参考https://mp.weixin.qq.com/s/N4u3bKf4KwDsRxaeU4rvRg

无论在机器学习还是深度领域中,损失函数都是一个非常重要的知识点。损失函数（Loss Function）是用来估量模型的预测值 f(x) 与真实值 y 的不一致程度。我们的目标就是最小化损失函数，让 f(x) 与 y 尽量接近。通常可以使用梯度下降算法寻找函数最小值。损失函数有许多不同的类型，没有哪种损失函数适合所有的问题，需根据具体模型和问题进行选择。一般来说，损失函数大致可以分成两类：回归（Regression）和分类（Classification）。

回归模型中的三种损失函数包括：

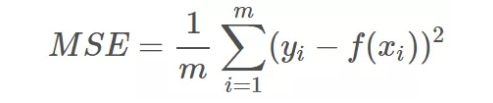
均方误差（Mean Square Error MSE）

平均绝对误差（Mean Absolute Error，MAE）、

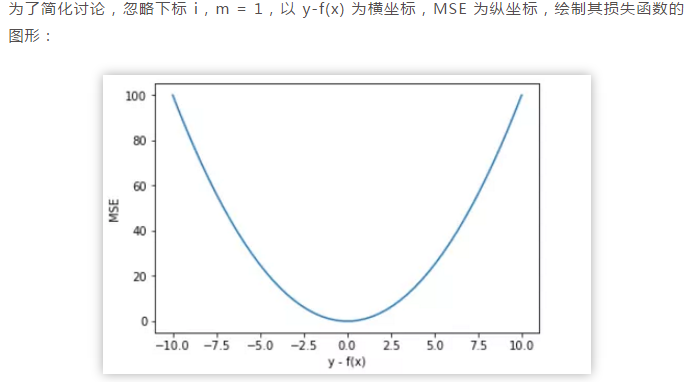
Huber Loss

1. **均方误差（Mean Square Error，MSE）**

均方误差指的就是模型预测值 f(x) 与样本真实值 y 之间距离平方的平均值。其公式如下所示：

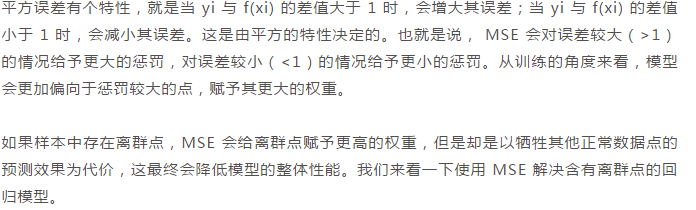


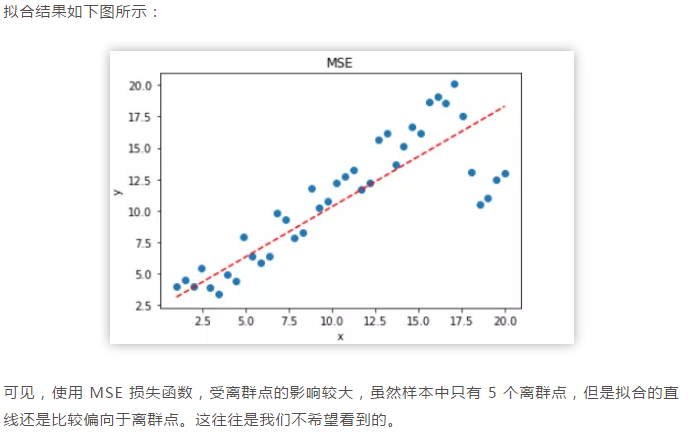
其中，yi 和 f(xi) 分别表示第 i 个样本的真实值和预测值，m 为样本个数。



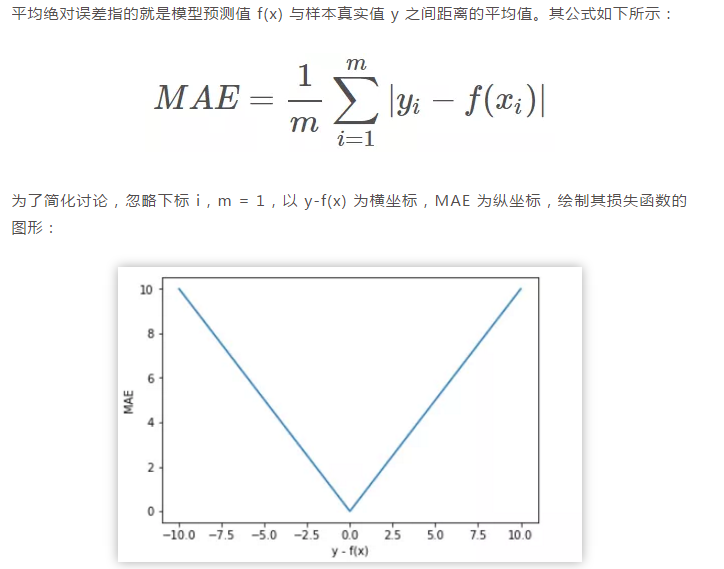
MSE 曲线的特点是光滑连续、可导，便于使用梯度下降算法，是比较常用的一种损失函数。而且，MSE 随着误差的减小，梯度也在减小，这有利于函数的收敛，即使固定学习因子，函数也能较快取得最小值，从图中可以观察到随着损失值的减小，函数的梯度也在不断的下降。

缺点：



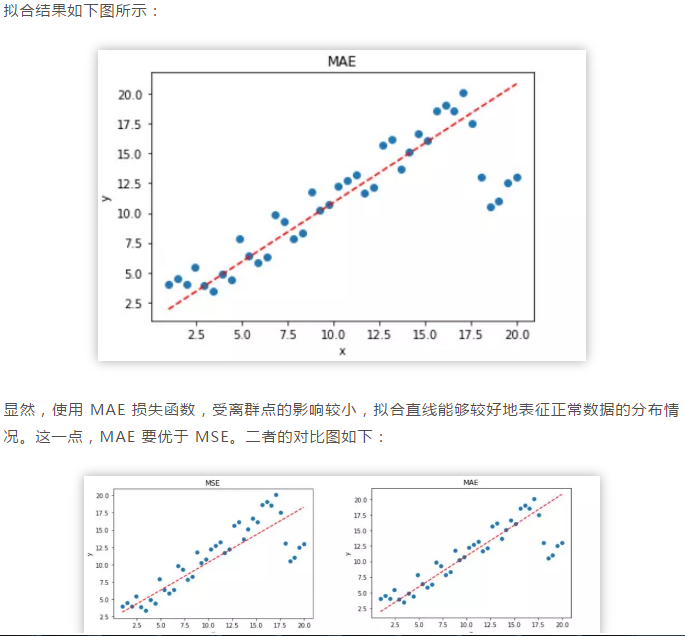


1. **平均绝对误差（Mean Absolute Error，MAE）**



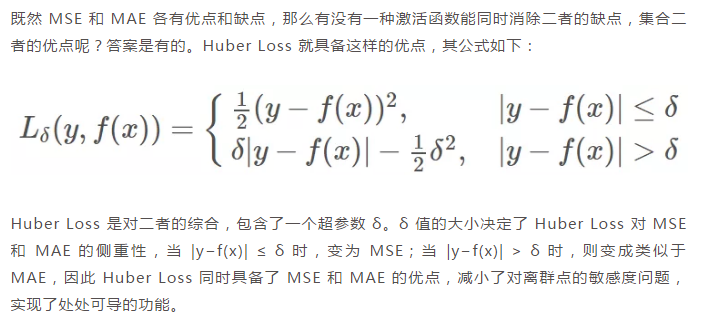
直观上来看，MAE 的曲线呈 V 字型，连续但在 y-f(x)=0 处不可导，计算机求解导数比较困难。而且 MAE 大部分情况下梯度都是相等的，这意味着即使对于小的损失值，其梯度也是大的。这不利于函数的收敛和模型的学习。

值得一提的是，MAE 相比 MSE 有个优点就是 MAE 对离群点不那么敏感，更有包容性。因为 MAE 计算的是误差 y-f(x) 的绝对值，无论是 y-f(x)>1 还是 y-f(x)<1，没有平方项的作用，惩罚力度都是一样的，所占权重一样。针对 MSE 中的例子，我们来使用 MAE 进行求解，看下拟合直线有什么不同。

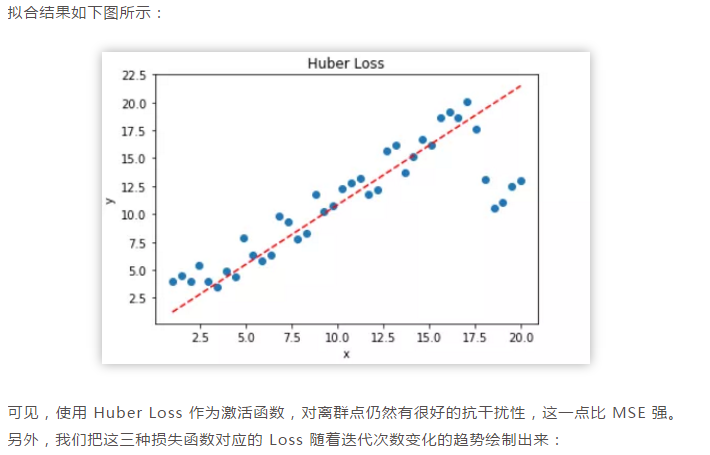


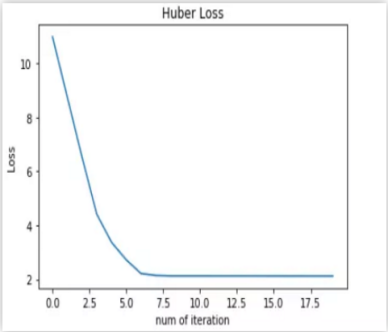
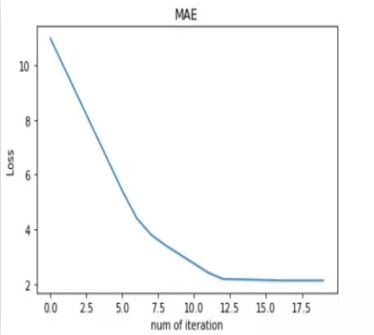
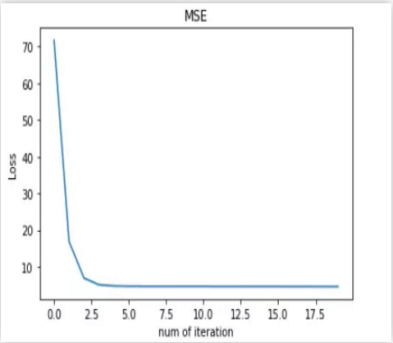
实际应用中，我们应该选择 MSE 还是 MAE 呢？从计算机求解梯度的复杂度来说，MSE 要优于 MAE，而且梯度也是动态变化的，能较快准确达到收敛。但是从离群点角度来看，如果离群点是实际数据或重要数据，而且是应该被检测到的异常值，那么我们应该使用MSE。另一方面，离群点仅仅代表数据损坏或者错误采样，无须给予过多关注，那么我们应该选择MAE作为损失。

1. **Huber Loss**

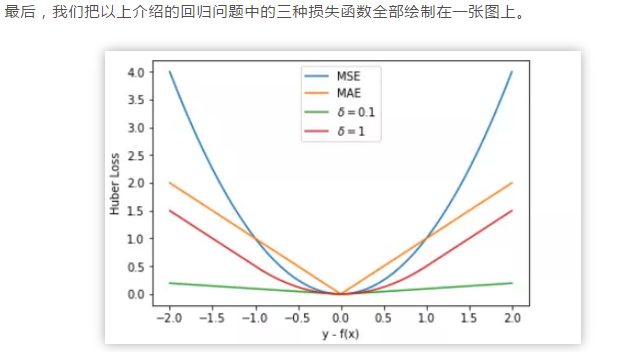


利用了均方误差在接近极值点的优势，快速收敛，处处可导，在误差较大的区域则选用近似MAE的做法，使得离群点的影响更小



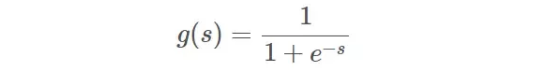


对比发现，MSE 的 Loss 下降得最快，MAE 的 Loss 下降得最慢，Huber Loss 下降速度介于 MSE 和 MAE 之间。也就是说，Huber Loss 弥补了此例中 MAE 的 Loss 下降速度慢的问题，使得优化速度接近 MSE。



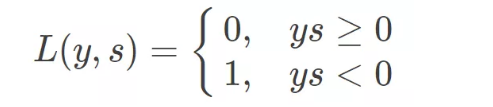
分类损失函数

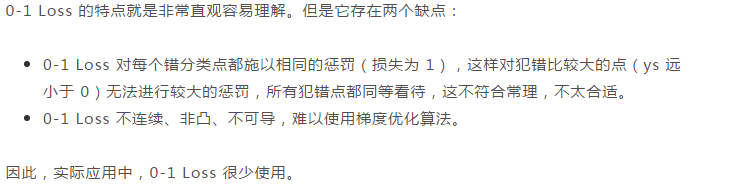
在讨论分类问题的损失函数之前，我想先说一下模型的输出 g(s)。一般来说，二分类机器学习模型包含两个部分：线性输出 s 和非线性输出 g(s)。其中，线性输出一般是模型输入 x 与 参数 w 的乘积，简写成：s = wx；非线性输出一般是 Sigmoid 函数，其表达式如下：



经过 Sigmoid 函数，g(s) 值被限定在 [0,1] 之间，若 s ≥ 0，g(s) ≥ 0.5，则预测为正类；若 s < 0，g(s) < 0.5，则预测为负类。

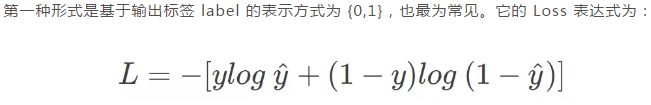
**0-1 Loss**





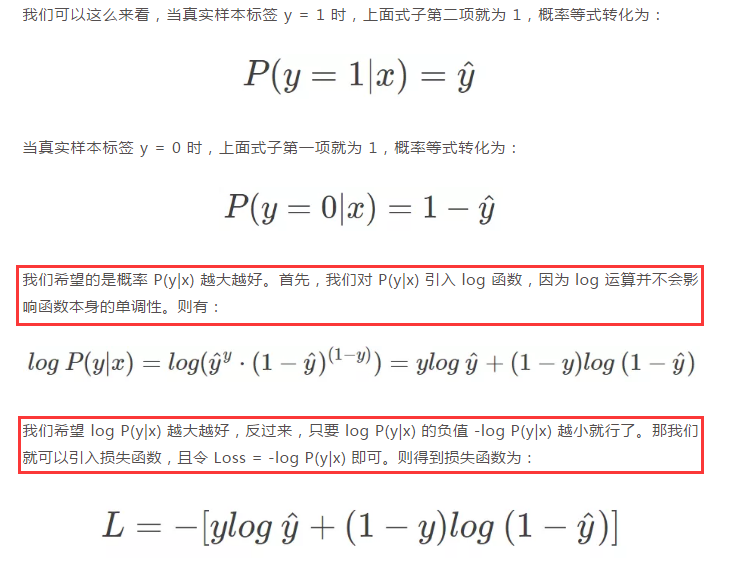
1. **Cross Entropy Loss**

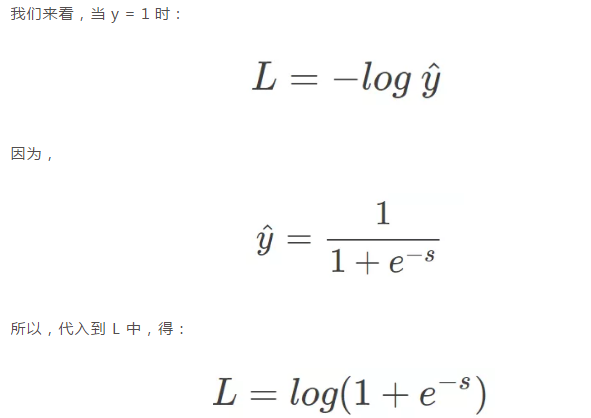
Cross Entropy Loss 是非常重要的损失函数，也是应用最多的损失函数之一。二分类问题的交叉熵 Loss 主要有两种形式，下面分别详细介绍。

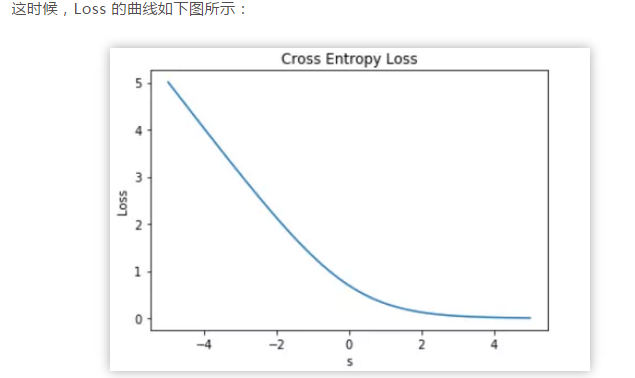


Y为样本真实值，y’为算法预测值，Y为正样本那么 1-Y为负样本，基于极大似然有







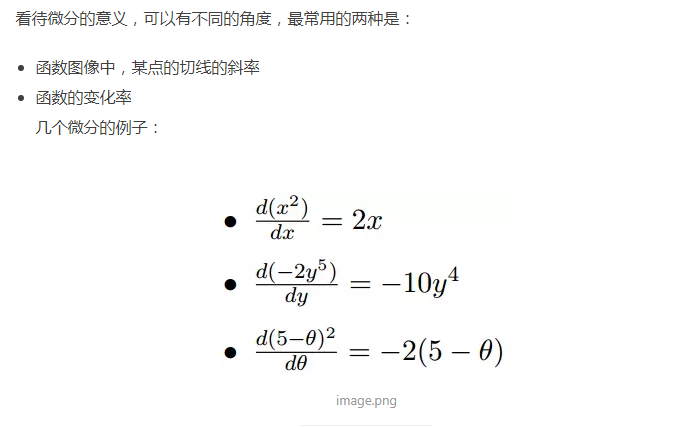


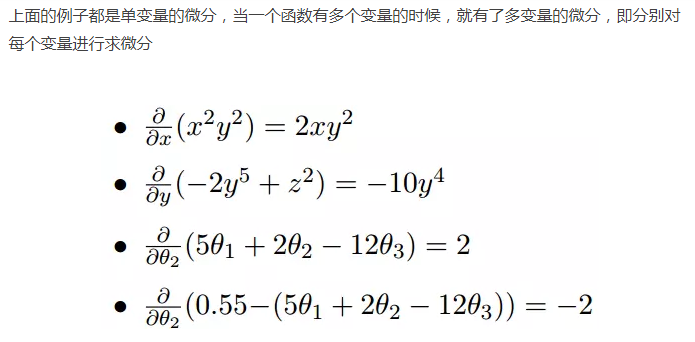
梯度下降

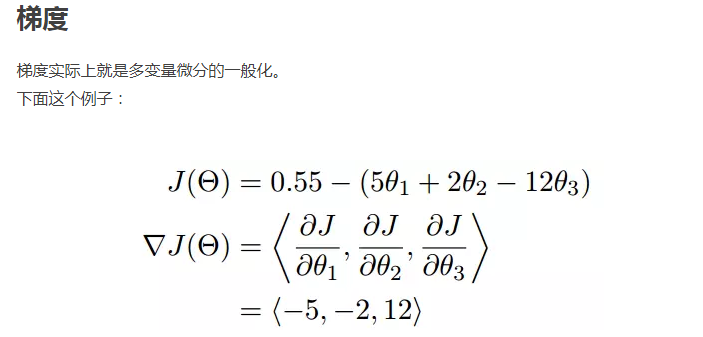
<https://www.jianshu.com/p/c7e642877b0e>

梯度下降的基本过程就和下山的场景很类似。首先，我们有一个可[微分](https://link.jianshu.com?t=https://en.wikipedia.org/wiki/Differentiable_function" \t "_blank)的函数。这个函数就代表着一座山。我们的目标就是找到这个函数的最小值，也就是山底。根据之前的场景假设，最快的下山的方式就是找到当前位置最陡峭的方向，然后沿着此方向向下走，对应到函数中，就是找到给定点的[梯度](https://link.jianshu.com?t=https://en.wikipedia.org/wiki/Gradient" \t "_blank) ，然后朝着梯度相反的方向，就能让函数值下降的最快！因为梯度的方向就是函数之变化最快的方向(在后面会详细解释)  
所以，我们重复利用这个方法，反复求取梯度，最后就能到达局部的最小值，这就类似于我们下山的过程。而求取梯度就确定了最陡峭的方向，也就是场景中测量方向的手段。

1微分







* 在单变量的函数中，梯度其实就是函数的微分，代表着函数在某个给定点的切线的斜率
* 在多变量函数中，梯度是一个向量，向量有方向，梯度的方向就指出了函数在给定点的上升最快的方向

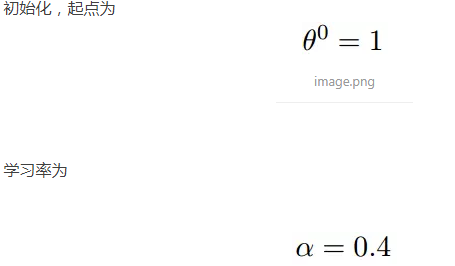


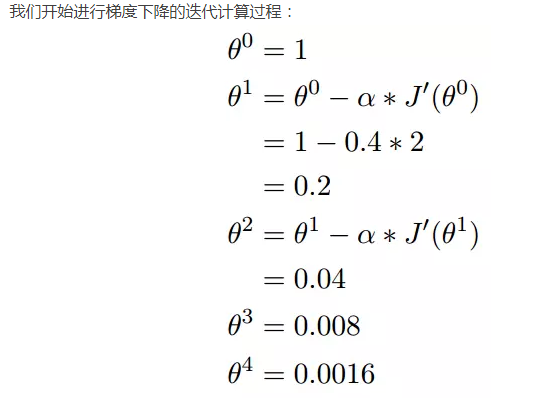
α是什么含义？  
α在梯度下降算法中被称作为**学习率**或者**步长**，意味着我们可以通过α来控制每一步走的距离，以保证不要步子跨的太大扯着蛋，哈哈，其实就是不要走太快，错过了最低点。同时也要保证不要走的太慢，导致太阳下山了，还没有走到山下。所以α的选择在梯度下降法中往往是很重要的！α不能太大也不能太小，太小的话，可能导致迟迟走不到最低点，太大的话，会导致错过最低点！

为什么要梯度要乘以一个负号？  
梯度前加一个负号，就意味着朝着梯度相反的方向前进！我们在前文提到，梯度的方向实际就是函数在此点上升最快的方向！而我们需要朝着下降最快的方向走，自然就是负的梯度的方向，所以此处需要加上负号

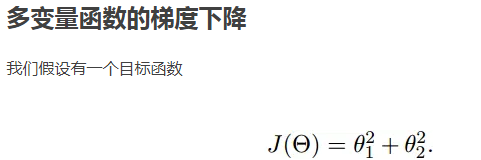
## 单变量函数的梯度下降

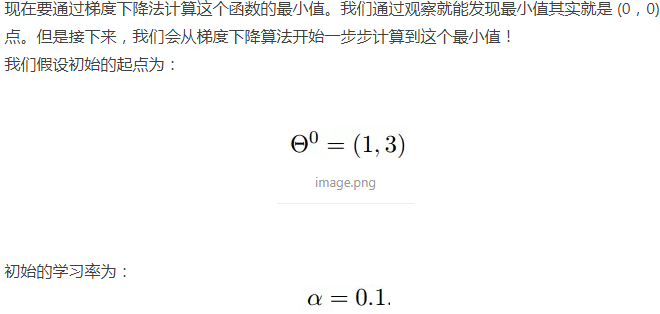




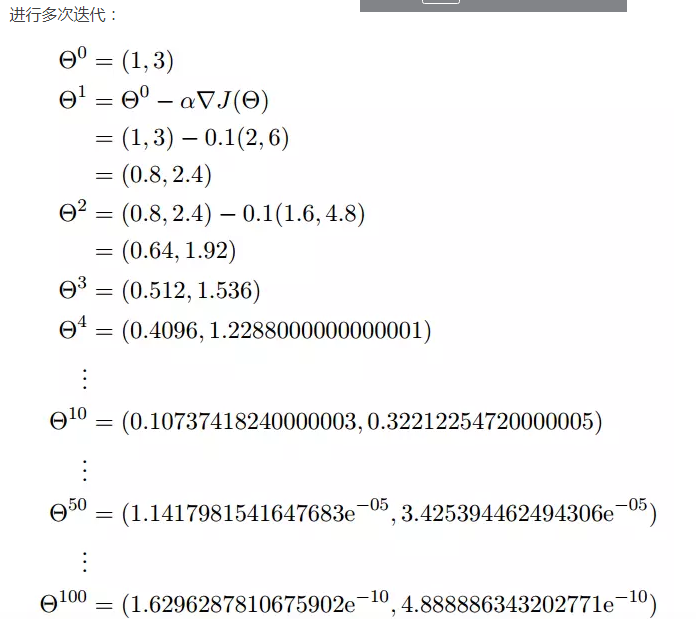


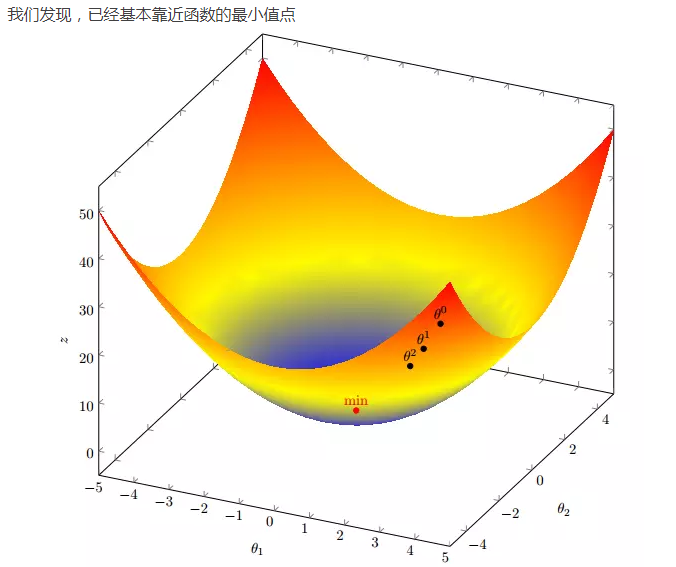
对于自变量O来说，只有给定一个O值，才能求出对应的负梯度值，这个值就是自变量变化的方向与大小



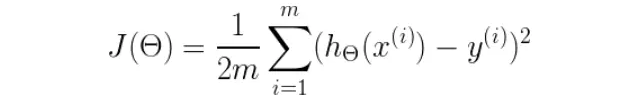


函数的梯度为





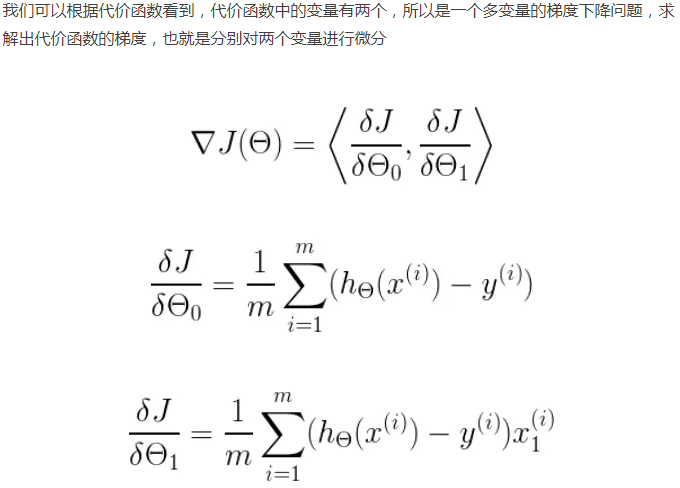
拿线性回归来举例子



这里自变量Xi 是特征值，O是权重，整个过程都是再调整权重O。

假设只有一维变量

有两个参数需要学习



权值完全可以给定初始随机值，然后开始调整

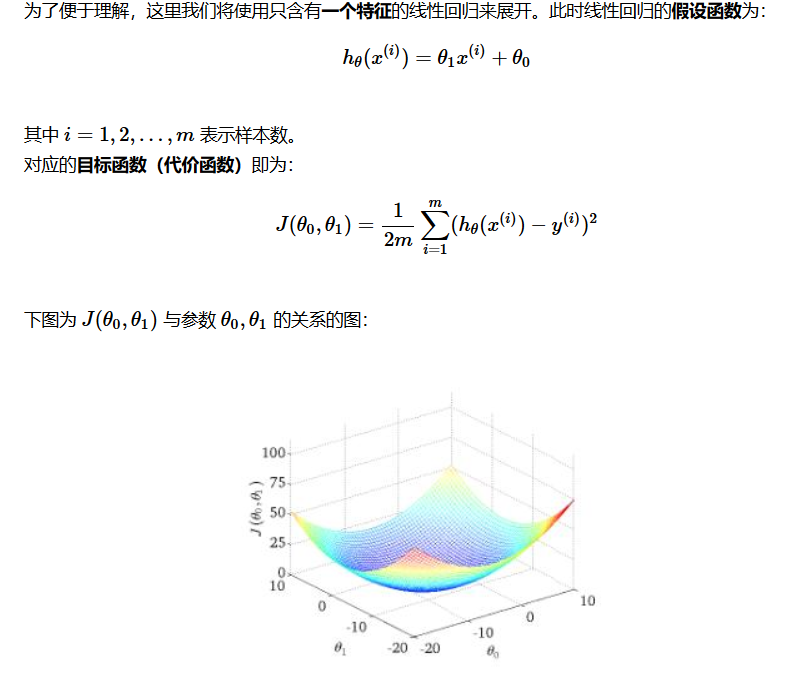
梯度下降法作为机器学习中较常使用的优化算法，其有着三种不同的形式：

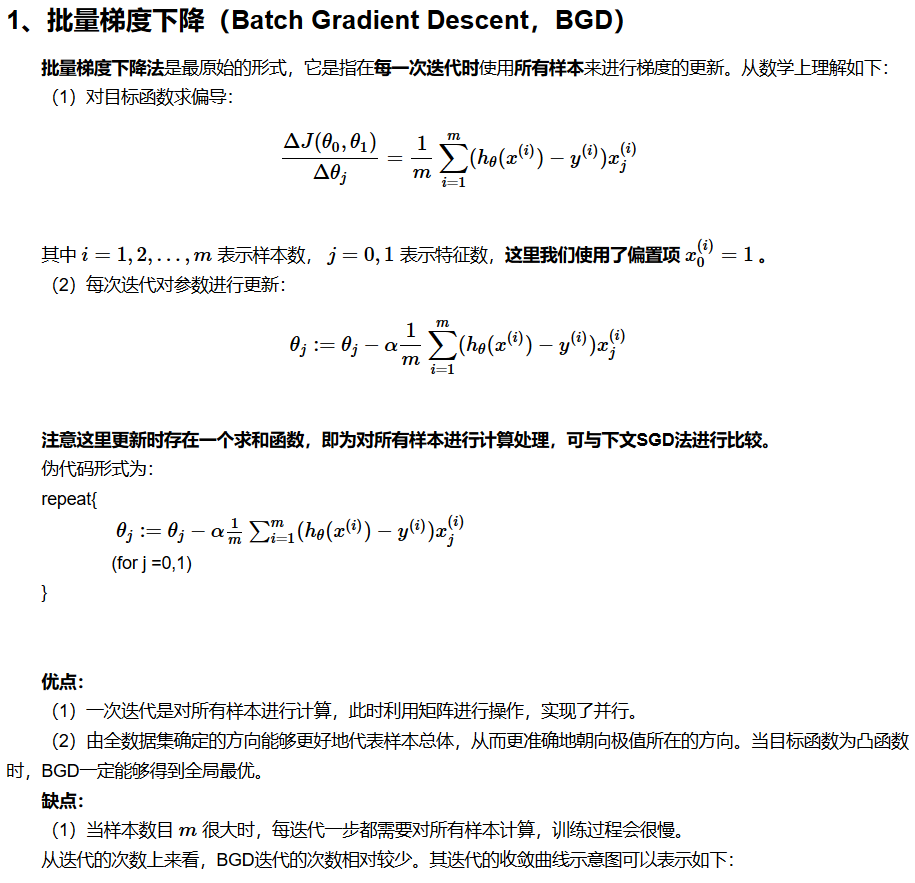
https://www.cnblogs.com/lliuye/p/9451903.html

**批量梯度下降（Batch Gradient Descent）、**

**随机梯度下降（Stochastic Gradient Descent）以及**

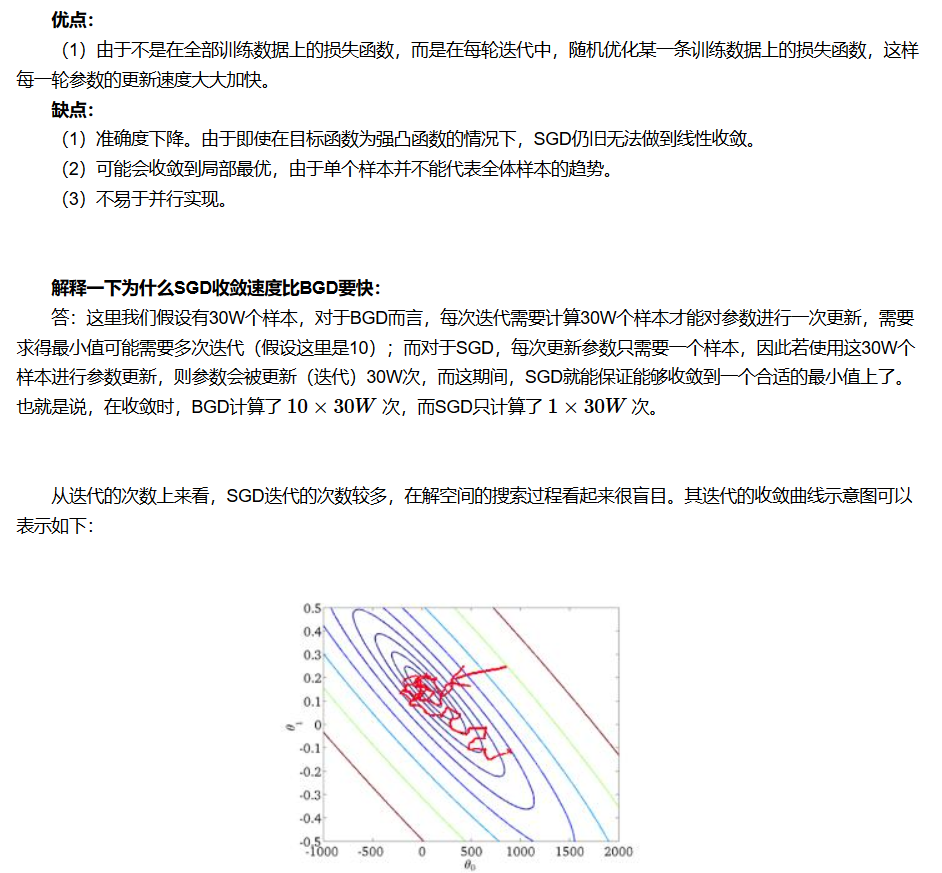
**小批量梯度下降（Mini-Batch Gradient Descent）**。其中小批量梯度下降法也常用在深度学习中进行模型的训练。接下来，我们将对这三种不同的梯度下降法进行理解。





## 2、随机梯度下降（Stochastic Gradient Descent，SGD）

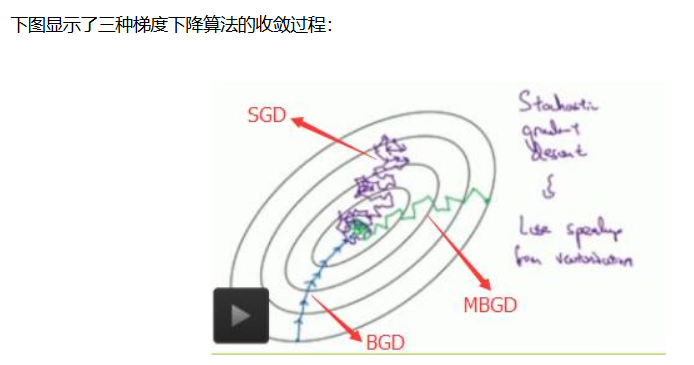
**随机梯度下降法**不同于批量梯度下降，随机梯度下降是**每次迭代**使用**一个样本**来对参数进行更新。使得训练速度加快。



## 3、小批量梯度下降（Mini-Batch Gradient Descent, MBGD）

**小批量梯度下降**，是对批量梯度下降以及随机梯度下降的一个折中办法。其思想是：**每次迭代** 使用 \*\* batch\_size\*\* 个样本来对参数进行更新。  
  这里我们假设 *batchsize*=10样本数 *m*=1000 。

**优点：**  
  （1）通过矩阵运算，每次在一个batch上优化神经网络参数并不会比单个数据慢太多。  
  （2）每次使用一个batch可以大大减小收敛所需要的迭代次数，同时可以使收敛到的结果更加接近梯度下降的效果。(比如上例中的30W，设置batch\_size=100时，需要迭代3000次，远小于SGD的30W次)  
  （3）可实现并行化。  
  **缺点：**  
  （1）batch\_size的不当选择可能会带来一些问题。  
  
  
  **batcha\_size的选择带来的影响：**  
  （1）在合理地范围内，增大batch\_size的好处：  
    a. 内存利用率提高了，大矩阵乘法的并行化效率提高。  
    b. 跑完一次 epoch（全数据集）所需的迭代次数减少，对于相同数据量的处理速度进一步加快。  
    c. 在一定范围内，一般来说 Batch\_Size 越大，其确定的下降方向越准，引起训练震荡越小。  
  （2）盲目增大batch\_size的坏处：  
    a. 内存利用率提高了，但是内存容量可能撑不住了。  
    b. 跑完一次 epoch（全数据集）所需的迭代次数减少，要想达到相同的精度，其所花费的时间大大增加了，从而对参数的修正也就显得更加缓慢。  
    c. Batch\_Size 增大到一定程度，其确定的下降方向已经基本不再变化。

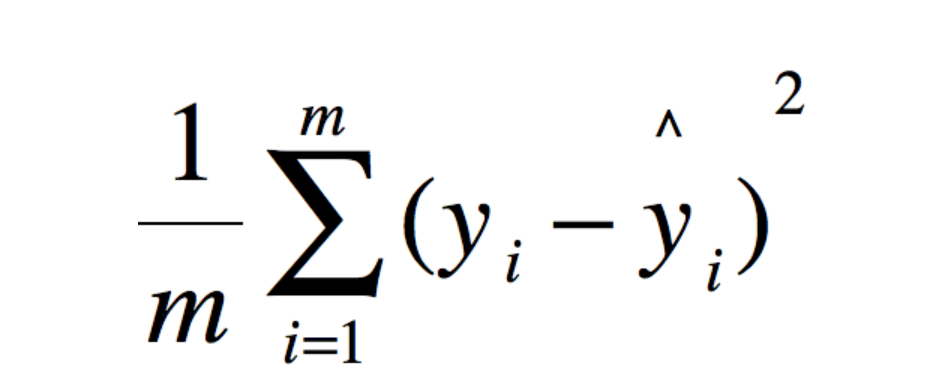


回归评价指标MSE、RMSE、MAE、R-Squared

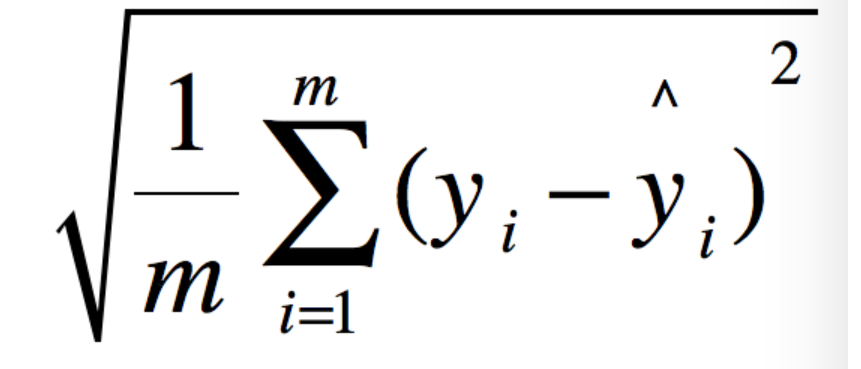
分类问题的评价指标是准确率，那么回归算法的评价指标就是MSE，RMSE，MAE、R-Squared。下面一一介绍

均方误差（MSE）

MSE （Mean Squared Error）叫做均方误差。看公式



# 均方根误差（RMSE）



不就是MSE开个根号么。有意义么？其实实质是一样的。只不过用于数据更好的描述。

例如：要做房价预测，每平方是万元（真贵），我们预测结果也是万元。那么差值的平方单位应该是 千万级别的。那我们不太好描述自己做的模型效果。怎么说呢？我们的模型误差是 多少千万？。。。。。。于是干脆就开个根号就好了。我们误差的结果就跟我们数据是一个级别的可，在描述模型的时候就说，我们模型的误差是多少万元。

