机器学习中的范数规则化之（一）L0、L1与L2范数博客的学习笔记，对一些要点进行摘录。规则化也有其他名称，比如统计学术中比较多的叫做增加惩罚项；还有现在比较多的正则化。

一、正则化背景

监督机器学习问题无非就是“minimizeyour error while regularizing your parameters”，也就是在规则化参数的同时最小化误差。最小化误差是为了让我们的模型拟合我们的训练数据，

而规则化参数是防止我们的模型过分拟合我们的训练数据。

问题背景：参数太多，会导致我们的模型复杂度上升，容易过拟合。

作用：

1、约束参数，降低模型复杂度。

2、规则项的使用还可以约束我们的模型的特性。这样就可以将人对这个模型的先验知识融入到模型的学习当中，强行地让学习到的模型具有人想要的特性，例如稀疏、低秩、平滑等等。

二、目标函数

一般来说，监督学习可以看做最小化下面的目标函数：

$$w^* = \arg\min_w \sum_i L(y_i, f(x_i; w)) + \lambda\Omega(w)$$

其中，第一项L(yi,f(xi;w)) 就是误差平方和；第二项则为惩罚项，对参数w的规则化函数Ω(w)去约束我们的模型尽量的简单。

机器学习的大部分带参模型都和这个不但形似，而且神似。是的，其实大部分无非就是变换这两项而已。

1、第一项-Loss函数

如果是Square loss，那就是最小二乘了；

如果是Hinge Loss，那就是著名的SVM了；

如果是exp-Loss，那就是牛逼的 Boosting了；

如果是log-Loss，那就是Logistic Regression了；还有等等。不同的loss函数，具有不同的拟合特性，这个也得就具体问题具体分析的。但这里，我们先不究loss函数的问题，我们把目光转向“规则项Ω(w)”。

2、第二项-规则化函数Ω(w)
一般是模型复杂度的单调递增函数，模型越复杂，规则化值就越大。比如，规则化项可以是模型参数向量的范数。然而，不同的选择对参数w的约束不同，取得的效果也不同，但我们在论文中常见的都聚集在：零范数、一范数、二范数、迹范数、Frobenius范数和核范数等等。这么多范数，到底它们表达啥意思？具有啥能力？什么时候才能用？什么时候需要用呢？不急不急，下面我们挑几个常见的娓娓道来。

——

三、L0/L1范数

1、分别定义

L0范数是指向量中非0的元素的个数。如果我们用L0范数来规则化一个参数矩阵W的话，就是希望W的大部分元素都是0。都为稀疏。

L1范数是指向量中各个元素绝对值之和，也有个美称叫“稀疏规则算子”（Lasso regularization）。

2、两者关系：

为什么L1范数会使权值稀疏？有人可能会这样给你回答“它是L0范数的最优凸近似”。

任何的L0规则化算子，如果他在Wi=0的地方不可微（L1），并且可以分解为一个“求和”的形式，那么这个规则化算子就可以实现稀疏。

L1范数和L0范数可以实现稀疏，L1因具有比L0更好的优化求解特性而被广泛应用。

3、参数稀疏的好处

1）特征选择(Feature Selection)：

大家对稀疏规则化趋之若鹜的一个关键原因在于它能实现特征的自动选择。一般来说，xi的大部分元素（也就是特征）都是和最终的输出yi没有关系或者不提供任何信息的，在最小化目标函数的时候考虑xi这些额外的特征，虽然可以获得更小的训练误差，但在预测新的样本时，这些没用的信息反而会被考虑，从而干扰了对正确yi的预测。稀疏规则化算子的引入就是为了完成特征自动选择的光荣使命，它会学习地去掉这些没有信息的特征，也就是把这些特征对应的权重置为0。

2）可解释性(Interpretability)：

另一个青睐于稀疏的理由是，模型更容易解释。例如患某种病的概率是y，然后我们收集到的数据x是1000维的，也就是我们需要寻找这1000种因素到底是怎么影响患上这种病的概率的。假设我们这个是个回归模型：y=w1*x1+w2*x2+…+w1000*x1000+b（当然了，为了让y限定在[0,1]的范围，一般还得加个Logistic函数）。通过学习，如果最后学习到的w*就只有很少的非零元素，例如只有5个非零的wi，那么我们就有理由相信，这些对应的特征在患病分析上面提供的信息是巨大的，决策性的。也就是说，患不患这种病只和这5个因素有关，那医生就好分析多了。但如果1000个wi都非0，医生面对这1000种因素，累觉不爱。

四、L1（Lasso）、L2（岭回归）范数

L2范数是指向量各元素的平方和然后求平方根。我们让L2范数的规则项||W||2最小，可以使得W的每个元素都很小，都接近于0，但与L1范数不同，它不会让它等于0，而是接近于0。

L2的作用=参数变小=模型变简单≈模型参数信息变少。

L2的作用：

1、L2范数不但可以防止过拟合，还可以让我们的优化求解变得稳定和快速。

2、优化计算的角度。L2范数有助于处理 condition number不好的情况下矩阵求逆很困难的问题。

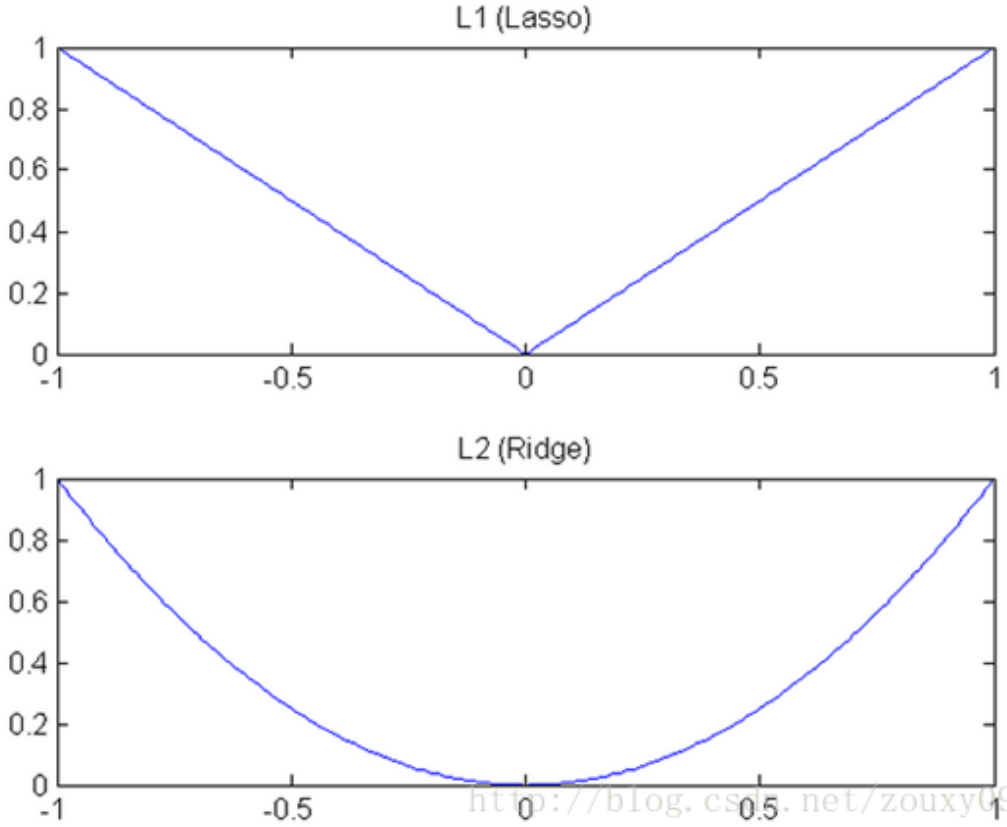
（condition number：condition number衡量的是输入发生微小变化的时候，输出会发生多大的变化。也就是系统对微小变化的敏感度。condition number值小的就是well-conditioned的，大的就是ill-conditioned的。

如果一个矩阵的condition number在1附近，那么它就是well-conditioned的，如果远大于1，那么它就是ill-conditioned的，如果一个系统是ill-conditioned的，它的输出结果就不要太相信了。）

五、Lasso算法和岭回归算法区别

1、梯度下降速度

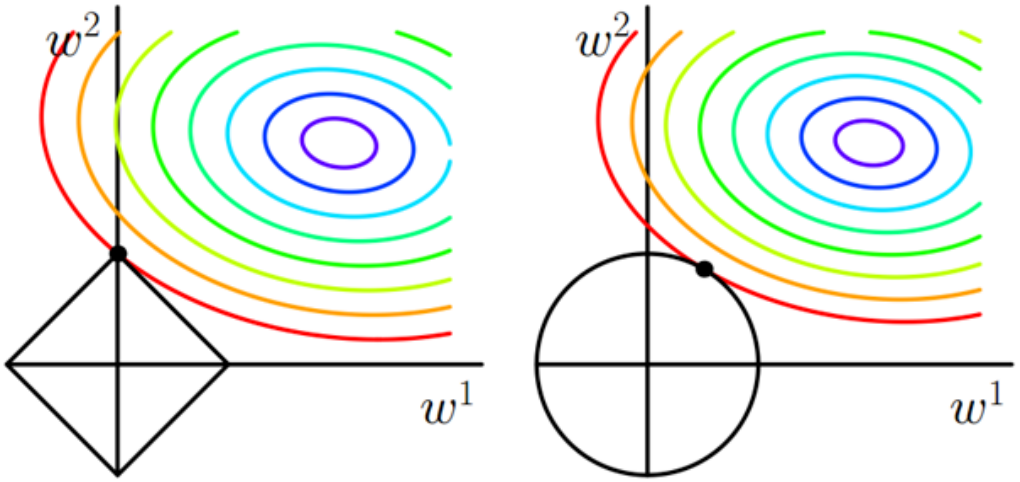
L1和L2的差别就在于这个“坡”不同，如下图：L1就是按绝对值函数的“坡”下降的，而L2是按二次函数的“坡”下降。所以实际上在0附近，L1的下降速度比L2的下降速度要快。所以会非常快得降到0。



L1在江湖上人称Lasso，L2人称Ridge。不过这两个名字还挺让人迷糊的，看上面的图片，Lasso的图看起来就像ridge，而ridge的图看起来就像lasso。

2、模型空间的限制

为了便于可视化，我们考虑两维的情况，在(w1, w2)平面上可以画出目标函数的等高线，而约束条件则成为平面上半径为C的一个 norm ball 。等高线与 norm ball 首次相交的地方就是最优解：



(a) ℓ_1-ball meets quadratic function. ℓ_1-ball has corners. It's very likely that the meet-point is at one of the corners.

(b) ℓ_2-ball meets quadratic function. ℓ_2-ball has no corner. It is very unlikely that the meet-point is on any of axes.

可以看到，L1-ball 与L2-ball 的不同就在于L1在和每个坐标轴相交的地方都有“角”出现，而目标函数的测地线除非位置摆得非常好，大部分时候都会在角的地方相交。

注意到在角的位置就会产生稀疏性，例如图中的相交点就有w1=0，而更高维的时候（想象一下三维的L1-ball 是什么样的？）除了角点以外，还有很多边的轮廓也是既有很大的概率成为第一次相交的地方，又会产生稀疏性。

相比之下，L2-ball 就没有这样的性质，因为没有角，所以第一次相交的地方出现在具有稀疏性的位置的概率就变得非常小了。这就从直观上来解释了为什么L1-regularization能产生稀疏性，而L2-regularization 不行的原因了。

因此，一句话总结就是：L1会趋向于产生少量的特征，而其他的特征都是0，而L2会选择更多的特征，这些特征都会接近于0。Lasso在特征选择时候非常有用，而Ridge就只是一种规则化而已。

延伸一：L1&L2正则化一起结合的Elastic Nets效果真的很好吗？

一般来说，如果L1和L2对比，L2比L1要好一些，因为L2之后，精度更好且较好适应、拟合。L1的效果在处理稀疏数据时候比较棒，且有利于稀疏数据的特征。

那么从理论上来说，L1+L2=Elastic Nets的办法，既可以处理稀疏问题，同时也可以保证精度。

但是，实际上引入超参数会难以适当，至少在超参数搜索上成本较高，**实际案例中很少有L1+L2的效果优于L2的。**Elastic Nets会更为复杂，不过如果你看中的是Elastic Nets网络中的特征选择功能，那么复杂度提升+精度下降可能也是可以接受的。

内容来源：Quora问答、Xavier Amatriain