# 11.特征选择与稀疏学习

## 11.1.子集搜索与评价

处理高维数据的两大主流技术：**降维与特征选择**。

降维从一定程度起到了提炼优质低维属性和降噪的效果，特征选择则是直接剔除那些与学习任务无关的属性而选择出最佳特征子集。

先介绍几个概念

“特征”（feature）：属性

“相关特征”（relevant feature）：对当前学习任务有用的属性

“无关特征”（irrelevant feature）：对当前学习任务没什么用的属性

“特征选择”（feature selection）：从给定的特征集合中选择出相关特征子集的过程

特征选择是一个重要的“数据预处理”（data preprocessing）过程。在现实机器学习任务中，获得数据之后通常先进行特征选择，此后再训练学习器。那么，为什么要进行特征选择呢？

有两个很重要的原因：

首先，我们在现实任务中经常会遇到维数灾难问题，这是由于属性过多而造成的，若能从中选择出重要的特征，使得后续学习过程仅需在一部分特征上构建模型，则维数灾难问题会大为减轻。

第二个原因是，去除不相关特征往往会降低学习任务的难度，这就像侦探破案一样，若将纷繁复杂的因素抽丝剥茧，只留下关键因素，则真相往往更易看清.

需注意的是，特征选择过程必须确保不丢失重要特征，否则后续学习过程会因为重要信息的缺失而无法获得好的性能.给定数据集，若学习任务不同，则相关特征很可能不同，因此，特征选择中所谓的“无关特征”是指与当前学习任务无关（或“冗余特征”（redundant feature））。

欲从初始的特征集合中选取一个包含了所有重要信息的特征子集，若没有任何领域知识作为先验假设，那就只好遍历所有可能的子集了；然而这在计算上却是不可行的，因为这样做会遭遇组合爆炸，特征个数稍多就无法进行。

这时就涉及到了两个关键环节：

1.“子集搜索”（subset search）问题；

2.“子集评价”（subset evaluation）问题。

首先，对于如何生成候选子集，可以利用贪心策略（每一步都选择最好的）

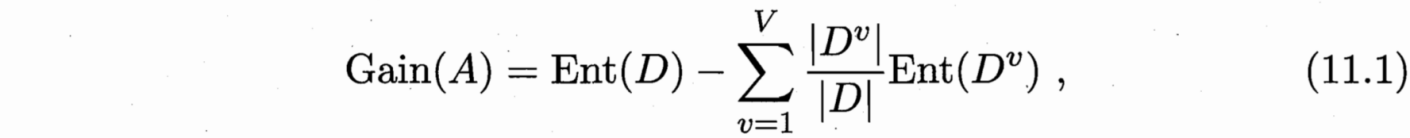
**前向搜索（forward）：**初始将每个特征当做一个候选特征子集，然后从当前所有的候选子集中选择出最佳的特征子集；接着在上一轮选出的特征子集中添加一个新的特征，同样地选出最佳特征子集；最后直至选不出比上一轮更好的特征子集。

**后向搜索（backward）：**初始将所有特征作为一个候选特征子集；接着尝试去掉上一轮特征子集中的一个特征并选出当前最优的特征子集；最后直到选不出比上一轮更好的特征子集。

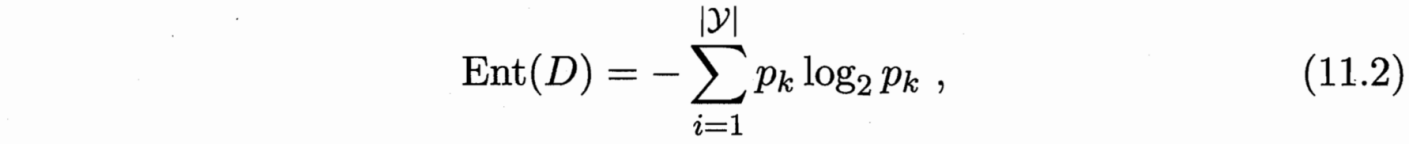
**双向搜索（bidirectional）：**将前向搜索与后向搜索结合起来，即在每一轮中既有添加操作也有剔除操作。

接着，对于如何评价候选子集的好坏，可以通过信息增益进行评价。

给定数据集，对属性子集，假定根据其取值将分成了个子集，每个子集中的样本在上取值相同，则属性的信息增益



其中信息熵定义为



信息增益越大，意味着特征子集包含的有助于分类的特征越多。故对于每个候选特征子集，我们可以基于训练数据集来计算其信息增益，以此作为评价准则。

以上，将特征子集搜索机制和子集评价机制相结合，即可得到特征选择方法。

值得一提的是，若将前向搜索和信息熵相结合，则与ID3决策树算法非常相似。事实上，决策树可用于特征选择，树结点的划分属性所组成的集合就是选择出的特征子集。

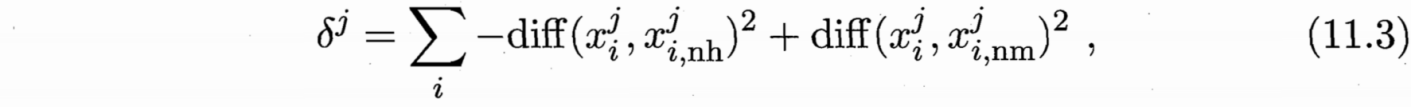
常见的特征选择方法大致可分为三类：过滤式（filter）、包裹式（wrapper）和嵌入式（embedding）.

## 11.2.过滤式选择

过滤式方法是一种将特征选择与学习器训练相分离的特征选择技术，这相当于先用特征选择过程对初始特征进行“过滤”，再用过滤后的特征来训练模型.

Relief是其中著名的代表性算法，它使用一个“相关统计量”来度量特征的重要性，该统计量是一个向量，其中每个分量代表着相应特征的重要性，因此我们最终可以根据这个统计量各个分量的大小来选择出合适的特征子集。

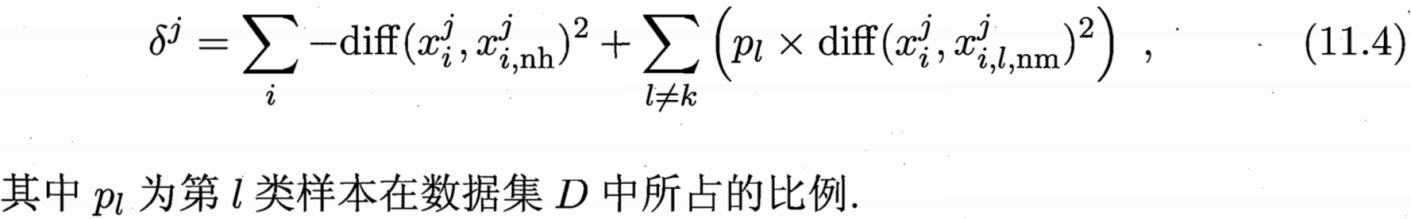
Relief的关键在于如何确定相关统计量。对于训练集的每个示例，Relief先在的同类样本中寻找其最近邻，称为“猜中近邻”(near-hit)，再从的异类样本中寻找其最近邻，称为“猜错近邻”(near-miss)。于是，相关统计量对应于特征的分量为：



直观上理解：对于猜中近邻，两者属性的距离越小越好，对于猜错近邻，属性距离越大越好。

标准Relief算法只适用于二分类问题，后续产生的拓展变体Relief-F则解决了多分类问题。

假定数据集中的样本来自个类别。对示例，若它属于第类，则Relief-F先在此类中寻找的最近邻示例作为猜中近邻，然后在第k类之外的每个类中找到一个的最近邻 作为猜错近邻，于是，对于分量，新的计算公式为：



## 11.3.包裹式选择

过滤式特征选择方法，不考虑后续学习器。包裹式特征选择直接把最终将要使用的学习器的性能作为特征子集的评价准则。换言之，包裹式特征选择的目的就是为给定学习器选择最有利于其性能、“量身定做”的特征子集。

学习器性能上看，包裹式选择肯定好；但多次训练导致计算开销大。

LVW（Las Vegas Wrapper）是一个经典的包裹式特征选择方法，它在拉斯维加斯框架下使用随机策略来进行特征子集的搜索。具体过程为

1.特征集中随机产生特征子集；

2.使用交叉验证的方法估计学习器在特征子集上的误差

3.若该误差小于之前获得的最小误差，或者与之前的最小误差相当但 中包含的特征数更少，则将保留下来。

4.重复以上三步，若连续T轮未更新则算法停止。

由于 LVW 算法每次评价子集 A′A′ 时，都需要重新训练学习器，计算开销很大，当特征数很多并且T设置得很大时，可能算法运行很长时间都不能停止。

下面科普一下拉斯维加斯方法和蒙特卡罗方法。

拉斯维加斯方法和蒙特卡罗方法是两个以著名赌城名字命名的随机化方法。这两个词本身是两座著名赌城，因为赌博中体现了许多随机算法，所以借过来命名。

蒙特卡罗算法：采样越多，越近似最优解，一定会给出解，但给出的解不一定是正确解；  
 拉斯维加斯算法：采样越多，越有机会找到最优解，不一定会给出解，且给出的解一定是正确解。

举个例子，假如筐里有100个苹果，让我每次闭眼拿1个，挑出最大的。于是我随机拿1个，再随机拿1个跟它比，留下大的，再随机拿1个……我每拿一次，留下的苹果都至少不比上次的小。拿的次数越多，挑出的苹果就越大，但我除非拿100次，否则无法肯定挑出了最大的。这个挑苹果的算法，就属于蒙特卡罗算法——尽量找好的，但不保证是最好的。

而拉斯维加斯算法，则是另一种情况。假如有一把锁，给我100把钥匙，只有1把是对的。于是我每次随机拿1把钥匙去试，打不开就再换1把。我试的次数越多，打开（最优解）的机会就越大，但在打开之前，那些错的钥匙都是没有用的。这个试钥匙的算法，就是拉斯维加斯的——尽量找最好的，但不保证能找到。

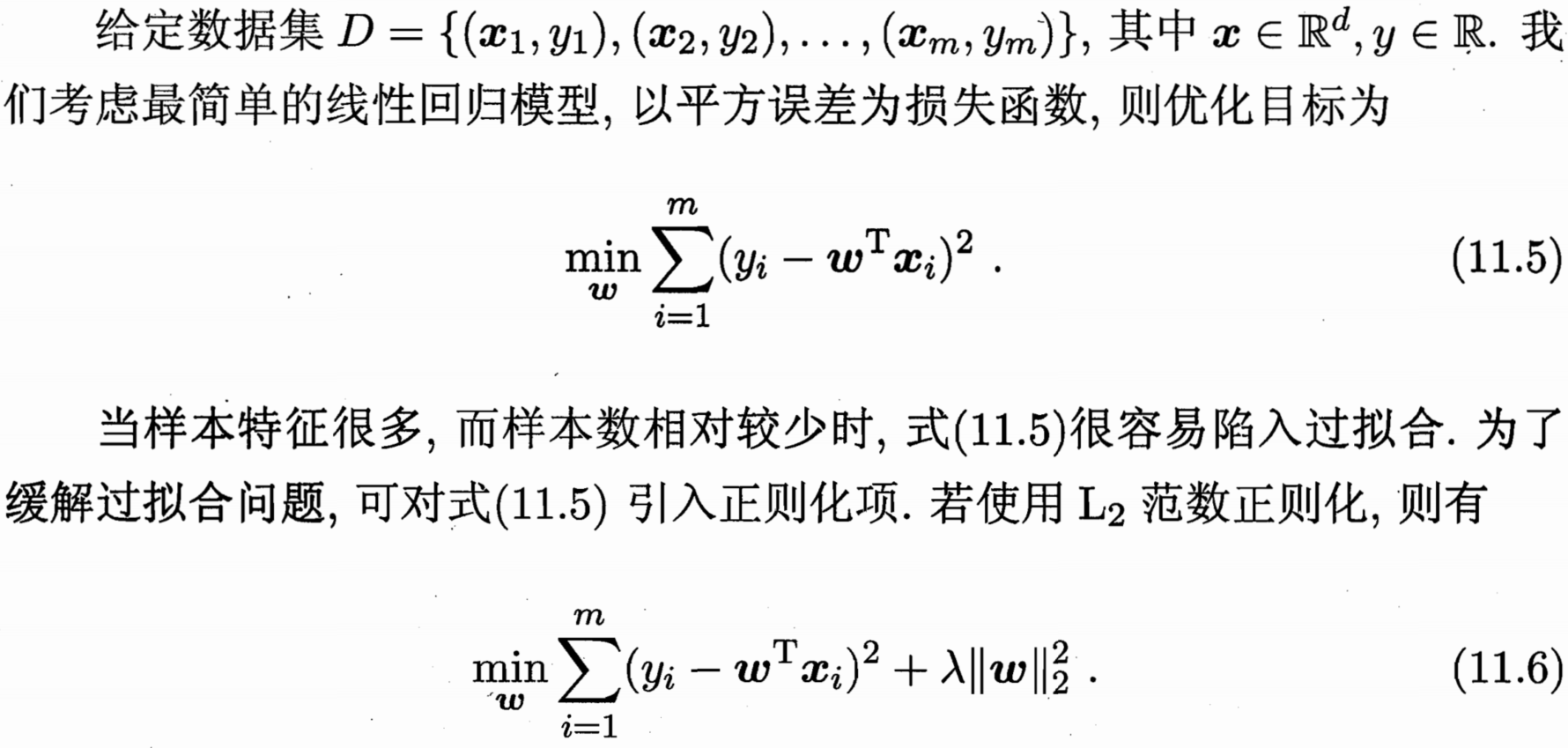
## 11.4.嵌入式选择与正则化

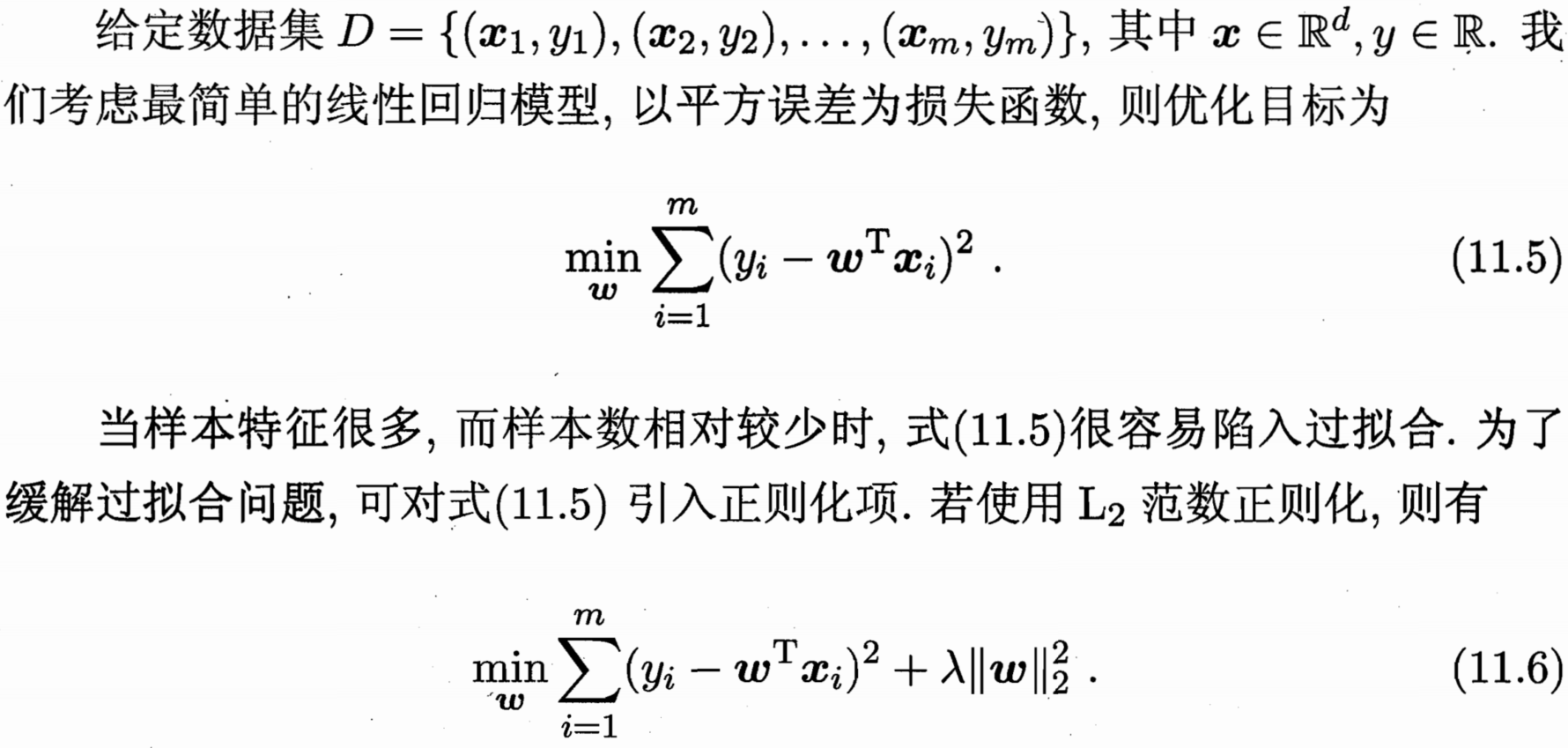
前面提到了的两种特征选择方法：过滤式中特征选择与后续学习器完全分离，包裹式则是使用学习器作为特征选择的评价准则；**嵌入式**是一种将特征选择与学习器训练完全融合的特征选择方法，即两者同在一个优化过程中完成。

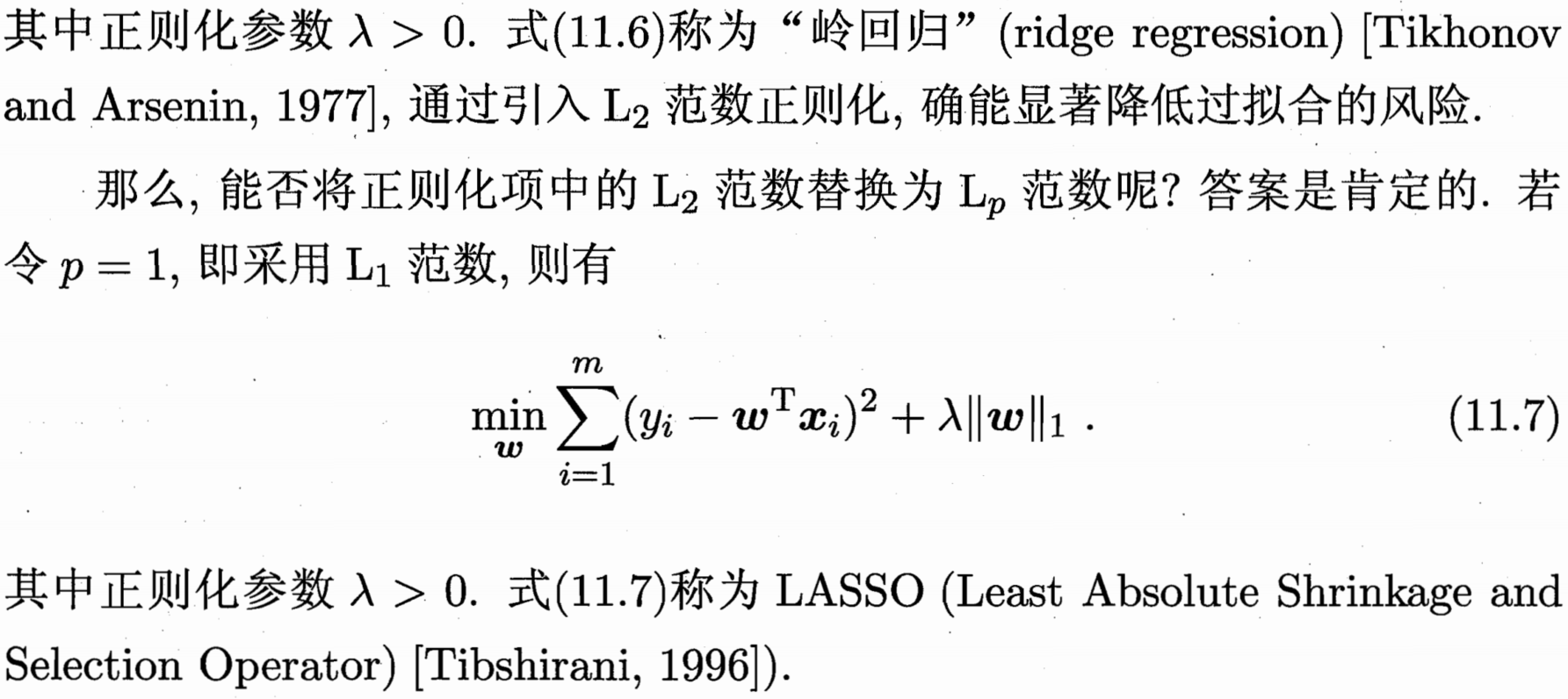
最常用的嵌入式特征选择的模型：树模型和带正则项的模型

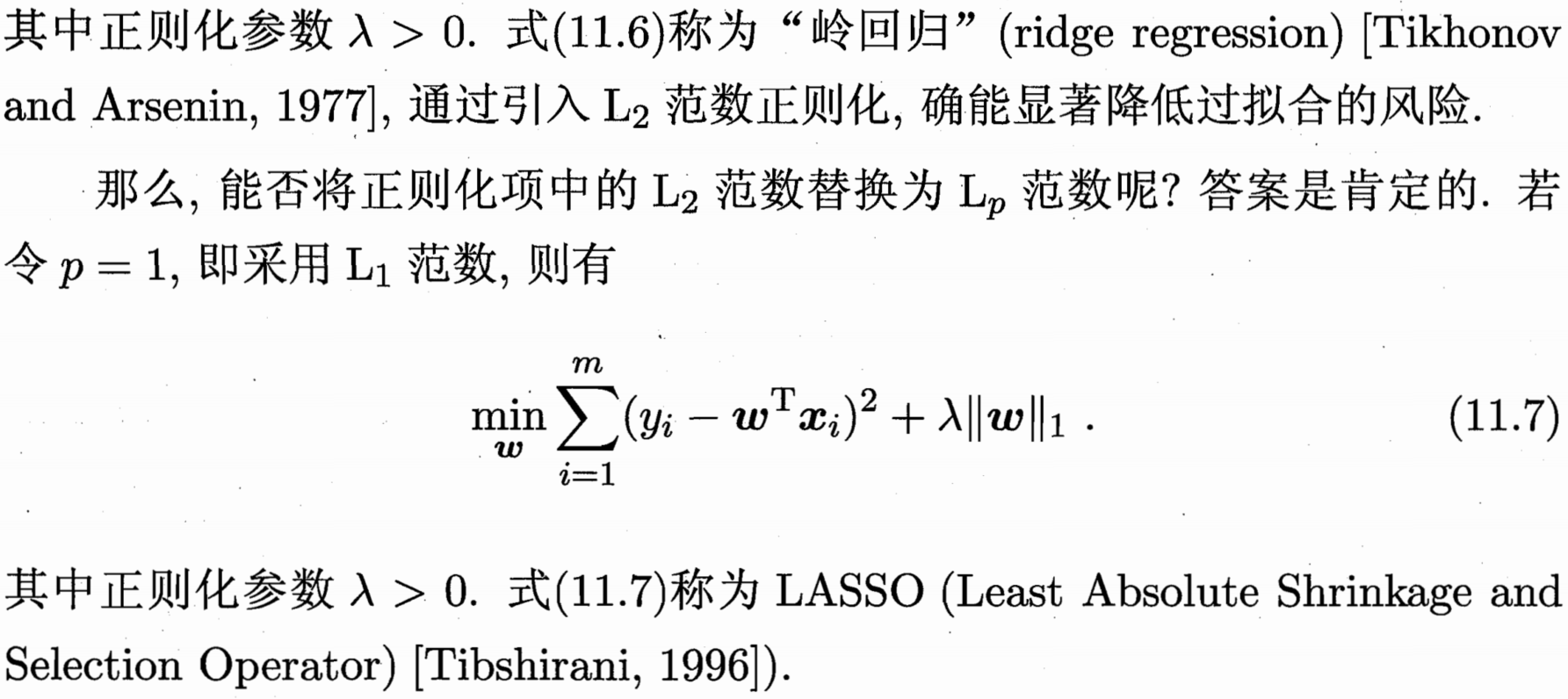
树模型中树节点的划分特征所组成的集合就是选择出的特征子集

带正则项的模型常用的是范数正则化，有助于降低过拟合风险，范数通过对参数向量各元素平方和，使缩小了回归系数。 而范数比范数更易获得“稀疏”解，即范数求得的会有更少的非零分量，所以范数可用于特征选择，而范数在参数规则化时经常用到。同时，也有范数，用来表示向量中非0元素的个数，所以范数是尽量减少非0元素的个数，与范数类似，获得稀疏解。



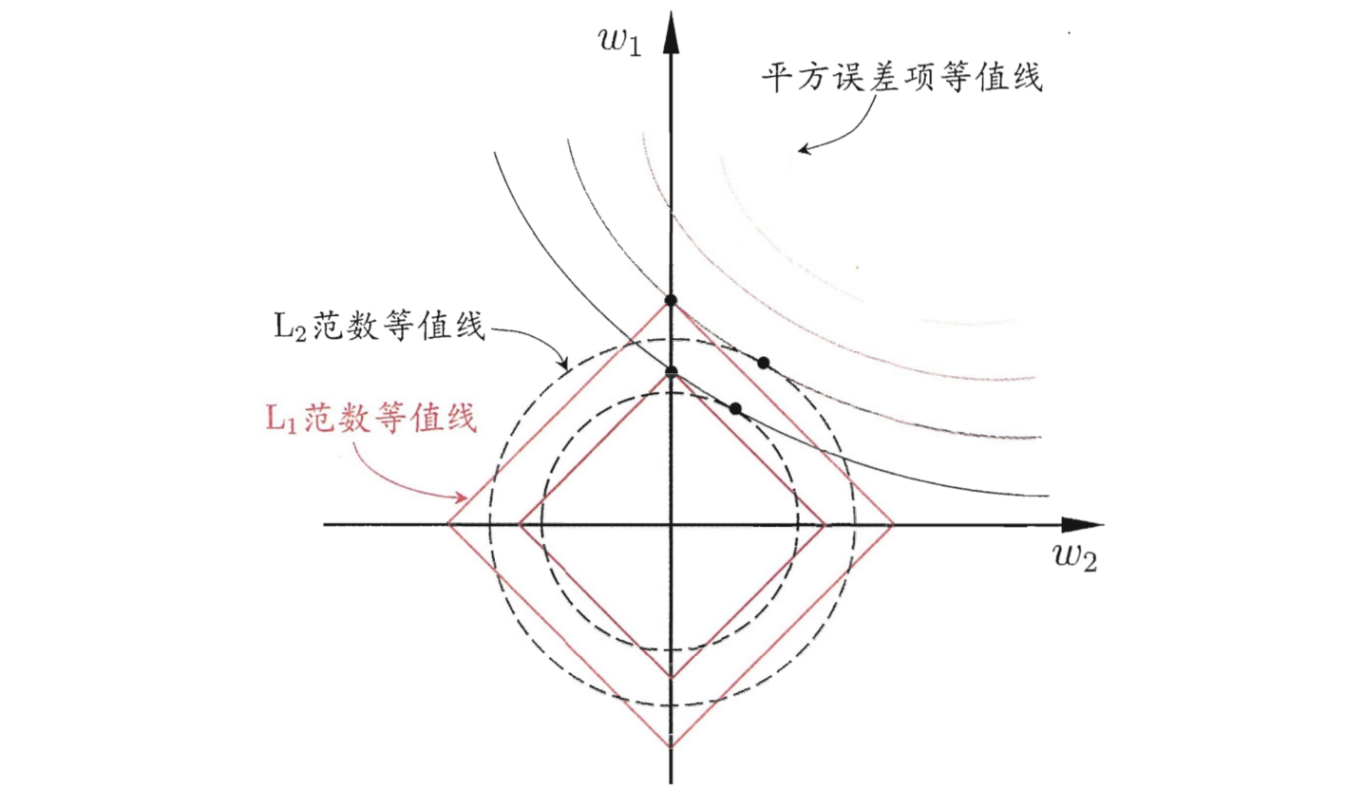






为何范数比范数更易获得“稀疏”解？

假定仅有两个属性，则有两个分量，即，我们将其作为两个坐标轴，然后绘出损失函数的“等值线”，再分别绘制出范数与范数的等值线，如图所示



式（11.6）与（11.7）的解要在平方误差项与正则化项之间折中，即出现在图中平方误差项等值线与正则化项等值线相交处。可看出采用范数时交点常出现在坐标轴上，即或为0，而在采用范数时交点常出现在某个象限中，即与均非0；换言之，采用范数比范数更易于得到稀疏解。

注意到对应着的非零分量的特征才会出现在最终模型中，即范数是仅采用一部分初始特征的模型。

其中为超参数，用来表示正则项的作用强度，越大，则正则的强度越大，即越趋近于0，也就更容易出现欠拟合；越小，则正则的强度越小，即越大，也就容易出现过拟合。所以需要选择合适的，可以用前面的评估方法进行的选择。

范数还有另一个作用，就是提高模型的可解释性，例如有100个特征，最后的决策到底由哪些特征决定呢，通过范数，可以看出最终对输出产生影响的有效特征。

如何求解范数与范数中的参数？

对于范数，由于正则项本身为凸函数，为连续可导函数，所以可以使用梯度下降法直接求解。

对于范数由于正则项为一个绝对值，即，是不连续的函数，所以比较难求解，对于绝对值函数求导时，其导数为



对于这样的问题可以利用**坐标下降法**（coordinate descent）进行求解，坐标下降法属于一种非梯度优化的方法，它在每步迭代中沿一个坐标的方向进行搜索，通过循环使用不同的坐标方法来达到目标函数的局部极小值。即求导时只对一个维度（坐标轴方向）进行求导，而固定其它维度，这样每次只优化一个分量。

对于目标函数



由于为维向量，代入上式得



其中为第个样本的第个特征，利用坐标下降法，设当前只对进行优化，则对于之外的值均为已知，对求偏导得











由于为常数，令其为，也为常数，令其为，且，则上式可化为





当时，令偏导等于0可得





由于，则有即

当时，令偏导等于0可得





由于，则有即

所以当时有

综上，得出的迭代表达式为



从上式可以看出，当时，的值为0，即在分量的特征对结果没有影响，因此，范数更易得到稀疏解。

上面是对的更新过程，当更新完成后，再对下一个权重进行更新，依次迭代直至收敛。

对于范数的求解还有很多其他算法，例如最小角回归（LARS）、近端梯度下降（PGD）、前向选择法等。

另外，范数与范数也可以共同使用，这里设目标函数为关于的函数，则可以有



其中，和用来调节和的重要性。例如

场景一：神经网络中，假设神经元可以划分为n个区域，对应参数为，当输入不同时，由某个（或几个）区域共同响应并得到结果，例如人的大脑，有负责视觉的区域，负责听觉的区域，还有负责味觉的区域等。而且，相邻的区域的作用比较类似，那么，可以用范数进行过滤，去除掉大部分作用较小的区域，再利用范数来约束相邻区域之间差别尽量小。则可以将目标函数写为



场景二：推荐系统，用户与商品的矩阵进行SVD分解后，会产生用户兴趣矩阵和商品矩阵（这里不展开讲解），由于推荐系统为动态系统，即矩阵中用户和商品的向量是在随着时间的推移而改变（如用户兴趣的变换，商品价格的变化），假设第个用户的对第个商品的真实兴趣程度为，第个用户的兴趣向量为，第个商品的属性向量为，则第个用户的对第个商品的预测兴趣程度为，则有



其中，后两项分别防止用户与商品的过拟合，但是在实际情况中，用户的兴趣和商品的属性在相邻时间段不能差别太大，令相邻时间段为和，则可以再次加入正则为



因此，可以看出正则的使用非常灵活与广泛，不仅可以防止过拟合和产生稀疏的特征，也可以在需要对参数有一定限制时，可以根据需求使用不同的正则。