# 机器学习简介

# 回归模型

**回归任务：**

给定训练数据𝒟 = {𝐱i ,𝑦i }Ni=1，其中𝑁为训练样本数目，𝑖为样本索引，𝐱i为第𝑖个样本的输入特征，𝑦i 为第𝑖个样本的输出/响应，𝑦 i ∈ **R**

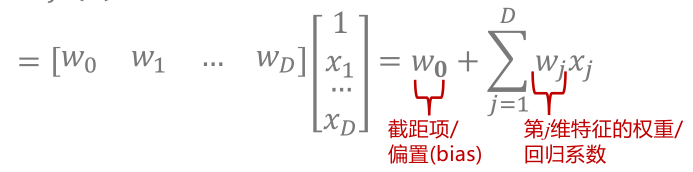
**回归**：根据训练样本𝒟，学习一个从输入𝐱到输出𝑦的映射f

对新的测试数据𝐱 ，用学习到的映射𝑦对其进行预测：。表示预测值

## 线性回归

### 线性回归定义

假设输出𝑦与输入𝐱之间的关系为线性关系，即：，展开为：

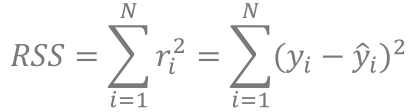


* 𝐷: 特征维数
* 𝑗: 特征索引
* 𝐱：在𝐷维特征的基础上，增加一个常数项1（用于表示截距项）𝐱 = (1,𝑥1 ,…,𝑥 D ) T

### 如何判断最佳直线

**预测残差(Residual)：真实值和预测值之间的差异。**

**残差平方和(Residual Sum of Squares, RSS )：𝑅𝑆𝑆最小，则表示直线最佳。可理解为方差。**



***~~(个人感觉这里应该理解为损失函数值最小时，表示直线最佳，因为有多种损失函数)~~***

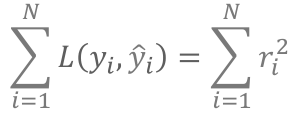
### 损失函数

在机器学习中，我们用损失函数（loss function）度量样本真实值与模型预测值之间差异，上节讲到的RSS，我们称为L2损失函数。

* **L2损失函数：**

**当损失函数取残差平方时，我们称之为**L2损失**，记为：**

**目标函数：最小训练集上的损失（经验风险最小）**

****

**L2损失处处可导，优化计算方便（参见优化计算小节），但对噪声敏感。如下例：**

****

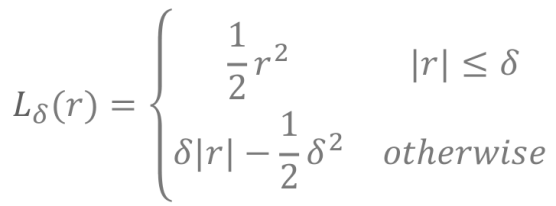
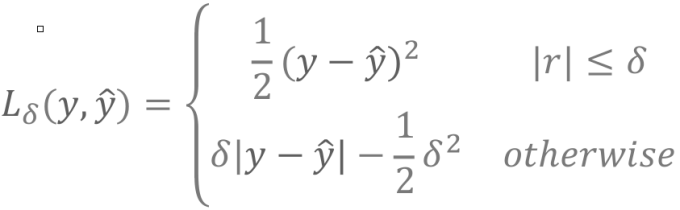
**参考上表，当在线广告费用x=2.2时，其残差平方和为3957.27；这个值对损失的描述影响较大。由此可知L2损失函数对噪声敏感。但它因为平方函数，处处可导，优化计算方便。**

* **L1损失函数：**

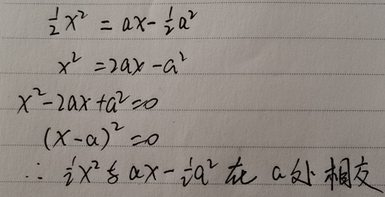
**当损失函数取残差绝对值时，我们称之为**L1损失**，记为；对比L2损失函数，L1损失对噪声不敏感，但在0处不可导，在优化计算不方便。**

* **Huber损失函数：**

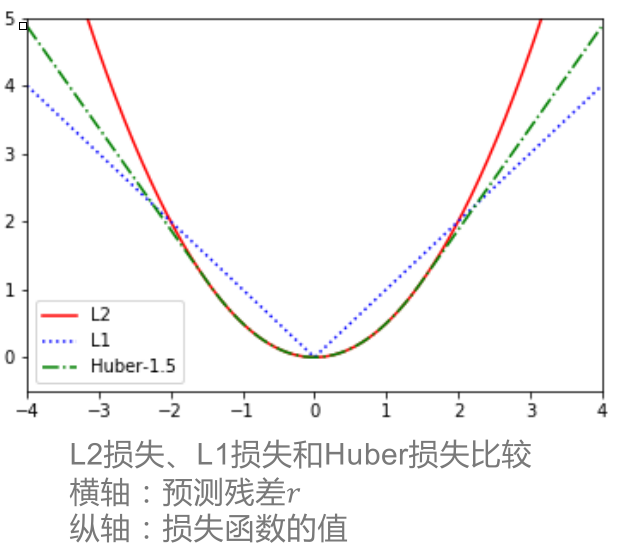
**综合L2损失和L1损失的优点，得到Huber损失：**

** 或**

* **δ(de·er·ta)是一个比较小的数**
* **|r|<δ时，取L2损失；这里因为优化计算时会求导前面加了一个1/2，的系数1/2是为了求导时消掉系数。**
* **其他情况，取L1损失；最后减去是为了这个符合函数能够在δ处连续，上式变换如下：**

****

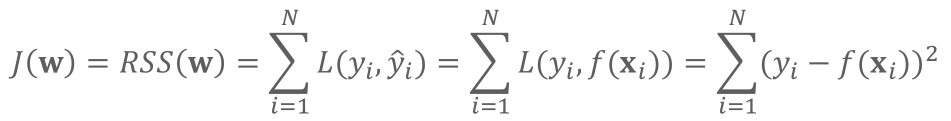
**Huber损失和L1损失L2损失的比较图：**

****

**Huber Loss 是一个用于回归问题的带参损失函数, 优点是能增强平方误差损失函数(MSE, mean square error)对离群点的鲁棒性。**

### ！！！回归模型中损失函数的概率解释

上文中我们提到，最佳模型为残差平方和最小，或叫L2损失最小，模型的目标函数为：

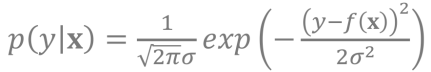


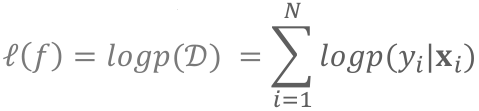
* ***J*为目标函数**
* **残差平方和最小也被称为最小二乘（Ordinary Least Square, OLS）**
* **噪声分布**

在回归任务中，令模型预测值和真实值之间的差异为噪声𝜀，假设噪声𝜀的分布为0均值的正态分布：𝜀~𝒩(0, 𝜎2 ) ， 其中𝒩为正态分布函数，括号内0代表均值，𝜎2代表方差。那么真实值y的函数可写为：

𝑦 = 𝑓 (𝐱) + 𝜀 ； 那么 𝑦|𝐱~𝒩(𝑓(𝐱) , 𝜎2 )

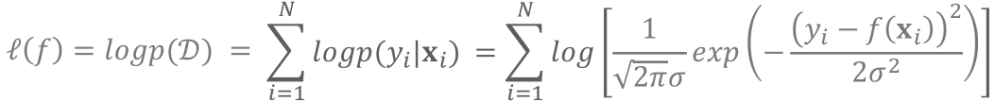
* **为什么𝑦|𝐱服从正态分布𝒩(𝑓(𝐱) , 𝜎2 )，这里需要注意在给定𝐱的情况下，𝑓(𝐱) 为常数，随机变量𝜀加上一个常数，只改变分布的均值，不改变分布的方差**
* **极大似然估计**

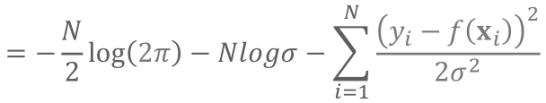
**𝑦|𝐱分布的概率密度函数可写为：**

**那么，**log似然估计**为：**

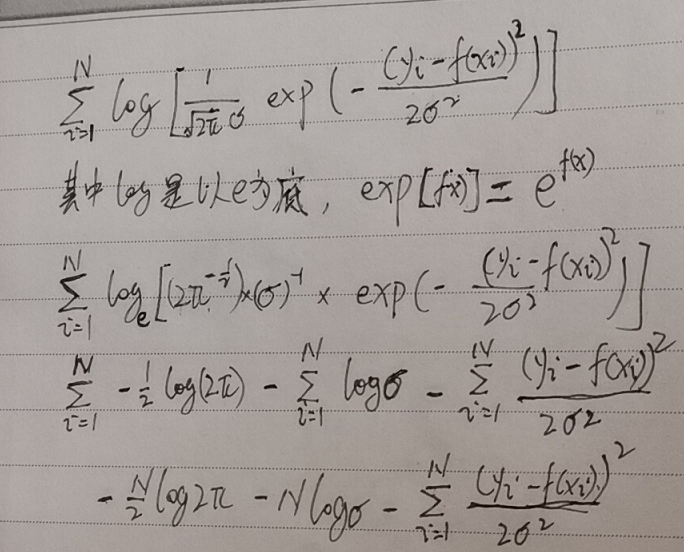
* **其中的样本为独立分布样本(独立同分布样本指随机过程中，任何时刻的取值都为随机变量，如果这些随机变量服从同一分布，并且互相独立，那么这些随机变量是独立同分布)**
* **那么𝑝(𝒟)为样本的联合概率(联合概率指的是包含多个条件且所有条件同时成立的概率，记作P(X=a,Y=b)或P(a,b)或P(ab))**

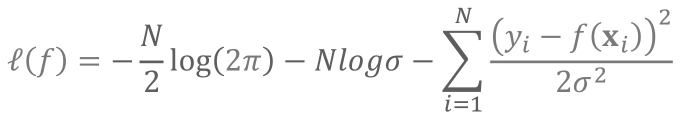
相乘取log等于先取log再相加，所以上式可变为：

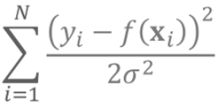




推导过程如下图：



由上面推导过程最终可得log似然函数为：

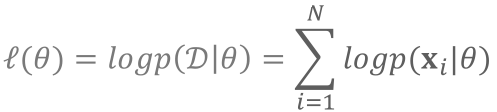
当log似然函数取最大值时， 取最小值，而刚好是残差平方和/训练集上的L2损失之和，所以极大似然估计等价于最小二乘。

* **极大似然估计等价于负log似然最小，因此负log似然也被称为一种损失函数：负log似然损失。**
* **L2损失也是负log似然损失。**
* **分类任务中Logistic回归中采用的损失函数也是负log似然损失。**

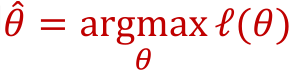
### ！！！极大似然估计

给定数据， 似然函数定义为数据出现的概率，通常我们假定数据是独立同分布样本，因此所有数据出现的概率等于每个数据点出现的概率相乘。

实际计算中，通常对似然函数取log运算（ log函数为单调函数，不影响取极值的位置；很多分布的概率密度函数为指数函数形式，log运算数值计算更稳定），得到log似然函数：



* **其中θ为分布的参数。**

统计中我们需要根据观测数据估计分布的参数**θ**。一种常用的参数估计为极大似然估计，即。

最大似然估计参考：<https://blog.csdn.net/zengxiantao1994/article/details/72787849>

### 过拟合

原则上任意复杂的模型能完全拟合训练数据。我们称之为过拟合。

* **过拟合（ overfitting ）：过于复杂的模型与训练数据拟合得太好，但和测试数据拟合得不好。**
* **欠拟合（underfitting）：过于简单的模型与训练数据拟合得欠佳（和测试数据自然也拟合得不好）**

线性回归中采用线性模型。而线性模型是很简单的模型，所以当特征维数不是太高时，线性回归的过拟合现象通常不太严重。

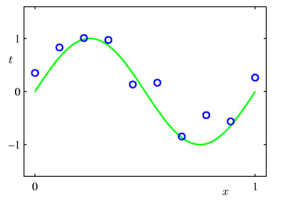
#### ****抑制过拟合的方法****

1. **方法一：增加训练的样本数量**
2. **方法二：减少训练数据的方差**
3. **方法三：增加正则项**

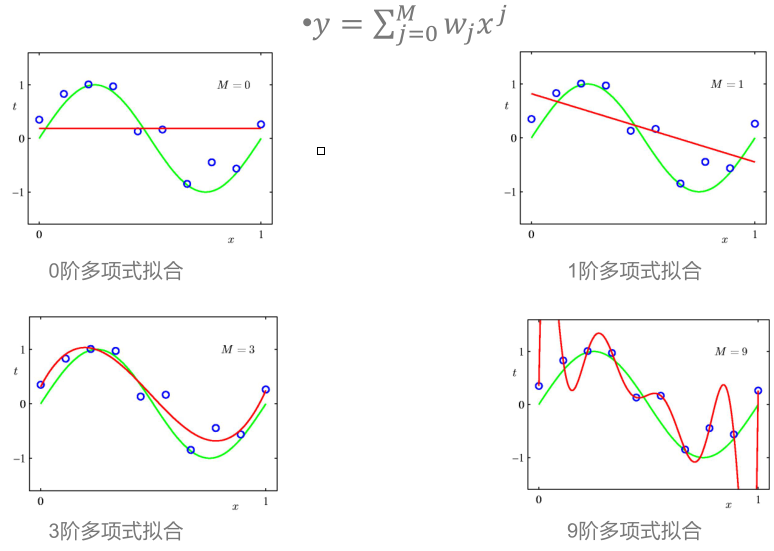
**方法1和方法2比较好理解，这里不详述，下面我们看增加正则项。**

#### ****增加正则项****

假设数据产生模型为：，给定𝑁 = 10个样本点，模型如下图：

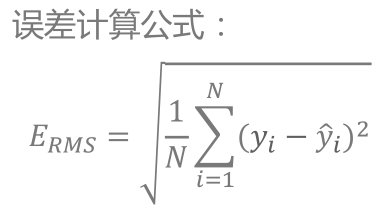
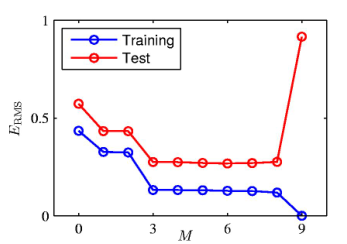


采用𝑀阶多项式拟合数据：，下图是0阶、1阶、3阶和9阶拟合曲线。



上图可以看出，9阶拟合误差较大，3阶拟合效果最好。

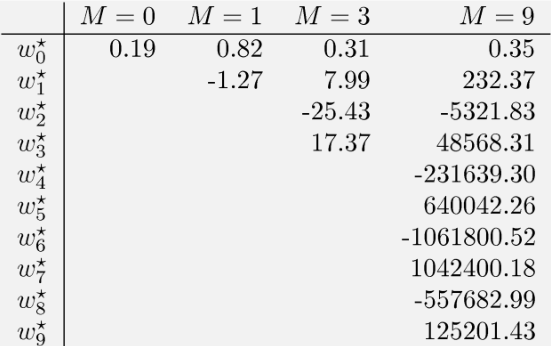
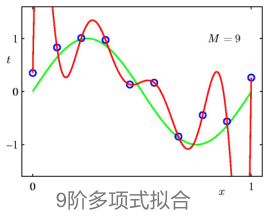
**根据**误差计算公式**，对比训练误差 vs. 测试误差（另外生成100个测试样本）：**

** **

根据误差-阶的图像我们可以看出，4~8阶的拟合效果与3阶基本一致，到9阶时拟合误差与实际误差相差甚远，我们称4~9阶时为过拟合。

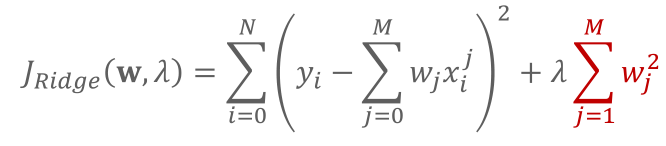
需要注意的是，评价模型性能不能在训练集上评估，而应该在新的测试数据上进行评估：推广性/泛化能力（ generalization ）。

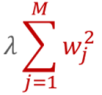
继续分析过拟合时w的值，如下图：

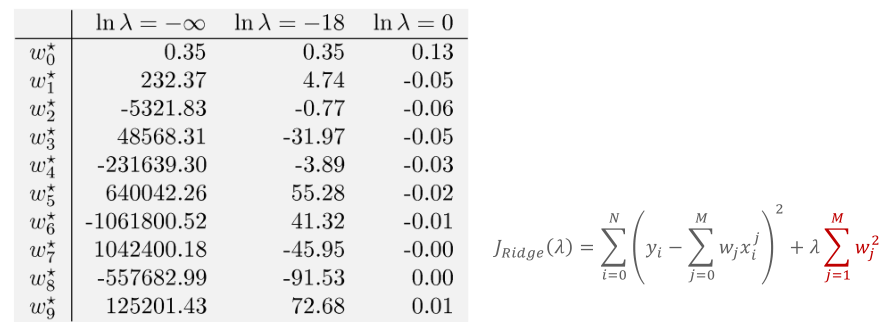
根据上表我们看到，发生过拟合时，回归系数（绝对值/平方）很大，即输入𝑥的很小变化可能带来输出𝑦较大的变化，函数变化剧烈。

根据发生过拟合时，回归系数（绝对值/平方）很大的这个现象，我们考虑将在损失函数中增加对w的限制，以L2损失函数为例，公式如下：



我们在L2损失函数中增加关于系数w的相关计算来评判目标函数的优劣，就是L2的正则项，其中的λ表示系数w的权重(后面会有λ取值方法介绍，本节不讨论λ的如何取值)。

对L2函数增加正则项的函数，我们称为岭回归，下图看一下岭回归系数：



上图我们可以看到，λ越大则******在整个损失函数中所占的比重越大，所以训练出的w就越小，若λ过大，则回归系数也会比较大，其拟合效果也必然不好。总结就是：𝜆越大，回归系数（绝对值/平方）越小，模型越简单。

**常用的正常项：**

**L2正则：**

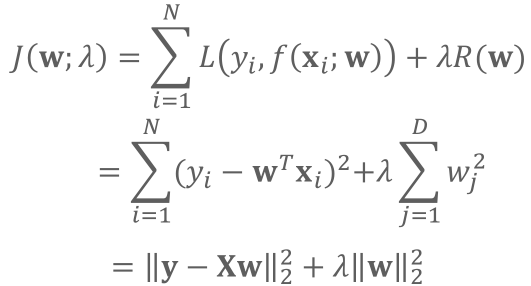
**L1正则：**

**注意：其中𝐰 为模型参数， 𝐷为参数的维数（对截距项不惩罚）。**

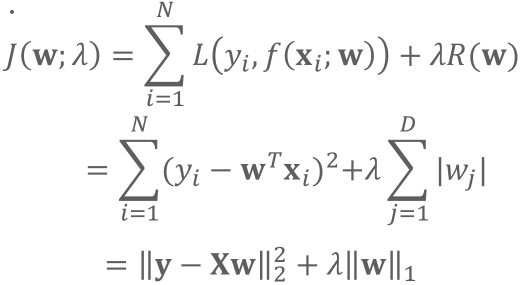
* **注意：L1范数是L0范数的最优凸近似，更方便优化计算。所以课件中我们不单独讨论L0正则。**

#### ****常用正则项****

* **岭回归(Ridge - L2损失+L2正则项)**
* **当损失函数取L2损失： ，**
* **正则项取L2正则： (注意j从1开始，表示**对截距项不施加惩罚**)**
* **模型取线性回归： (**增加常数项**𝑥0 = 1表示截距项)**
* **得到岭回归的目标函数为，变换过程如下：**

****

* **Lasso（L2损失 + L1正则）**
* **当损失函数取L2损失：**
* **正则项取L1正则：**
* **模型取线性回归：**
* **得到Lasso (Least Absolute Shrinkage and Selection Operator) 的目标函数为：**

****

* **弹性网络—(L2损失 + （L1正则 + L2正则）)**

弹性网络的目标函数为：

* 𝜌表示L1正则的比例
* **正则的效果**

L2正则使得线性回归系数收缩，模型稳定。当输入特征之间存在共线性时使用L2正则。共线性也就是特征之间存在相关性。

L1正则也会收缩回归系数。当正则参数取合适值时，L1正则使得有些线性回归系数为0，得到稀疏模型。当输入特征多，有些特征与目标变量之间相关性很弱时， L1正则可能只选择强相关的特征，模型解释性好。

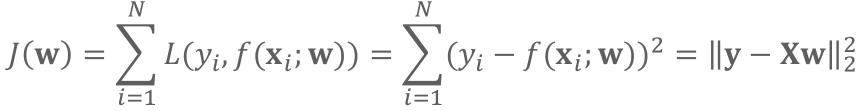
注意：由于正则项中对不同维度的wj 同等对待，对输入特征𝐗最好做去量纲（scaling）处理，使得不同维度的特征取值范围大致相同。

#### ****正则项的概率解释（--待补充--）****

待补充

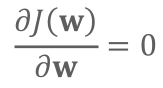
### 线性回归模型解析求解

#### ****最小二乘（OLS，Ordinary Least Square）****

最小二乘（OLS）线性回归：

把**w**看成变量，其他符号看成标量，那么就是需要求**w**使得J(w)最小：



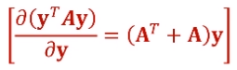
根据导数定义，求极值就是求一阶导数为0处：

OLS目标函数（矩阵形式）（疑问点）：





由于：

1. **w**T **X = w****X**T= Σ**w**i**x**i （**w**和**x**都是一行数据，同理**x**T**x**=**xx**T）
2. 矩阵求导公式：

所以上式对w求导得:



所以需要求：

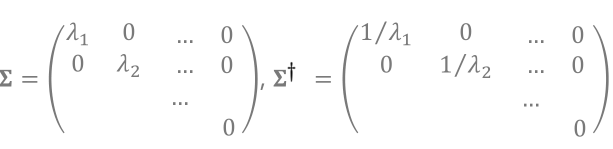
变换得到：，此式也叫作正规方程组，两边乘以，最终得：



OLS目标函数：最小，相当于求**y**=**Xw（**AB的逆，等于 B的逆乘以A的逆**）**，如果**X**为方阵则**X**有逆矩阵**X**-1，若X不是方阵则可求广义逆：**w**=**X**+**y。**

广义逆可采用奇异值分解(SVD)：

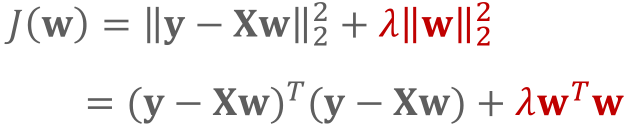
m\*n的矩阵A可分解为：**UΣV**T ，U和V是m\*n的正交阵，**Σ**为m\*n矩阵，那么**A**的伪逆**A**+=V**Σ+U**T，其中：

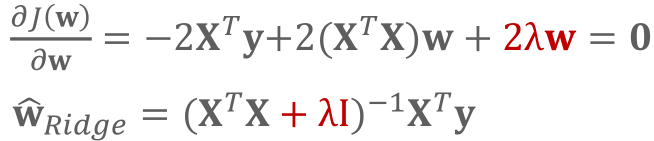


Σ的前r=r(A)个“对角元”为A的奇异值，λ1≥λ2≥…≥0。

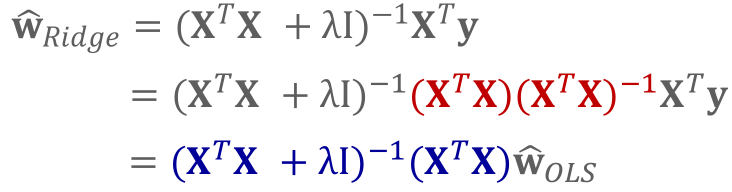
#### ****岭回归****

岭回归的目标函数为：



偏导数：

岭回归的解：



#### ****对比OLS和岭回归****

OLS的解为：，需要对矩阵𝐗T𝐗求逆。当输入特征存在共线性（某些特征可以用其他特征的线形组合表示），矩阵𝐗是接近不满秩，矩阵𝐗T𝐗接近奇异，求逆不稳定。岭回归的解为：，对矩阵(𝐗T𝐗 + λI)求逆。即使输入特征存在共线性，矩阵𝐗不满秩，矩阵𝐗 B 𝐗对角线存在等于0或接近于0的元素，但0 + 𝜆 ≠ 0， (𝐗 B 𝐗 + λI)求逆仍可得到稳定解。因此岭回归在输入特征存在共线性的情况仍然能得到稳定解。

### 线性回归模型梯度下降法求解

#### ****什么是梯度下降****

使用解析解析求解法求解N\*D维的矩阵X，进行SVD的复杂度为O(N2D), 当样本数目𝑁很大或者特征维数𝐷很大时， SVD计算复杂度高，或者机器的内存根本不够。此时我们可以采用梯度下降法、随机梯度下降法、次梯度法、坐标轴下降法等求解。

梯度下降法（Gradient Descent）是求解无约束优化问题最常采用的方法之一。

在微积分中，一元函数𝑓(𝑥) 在𝑥处的梯度为函数在该点的导数𝑑𝑓⁄𝑑𝑥 。对多元函数𝑓(𝑥1 - ,…,𝑥D )，在点𝐱 = (𝑥1 ,…,𝑥D )处，共有𝐷个偏导数𝜕𝑓/𝜕𝑥1 ,..,𝜕𝑓/𝜕𝑥D 。将这𝐷个偏导数组合成一个𝐷维矢量(𝜕𝑓/𝜕𝑥1 ,..,𝜕𝑓/𝜕𝑥D) T ，称为函数𝑓(𝑥1 ,…,𝑥D )在𝐱处的梯度。梯度一般记为𝛻或𝑔𝑟𝑎𝑑，即：

𝛻𝑓 (𝑥1 ,…,𝑥D) = 𝑔𝑟𝑎𝑑𝑓(𝑥1 ,…,𝑥D) = (𝜕𝑓/𝜕𝑥1 ,..,𝜕𝑓/𝜕𝑥D ) T

从几何意义上讲，某点的梯度是函数在该点变化最快的地方。沿着梯度向量的方向，函数增加最快，更容易找到函数的最大值。沿着负梯度方向，函数减少最快，更容易找到函数的最小值。

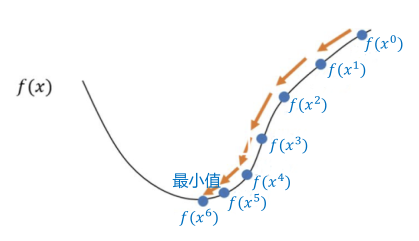
∇：发音为nabla，表示微分，不属于希腊字母，只是一个记号

##### 梯度下降法原理

1. 从t = 0开始，初始化x0 = 为随机值；

2. 找到下一个点xt+1 ，使得函数值越来越小，即𝑓 (xt+1) < 𝑓 (xt )；

3. 重复，直到函数值不再减小，则已经找到函数的最小值。



注意：

* x0为随机值
* 该方法只能找到局部极小值

##### 梯度下降法计算方法

对函数𝑓(𝑥)进行一阶泰勒（Taylor）展开，得到：, 𝛻𝑓(𝑥)为𝑓(x)一阶导数。要找到函数的最小值，需要𝑓(𝑥 + ∆𝑥) < 𝑓(𝑥) ，因此∆𝑥𝛻𝑓 𝑥 < 0

可选择∆𝑥 = −𝜂𝛻𝑓 (𝑥) ， 𝜂 > 0。

其中步长𝜂是一个较小的正数，从而∆𝑥𝛻𝑓(𝑥)= −𝜂 𝛻(𝑓(𝑥) )2< 0。因此， 𝑥的更新：沿负梯度方向走一小步，会使得𝑓(𝑥t+1) < 𝑓(𝑥t)。

##### OLS的梯度下降

上面我们已经得到了OLS对w求导：，根据梯度下降计算方法可得：𝐰t+1 = 𝐰t − 𝜂𝛻𝐽 (𝐰t)= ,其中就是预测残差r。

OLS梯度下降法计算步骤：

1. 从𝑡 = 0开始，初始化𝐰 (0) 为较小的随机值（或0）

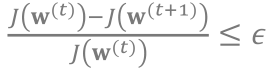
2. 计算目标函数𝐽(𝐰) 在当前值的梯度：𝛻𝐽(𝐰t)

3. 根据学习率𝜂 ，更新参数：𝐰 (t+1) = 𝐰 (t) − 𝜂𝛻𝐽 (𝐰t)

4. 判断是否满足迭代终止条件。如果满足，循环结束，返回最佳参数𝐰t+1 和目标函数极小值𝐽(𝐰t+1) ；否则转到第2步。

• 迭代终止条件：

1. 迭代次数达到最大次数；

2. 目标函数值变化很小：

##### 岭回归与Lasso的梯度下降法

岭回归的梯度下降计算步骤同OLS，不再详述。

Lasso目标函数为：，其中绝对值函数 ||𝐰||1原点𝐰 = 𝟎处不可导，无法用梯度下降求解。Lasso函数求解可以用其他方法：

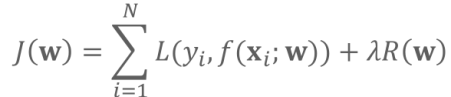
* 用次梯度概念替换梯度，得到次梯度法
* 用坐标轴下降求解

##### 梯度下降法实用技巧

1. 学习率𝜂需小心设置。太大可能引起目标函数值震荡，太小收敛速度慢。可采用自适应学习率（深度学习部分会学习更多自适应学习率算法）。

2. 梯度下降对特征的取值范围敏感。建议对输入特征**x**做去量纲（可用sklearn.preprocessing.StandardScaler 实现）。

#### ****随机梯度下降法****

我们把目标函数写成累加的形式：，那么其梯度公式如下：



由上式可以看出，若要求解 梯度 𝛻𝐽(𝐰t) ，**那么我们需要计算**𝛻L的累加。由于当样本中存在信息冗余（正负抵消或梯度相似）时效率不高，为了加快计算速度，我们可以选择一个随机的i (N为样本数，选择一个i就是计算一个样本上的梯度)，计算梯度：



为了确保收敛，相比于同等条件下的梯度下降，随机梯度下降需要采用更小的步长和更多的迭代轮数。相对于非随机算法，随机梯度下降在前期迭代效果卓越。

小批量梯度下降法：介于一次使用所有样本（批处理梯度下降）和一次只使用一个样本（随机梯度下降），实践中常采用小批量样本（mini-batch）

参考：<https://zhuanlan.zhihu.com/p/28060786>

#### ****次梯度法****

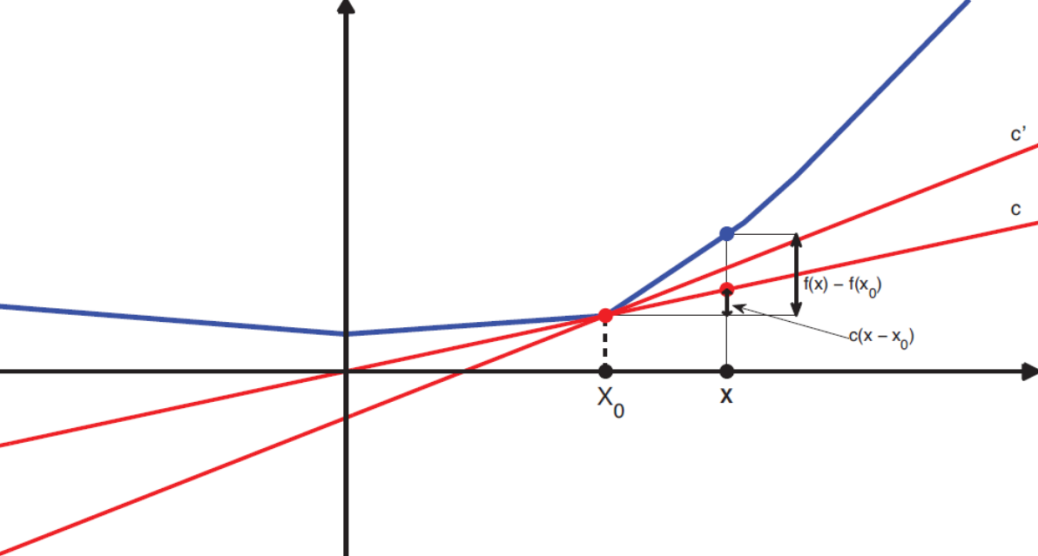
学习次梯度法前我们先熟悉几个知识点：

* 函数可导不可导怎么判断？

函数可导的条件是在定义域内必须是连续的，可导函数都是连续的,但是连续函数不一定是可导函数。例如，y=|x|,在x=0上不可导。即使这个函数是连续的,但是lim(x趋向0+)y'=1,lim(x趋向0-)y'=-1，两个值不相等，所以不是可导函数。也就是说在每一个点上导数的左右极限都相等的函数是可导函数，反之不是。

* 次导数

如果函数f(x)在某点不可导，那么在该点函数f(x)就不存在导数，但在该点会存在次导数。 次导数，即通过该不可导点(x0,f(x0))的一条直线的斜率，该直线要在函数图像之下或者和函数图像重合，但不能超过函数图像或与函数图像相交。即可以用下图来表示：



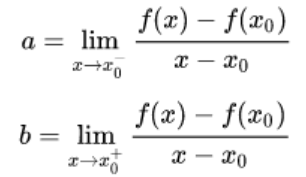
上图蓝线为f(x),可以看出在x0处，从左侧趋近x0和从右侧趋近x0其导数并不一样，所以x0处不可导；但x0处存在次导数，如图中红线的斜率就是f(x)在x0处的次导数，需要注意的是，次导数不只一个。

严格地说，凸函数f:I→R在点x0的次导数，是标量g使得：



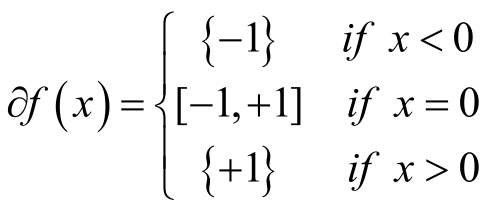
其中I为包含𝑥 0 的某个区间。

对于I内所有的x，我们可以证明，在点x0的次导数的集合是一个非空闭区间[a, b]，其中a和b是单侧极限。



它们一定存在并满足a≤b。 所有次导数的集合[a，b]称为函数f在x0处的*次微分*。

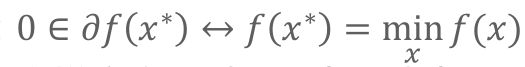
例如：对凸函数𝑓 (𝑥) = |𝑥|



当函数在𝑥 0 处可导时，次微分只有一个点组成，这个点就是函数在𝑥 0 处的导数。Python可用numpy.sign函数实现绝对值函数的次梯度

次导数的极值点

我们知道当导数为0时，可计算极值点，那么对应某点次导数为一个区间，当0属于此区间内时，则此点为极值点：



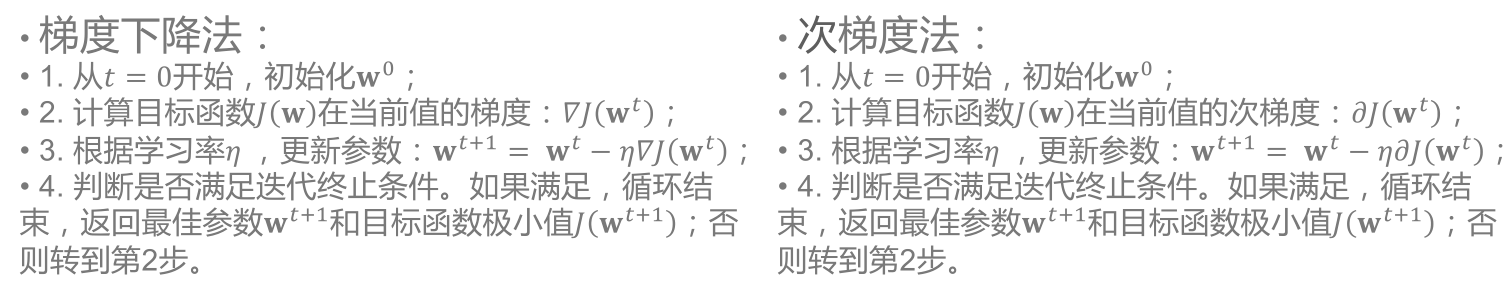
既当且仅当0属于函数𝑓在点𝑥∗ 处次梯度集合时，𝑥∗ 为极值点（𝑥∗表示不可导点）。

* 什么是次梯度

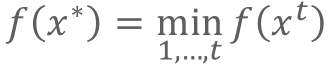
通过求函数在某点每个分量的次微分，可以求出函数在该点的次梯度。

##### 次梯度法求解线性回归模型

将梯度下降法中的梯度换成次梯度，就得到次梯度法。



与梯度下降算法不同，次梯度算法并不是下降算法（每次对参数的更新，并不能保证目标函数单调递减），因此一般情况下我们选择：



虽然不能保证次梯度法中目标函数单调下降，可以证明对满足一定条件的凸函数，次梯度法是收敛的，只是收敛速度比梯度下降法慢。

#### ****坐标轴下降法****

次梯度法收敛速度慢，Lasso求解推荐使用坐标轴下降法。

坐标轴下降法：沿坐标轴方向搜索

• 在每次迭代中，在当前点处沿一个坐标轴方向进行一维搜索。

• 循环使用不同的坐标轴。一个周期的一维搜索迭代过程相当于一个梯度迭代。

• 利用当前坐标系统进行搜索，无需计算目标函数的导数，只按照某一坐标方向进行搜索最小值。

• 梯度下降法：沿目标函数负梯度的方向搜索，梯度方向通常不与任何坐标轴平行。

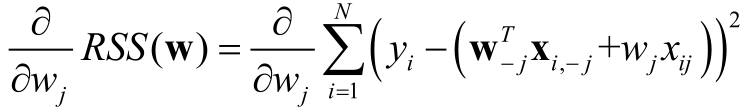
• 坐标下降法在稀疏矩阵上的计算速度非常快。

##### Lasso的坐标轴下降法求解

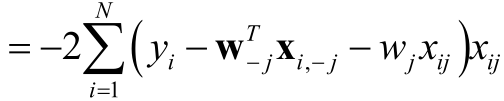
Lasso的目标函数为：

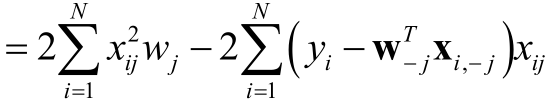
坐标轴下降法：每次搜索一个维度；

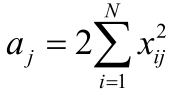
设𝐰-j, 表示𝐷维向量𝐰中去掉第𝑗维的其他(𝐷 − 1)维向量；类似的，𝐱i,-j 表示𝐷维向量𝐱-j 中去掉第𝑗维的其他(𝐷 − 1)维向量；对第𝑗维，目标函数第一部分为残差平方和𝑅𝑆𝑆(𝐰)：

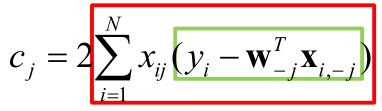


复合函数求导：f[g(x)]中,设g(x)=u,则f[g(x)]=f(u)，从而（公式）：f'[g(x)]=f'(u)\*g'(x)，那么上式可看成g(x)= ，可计算f[g(x)]=[g(x)]2导数，得：





设： 

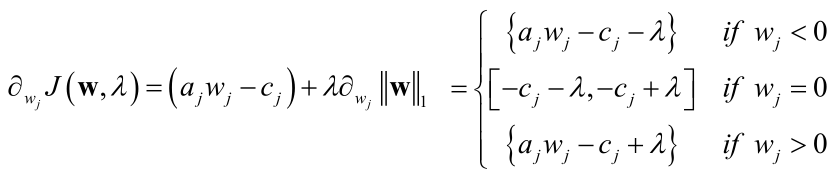


**第j维特征与残差的相关性**

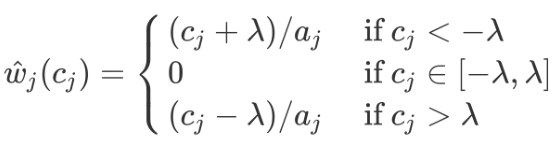
**利用去掉第j维后其他特征得到的预测的残差**

上式可写为：

Lasso目标函数为，其对应的的次梯度函数为：

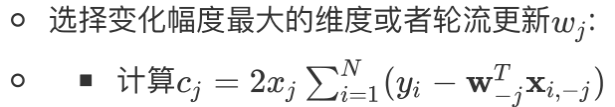


当0 ∈ 𝜕wj𝐽(𝐰)时，有极值，那么可将上述三种情况分别等于零，求wj。



以这种情况为例：ajwj-cj-λ=0，可得wj=(cj+λ)/aj，所以当wj取 (cj+λ)/aj时有极值点，但这种情况是wj小于0，所以(cj+λ)/aj < 0；因为aj 为平方和，必大于0；所以(cj+λ)<0,得到条件cj < λ。

##### 计算步骤（—需看代码补充理解—）

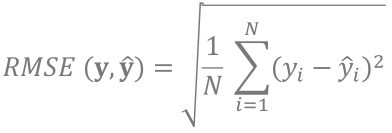
1. 预计算zj=∑Ni=1hj(xi)2
2. 初始化参数w（全0或随机）
3. 循环直到收敛: 

注意：当特征与预测残差弱相关，即𝑐j ∈ [−𝜆,𝜆]时，𝑤j = 0。当时，所有𝑤j = 0。其中𝐱 :j 表示所有样本第𝑗维的特征值， 𝐲表示所有样本的标签。

### 线性回归模型性能评价指标

回归模型的评价指标：

• 开方均方误差（ Rooted Mean Squared Error，RMSE ）：



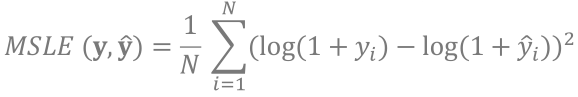
•平均绝对误差（ Mean Absolute Error，MAE ）：



• 绝对误差中值—对噪声不敏感（ Median Absolute Error，MedAE ）：

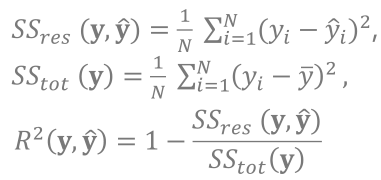


• 平均平方log误差（Mean Squared Logarithmic Error, MSLE）：

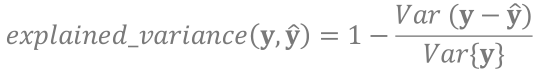


- 当𝑦呈指数增长时可以使用平均平方log误差（如计数、一年的平均销量…）

• R2分数（R2 > score）：既考虑了预测值与真值之间的差异，也考虑了问题本身真值之间的差异（ Scikit-Learn 线性回归模型的缺省评价准则）。最佳分数是1，越小越不好，可能为负值。



• 已解释的方差分数（Explained variance score）：最佳分数为1，越小越不好。



* 问题：已解释的方差分数为 1 – 预测模型方差/数据本身的方差 ；与R2相比有什么区别？？，R2分母是预测值的均值，已解释的方差分数分母是数据本身的均值？？？

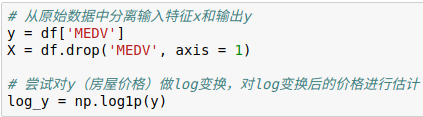
### 线性回归的基本步骤

1. 数据分析(DA—Data Analysis)
2. 工程特征(FE—feature engineering)

* 数据是否缺失：pandas.info() 查看是否有缺失数据
* 是否需要去噪声



* 数据分离 ：将输入特征X和输出Y进行分离，变成两组数



* 离散型数据编码：独热编码
* 数据型特征处理：去量纲

### 练习

1. 对数据做数据探索分析（可参考EDA\_BikeSharing.ipynb，不计分）

2. 适当的特征工程（可参考FE\_BikeSharing.ipynb，不计分）

3. 对全体数据，随机选择其中80%做训练数据，剩下20%为测试数据，评价指标为RMSE。（10分）

4. 用训练数据训练最小二乘线性回归模型（20分）、岭回归模型、Lasso模型，其中岭回归模型（30分）和Lasso模型（30分），注意岭回归模型和Lasso模型的正则超参数调优。

5. 比较用上述三种模型得到的各特征的系数，以及各模型在测试集上的性能。并简单说明原因。（10分）

机器学习开放课程：一、使用Pandas探索数据分析: <https://www.jqr.com/article/000079>

英文地址：<https://medium.com/open-machine-learning-course/open-machine-learning-course-topic-1-exploratory-data-analysis-with-pandas-de57880f1a68>