# Regression

Prevalenter

2020 年 2 月 2 日

## 1 简介

本文主要介绍了李宏毅老师的机器学习课程中的 Demo1 的例子，对 Gradient Descent 与 Adagrad 进行了简单的公式介绍，并对两种方式进行了编程仿真，其仿真源码对于代码文件的 Regrassion 中。

## 2 公式推导

### 2.1 定义模型

李宏毅老师在课程中定义的模型为线性的，如公式 1所示，其中 w 代表权重，b 代表偏置值，x 代表输入值，y 代表输出值。

$$y = b + w * x \tag{1}$$

当写出矩阵的形式，如公式 2所示。

$$y = \begin{pmatrix} w & b \end{pmatrix} \begin{pmatrix} x \\ 1 \end{pmatrix} \tag{2}$$

### 2.2 定义损失函数

直接通过计算预测得到的直线与标记后的数据的距离，来定义损失函数，公式如 3所示。

$$\begin{aligned} L(f) &= L(w, b) \\ &= \sum_{n=1}^{10} (\hat{y} - (b + w * x^n))^2 \end{aligned} \tag{3}$$

## 2.3 迭代方法

当我们通过公式定义了模型，通过公式定义了损失函数后，实际上就可以通过迭代的方法寻找 $f^*$，即 $w^*$ 与 $b^*$, 可用公式表达。

$$f^* = \arg\min(f) \tag{4}$$

$$\begin{aligned} w^*, b^* &= \arg\min_{w,b} L(w,b) \\ &= \arg\min_{w,b} (\hat{y} - (b + w * x^n))^2 \end{aligned} \tag{5}$$

但是选择正确迭代方法，对于能否找到合适的 w 与 b 值是至关重要的，视频中介绍的方法主要有 Gradient Descent 与 Adagrad 两种方法。

### 2.3.1 Gradient Descent

Gradient Descent 又称梯度下降，是最基础的一种迭代方法。其大致思路为：随机或人工设定 $w$ 与 $b$ 的初始参数 $w_0$ 与 $b_0$，再通过公式 6进行迭代。其中 $\eta$ 为学习率，在迭代中我们可以对参数 w 与 b 设置不同的学习率。

$$\begin{cases} w^{i+1} = w^i - \eta \frac{\mathrm{d}L}{\mathrm{d}w}|_{w=w^i} \\ b^{i+1} = b^i - \eta \frac{\mathrm{d}L}{\mathrm{d}b}|_{b=b^i} \\ \frac{\mathrm{d}L}{\mathrm{d}w} = \sum_{n=1}^{10} 2(\hat{y} - (b + w * x^n))(-x^n) \\ \frac{\mathrm{d}L}{\mathrm{d}b} = \sum_{n=1}^{10} -2(\hat{y} - (b + w * x^n)) \end{cases} \tag{6}$$

### 2.3.2 Adagrad

在 Gradient Descent 中的学习率 $\eta$ 是恒定不变的，是根据人的经验去选取，这可能会导致由于参数学习率 $\eta$ 的选取不合适从而导致无法得到满意的收敛结果。Adagrad 就是一种自适应的迭代方法，可以根据之前的梯度自适应一个学习率 $\eta^i$。

$$w^{t+1} = w^t - \frac{\eta^t}{\sigma^t} g^t \tag{7}$$

$$\eta^t = \frac{\eta}{\sqrt{t+1}} \tag{8}$$

$$\sigma^t = \sqrt{\frac{1}{1+t} \sum_{i=0}^{t} (g^i)^2} \tag{9}$$

将公式 8与公式 9代入公式 7可以得到公式 10, 其中 $i$ 代表第 i 个迭代回合的梯度，与 Gradient Descent 相比较，可知 Adagrad 的学习率除以了之前迭代回合的和，从而达到自适应学习率的效果。

$$w^{t+1} = w^t - \frac{\eta}{\sum_{i=0}^{t}(g^i)^2} g^t \tag{10}$$

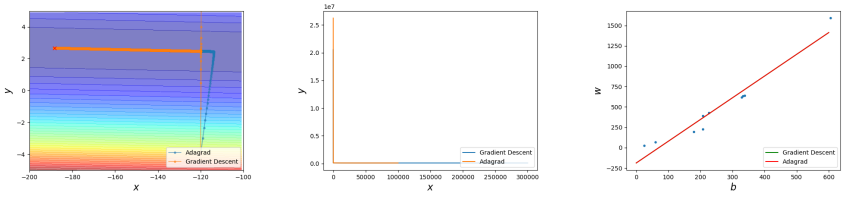
# 3 运行结果

通过视频中的得到的数据如表 1所示。

表 1: 测试数据

| x | 33 | 333 | 328 | 207 | 226 | 25 | 179 | 60 | 208 | 606 |
|---|---|---|---|---|---|---|---|---|---|---|
| y | 640 | 633 | 619 | 393 | 428 | 27 | 93 | 66 | 226 | 1591 |

通过 Gradient Descent 与 Adagrad 两种方式，我们可以得到的结果如图 1所示。通过图 1(a)可以看出两种方法都最终将参数 $w$ 与 $b$ 收敛至同一点，回归的曲线如图 1(c)所示，由于参数参数 $w$ 与 $b$ 几乎是相等，所以曲线重叠。两种方式迭代过程的损失如图 1(b)所示，并且我们可以发现其实损失函数在最开始的几个迭代回合下降的比较快，后面的则十分缓慢，这与图 1(a)的地形图在后面区域较平坦是一致的。




(a) 参数迭代图　(b) 损失迭代　(c) 回归后结果曲线

图 1: 两种方式迭代后结果

在图 1(a)我们还可以看到，使用 Gradient Descent 存在着振荡的现象，但是使用 Adagrad 的不存在震荡现象，但是 Adagrad 的梯度下降速率在后面的迭代回合几乎趋近于零,所以使用 Adagrad 方式能以较快速度和较好的准确性趋近于最优解，但是要想无限趋近于最优解，相比 Descent Gradient 需要较多的迭代回合。