Guide of Python

内容概要： 数学建模算法

创建时间： 2022/4/7 13:41　　**更新时间：** 2022/4/17 15:27

作者： **TwinkelStar**

梯度下降算法
Gradient Descent Algorithm

1、操作系统相关环境

1）硬件环境：

- 电脑

2）软件环境：

- Python3.7(向下兼容 Python3)(程序设计语言)
- Numpy1.19.5(兼容大部分版本)(科学计算库)

3）操作系统(2 选 1)：

- Windows7
- Windows10
- Windows11

2、梯度下降

梯度下降是迭代法的一种，可以用于求解最小二乘问题(线性和非线性的问题都可以求解)。在求解机器学习算法的模型参数，即无约束优化问题时，梯度下降（Gradient Descent）是最常采用的方法之一，另一种常用的方法是最小二乘法。在求解损失函数的最小值时，可以通过梯度下降法来一步步的迭代求解，得到最小化的损失函数和

模型参数值。反过来，如果我们需要求解损失函数的最大值，这时就需要用梯度上升法来迭代了。在机器学习中，基于基本的梯度下降法发展了两种梯度下降方法，分别为随机梯度下降法和批量梯度下降法。

1）应用场景

梯度下降法是机器学习中一种常用到的算法，但其本身不是机器学习算法，而是一种求解的最优化算法。主要解决求最小值问题，其基本思想在于不断地逼近最优点，每一步优化方向就是梯度的方向。

机器学习的本质就是“喂”给模型数据，让模型不断地去学习，而这个学习的过程就是利用梯度下降法不断去优化的过程，目前最为常见的深度神经网络便是利用梯度的反向传播，反复更新模型参数直至收敛，从而达到优化模型的目的。

3、算法原理

对于最简单线性模型，如（1）：

$$h(\theta) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \ldots + \theta_i x_i \qquad (1)$$

我们假设其损失函数为（2）：

$$J(\theta) = \frac{1}{2}[h_t(x) - y]^2 \qquad (2)$$

那么梯度下降的基本形式为（3）：

$$\theta_{n+1} = \theta_n - \alpha J^{'}(\theta) \qquad (3)$$

其中，α为学习率（learning_rate）。下一步便是要将损失函数最小化，需要对函数求导：

$$J^{'}(\theta) = \frac{\partial J(\theta)}{\theta} = [h_\theta(x) - y] * h_\theta^{'} \qquad (4)$$

求解可知：

$$J^{'}(\theta) = [h_\theta(x) - y] * x \qquad (5)$$

即：

$$\theta_{n+1}=\theta_n-\alpha[h_\theta(x^{(i)})-y^{(i)}]*x^{(i)} \tag{6}$$

常见的批量梯度下降法(Batch Gradient Descent, BGD) 、随机梯度下降法(Stochastic Gradient Descent, SGD)、小批量梯度下降法(Mini-batch Gradient Descent, MBGD)等，被广泛的应用在线性回归（非线性）、神经网络等场景。

在使用梯度下降法时，需要选择合适的步长以及迭代次数，如果选择的步长过短，不仅会使得迭代的时间过长，还有可能让模型陷入局部最小值，无法跳出去。如果设置步长过长，模型无法找到最优解，损失函数无法下降。下面对各种梯度下降法进行介绍。

1） 批量梯度下降法

使用所有样本在当前点的梯度值来对变量参数进行更新操作，其更新公式为：

$$\theta^{k+1}=\theta^k-\alpha\sum_{i=1}^{m}\nabla f_{\theta^k}(x^i) \tag{7}$$

批量梯度下降，顾名思义就是将所有样本一次性喂入公式进行更新操作，迭代速度快。

2） 随机梯度下降法

在更新变量参数的时候，不再像批量梯度下降一样，将所有的样本喂入公式迭代，而是随机选取一个样本的梯度值来更新参数，更新公式为：

$$\theta^{k+1}=\theta^k-\alpha\nabla f_{\theta^k}(x^i) \tag{8}$$

3） 小批量梯度下降法

小批量梯度下降法结合了 BGD 和 SGD 的特性，从原始数据中，每次选择 n 个样本来更新参数值，更新公式为：

$$\theta^{k+1}=\theta^k-\alpha\sum_{i=t}^{t+n-1}\nabla f_{\theta^k}(x^i) \tag{9}$$

4、例题

1）拟合函数$y = 3x_1 + 4x_2$，系数矩阵为[3,4]。

2）解题步骤

- 初始化样本集；
- 设置步长以及迭代次数；
- 根据公式（6）的推导更新权值；
- 计算损失函数；
- 得出结果。

3）程序设计

导入相关依赖包，如图 1 所示：

```
#导入依赖包
import numpy as np
import matplotlib.pyplot as plt
```

图 1.导入相关依赖
fig1.import package

初始化样本集合，即 x1，x2 的样本矩阵、权值矩阵[3,4]，函数 y，x1，x2 为一组等差数列，与权值矩阵相乘后得到函数 y，作为梯度下降的真实值，如图 2 所示：

```
#初始化样本集
sample_num=100
x1 = np.linspace(0, 9, sample_num)
x2 = np.linspace(4, 13, sample_num)
x = np.concatenate(([x1], [x2]), axis=0).T
y = np.dot(x, np.array([3, 4]).T)
```

图 2.初始化参数
fig2.init parameter

根据公式（7）的推导，撰写代码，具体代码例程运行 BGD.py 文件，核心程序如图 3 所示：

```
#迭代更新
while abs(loss) > 0.0001 and iter_count < max_iter_count:
    w[0] -= step_size * np.sum((w[0] * x1 + w[1] * x2 - y) * x1) / x.shape[0]
    w[1] -= step_size * np.sum((w[0] * x1 + w[1] * x2 - y) * x2) / x.shape[0]
    loss = np.sum(w[0] * x1 + w[1] * x2 - y)
    iter_count += 1
    print("iter_count:%d    the loss:%f" % (iter_count, loss))

    loos.append(abs(loss))
    plt.plot(loos, lw=3, label='loss')
    plt.legend()
    plt.show()
```

图 3.梯度下降
fig3. gradient descent

运行结果如图 4 所示：

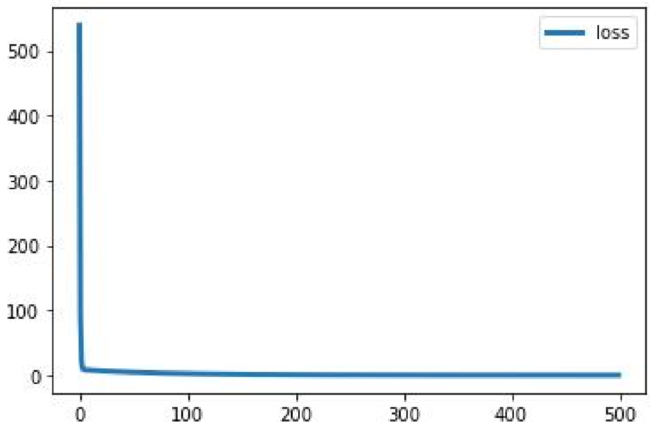

图 3.BGD 迭代 500 次结果
fig3. BGD results of 500 iterations

经过 500 次的迭代，loss 值从-21.205183 下降到-0.012776，拟合的权值矩阵最终结果为[3.00035, 3.9998]，拟合数据逼近初始化的权值矩阵[3,4]。更多实例参考 example.py 文件。