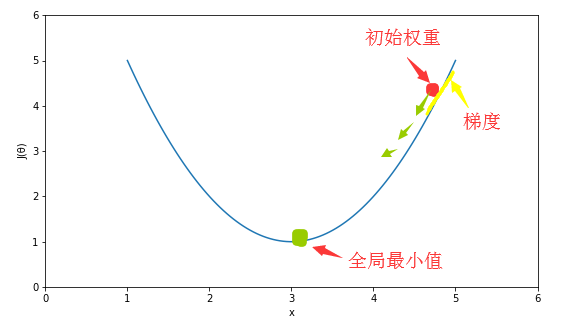
**随机梯度下降算法**

1. 相关概念
2. 首先定义目标函数，也就是需要做最小化处理的代价函数，将代价函数J定义为通过模型得到的输出与实际类标之间的误差平方和（SSE）：

m是训练集的样本数。

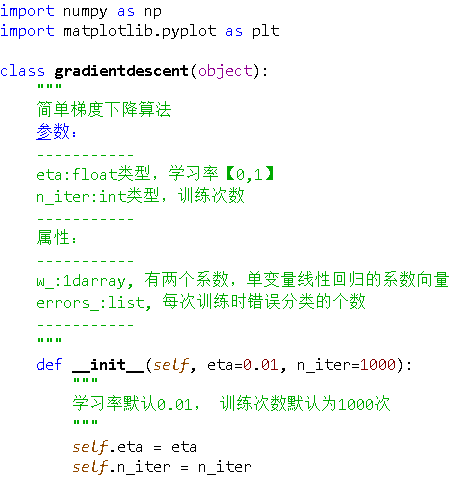
1. 这个函数是一个凸函数（这里与数学中对凸函数的定义正好相反，哈哈，理解为数学的凹函数吧），这样我们可以通过梯度下降算法，根据给定的学习速率和梯度的斜率，一步步沿着梯度方向移动，直至到达最小值。

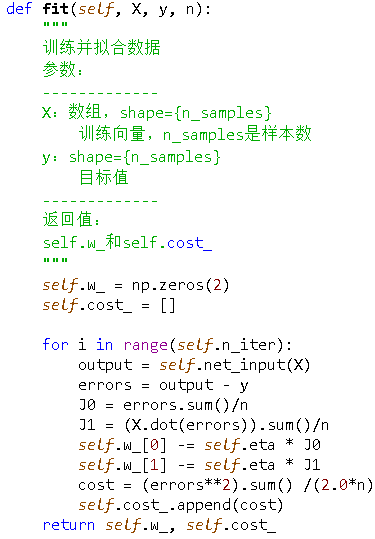


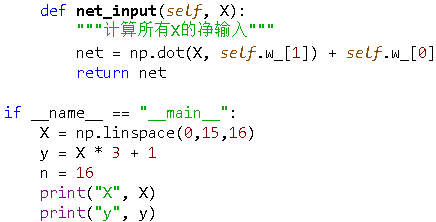
基于代价函数J(θ)沿梯度方向做一次权重更新：

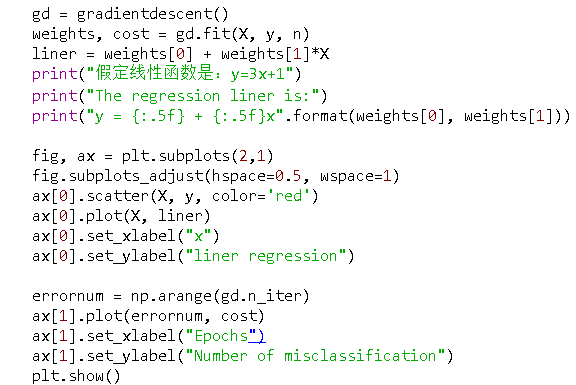
η是学习率，而且

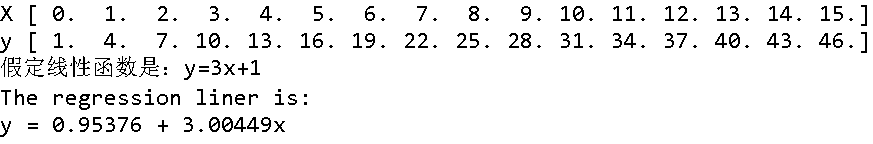
1. 对单变量线性回归的系数分别求偏导，可得到：
2. 可以看到梯度下降算法与感知器模型的算法很相似，但是在感知器模型中使用的类似于梯度上升法用于找到最大值，而梯度下降法用于找到最小值。而且在这里权重的更新用到了所有的训练样本，而感知器模型是渐进更新权重。
3. 利用Python尝试实现梯度下降算法：

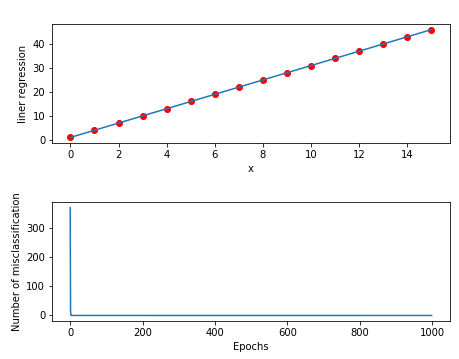












1. 讨论
2. 如果学习率太小，会导致需要更多的迭代次数才可以达到局部最优；如果学习率太大，可能会跳过全局最优解。
3. 如果对特征进行缩放，使其具备标准正态分布的特征：均值为0，方差为1。