PyTorch

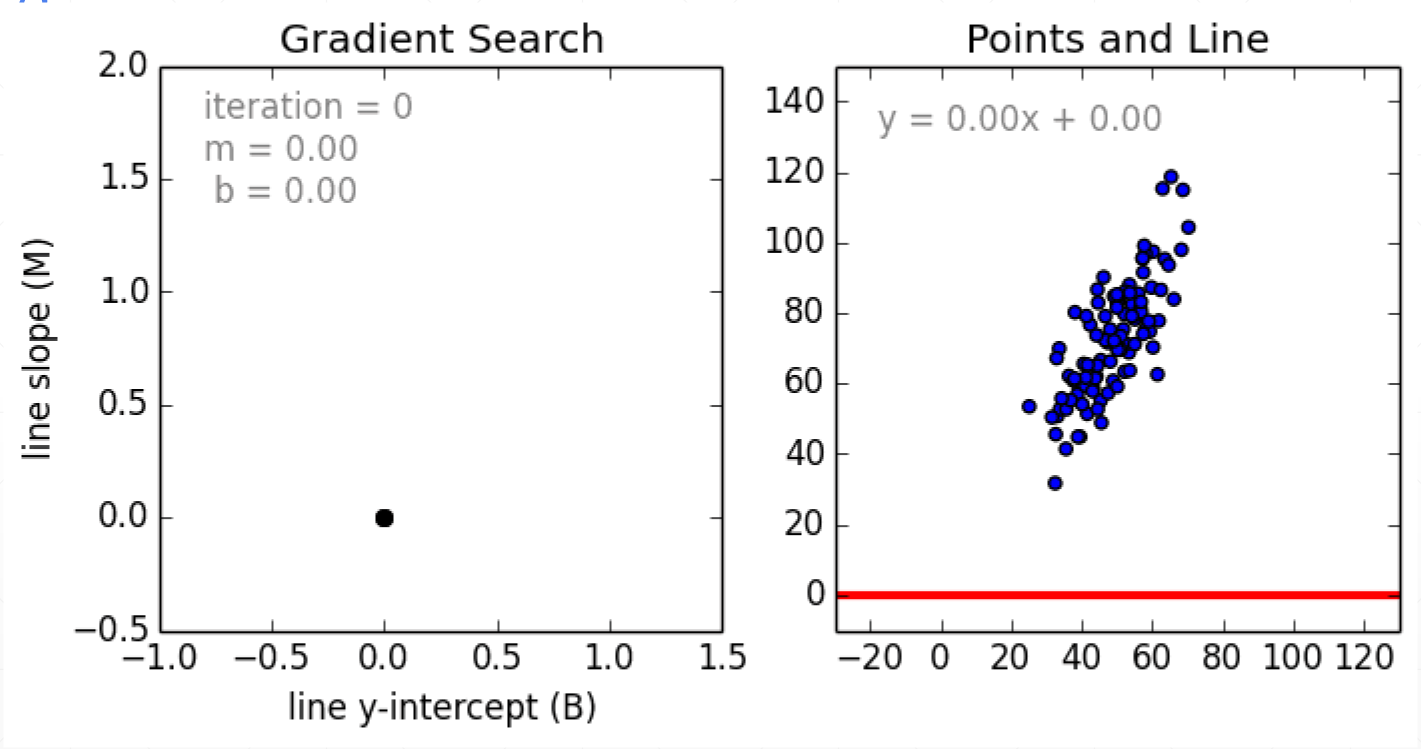
你好，梯度-II

主讲人：龙良曲

Recap

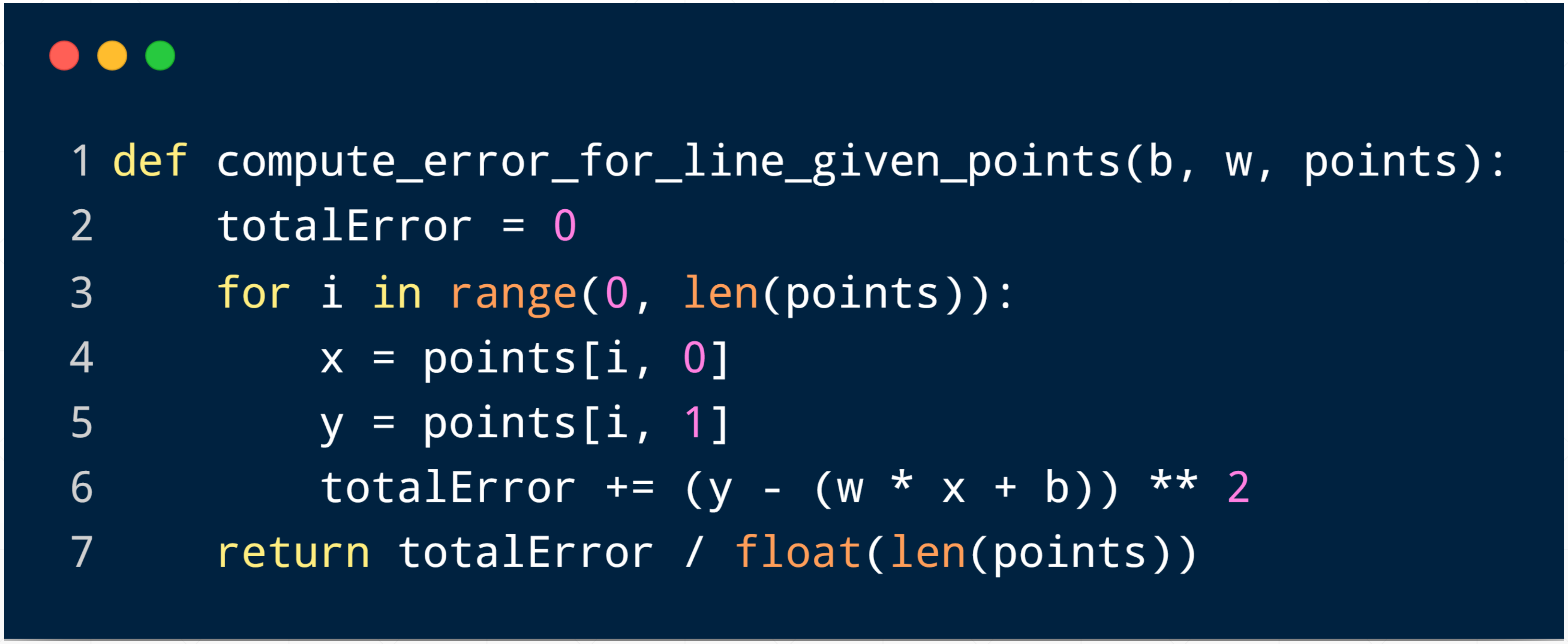
- Linear regression

背景：有一堆数据
目的：找到最能够表示这堆数据的模型
模型构建:
导入数据
预测模型
训练模型
　设置损失
　计算梯度
　梯度迭代
应用模型



$$\boldsymbol{loss = (WX + b - y)^2}$$

$$loss = \sum_i (w * x_i + b - y_i)^2$$



```
def compute_error_for_line_given_points(b, w, points):
    totalError = 0
    for i in range(0, len(points)):
        x = points[i, 0]
        y = points[i, 1]
        totalError += (y - (w * x + b)) ** 2
    return totalError / float(len(points))
```

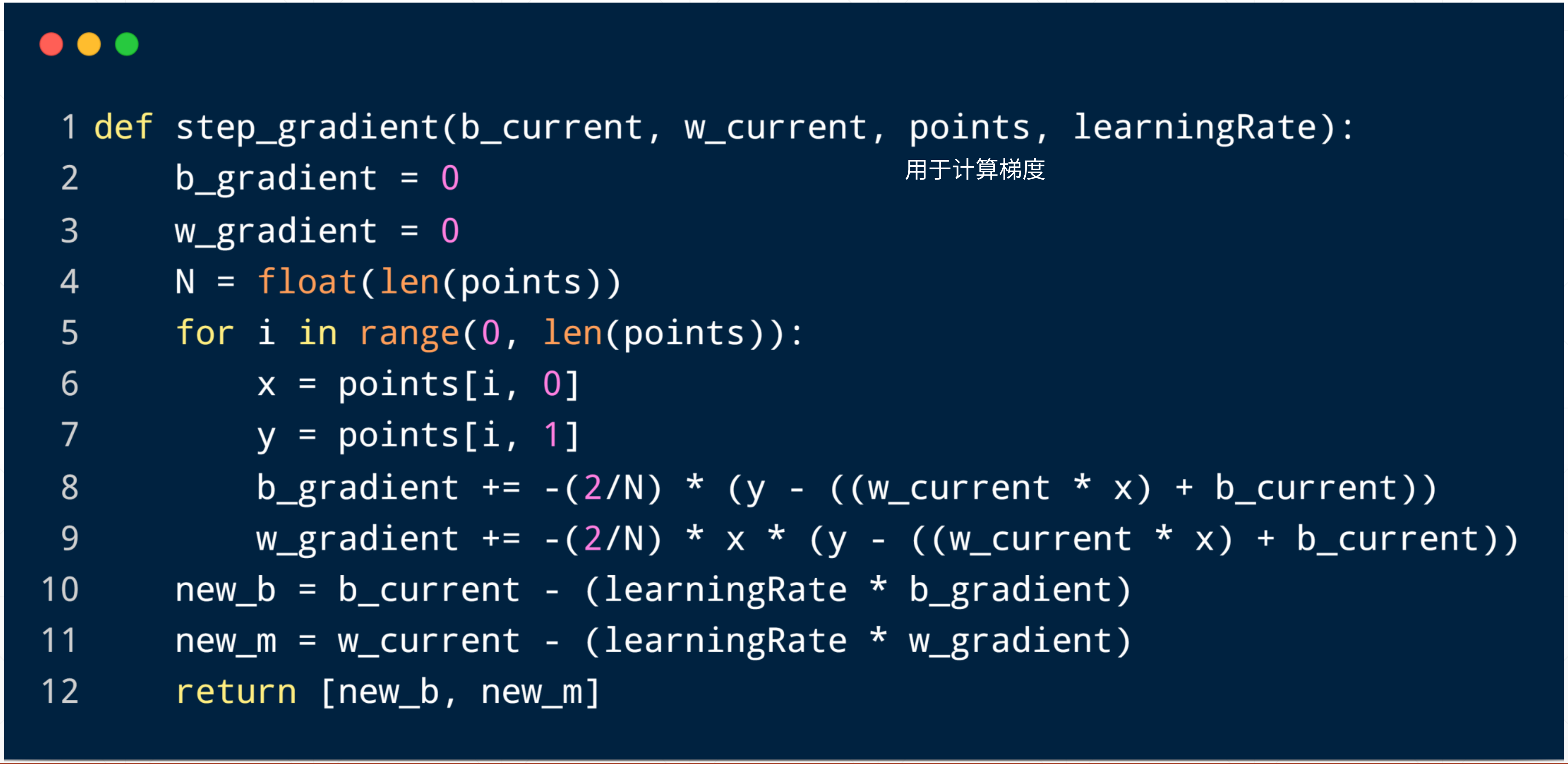
定义返回值

计算损失

返回结果

$$w' = w - lr * \frac{\nabla loss}{\nabla w}$$

$$loss = (WX + b - y)^2$$



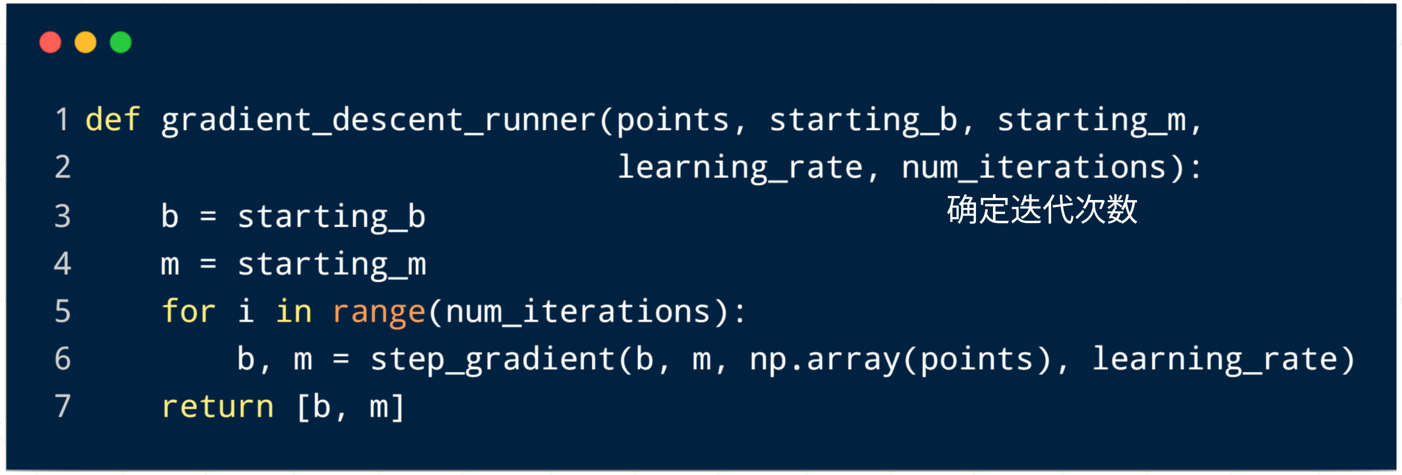
```
def step_gradient(b_current, w_current, points, learningRate):
    b_gradient = 0
    w_gradient = 0
    N = float(len(points))
    for i in range(0, len(points)):
        x = points[i, 0]
        y = points[i, 1]
        b_gradient += -(2/N) * (y - ((w_current * x) + b_current))
        w_gradient += -(2/N) * x * (y - ((w_current * x) + b_current))
    new_b = b_current - (learningRate * b_gradient)
    new_m = w_current - (learningRate * w_gradient)
    return [new_b, new_m]
```

定义中间变量

计算梯度

单次梯度迭代

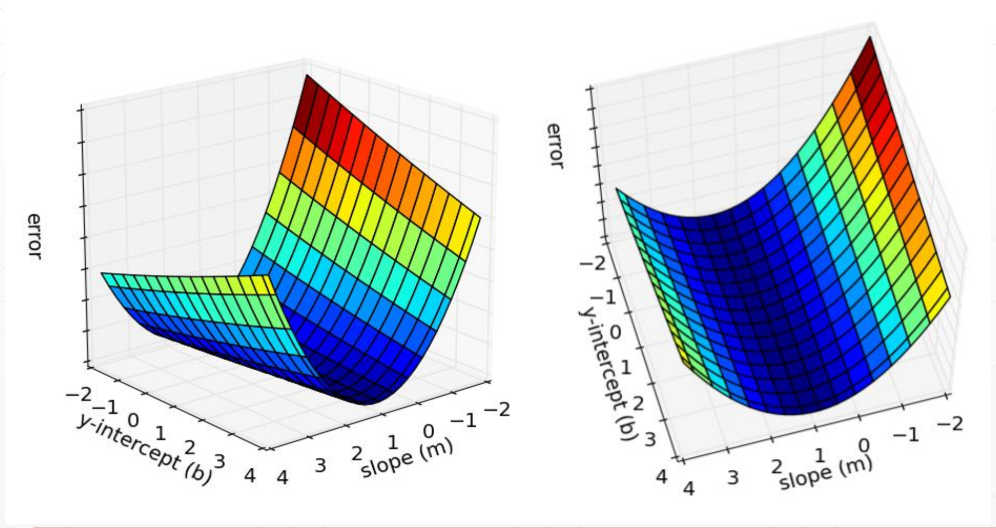
Iterate to optimize

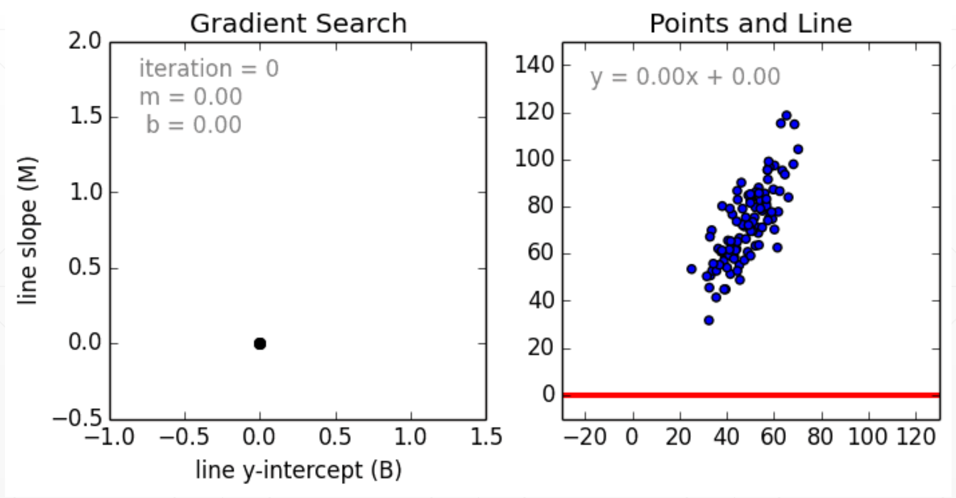


```python
def gradient_descent_runner(points, starting_b, starting_m,
                            learning_rate, num_iterations):
    b = starting_b
    m = starting_m
    for i in range(num_iterations):
        b, m = step_gradient(b, m, np.array(points), learning_rate)
    return [b, m]
```

确定迭代次数

多次梯度迭代





下一课时

Hello, MNIST

Thank You.