Gradient Descent Methods

FrankZhou-jun*

2019 年 11 月 26 日

梯度下降法非常重要，下面讲一的原理，最后使用代码实现，首先从单一变量的梯度下降法计算，然后扩展到多变量，最后使用线性回归的例子来表达梯度下降法的应用。

梯度下降法，顾名思义就是初始值在某一个点，然后沿着梯度方向，快速下降，到达最低点。如果是沿着梯度上升方向，就会达到最大值。举一个有关梯度下降法的例子

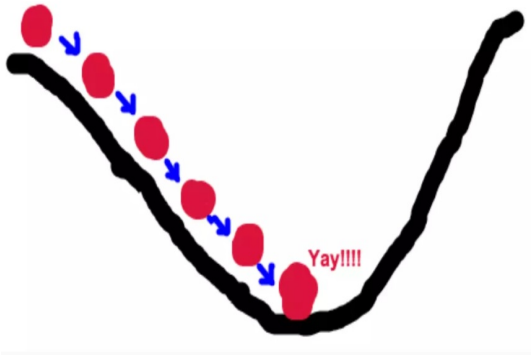


图 1: 石头滑坡

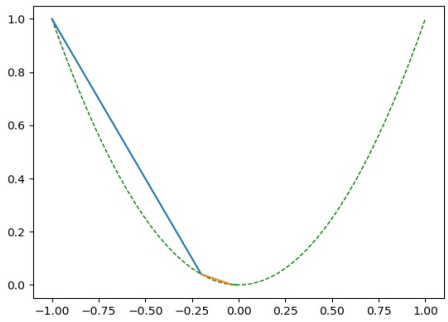


图 2: 数学模型

如图中所示的例子，石头初始在一个位置，然后滚落至山谷位置，为啥会向下滚，受重力影响，石头沿着斜坡相切的方向滚，这个斜坡相切的方向其实就是梯度方向，对斜面的曲线求一阶导函数，就可以该位置的梯度了。

当函数为最均方误差函数，求最小值过程中，就是在求估计值和目标值的误差最小的情况，从而可以进行参数优化，应用在线性回归中。

1 单变量梯度下降法

下面就开始讲解单变量的梯度了，假设斜坡的曲线函数为：

$$\mathbf{J}(\theta) = \theta^2$$

*研究方向：信号处理，机械故障诊断，深度学习，强化学习，邮箱:zhoujun14@yeah.net

由于其变量只有一个，只对其中一个变量求偏导，也就是求导，得到梯度为：

$$\nabla \mathbf{J}(\theta) = 2\theta$$

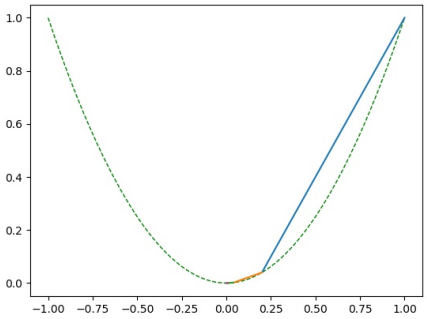
假设初始值在 $\theta_0 = 1$，学习率为 $\alpha = 0.4$，得到迭代公式为：

$$\theta_{k+1} = \theta_k - \alpha \nabla \mathbf{J}(\theta_k), k = 0, 1, 2, 3...$$

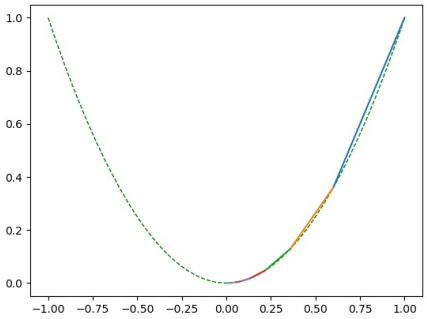
如下图所示：

表 1: 迭代过程 $\alpha = 0.4$

k	θ_k
0	1
1	0.19999999999999996
2	0.03999999999999998
3	0.007999999999999993
4	0.0015999999999999981
5	0.00031999999999999954
6	6.399999999999988e-05
7	1.2799999999999972e-05
8	2.559999999999994e-06
9	5.119999999999987e-07



当学习率为 0.2 时，如下图所示：



可以看到学习率的大小直接影响学习的快慢。一般学习率理解为步长，步长太小，容易导致训练时间加强，但步长过大，容易偏离轨道错过最优解。

单变量梯度下降法至此讲解完毕，下面进入多变量讲解。

表 2: 迭代过程 $\alpha = 0.2$

k	θ_k
0	1
1	0.6
2	0.36
3	0.216
4	0.1296
5	0.07776
6	0.046655999999999996
7	0.027993599999999997
8	0.016796159999999997
9	0.010077695999999997

2 多变量梯度下降法

当存在多变量是，求梯度过程就是对函数的每一个变量求偏导, 假设函数为：

$$\mathbf{J}(\theta_0, \theta_1, ..., \theta_n) = \theta_0^2 + \theta_1^2 + ... + \theta_n^2$$

则梯度为

$$\nabla \mathbf{J} = < \frac{\partial \mathbf{J}}{\partial \theta_0}, \frac{\partial \mathbf{J}}{\partial \theta_1}, ..., \frac{\partial \mathbf{J}}{\partial \theta_n} > = < 2\theta_0, 2\theta_1, ..., 2\theta_n >$$

假设目标函数为：

$$\mathbf{J}(\theta_0, \theta_1) = \theta_0^2 + \theta_1^2$$

可以啊看到最小值为 (0,0), 梯度为：

$$\nabla \mathbf{J}(\theta_0, \theta_1) = (2\theta_0, 2\theta_1)$$

迭代公式为：

$$\Theta_{k+1} = \Theta_k - \alpha \nabla \mathbf{J}(\Theta)$$

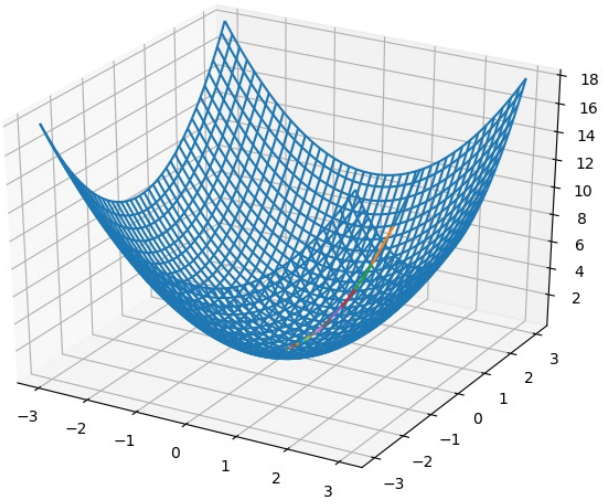
即：

$$(\theta_0^{k+1}, \theta_1^{k+1}) = (\theta_0^k, \theta_1^k) - \alpha(2\theta_0^k, 2\theta_1^k)$$

初始值为 $(\theta_0, \theta_1) = (1, 3)$, 学习率 $\alpha = 0.1$, 可得：

表 3: 迭代过程，$\alpha = 0.1$

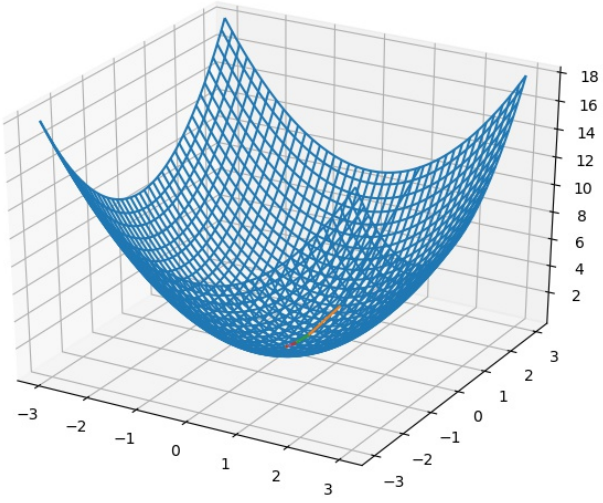
k	(θ_0, θ_1)
0	[1, 3]
1	[0.8 2.4]
2	[0.64 1.92]
3	[0.512 1.536]
4	[0.4096 1.2288]
5	[0.32768 0.98304]
6	[0.262144 0.786432]
7	[0.2097152 0.6291456]
8	[0.16777216 0.50331648]
9	[0.13421773 0.40265318]
10	[0.10737418 0.32212255]
11	[0.08589935 0.25769804]
12	[0.06871948 0.20615843]
13	[0.05497558 0.16492674]
14	[0.04398047 0.1319414]
15	[0.03518437 0.10555312]
16	[0.0281475 0.08444249]
17	[0.022518 0.06755399]
18	[0.0180144 0.0540432]
19	[0.01441152 0.04323456]
20	[0.01152922 0.03458765]
21	[0.00922337 0.02767012]
22	[0.0073787 0.02213609]
23	[0.00590296 0.01770887]
24	[0.00472237 0.0141671]
25	[0.00377789 0.01133368]
26	[0.00302231 0.00906694]
27	[0.00241785 0.00725355]
28	[0.00193428 0.00580284]
29	[0.00154743 0.00464228]
30	[0.00123794 0.00371382]
31	[0.00099035 0.00297106]
32	[0.00079228 0.00237684]
33	[0.00063383 0.00190148]
34	[0.00050706 0.00152118]
35	[0.00040565 0.00121694]
36	[0.00032452 0.00097356]
37	[0.00025961 0.00077884]
38	[0.00020769 0.00062308]
39	[0.00016615 0.00049846]
40	[0.00013292 0.00039877]



但学习率变为 0.3 时：

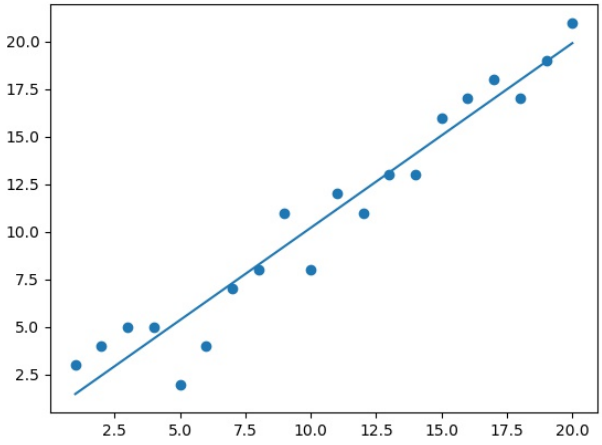
表 4: 迭代过程，$\alpha = 0.3$

k	(θ_0, θ_1)
0	[1, 3]
1	[0.4 1.2]
2	[0.16 0.48]
3	[0.064 0.192]
4	[0.0256 0.0768]
5	[0.01024 0.03072]
6	[0.004096 0.012288]
7	[0.0016384 0.0049152]
8	[0.00065536 0.00196608]
9	[0.00026214 0.00078643]
10	[0.00010486 0.00031457]



3 梯度下降法在线性回归中的应用

从上面的例子可以看出，通过梯度下降法可以较好的得到函数最小值的解，好像和初始值的设定没有关系，最终都可以通过梯度下降方法得到最小值的解，但是在神经网络中，使用初始值的设定有时候影响很大，所以这里有待分析。



假设需要使用梯度下降法，求解函数的最小值的解，构造出均方误差代价函数：

$$\mathbf{J}(\Theta) = \frac{1}{2m}\sum_{i=1}^{m}(\hat{h}_\Theta(x_i) - y_i)^2$$

我们要求的是当均方误差值最小时，参数 Θ 的值，设预测函数为一次线性函数 (有可能是多次)：

$$\hat{h}_\Theta(x_i) = \hat{\theta}_0 + \hat{\theta}_1 * x_i$$

这里有两个变量 θ_0, θ_1，则他的梯度为

$$\nabla\mathbf{J}(\Theta) =< \frac{1}{2}\sum_{i=1}^{m}(\hat{h}_\Theta(x_i) - y_i), \frac{1}{2}\sum_{i=1}^{m}(\hat{h}_\Theta(x_i) - y_i)x_i >$$

使用矩阵形式计算

$$\hat{h}_\Theta(x_i) = \hat{\theta}_0 + \hat{\theta}_1 * x_i = \underbrace{\begin{bmatrix}1 & x_i\end{bmatrix}}_{\mathbf{X}}\underbrace{\begin{bmatrix}\hat{\theta}_0 \\ \hat{\theta}_1\end{bmatrix}}_{\Theta} = \mathbf{X}\Theta$$

均方误差计算：

$$\mathbf{J}(\Theta) = \frac{1}{2m}\sum_{i=1}^{m}(\mathbf{X}\Theta - \vec{y})^T(\mathbf{X}\Theta - \vec{y})$$

梯度

$$\nabla\mathbf{J}(\Theta) = \frac{1}{m}\mathbf{X}^T(\mathbf{X}\Theta - \vec{y})$$

迭代公式为：

$$\Theta_{k+1} = \Theta_k - \alpha\nabla\mathbf{J}(\Theta)$$