# 一 范数

## 1 机器学习中为什么要使用范数

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（又叫正则化）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。

因为参数太多，会导致我们的模型复杂度上升，容易过拟合，也就是我们的训练误差会很小。但训练误差小并不是我们的最终目标，我们的目标是希望模型的测试误差小，也就是能准确的预测新的样本。所以，我们需要保证模型“简单”的基础上最小化训练误差，这样得到的参数才具有好的泛化性能（也就是测试误差也小）.

而模型“简单”就是通过规则函数来实现的。另外，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。要知道，有时候人的先验是非常重要的。前人的经验会让你少走很多弯路，这就是为什么我们平时学习最好找个大牛带带的原因。一句点拨可以为我们拨开眼前乌云，还我们一片晴空万里，[醍醐灌顶](https://www.baidu.com/s?wd=%E9%86%8D%E9%86%90%E7%81%8C%E9%A1%B6&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)。对机器学习也是一样，如果被我们人稍微点拨一下，它肯定能更快的学习相应的任务。只是由于人和机器的交流目前还没有那么直接的方法，目前这个媒介只能由规则项来担当了。

还有几种角度来看待规则化的。规则化符合奥卡姆剃刀(Occam’s razor)原理。不过它的思想很平易近人：在所有可能选择的模型中，我们应该选择能够很好地解释已知数据并且十分简单的模型。从贝叶斯估计的角度来看，规则化项对应于模型的先验概率。民间还有个说法就是，规则化是结构风险最小化策略的实现，是在经验风险上加一个正则化项(regularizer)或惩罚项(penalty term)。

## 2 什么是范数？

我们知道距离的定义是一个宽泛的概念，只要满足非负、自反、三角不等式就可以称之为距离。范数是一种强化了的距离概念，它在定义上比距离多了一条数乘的运算法则。有时候为了便于理解，我们可以把范数当作距离来理解。

在数学上，范数包括向量范数和矩阵范数，向量范数表征向量空间中向量的大小，矩阵范数表征矩阵引起变化的大小。一种非严密的解释就是，对应向量范数，向量空间中的向量都是有大小的，这个大小如何度量，就是用范数来度量的，不同的范数都可以来度量这个大小，就好比米和尺都可以来度量远近一样；对于矩阵范数，学过线性代数，我们知道，通过运算AX=BAX=BAX=B，可以将向量X变化为B，矩阵范数就是来度量这个变化大小的。

## 2.常见的范数

### 2.1 L0范数与L1范数

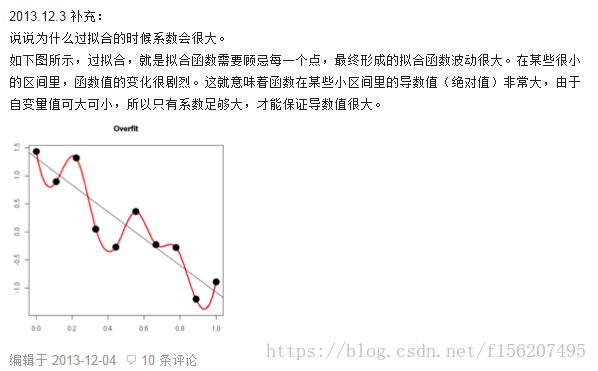
L0范数是指向量中非0的元素的个数。如果我们用L0范数来规则化一个参数矩阵W的话，就是希望W的大部分元素都是0。这太直观了，太露骨了吧，换句话说，让参数W是稀疏的。OK，看到了“稀疏”二字，大家都应该从当下风风火火的“压缩感知”和“稀疏编码”中醒悟过来，原来用的漫山遍野的“稀疏”就是通过这玩意来实现的。但你又开始怀疑了，是这样吗？看到的papers世界中，稀疏不是都通过L1范数来实现吗？脑海里是不是到处都是||W||1影子呀！几乎是[抬头不见低头见](https://www.baidu.com/s?wd=%E6%8A%AC%E5%A4%B4%E4%B8%8D%E8%A7%81%E4%BD%8E%E5%A4%B4%E8%A7%81&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)。没错，这就是这节的题目把L0和L1放在一起的原因，因为他们有着某种不寻常的关系。那我们再来看看L1范数是什么？它为什么可以实现稀疏？为什么大家都用L1范数去实现稀疏，而不是L0范数呢？  
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）。现在我们来分析下这个价值一个亿的问题：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”。实际上，还存在一个更美的回答：任何的规则化算子，如果他在Wi=0的地方不可微，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。这说是这么说，W的L1范数是绝对值，|w|在w=0处是不可微。后面会将L1和L2进行对比分析得出为什么L1可以实现稀疏。  
对了，上面还有一个问题：既然L0可以实现稀疏，为什么不用L0，而要用L1呢？上面也提到了一是因为L0范数很难优化求解（NP难问题），二是L1范数是L0范数的最优凸近似，而且它比L0范数要容易优化求解。所以大家才把目光和万千宠爱转于L1范数。  
OK，来个一句话总结：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。

好，到这里，我们大概知道了L1可以实现稀疏，但我们会想呀，为什么要稀疏？让我们的参数稀疏有什么好处呢？这里扯两点  
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。

### 2.2 L2范数

除了L1范数，还有一种更受宠幸的规则化范数是L2范数: ||W||2（L0范数：||W||0、 L1范数：||W||1）。它也不逊于L1范数，它有两个美称，在回归里面，有人把有它的回归叫“岭回归”（Ridge Regression），有人也叫它“权值衰减weight decay”。这用的很多吧，因为它的强大功效是改善机器学习里面一个非常重要的问题：过拟合。  至于过拟合是什么，上面也解释了，就是模型训练时候的误差很小，但在测试的时候误差很大，也就是我们的模型复杂到可以拟合到我们的所有训练样本了，但在实际预测新的样本的时候，糟糕的[一塌糊涂](https://www.baidu.com/s?wd=%E4%B8%80%E5%A1%8C%E7%B3%8A%E6%B6%82&tn=24004469_oem_dg&rsv_dl=gh_pl_sl_csd)。通俗的讲就是应试能力很强，实际应用能力很差。擅长背诵知识，却不懂得灵活利用知识。  
OK，那现在到我们非常关键的问题了，为什么L2范数可以防止过拟合？  
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，这里是有很大的区别的哦。而越小的参数说明模型越简单，越简单的模型则越不容易产生过拟合现象。为什么越小的参数说明模型越简单？



通过L2范数，我们可以实现了对模型空间的限制，从而在一定程度上避免了过拟合。

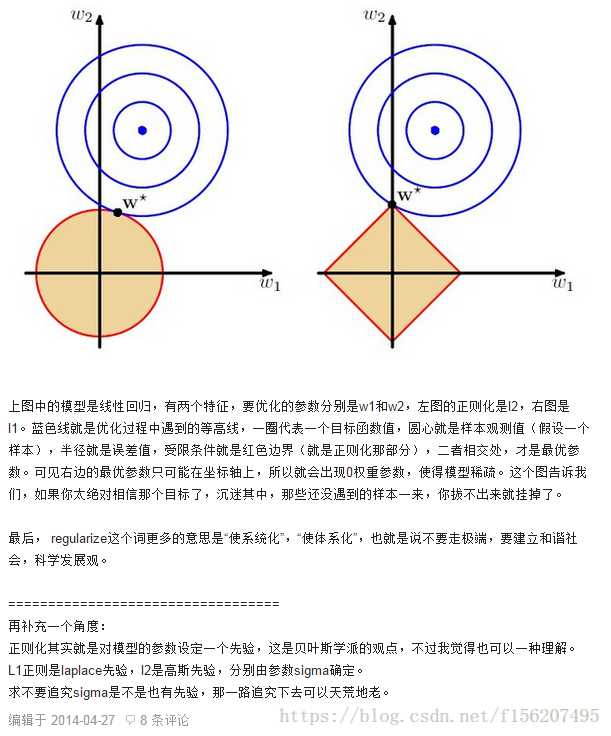
L2范数的好处是什么呢？这里也扯上两点：

1）学习理论的角度：

从学习理论的角度来说，L2范数可以防止过拟合，提升模型的泛化能力。

2）优化计算的角度：

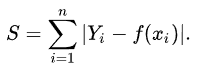
从优化或者数值计算的角度来说，L2范数有助于处理 condition number不好的情况下矩阵求逆很困难的问题。



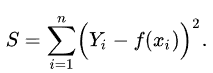
可以简化为一元函数理解原理。

### 作为损失函数

L1范数损失函数，也被称为最小绝对值偏差（LAD），最小绝对值误差（LAE）。总的说来，它是把目标值与估计值的绝对差值的总和最小化：



L2范数损失函数，也被称为最小平方误差（LSE）。总的来说，它是把目标值与估计值的差值的平方和最小化：

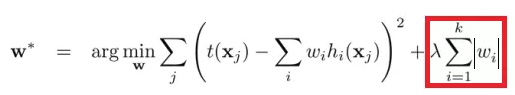


| **L2损失函数** | **L1损失函数** |
| --- | --- |
| 不是非常的鲁棒（robust） | 鲁棒 |
| 稳定解 | 不稳定解 |
| 总是一个解 | 可能多个解 |

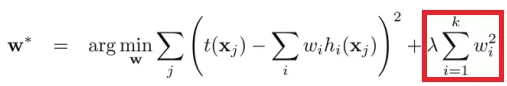
### 作为正规化

在机器学习中，正规化是防止过拟合的一种重要技巧。从数学上讲，它会增加一个正则项，防止系数拟合得过好以至于过拟合。L1与L2的区别只在于，L2是权重的平方和，而L1就是权重的和。如下：

最小平方损失函数的L1正则化：



最小平方损失函数的L2正则化：



| **L2正则化** | **L1正则化** |
| --- | --- |
| 计算效率高（因为有解析解） | 在非稀疏情形下计算效率低 |
| 非稀疏输出 | 稀疏输出 |
| 无特征选择 | 内置特征选择 |

# 机器学习中 L1 和 L2 正则化的直观解释

机器学习中，如果参数过多，模型过于复杂，容易造成过拟合（overfit）。即模型在训练样本数据上表现的很好，但在实际测试样本上表现的较差，不具备良好的泛化能力。为了避免过拟合，最常用的一种方法是使用使用正则化，例如 L1 和 L2 正则化。但是，正则化项是如何得来的？其背后的数学原理是什么？L1 正则化和 L2 正则化之间有何区别？本文将给出直观的解释。

1. L2 正则化直观解释

L2 正则化公式非常简单，直接在原来的损失函数基础上加上权重参数的平方和：



其中，Ein 是未包含正则化项的训练样本误差，λ 是正则化参数，可调。但是正则化项是如何推导的？接下来，我将详细介绍其中的物理意义。

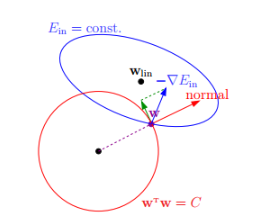
我们知道，正则化的目的是限制参数过多或者过大，避免模型更加复杂。例如，使用多项式模型，如果使用 10 阶多项式，模型可能过于复杂，容易发生过拟合。所以，为了防止过拟合，我们可以将其高阶部分的权重 w 限制为 0，这样，就相当于从高阶的形式转换为低阶。

为了达到这一目的，最直观的方法就是限制 w 的个数，但是这类条件属于 NP-hard 问题，求解非常困难。所以，一般的做法是寻找更宽松的限定条件：



上式是对 w 的平方和做数值上界限定，即所有w 的平方和不超过参数 C。这时候，我们的目标就转换为：最小化训练样本误差 Ein，但是要遵循 w 平方和小于 C 的条件。

下面，我用一张图来说明如何在限定条件下，对 Ein 进行最小化的优化。



如上图所示，蓝色椭圆区域是最小化 Ein 区域，红色圆圈是 w 的限定条件区域。在没有限定条件的情况下，一般使用梯度下降算法，在蓝色椭圆区域内会一直沿着 w 梯度的反方向前进，直到找到全局最优值 wlin。例如空间中有一点 w（图中紫色点），此时 w 会沿着 -∇Ein 的方向移动，如图中蓝色箭头所示。但是，由于存在限定条件，w 不能离开红色圆形区域，最多只能位于圆上边缘位置，沿着切线方向。w 的方向如图中红色箭头所示。

那么问题来了，存在限定条件，w 最终会在什么位置取得最优解呢？也就是说在满足限定条件的基础上，尽量让 Ein 最小。

我们来看，w 是沿着圆的切线方向运动，如上图绿色箭头所示。运动方向与 w 的方向（红色箭头方向）垂直。运动过程中，根据向量知识，只要 -∇Ein 与运行方向有夹角，不垂直，则表明 -∇Ein 仍会在 w 切线方向上产生分量，那么 w 就会继续运动，寻找下一步最优解。只有当 -∇Ein 与 w 的切线方向垂直时，-∇Ein在 w 的切线方向才没有分量，这时候 w 才会停止更新，到达最接近 wlin 的位置，且同时满足限定条件。



-∇Ein 与 w 的切线方向垂直，即 -∇Ein 与 w 的方向平行。如上图所示，蓝色箭头和红色箭头互相平行。这样，根据平行关系得到：

−∇Ein+λw=0

−∇Ein+λw=0

移项，得：

∇Ein+λw=0

∇Ein+λw=0

这样，我们就把优化目标和限定条件整合在一个式子中了。也就是说只要在优化 Ein 的过程中满足上式，就能实现正则化目标。

接下来，重点来了！根据最优化算法的思想：梯度为 0 的时候，函数取得最优值。已知 ∇Ein 是 Ein 的梯度，观察上式，λw 是否也能看成是某个表达式的梯度呢？

当然可以！λw 可以看成是 1/2λw\*w 的梯度：



这样，我们根据平行关系求得的公式，构造一个新的损失函数：



之所以这样定义，是因为对 Eaug 求导，正好得到上面所求的平行关系式。上式中等式右边第二项就是 L2 正则化项。

这样， 我们从图像化的角度，分析了 L2 正则化的物理意义，解释了带 L2 正则化项的损失函数是如何推导而来的。

2. L1 正则化直观解释

L1 正则化公式也很简单，直接在原来的损失函数基础上加上权重参数的绝对值：

L=Ein+λ∑j|wj|

L=Ein+λ∑j|wj|

我仍然用一张图来说明如何在 L1 正则化下，对 Ein 进行最小化的优化。

Ein 优化算法不变，L1 正则化限定了 w 的有效区域是一个正方形，且满足 |w| < C。空间中的点 w 沿着 -∇Ein 的方向移动。但是，w 不能离开红色正方形区域，最多只能位于正方形边缘位置。其推导过程与 L2 类似，此处不再赘述。