Regularization正则化

笔记本:	机器学习		
创建时间:	2017/12/19 21:16	**更新时间:**	2017/12/20 7:45
作者:	beyourselfwb@163.com		

参考：http://blog.csdn.net/jinping_shi/article/details/52433975
参考：https://www.cnblogs.com/jianxinzhou/p/4083921.html

但更一般地说，如果我们像惩罚 θ_3 和 θ_4 这样惩罚其它参数，那么我们往往可以得到一个相对较为简单的假设。

实际上，这些参数的值越小，通常对应于越光滑的函数，也就是更加简单的函数。因此 就不易发生过拟合的问题。

我知道，为什么越小的参数对应于一个相对较为简单的假设，对你来说现在不一定完全理解，但是在上面的例子中使 θ_3 和 θ_4 很小，并且这样做能给我们一个更加简单的假设，这个例子至少给了我们一些直观感受。

来让我们看看具体的例子，对于房屋价格预测我们可能有上百种特征，与刚刚所讲的多项式例子不同，我们并不知道 θ_3 和 θ_4 是高阶多项式的项。所以，如果我们有一百个特征，我们并不知道如何选择关联度更好的参数，如何缩小参数的数目等等。

因此在正则化里，我们要做的事情，就是把减小我们的代价函数（例子中是线性回归的代价函数）所有的参数值，因为我们并不知道是哪一个或哪几个要去缩小。

因此，我们需要修改代价函数，在这后面添加一项，就像我们在方括号里的这项。当我们添加一个额外的正则化项的时候，我们收缩了每个参数。

Regularization.

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

$$\min_\theta J(\theta)$$

顺便说一下，按照惯例，我们没有去惩罚 θ_0，因此 θ_0 的值是大的。这就是一个约定从 1 到 n 的求和，而不是从 0 到 n 的求和。但其实在实践中这只会有非常小的差异，无论你是否包括这 θ_0 这项。但是按照惯例，通常情况下我们还是只从 θ_1 到 θ_n 进行正则化。

下面的这项就是一个正则化项

$$\lambda \sum_{j=1}^{n} \theta_j^2$$

并且 λ 在这里我们称做正则化参数。

λ 要做的就是控制在两个不同的目标中的平衡关系。

第一个目标就是我们想要训练，使假设更好地拟合训练数据。我们希望假设能够很好的适应训练集。

而第二个目标是我们想要保持参数值较小。（通过正则化项）

而 λ 这个正则化参数需要控制的是这两者之间的平衡，即平衡拟合训练的目标和保持参数值较小的目标。从而来保持假设的形式相对简单，来避免过度的拟合。

3. 最小二乘法与梯度下降法（最小均方法）

相同点：

a. 两种方法都是在给定已知数据（independent & dependent variables）的前提下对dependent variables算出出一个一般性的估值函数。然后对给定新数据的dependent variables进行估算。

b. 都是在已知数据的框架内，使得估算值与实际值的总平方差尽量更小。

不同点：

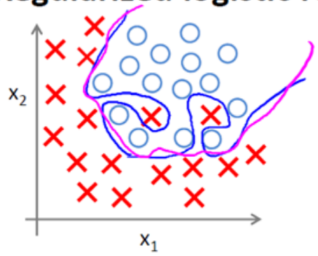
实现方法和结果不同：最小二乘法是直接对参数求导找出**全局最小**，是非迭代法。而梯度下降法是一种迭代法，先给定参数初始值，然后向下降最快的方向调整，在若干次迭代之后可能找到**局部最小**。梯度下降法的缺点是到最小点的时候收敛速度变慢，并且对初始点的选择极为敏感，其改进大多是在这两方面下功夫。

在接下来的视频中，我们将把这种正则化的想法应用到 Logistic 回归，这样我们就可以让 logistic 回归也避免过度拟合，从而表现的更好。

4. Regularized Logistic Regression

Regularized Logistic Regression 实际上与 Regularized Linear Regression 是十分相似的。

Regularized logistic regression.



$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_1^2 + \theta_3 x_1^2 x_2 + \theta_4 x_1^2 x_2^2 + \theta_5 x_1^2 x_2^3 + \dots)$$

Cost function:

$$J(\theta) = -\left[\frac{1}{m}\sum_{i=1}^{m} y^{(i)} \log h_\theta(x^{(i)}) + (1-y^{(i)}) \log(1-h_\theta(x^{(i)}))\right] + \frac{\lambda}{2m}\sum_{j=1}^{n}\theta_j^2$$

$\boxed{\theta_1, \theta_2, \dots, \theta_n}$

同样使用梯度下降：

Gradient descent

Repeat {

$$\theta_0 := \theta_0 - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_0^{(i)}$$

$$\theta_j := \theta_j - \alpha \underbrace{\left[\frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} - \frac{\lambda}{m}\theta_j \right]}_{\frac{\partial}{\partial \theta_j} J(\theta)}$$

$$(j = \cancel{0}, 1, 2, 3, \ldots, n)$$

$$\theta_1 \ldots \theta_n$$

}

$$h_\theta(x) = \frac{1}{1+e^{-\theta^T x}}$$

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***L2正则化***，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正则化的模型建叫做Lasso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，例如文本处理时，如果将一个词组（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的模型是一个稀疏模型，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，即使去掉对模型也没有什么影响），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：

$$J = J_0 + \alpha \sum_w |w| \tag{1}$$

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，因此J是不完全可微的。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做了一个约束。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$L = |w^1| + |w^2|$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：

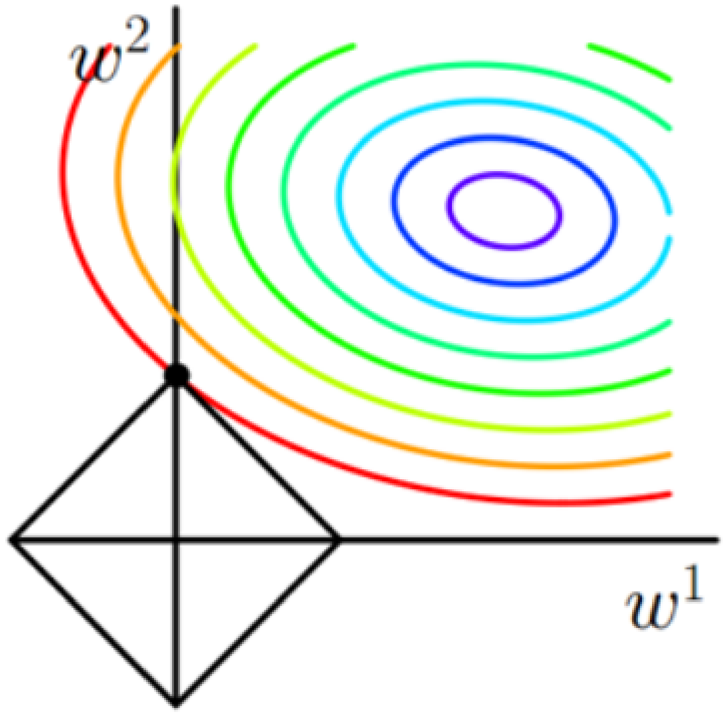


图1 L1正则化

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的一个顶点处相交，这个顶点就是最优解。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与这些角接触的机率会远大于与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，可以小到黑色方框只超出原点范围一点点，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：

$$J = J_0 + \alpha \sum_w w^2 \tag{2}$$

同样可以画出他们在二维平面上的图形，如下：

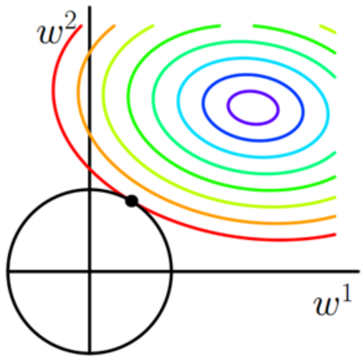


图2 L2正则化

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正则化不具有稀疏性的原因。

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，那么线性回归的代价函数如下：

$$J(\theta) = \frac{1}{2m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) \tag{3}$$

那么在梯度下降法中，最终用于迭代计算参数θ的迭代式为：

$$\theta_j := \theta_j - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \tag{4}$$

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：

$$\theta_j := \theta_j (1 - \alpha \frac{\lambda}{m}) - \alpha \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x^{(i)}) - y^{(i)}) x_j^{(i)} \tag{5}$$

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，因此总得来看，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正则化类似的效果。

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，那么线性回归的代价函数如下：

$$J(\theta) = \frac{1}{2m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)}) \tag{3}$$

那么在梯度下降法中，最终用于迭代计算参数θ的迭代式为：

$$\theta_j := \theta_j - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \tag{4}$$

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：

$$\theta_j := \theta_j(1 - \alpha\frac{\lambda}{m}) - \alpha\frac{1}{m}\sum_{i=1}^{m}(h_\theta(x^{(i)}) - y^{(i)})x_j^{(i)} \tag{5}$$

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，因此总得来看，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正则化类似的效果。