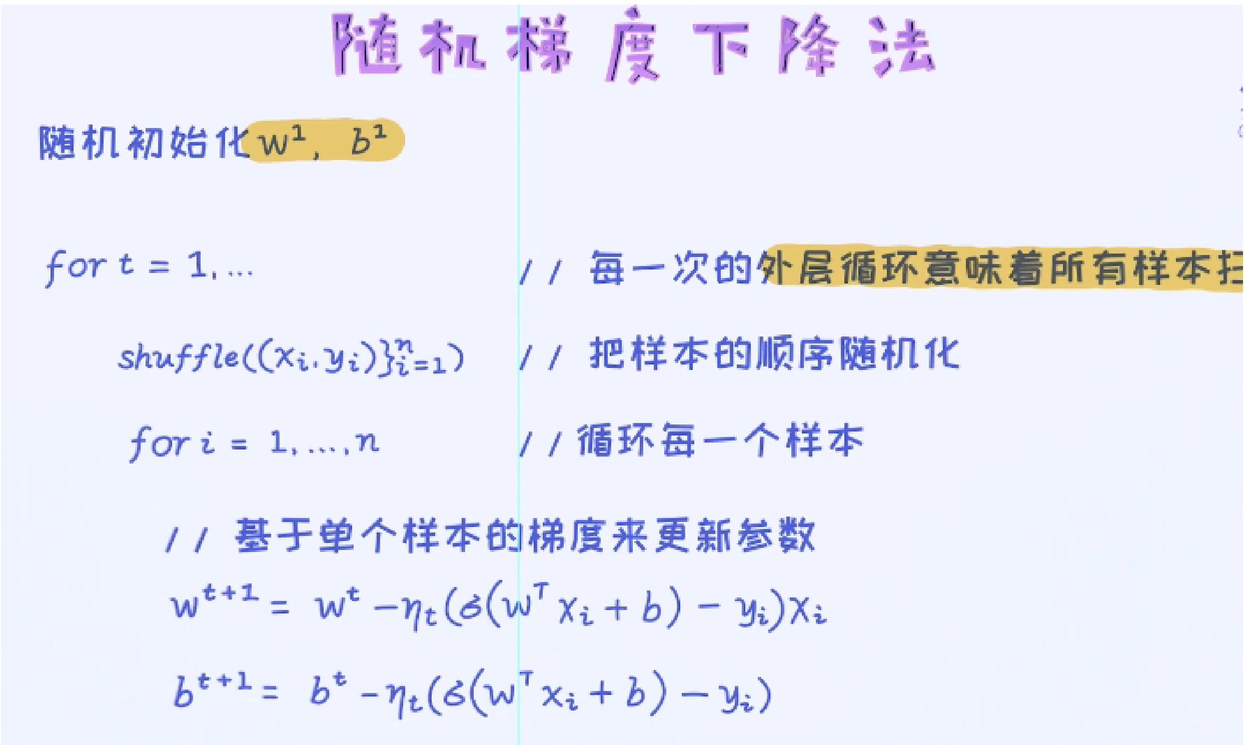
一、大数据的利器：随机梯度下降法

传统**梯度下降**全部样本循环迭代次数过多

随机梯度下降法核心：每一次的迭代更新不再依赖于所有样本的梯度之和，而是仅仅依赖于其中一个样本的梯度：

- 随机梯度下降法计算由于仅依赖其中一个样本，与使用整个样本计算的梯度会有偏差，即会产生梯度的噪声



小批量梯度下降法（mini-batch gradient descent）：它不依赖于所有的样本，但也不依赖于仅仅一个样本，而是它从所有样本中随机挑选一部分样本来计算梯度并更新参数--梯度下降和随机梯度下降的折中（两个极端的折中）

小批量梯度下降法

随机初始化 w^1, b^1

for $t = 1, \dots$ // 直到收敛

  $batch = subsample(D, m)$ // 从所有样本里随机采样m个样本

  // 基于 batch 计算梯度并更新参数

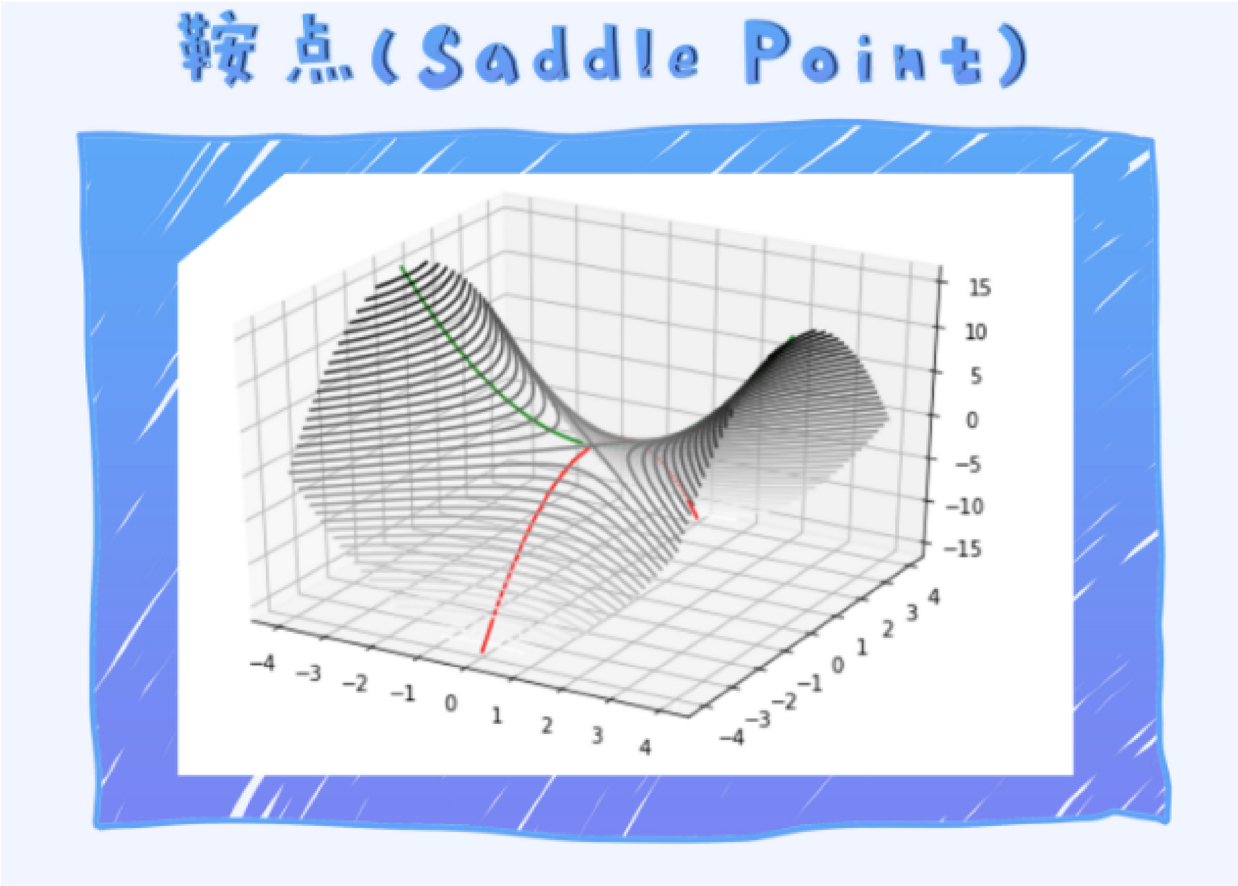
$$w^{t+1} = w^t - \eta_t \sum_{(x_i, y_i) \in batch} (\sigma(w^T x_i + b) - y_i) x_i$$

$$b^{t+1} = w^t - \eta_t \sum_{(x_i, y_i) \in batch} (\sigma(w^T x_i + b) - y_i)$$

三者比较：样本，时间复杂度，稳定性，计算速度，应用场景

梯度下降法	mini-batch梯度下降法	随机梯度下降法
N	m	1
O(N)	O(m)	O(1)
非常强	一般	弱
非常慢	快	快
一般	非常强	强

鞍点（saddle point），它的梯度为0，但并不是局部最优解，如上面图里绿色和红色的相交点：



二、多元逻辑回归

二分类改造成多分类：

- 不能单纯的添加标签数进行多分类，虽然概率落在（0，1）区间，但是SUM值不为1的话，改造就不合理
- 由此引进Softmax函数

多项逻辑回归

$$p(y=1|x,w)=\frac{e^{-(w_1^Tx+b_1)}}{\sum_{k=1}^{K}e^{-(w_k^Tx+b_k)}}$$

$$p(y=2|x,w)=\frac{e^{-(w_2^Tx+b_2)}}{\sum_{k=1}^{K}e^{-(w_k^Tx+b_k)}}$$

$$\cdots\cdots$$

$$p(y=K|x,w)=\frac{e^{-(w_K^Tx+b_K)}}{\sum_{k=1}^{K}e^{-(w_k^Tx+b_k)}}$$

这个叫Softmax函数

三、正则与过拟合

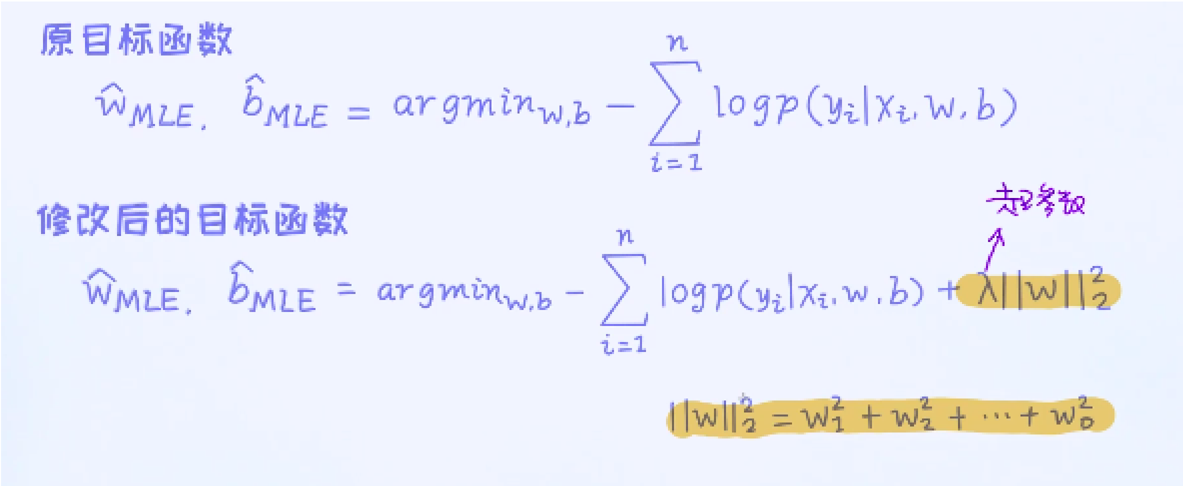
背景：在数据**线性可分**时，会有多种方法供选择，而算法会选择最大似然值（MLE），这时在逻辑回归建立函数模型时采用sigmoid，这时w变得非常大，趋近于∞，可以将两类完美分开，这时需要添加参数防止w过大。

$$P(y=1|x;w,b)=\frac{1}{1+e^{-(w^Tx+b)}}\approx 1$$

$$P(y=0|x;w,b)=\frac{e^{-(w^Tx+b)}}{1+e^{-(w^Tx+b)}}\approx 0$$

$$MLE\Rightarrow\sum_{i=1}^{n}\underbrace{y_i}_{\Downarrow\ 1}\log P(y_i=1|x_i;w,b)+\underbrace{(1-y_i)}_{\Downarrow\ 0}[1-\log P(y_i=1|x_i;w,b)]$$

解决办法：增加L2范数，防止w变得太大



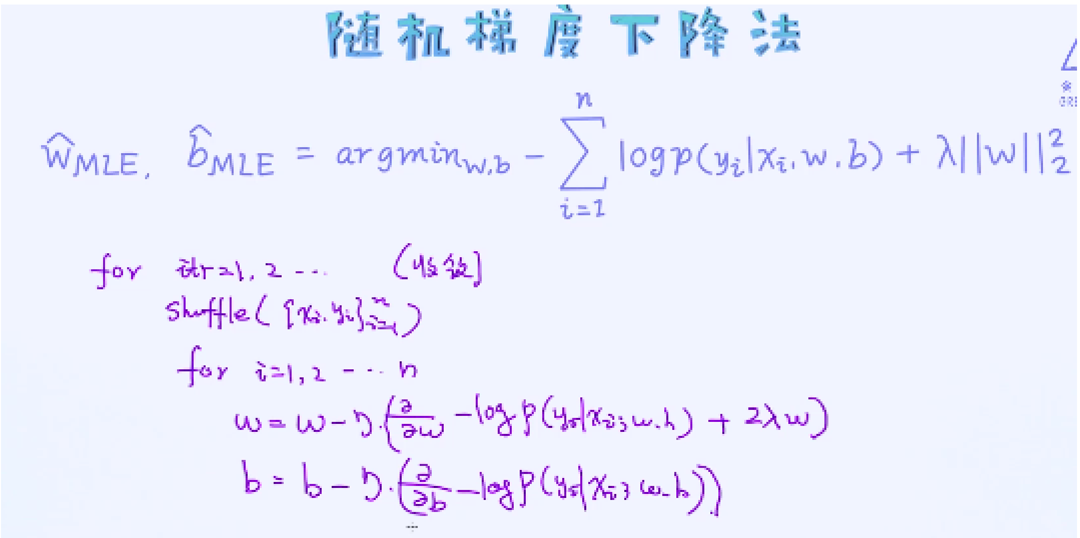
函数目的：求argmin 整个式子的最小值。

若λ->∞，导致w=0，即λ很大的时候，正则会很大即整个式子值很大这就违背了argmin。

λ=0，则w无限制，但w还是会变得非常大，这就是**模型的过拟合**（训练表现现的太好，而实际测试时并不是这样，**泛化能力差**）。

因此λ值要选择合适得值，一般采用交叉验证法来选取。

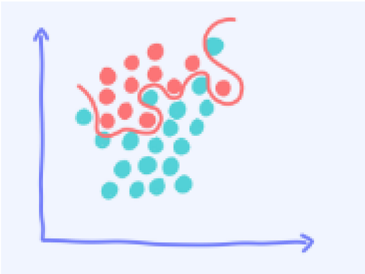
加完正则后随机梯度下降法：



构建泛化能力强的模型要点：

- 选择正确的数据
- 选择合适的模型
- 选择合适的优化算法
- 避免模型的过拟合

越复杂的模型越容易过拟合



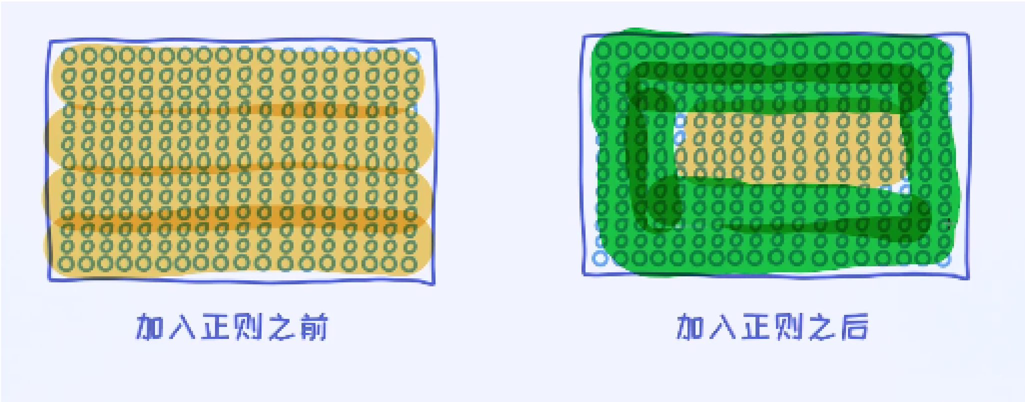
后果：模型的决策边界非常陡峭，而且这种陡峭的决策边界会带来不太稳定的模型效果。比如其中的一些点上加了一些噪声，让点的位置稍微偏移一点点，很有可能就会判断错误。

避免过拟合：

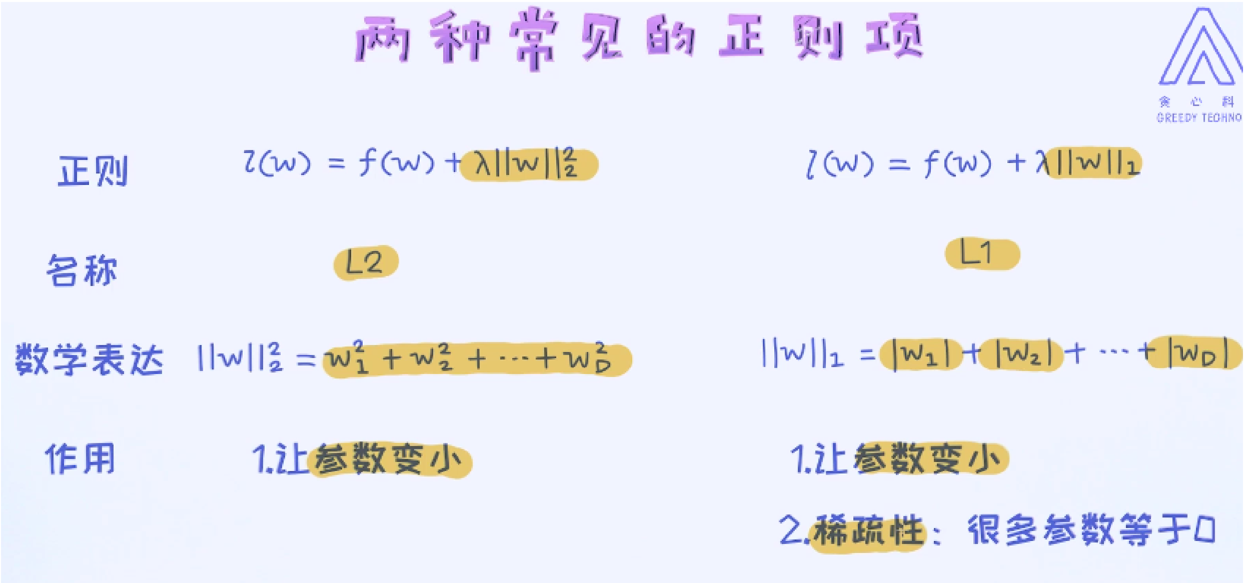
- 数据量的增加
- 使用更简单的模型
- 加入正则项

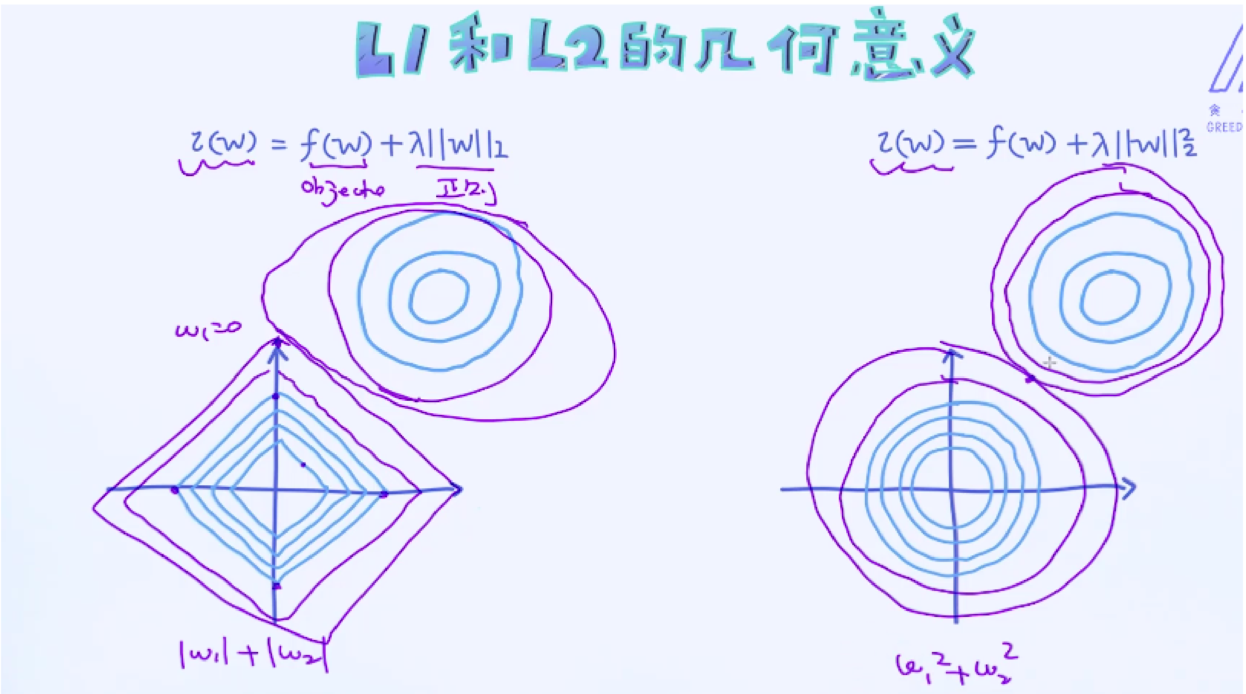
正则的作用：

- 对可行解空间的限制即缩小可行解空间
- 被丢弃的解空间是比较容易产生过拟合的（约束过）
- 正则是一种能够把先验知识加入到模型里的最直接的方式


加入正则之前
加入正则之后

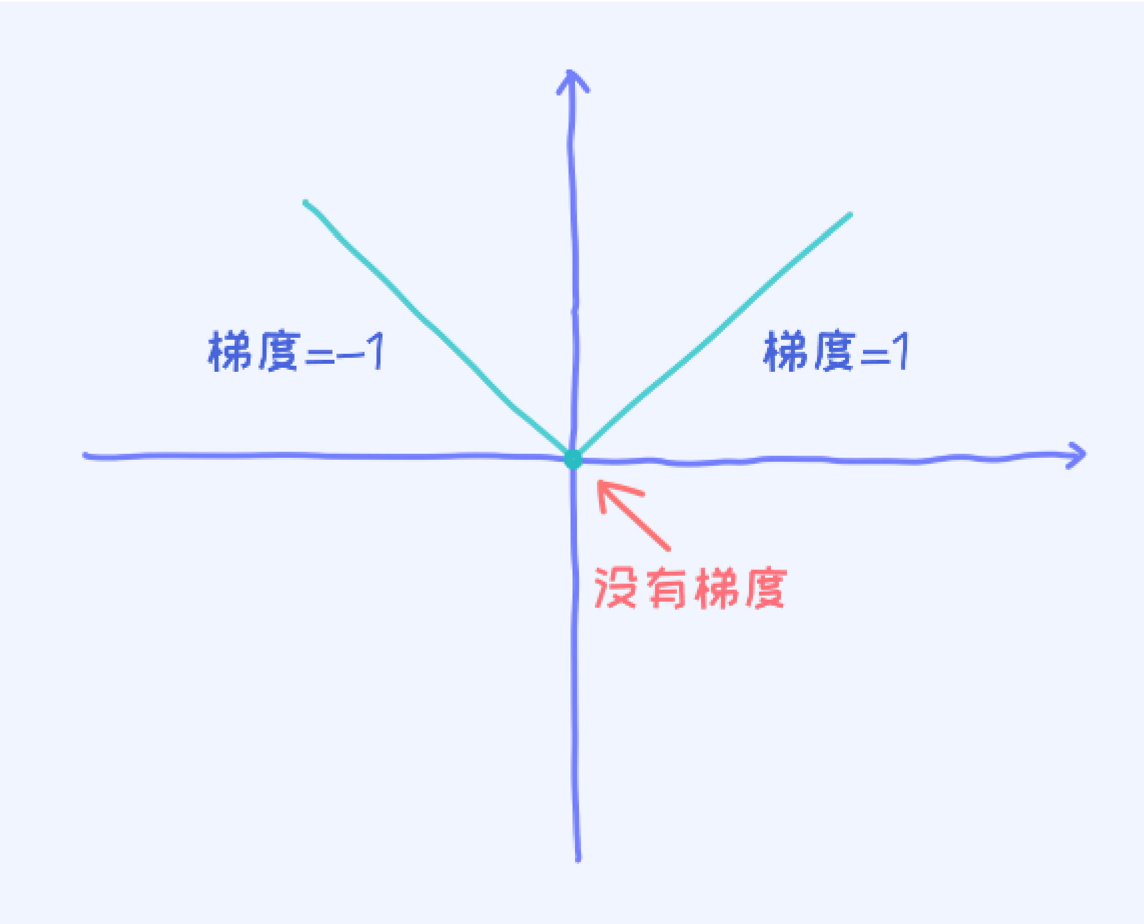
四、正则与后验概率


两种常见的正则项
正则
$l(w) = f(w) + \lambda\|w\|_2^2$
$l(w) = f(w) + \lambda\|w\|_1$
名称
L2
L1
数学表达
$\|w\|_2^2 = w_1^2 + w_2^2 + \cdots + w_D^2$
$\|w\|_1 = |w_1| + |w_2| + \cdots + |w_D|$
作用
1.让参数变小
1.让参数变小
2.稀疏性：很多参数等于0


L1和L2的几何意义
$l(w) = f(w) + \lambda\|w\|_1$
正则
$w_1=0$
$|w_1|+|w_2|$
$l(w) = f(w) + \lambda\|w\|_2^2$
$w_1^2+w_2^2$

由于L1与L2的几何图像的特殊性，L1带有顶点，与f（w)相交时，可能使某个wi参数为0，这就造成了L1参数会产生稀疏性。

比起L2，L1确实有更多的功效。但从计算的角度来讲，L1范数的挑战要大很多。一旦目标函数里包含了L1的正则，则优化起来会比较麻烦。主要的原因是L1范数在0点不具备梯度，所以需要做一些特殊处理，比如使用subgradient来代替梯度。



L1范数虽然有特征选择的功能，但也有一些不足。比如多个特征具有强相关性，那通过L1正则选出来的特征可能是这些特征里的任意个（**稀疏性的原因**），但实际上特征还是有好有坏。

扩展：联合L1和L2正则一起使用，这个模型就是非常著名的**ElasticNet，**ElasticNet又叫弹性网络回归:

线性回归最终目标是要最小化以下损失函数，基本思想还是最小二乘法：

$$\min_{\theta}\frac{1}{2m}\sum_{i=1}^{m}(\theta^{T}x^{(i)}-y^{(i)})^{2}$$

Lasso 回归是在该损失函数上加上 L1 正则化项：

$$\min_{\theta}\frac{1}{2m}\left[\sum_{i=1}^{m}(\theta^{T}x^{(i)}-y^{(i)})^{2}+\lambda\sum_{j=1}^{n}||\theta||\right]$$

岭回归是在该损失函数上加上 L2 正则化项：

$$\min_{\theta}\frac{1}{2m}\left[\sum_{i=1}^{m}(\theta^{T}x^{(i)}-y^{(i)})^{2}+\lambda\sum_{j=1}^{n}||\theta||_{2}^{2}\right]$$

有的时候，我们无法权衡好，到底是 L1 还是 L2 正则化对参数更新更有利，那么为什么不结合两者呢？这就是 ElasticNet 的思想：

$$\min_{\theta}\frac{1}{2m}\left[\sum_{i=1}^{m}(\theta^{T}x^{(i)}-y^{(i)})^{2}+\lambda_{1}\sum_{j=1}^{n}||\theta||+\lambda_{2}\sum_{j=1}^{n}||\theta||_{2}^{2}\right]$$

即同时结合了 L1 和 L2 正则化。

Sklearn库中有sklearn.linear_model.ElasticNetCV和sklearn.linear_model.ElasticNet两个函数可供选择，前者可以通过迭代选择最佳的 λ1 和λ2 （当然你可以指定一组值），后者需要你指定λ1 和λ2的值。