \nabla f(x,y)=\frac{\partial f}{\partial x}i+\frac{\partial f}{\partial y}j$$

其中，i是沿x轴的单位向量(1,0)，j是沿y轴的单位向量(0,1)。

还可以将梯度写成如下形式：

$$\mathbf{grad}f(x,y)=(\frac{\partial f}{\partial x},\frac{\partial f}{\partial y})$$

梯度本质上是一个向量。

什么是梯度?

梯度的数学表达式:

$$\mathbf{grad}f(x,y) = (\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y})$$

- 梯度：函数增加（上升）最快的方向。沿着梯度的方向，可以找到这个函数的最大值。
- 在机器学习中，损失函数（或者代价函数）的值代表误差，误差肯定是越小越好，也就是要求损失函数的最小值。
- 梯度下降：函数减少（下降）最快的方向。沿着梯度下降的方向，就可以找到一个函数的最小值。

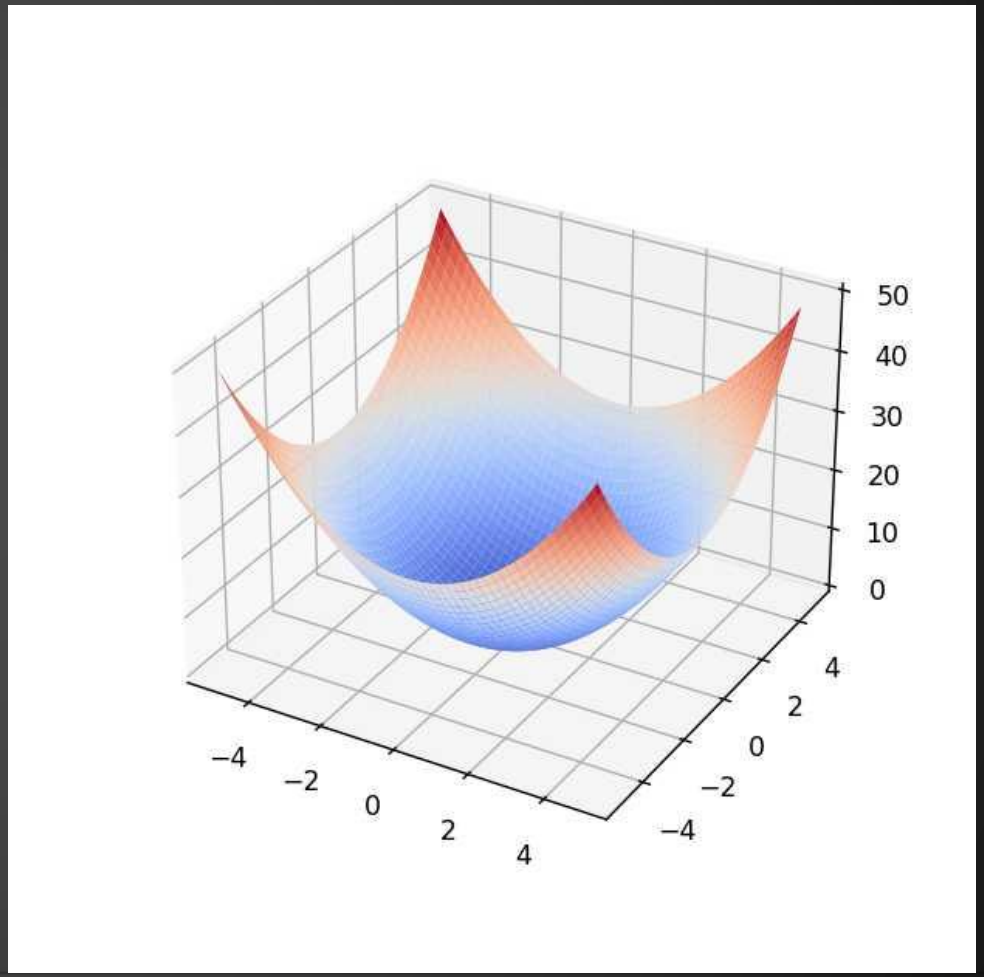
举个例子：二元函数的梯度

- 例如，给定一个二元函数，

$$f(x,y) = x^2 + y^2$$

- 该函数的梯度函数为：

$$\mathbf{grad} f(x,y) = \left(\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right) = (2x, 2y)$$



梯度下降基本步骤

1. 首先，我们有一个可导函数（损失函数），现在的目标就是找到这个函数的最小值，也就是山底。
2. 根据之前的场景假设，最快的下山的方式就是找到当前位置最陡峭的方向，然后沿着此方向向下走(梯度下降)，就能让函数值下降得最快！
3. 所以，我们重复利用这个方法，反复求取梯度，最后就能到山底，即获得函数的最小值，这就类似于我们下山的过程。

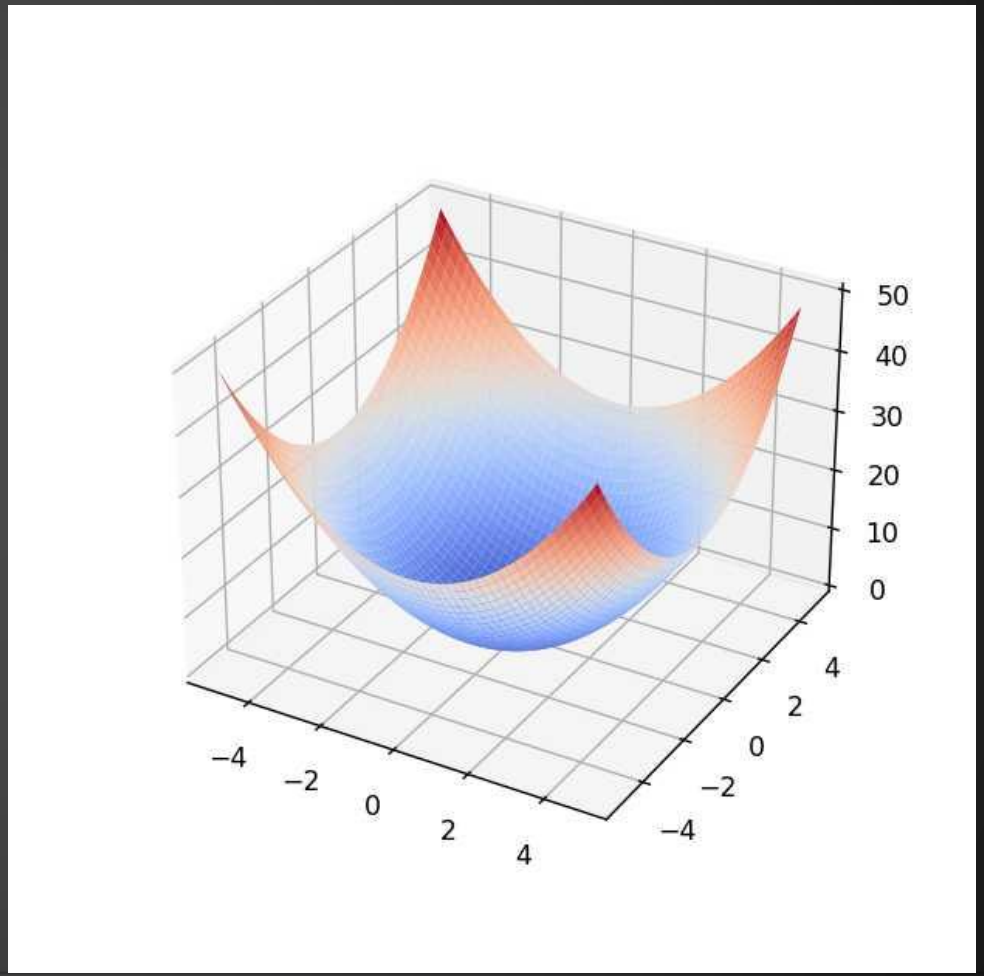
梯度下降实例：二元函数

- 给定目标函数（损失函数）：

$$f(x,y)=x^2+y^2$$

- 该函数的梯度函数为：

$$\mathbf{grad}f(x,y)=\left(\frac{\partial f}{\partial x},\frac{\partial f}{\partial y}\right)=(2x,2y)$$



梯度下降实例

- 给定一个初始点、学习率（也叫步长）
- 初始点：梯度下降开始的点；学习率：用来控制每一步下降的距离，每一步下降的距离为学习率 $\times$ $\mathbf{grad}\ f(x,y)$
- 假设初始点选为$X_0 = (2,4)$，学习率：0.01
- 在初始点，函数值$f(2,4) = 20$，梯度为：$\mathbf{grad}\ f(2,4) = (4,8)$

- 接着，沿着梯度下降的方向走。
- 第1步：$X_1 = X_0 - 0.01 \times \mathbf{grad}\ f(2,4) = X_0 - 0.01 \times (4,8) = (1.96,3.92)$
- 说明：以上负号（减号）表示梯度下降。
- 此时，函数值$f(1.96,3.92) = 19.21$

梯度下降实例

- 第2步：$X_2 = X_1 - 0.01 \times \mathbf{grad}\, f(1.96,3.92) = X_1 - 0.01 \times (3.92,7.84) = (1.92,3.84)$

此时，函数值$f(1.92,3.84) = 18.45$

- 第3步：$X_3 = X_2 - 0.01 \times \mathbf{grad}\, f(1.92,3.84) = X_2 - 0.01 \times (3.84,7.68) = (1.88,3.76)$
- 此时，函数值$f(1.88,3.76) = 17.72$

继续……

- 第100步：，此时，已经下降到了点$(0.27, 0.53)$，此时，函数值$f(0.27, 0.53) = 0.35$

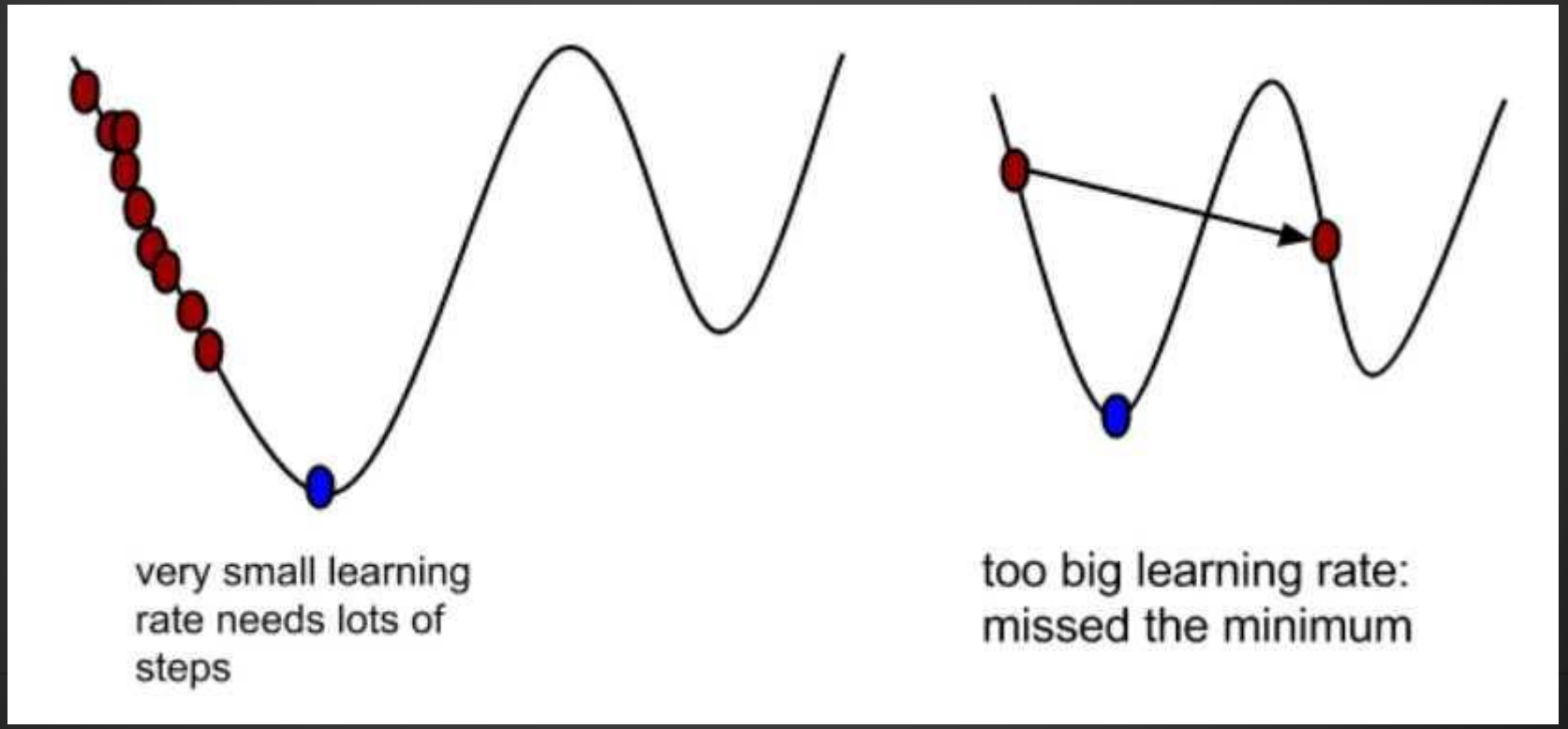
可以看到，当下降100步的时候，函数值已经很小了。

- 第200步：函数值为0.006，已经非常接近真实最小值0了，此时误差小于0.01。

这就是用梯度下降求解这个二元函数的一个基本过程。

关于学习率的说明

- 学习率对于下降的速度有着很大的影响，有人可能会说，把学习率设置得大一点，不就能很快下降到最小值了吗？俗话说，“步子太大，容易扯着蛋”，如果学习率过大，可能会错过最低点，见下图。



梯度下降：Python实现

1. 定义目标函数（损失函数）
2. 绘制函数图像
3. 定义梯度函数
4. 给定初始值、学习率，并进行3次梯度下降
5. 梯度下降：多次迭代
6. 绘制损失曲线图

梯度下降算法的一般形式

对于多元线性回归，回归方程为

$$h_w(x_1, x_2, ..., x_n) = w_1x_1 + w_2x_2 + ... + w_nx_n + b$$

为了方便表示，回归方程可以写成如下形式：

$h_w(x_1, x_2, ..., x_n, 1) = w_1x_1 + w_2x_2 + ... + w_nx_n + w_0x_0$，其中，$w_0 = b, x_0 = 1$

当然，也可以把这一项放到前面去，写成：

$h_w(x_0, x_1, x_2, ..., x_n) = w_0x_0 + w_1x_1 + w_2x_2 + ... + w_nx_n$，其中，$w_0 = b, x_0 = 1$

假设有m个样本，$(x_1^{(1)}, x_2^{(1)}, ...x_n^{(1)}, 1, y_1), (x_1^{(2)}, x_2^{(2)}, ...x_n^{(2)}, 1, y_2), ...(x_1^{(m)}, x_2^{(m)}, ...x_n^{(m)}, 1, y_m)$

可以把上面的多元线性回归的回归方程叫作**假设函数**，对于这个假设函数，写出它的损失函数：

$$l(w_0, w_1, ..., w_n) = \frac{1}{2m}\sum_{j=1}^{m}(h_w(x_1^{(j)}, x_1^{(j)}, \ldots, x_n^{(j)}, 1) - y_j)^2$$

说明：这里除以m表示平均损失，又除了一个2是为了方便后面的计算，并不影响对于目标函数的优化。

梯度下降算法的一般形式

损失函数：

$$l(w_0, w_1, ..., w_n) = \frac{1}{2m}\sum_{j=1}^{m}(h_w(x_1^{(j)}, x_1^{(j)}, \dots, x_n^{(j)}, 1) - y_j)^2$$

$$= \frac{1}{2m}\sum_{j=1}(w_0x_0^{(j)} + w_1x_1^{(j)} + w_2x_2^{(j)} + \cdots + w_nx_n^{(j)} - y_j)^2$$

求损失函数的梯度，对于每一个参数w_i，其梯度表达式为：

$$\frac{\partial}{\partial w_i}l(w_0, w_1, w_2, \cdots, w_n) = \frac{1}{m}\sum_{j=0}^{m}(h_w(x_0^{(j)}, x_1^{(j)}, ..., x_n^{(j)}) - y_j)x_i^{(j)}, i = 0, 1, \cdots, n$$

说明：

上式中，$w_0 = b$，$x_0^{(j)} = 1$

接下来,利用梯度下降算法求这个损失函数的最小值。

梯度下降算法的一般形式

给定初始值和学习率（步长），假设初始值：$w_0, w_1, w_2, ..., w_n$，学习率：α。

1. 对于每一个参数w_i，其梯度表达式为：
   $\frac{\partial}{\partial w_i} l(w_0, w_1, w_2, \cdots, w_n) = \frac{1}{m} \sum_{j=0}^{m} (h_w(x_0^{(j)}, x_1^{(j)}, ..., x_n^{(j)}) - y_j) x_i^{(j)}, i = 0, 1, \cdots, n$
2. 用步长乘以损失函数的梯度就是当前位置下降的距离
   $\alpha \times \frac{\partial}{\partial w_i} l(w_0, w_1, w_2, ..., w_n), i = 0, 1, \cdots, n$
3. 更新参数，对于参数w_i，其更新表达式为：
   $w_i = w_i - \alpha \frac{\partial}{\partial w_i} l(w_0, w_1, w_2, ..., w_n) = w_i - \alpha \frac{1}{m} \sum_{j=0}^{m} (h_w(x_0^{(j)}, x_1^{(j)}, ..., x_n^{(j)}) - y_j) x_i^{(j)}$
   其中，$i = 0, 1, \cdots, n$
4. 确定是否所有的参数w_i梯度下降的距离都小于ϵ，如果小于ϵ则算法终止，否则进入步骤3。当然也可以设置一个最大迭代次数作为终止迭代的条件。

梯度下降算法家族成员

1、批量梯度下降

- 批量梯度下降法是梯度下降法最常用的形式，具体做法也就是在更新参数时使用所有的样本来进行更新，这个方法对应于前面说的线性回归的梯度下降算法。

$$w_i = w_i - \alpha \frac{1}{m} \sum_{j=0}^{m} (h_w(x_0^{(j)}, x_1^{(j)}, ..., x_n^{(j)}) - y_j) x_i^{(j)}$$

2、随机梯度下降

- 随机梯度下降法和批量梯度下降法原理类似，区别在于求梯度时没有用所有的m个样本的数据，而是选取一个样本来求梯度，对应的参数更新公式为：

$$w_i = w_i - \alpha (h_w(x_0^{(j)}, x_1^{(j)}, ..., x_n^{(j)}) - y_j) x_i^{(j)}$$

- 随机梯度下降法和批量梯度下降法是两个极端，一个采用所有数据来梯度下降，另一个用一个样本来梯度下降。自然各自的优缺点都非常突出。

梯度下降算法家族成员

3、小批量梯度下降

- 小批量梯度下降法是批量梯度下降法和随机梯度下降法的折衷，也就是对于m个样本，采用其中的一部分，例如选取x个样本来迭代，1<x<m。一般可以取x=10，当然根据样本的数据，可以调整这个x的值。对应的参数更新公式是：

$$w_i = w_i - \alpha \sum_{j=t}^{t+x-1} (h_w(x_0^{(j)}, x_1^{(j)}, ..., x_n^{(j)}) - y_j) x_i^{(j)}$$