机器学习中正则化项L1和L2的直观理解

正则化（Regularization）

机器学习中几乎都可以看到损失函数后面会添加一个额外项，常用的额外项一般有两种，一般英文称作 ℓ_1**-norm** 和 ℓ_2**-norm**，中文称作 **L1正则化** 和 或者 **L1范数** 和 **L2范数**。

L1正则化和L2正则化可以看做是损失函数的惩罚项。所谓『惩罚』是指对损失函数中的某些参数做一些限制。对于线性回归模型，使用L1正则化的模 回归，使用L2正则化的模型叫做Ridge回归（岭回归）。下图是Python中Lasso回归的损失函数，式中加号后面一项 $\alpha||w||_1$即为L1正则化项。

$$\min_w \frac{1}{2n_{samples}}||Xw-y||_2^2+\alpha||w||_1$$

下图是Python中Ridge回归的损失函数，式中加号后面一项 $\alpha||w||_2^2$ 即为L2正则化项。

$$\min_w ||Xw-y||_2{}^2+\alpha||w||_2{}^2$$

一般回归分析中回归w表示特征的系数，从上式可以看到正则化项是对系数做了处理（限制）。**L1正则化和L2正则化的说明如下：**

- L1正则化是指权值向量w中各个元素的**绝对值之和**，通常表示为$||w||_1$
- L2正则化是指权值向量w中各个元素的**平方和然后再求平方根**（可以看到Ridge回归的L2正则化项有平方符号），通常表示为$||w||_2$

一般都会在正则化项之前添加一个系数，Python中用 α 表示，一些文章也用 λ 表示。这个系数需要用户指定。

那添加L1和L2正则化有什么用？**下面是L1正则化和L2正则化的作用**，这些表述可以在很多文章中找到。

- L1正则化可以产生稀疏权值矩阵，即产生一个稀疏模型，可以用于特征选择
- L2正则化可以防止模型过拟合（overfitting）；一定程度上，L1也可以防止过拟合

稀疏模型与特征选择

上面提到L1正则化有助于生成一个稀疏权值矩阵，进而可以用于特征选择。为什么要生成一个稀疏矩阵？

稀疏矩阵指的是很多元素为0，只有少数元素是非零值的矩阵，即得到的线性回归模型的大部分系数都是0. 通常机器学习中特征数量很多，例如文本处 一个词组（term）作为一个特征，那么特征数量会达到上万个（bigram）。在预测或分类时，那么多特征显然难以选择，但是如果代入这些特征得到 稀疏模型，表示只有少数特征对这个模型有贡献，绝大部分特征是没有贡献的，或者贡献微小（因为它们前面的系数是0或者是很小的值，即使去掉对 么影响），此时我们就可以只关注系数是非零值的特征。这就是稀疏模型与特征选择的关系。

L1和L2正则化的直观理解

这部分内容将解释**为什么L1正则化可以产生稀疏模型（L1是怎么让系数等于零的）**，以及**为什么L2正则化可以防止过拟合。**

##L1正则化和特征选择
假设有如下带L1正则化的损失函数：

$$J=J_0+\alpha\sum_w|w|$$

其中J_0是原始的损失函数，加号后面的一项是L1正则化项，α是正则化系数。注意到L1正则化是权值的**绝对值之和**，J是带有绝对值符号的函数，因
可微的。机器学习的任务就是要通过一些方法（比如梯度下降）求出损失函数的最小值。当我们在原始损失函数J_0后添加L1正则化项时，相当于对J_0
束。令$L=\alpha\sum_w|w|$，则$J=J_0+L$，此时我们的任务变成**在L约束下求出J_0取最小值的解**。考虑二维的情况，即只有两个权值w^1和w^2，此时L
$|w^2|$对于梯度下降法，求解J_0的过程可以画出等值线，同时L1正则化的函数L也可以在$w^1 w^2$的二维平面上画出来。如下图：

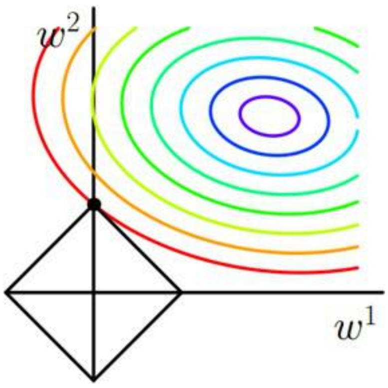

图1 L1正则化

图中等值线是J_0的等值线，黑色方形是L函数的图形。在图中，当J_0等值线与L图形首次相交的地方就是最优解。上图中J_0与L在L的一个顶处相交
是最优解。注意到这个顶点的值是$(w^1,w^2)=(0,w)$。可以直观想象，因为L函数有很多『突出的角』（二维情况下四个，多维情况下更多），J_0与
机率会远大于与L其它部位接触的机率，而在这些角上，会有很多权值等于0，这就是为什么L1正则化可以产生稀疏模型，进而可以用于特征选择。

而正则化前面的系数α，可以控制L图形的大小。α越小，L的图形越大（上图中的黑色方框）；α越大，L的图形就越小，可以小到 黑色 框只超出
点，这是最优点的值$(w1,w2)=(0,w)$中的w可以取到很小的值。

类似，假设有如下带L2正则化的损失函数：

$$J=J_0+\alpha\sum_w w^2$$

同样可以画出他们在二维平面上的图形，如下：

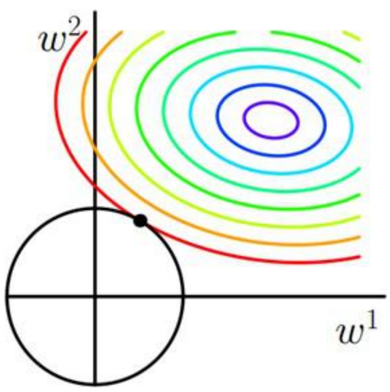

图2 L2正则化

二维平面下L2正则化的函数图形是个圆，与方形相比，被磨去了棱角。因此J_0与L相交时使得w^1或w^2等于零的机率小了许多，这就是为什么L2正则化的原因。

L2正则化和过拟合

拟合过程中通常都倾向于让权值尽可能小，最后构造一个所有参数都比较小的模型。因为一般认为参数值小的模型比较简单，能适应不同的数据集，也避免了过拟合现象。可以设想一下对于一个线性回归方程，若参数很大，那么只要数据偏移一点点，就会对结果造成很大的影响；但如果参数足够小，一点也不会对结果造成什么影响，专业一点的说法是『抗扰动能力强』。

那为什么L2正则化可以获得值很小的参数？

以线性回归中的梯度下降法为例。假设要求的参数为θ，$h_\theta(x)$是我们的假设函数。线性回归一般使用平方差损失函数。单个样本的平方差是$(h_\theta(x)$
考虑所有样本，损失函数是对每个样本的平方差求和，假设有m个样本，线性回归的代价函数如下，为了后续处理方便，乘以一个常数$\frac{1}{2}$：

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

在梯度下降算法中，需要先对参数求导，得到梯度。梯度本身是上升最快的方向，为了让损失尽可能小，沿梯度的负方向更新参数即可。

对于单个样本，先对某个参数 θ_j 求导：

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m}(h_\theta(x) - y)\frac{\partial}{\partial \theta_j} h_\theta(x)$$

注意到$h_\theta(x)$的表达式是$h_\theta(x) = \theta_0 x_0 + \theta_1 x_1 + \cdots + \theta_n x_n$.单个样本对某个参数 θ_j 求导，$\frac{\partial}{\partial \theta_j} h_\theta(x) = x_j$. 最终(3.1)式结果如下：

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m}(h_\theta(x) - y)x_j$$

在考虑所有样本的情况，将每个样本对 θ_j的导数求和即可，得到下式：

$$\frac{\partial}{\partial \theta_j} J(\theta) = \frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

梯度下降算法中，为了尽快收敛，会沿梯度的负方向更新参数，因此在(3.3)式前添加一个负号，并乘以一个系数 α （即学习率），得到最终用于迭代形式：

$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

其中 α 是learning rate. 上式是没有添加L2正则化项的迭代公式，如果在原始代价函数之后添加L2正则化，则迭代公式会变成下面的样子：

$$\theta_j := \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)}$$

其中 λ **就是正则化参数**。从上式可以看到，与未添加L2正则化的迭代公式相比，每一次迭代，θ_j 都要先乘以一个小于1的因子，从而使得 θ_j 不断减小来看，θ 是不断减小的。

最开始也提到L1正则化一定程度上也可以防止过拟合。之前做了解释，当L1的正则化系数很小时，得到的最优解会很小，可以达到和L2正则化类似的

正则化参数的选择

L1正则化参数

通常越大的 λ 可以让代价函数在参数为0时取到最小值。下面是一个简单的例子，这个例子来自Quora上的问答。为了方便叙述，一些符号跟这篇帖子致。

假设有如下带L1正则化项的代价函数：

$$F(x) = f(x) + \lambda ||x||_1$$

其中x是要估计的参数，相当于上文中提到的w以及 θ . 注意到L1正则化在某些位置是不可导的，当 λ 足够大时可以使得$F(x)$在$x = 0$时取到最小值。

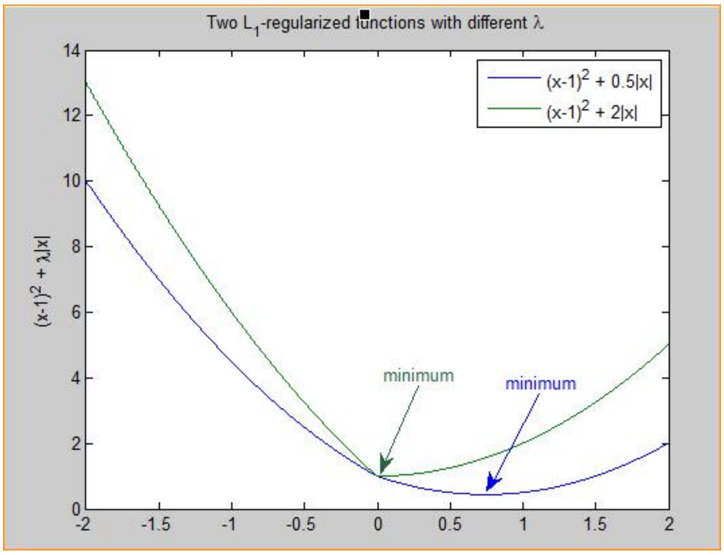

图3 L1正则化参数的选择

分别取$\lambda = 0.5$和$\lambda = 2$，可以看到越大的λ越容易使$F(x)$在$x = 0$时取到最小值。

L2正则化参数

从公式5可以看到，λ越大，θ_j衰减得越快。另一个理解可以参考图2，λ越大，L2圆的半径越小，最后求得代价函数最值时各参数也会变得很小。