梯度下降：

T1：

平方和损失函数：

这是一个函数，一般是

$$cost = 1/n * sum(y_i - y_i^{hat})^2$$

~~(hat不知道怎么打也是见笑子哈)~~

作用：

一般用于计算cost，目的是找到当前的损失（因为目标是将cost降到最小值），从而更好滴进行拟合函数（进行优化）。

T2：

梯度下降：

简单来说，我认为这是一种贪心算法，目的是寻找最小成本。 当得到一些成本函数值后，选择其中一个点，寻找当前能最快降低成本的方向(利用微积分，求导)，并朝 这个方向前进少许(即通过学习算式少量改变参数，得到新的参数)，再选刚刚得到的点，然后继续重复以 上行动，直到达到局部最小值。(局部最小值不是全局最小值，这也是贪心的弊端)。

学习率及其作用：

学习率可以看作梯度下降的幅度（每次下降多还是少）

权重更新公式（对w):

$$w' = w + aerfa * 1/n * sum((y_i - y_i^{hat}) * x_i)$$

ps:本来应该是2/n，但是可以把2合并到学习率aerfa中，所以这么写应该也没啥问题awa

T3:

学习率过大：

比如例题的0.1，学习率显然过大了。

这会导致每次梯度下降过多，从而超过了最优解。然后超过最优解后梯度可能更大，往回下降的又更多，最终导致越来越偏离最优解。

学习率过小：

这个导致需求循环次数更多，运行速度更慢，别人100次完成的效果你可能要10000次循环才能完成。

T4：

梯度下降:

每次使用全部样本进行计算。

优点：

更加全面，质量更好

缺点：

对于数据量大的情况下处理乏力。同时因为没有随机化优化，很容易陷入局部最优解

随机梯度下降：

在每次参数更新时使用单个训练样本计算梯度，并且立即更新参数

优点：

可以应对大数据，不容易陷入局部最优

缺点：

由于是随机化单个样本，拟合质量不高

T5：

什么是mini_batch：

可以视作梯度下降和随机梯度下降的折中版本，拥有两者的综合优点。

它在每次参数更新时使用一小批样本（既不是全部样本，也不是单个样本）来计算梯度。

和全量梯度下降、SGD的关系：

1.既不是全部样本，也不是单个样本

2.综合了两者的优缺点，是两者的折中方案