第一章　机器学习中的范数规则化之L0、L1与L2范数

今天我们聊聊机器学习中出现的非常频繁的问题：过拟合与规则化。我们先简单的来理解下常用的L0、L1、L2和核范数规则化。最后聊下规则化项参数的选择问题。这里因为篇幅比较庞大，为了不吓到大家，我将这个五个部分分成两篇博文。知识有限，以下都是我一些浅显的看法，如果理解存在错误，希望大家不吝指正。谢谢。

监督机器学习问题无非就是“minimize your 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。多么简约的哲学啊！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。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。这名字好霸气，razor！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)。

一般来说，监督学习可以看做最小化下面的目标函数：

$$w^* = \arg\min_w \sum_i L(y_i, f(x_i; w)) + \lambda\Omega(w) \tag{1.1}$$

其中，第一项$L(y_i, f(x_i; w))$衡量我们的模型（分类或者回归）对第i个样本的预测值$f(x_i; w)$和真实的标签yi之前的误差。因为我们的模型是要拟合我们的训练样本的嘛，所以我们要求这一项最小，也就是要求我们的模型尽量的拟合我们的训练数据。但正如上面说言，我们不仅要保证训练误差最小，我们更希望我们的模型测试误差小，所以我们需要加上第二项，也就是对参数w的规则化函数Ω(w)去约束我们的模型尽量的简单。

OK，到这里，如果你在机器学习浴血奋战多年，你会发现，哎哟哟，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。对于第一项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$\Omega(w)$”。

规则化函数$\Omega(w)$也有很多种选择，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。

1.1 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。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$W_i = 0$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，但这还是不够直观。这里因为我们需要和L2范数进行对比分析。所以关于L1范数的直观理解，请待会看看第二节。

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？个人理

解一是因为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。

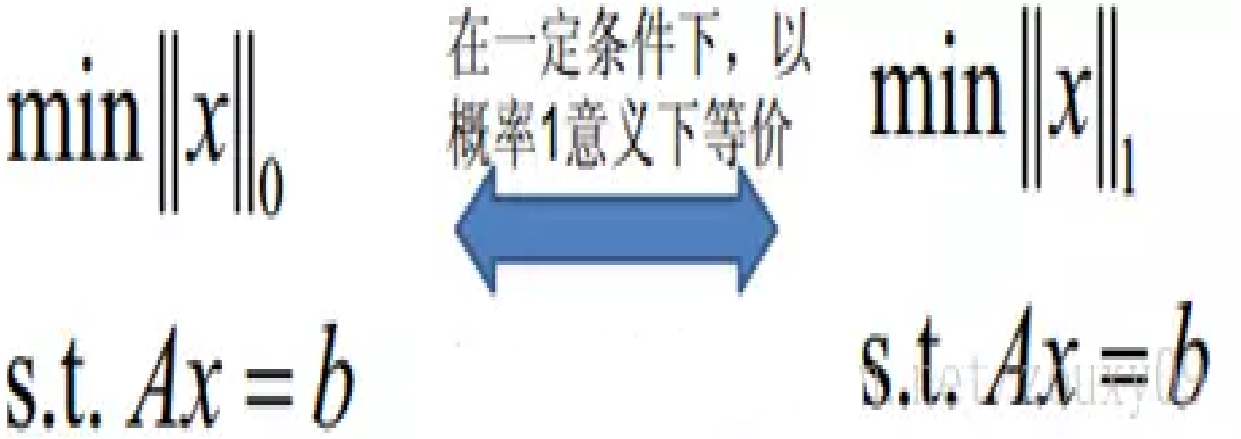


图 1.1: L0L1范数

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：

1.1.1 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，x_i的大部分元素（也就是特征）都是和最终的输出y_i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x_i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y_i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。

1.1.2 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：$y = w_1 * x_1 + w_2 * x_2 + \cdots + w_{1000} * x_{1000} + b$（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w_i，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w_i都非0，医生面对这1000种因素，累觉不爱。

1.2 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: $\|W\|_2$。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。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。例如下图所示（来自Ng的course）：

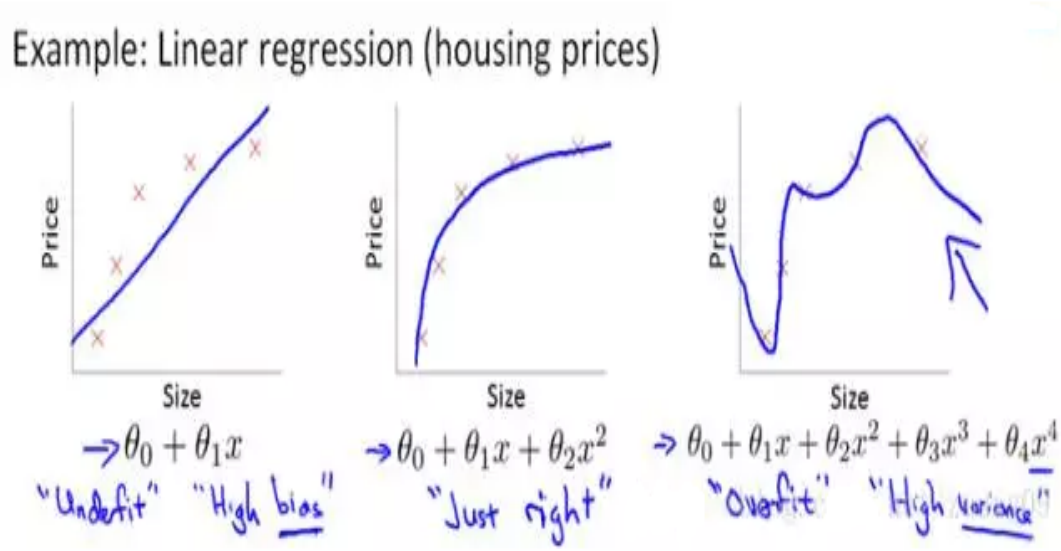


图 1.2: Overfitting linear regression

上面的图是线性回归，下面的图是Logistic回归，也可以说是分类的情况。从左到右分别是欠拟合（underfitting，也称High-bias）、合适的拟合和过拟合（overfitting，也称High variance）三种情况。可以看到，如果模型复杂（可以拟合任意的复杂函数），它可以让我们的模型拟合所有的数据点，也就是基本上没有误差。对于回归来说，就是我们的函数曲线通过了所有的数据点，如上图右。对分类来说，就是我们的函数曲线要把所有的数据点都分类正确，如下图右。这两种情况很明显过拟合了。

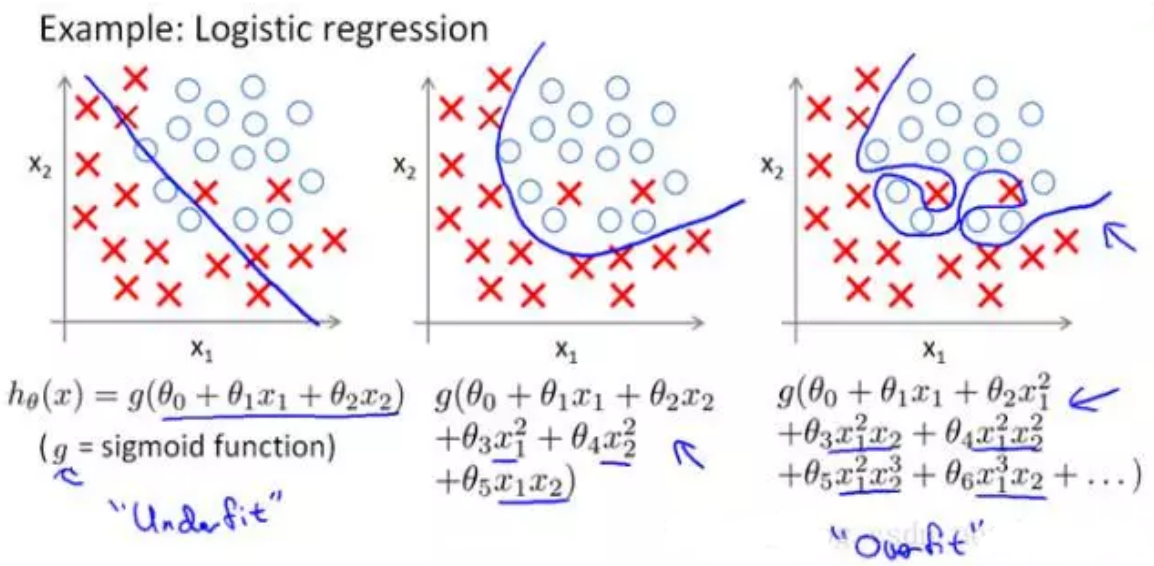


图 1.3: Overfitting logistic regression

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？回答这个问题之前，我们得先看看L2范数是个什么东西。

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，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？我也不懂，我的理解是：限制了参数很小，实际上就限制了多项式某些分量的影响很小（看上面线性回归的模型的那个拟合的图），这样就相当于减少参数个数。其实我也不太懂，希望大家可以指点下。

这里也一句话总结下：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。哎，等等，这condition number是啥？我先google一下哈。

这里我们也故作高雅的来聊聊优化问题。优化有两大难题，一是：局部最小值，二是：ill-condition病态问题。前者俺就不说了，大家都懂吧，我们要找的是全局最小值，如果局部最小值太多，那我们的优化算法就很容易陷入局部最小而不能自拔，这很明显不是观众愿意看到的剧情。那下面我们来聊聊ill-condition。ill-condition对应的是well-condition。那他们分别代表什么？假设我们有个方程组AX=b，我们需要求解X。如果A或者b稍微的改变，会使

得X的解发生很大的改变，那么这个方程组系统就是ill-condition的，反之就是well-condition的。我们具体举个例子吧：

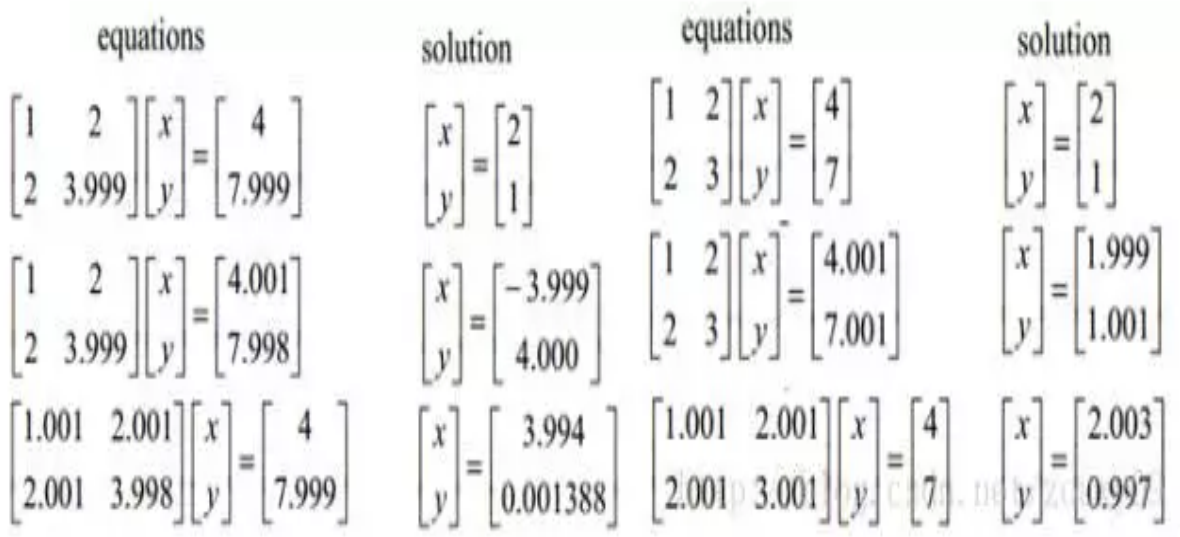


图 1.4: ill condition

咱们先看左边的那个。第一行假设是我们的AX=b，第二行我们稍微改变下b，得到的x和没改变前的差别很大，看到吧。第三行我们稍微改变下系数矩阵A，可以看到结果的变化也很大。换句话来说，这个系统的解对系数矩阵A或者b太敏感了。又因为一般我们的系数矩阵A和b是从实验数据里面估计得到的，所以它是存在误差的，如果我们的系统对这个误差是可以容忍的就还好，但系统对这个误差太敏感了，以至于我们的解的误差更大，那这个解就太不靠谱了。所以这个方程组系统就是ill-conditioned病态的，不正常的，不稳定的，有问题的，哈哈。这清楚了吧。右边那个就叫well-condition的系统了。

还是再啰嗦一下吧，对于一个ill-condition的系统，我的输入稍微改变下，输出就发生很大的改变，这不好啊，这表明我们的系统不能实用啊。你想想看，例如对于一个回归问题y=f(x)，我们是用训练样本x去训练模型f，使得y尽量输出我们期待的值，例如0。那假如我们遇到一个样本x'，这个样本和训练样本x差别很小，面对他，系统本应该输出和上面的y差不多的值的，例如0.00001，最后却给我输出了一个0.9999，这很明显不对呀。就好像，你很熟悉的一个人脸上长了个青春痘，你就不认识他了，那你大脑就太差劲了，哈哈。所以如果一个系统是ill-conditioned病态的，我们就会对它的结果产生怀疑。那到底要相信它多少呢？我们得找个标准来衡量吧，因为有些系统的病没那么重，它的结果还是可以相信的，不能一刀切吧。终于回来了，上面的condition number就是拿来衡量ill-condition系统的可信度的。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。

如果方阵A是非奇异的，那么A的condition number定义为：

$$\kappa(A) = \|A\|\|A^{-1}\| \tag{1.2}$$

也就是矩阵A的norm乘以它的逆的norm。所以具体的值是多少，就要看你选择的norm是什么了。如果方阵A是奇异的，那么A的condition number就是正无穷大了。实际上，每一个可逆方阵都存在一个condition number。但如果要计算它，我们需要先知道这个方阵的norm（范数）和Machine Epsilon（机器的精度）。为什么要范数？范数就相当于衡量一个矩阵的大小，我们知道矩阵是没有大小的，当上面不是要衡量一个矩阵A或者向量b变化的时候，我们的解x变化的大小吗？所以肯定得要有一个东西来度量矩阵和向量的大小吧？对了，他就是范数，表示矩阵大小或者向量长度。OK，经过比较简单的证明，对于$AX = b$，我们可以得到以下的结论：

$$\frac{\|\Delta x\|}{\|x\|} \leq \|A\| \cdot \|A^{-1}\| \cdot \frac{\|\Delta b\|}{\|b\|}, \frac{\|\Delta x\|}{\|x\|} \leq \kappa(A) \cdot \frac{\|\Delta b\|}{\|b\|}, \frac{\|\Delta x\|}{\|x + \Delta x\|} \leq \kappa(A) \cdot \frac{\|\Delta A\|}{\|A\|} \tag{1.3}$$

也就是我们的解x的相对变化和A或者b的相对变化是有像上面那样的关系的，其中$\kappa(A)$的值就相当于倍率，看到了吗？相当于x变化的界。

对condition number来个一句话总结：condition number是一个矩阵（或者它所描述的线性系统）的稳定性或者敏感度的度量，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。

好了，对这么一个东西，已经说了好多了。对了，我们为什么聊到这个的了？回到第一句话：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。因为目标函数如果是二次的，对于线性回归来说，那实际上是有解析解的，求导并令导数等于零即可得到最优解为：

$$\hat{w} = (X^T X)^{-1} X^T y \tag{1.4}$$

然而，如果当我们的样本X的数目比每个样本的维度还要小的时候，矩阵$X^T X$将会不是满秩的，也就是$X^T X$会变得不可逆，所以w*就没办法直接计算出来了。或者更确切地说，将会有无穷多个解（因为我们方程组的个数小于未知数的个数）。也就是说，我们的数据不足以确定一个解，如果我们从所有可行解里随机选一个的话，很可能并不是真正好的解，总而言之，我们过拟合了。

但如果加上L2规则项，就变成了下面这种情况，就可以直接求逆了：

$$w^* = (X^T X + \lambda I)^{-1} X^T y \tag{1.5}$$

这里面，专业点的描述是：要得到这个解，我们通常并不直接求矩阵的逆，而是通过解线性方程组的方式（例如高斯消元法）来计算。考虑没有规则项的时候，也就是 λ =0的情况，如

果矩阵X^TX的 condition number 很大的话，解线性方程组就会在数值上相当不稳定，而这个规则项的引入则可以改善condition number。

另外，如果使用迭代优化的算法，condition number 太大仍然会导致问题：它会拖慢迭代的收敛速度，而规则项从优化的角度来看，实际上是将目标函数变成 λ -strongly convex（ λ 强凸）的了。哎哟哟，这里又出现个 λ 强凸，啥叫 λ 强凸呢？

当f满足：

$$f(y) \geq f(x) + \leq \Delta f(x), y - x \geq + \frac{\lambda}{2}\|y - x\|^2 \tag{1.6}$$

时，我们称f为 λ -strongly convex函数，其中参数 λ >0。当 λ =0时退回到普通convex 函数的定义。

在直观的说明强凸之前，我们先看看普通的凸是怎样的。假设我们让f在x的地方做一阶泰勒近似（一阶泰勒展开忘了吗？$f(x) = f(a) + f'(a)(x - a) + o(\|x - a\|)$。

$$f(y) \geq f(x) + \leq \Delta f(x), y - x \geq + o(\|y - x\|) \tag{1.7}$$

直观来讲，convex 性质是指函数曲线位于该点处的切线，也就是线性近似之上，而 strongly convex 则进一步要求位于该处的一个二次函数上方，也就是说要求函数不要太“平坦”而是可以保证有一定的“向上弯曲”的趋势。专业点说，就是convex 可以保证函数在任意一点都处于它的一阶泰勒函数之上，而strongly convex可以保证函数在任意一点都存在一个非常漂亮的二次下界quadratic lower bound。当然这是一个很强的假设，但是同时也是非常重要的假设。可能还不好理解，那我们画个图来形象的理解下。

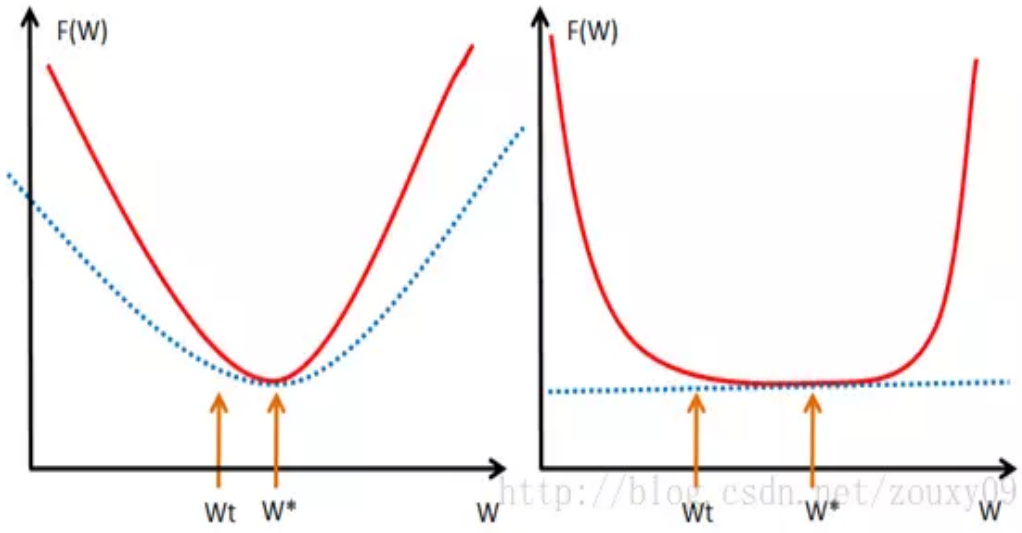


图 1.5: convex

大家一看到上面这个图就全明白了吧。不用我啰嗦了吧。还是啰嗦一下吧。我们取我们的最优解w^*的地方。如果我们的函数$f(w)$，见左图，也就是红色那个函数，都会位于蓝色虚

线的那根二次函数之上，这样就算wt和w^*离的比较近的时候，$f(w_t)$和f(w*)的值差别还是挺大的，也就是会保证在我们的最优解w^*附近的时候，还存在较大的梯度值，这样我们才可以在比较少的迭代次数内达到w^*。但对于右图，红色的函数f(w)只约束在一个线性的蓝色虚线之上，假设是如右图的很不幸的情况（非常平坦），那在wt还离我们的最优点w^*很远的时候，我们的近似梯度$(f(w_t) - f(w^*))/(w_t - w^*)$就已经非常小了，在wt处的近似梯度$\frac{\partial f}{\partial w}$就更小了，这样通过梯度下降$w_{t+1} = w_t - \alpha \cdot \frac{\partial f}{\partial w}$，我们得到的结果就是w的变化非常缓慢，像蜗牛一样，非常缓慢的向我们的最优点w*爬动，那在有限的迭代时间内，它离我们的最优点还是很远。

所以仅仅靠convex 性质并不能保证在梯度下降和有限的迭代次数的情况下得到的点w会是一个比较好的全局最小点w*的近似点（插个话，有地方说，实际上让迭代在接近最优的地方停止，也是一种规则化或者提高泛化性能的方法）。正如上面分析的那样，如果f(w)在全局最小点w*周围是非常平坦的情况的话，我们有可能会找到一个很远的点。但如果我们有“强凸”的话，就能对情况做一些控制，我们就可以得到一个更好的近似解。至于有多好嘛，这里面有一个bound，这个 bound 的好坏也要取决于strongly convex性质中的常数α的大小。看到这里，不知道大家学聪明了没有。如果要获得strongly convex怎么做？最简单的就是往里面加入一项$\frac{\alpha}{2} * \|w\|^2$。

呃，讲个strongly convex花了那么多的篇幅。实际上，在梯度下降中，目标函数收敛速率的上界实际上是和矩阵X^TX的 condition number有关，X^TX的 condition number 越小，上界就越小，也就是收敛速度会越快。

这一个优化说了那么多的东西。还是来个一句话总结吧：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。

好了，这里兑现上面的承诺，来直观的聊聊L1和L2的差别，为什么一个让绝对值最小，一个让平方最小，会有那么大的差别呢？我看到的有两种几何上直观的解析：

1）下降速度：

我们知道，L1和L2都是规则化的方式，我们将权值参数以L1或者L2的方式放到代价函数里面去。然后模型就会尝试去最小化这些权值参数。而这个最小化就像一个下坡的过程，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。不过我觉得这里解释的不太中肯，当然了也不知道是不是自己理解的问题。

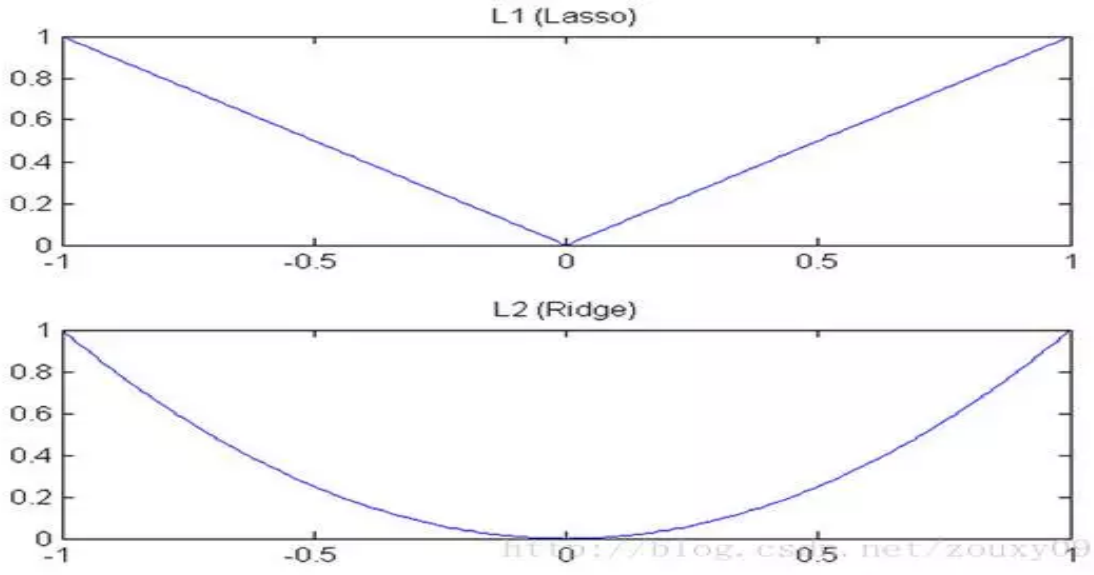


图 1.6: lasso ridge

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。

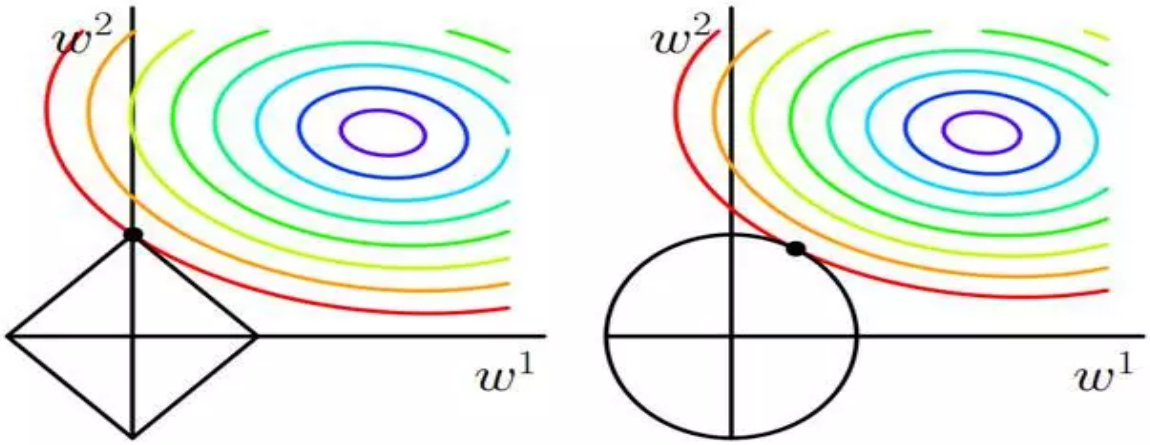
2）模型空间的限制：

实际上，对于L1和L2规则化的代价函数来说，我们可以写成以下形式：

$$Lasso : \min_{w} = \frac{1}{n}\|y - Xw\|^2, s.t. \|w\|_1 \leq C \tag{1.8}$$

$$Ridge : \min_{w} = \frac{1}{n}\|y - Xw\|^2, s.t. \|w\|_2 \leq C \tag{1.9}$$

也就是说，我们将模型空间限制在w的一个L1-ball 中。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.

图 1.7: L1L2 quadratic

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。