机器学习（L07）：特征选择

授课教师：王贝伦 / 助教：张嘉琦，黄旭，谈笑，徐浩卿

当一个机器学习模型的特征太多时，会带来计算量过大，学习性能太低，过拟合等问题，其中可能有成百上千个低相关性的特征，对模型的好坏没有太多的意义。所以我们需要特征选择，选择最相关的特征，来建立更好更快更容易理解的学习模型。从给定的特征集合中选择出相关特征子集的过程，称之为“特征选择”。

我们希望学习的模型拥有以下几个很好的性质：

- 很好的泛化性
  对噪声不敏感，更小的方差
- 计算上可扩展且高效
  更容易的训练（更少的样本），计算上更简单的使用
- 鲁棒性/可靠性/可解释性

1 过滤法

过滤法一般通过统计学、信息论等理论对特征或特征子集进行评分排名并进行过滤，与学习算法无关。分为单变量法和多变量法。

1.1 单变量法

单变量法指每次只考虑一个变量，对每个单个特征进行评估。下面举两个简单的例子

皮尔森相关系数 在回归问题中，我们常采用皮尔森相关系数来评估一个特征与标签的线性相关性。给定两个长度为 n 的序列 x 和 y，则 x 和 y 的相关系数为

$$r(x,y)=\frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2\times\sum_{i=1}^{n}(y_i-\bar{y})^2}} \tag{1}$$

其中 $\bar{x}$ 是 x 的平均值，$\bar{y}$ 是 y 的平均值。计算出每个特征与标签的皮尔森相关系数，便可以去掉一些相关系数较低的特征。但是皮尔森相关系数只能衡量线性相关性，若存在更复杂的相关性则无法体现出来。

T 检验 例如在二分类问题中，我们要考察某一特征与标签之间的相关性，可以用 T 检验来验证。T 检验有两个应用条件：

- 两个标签的样本群体都服从正态分布
- 两个样本群体的方差相等

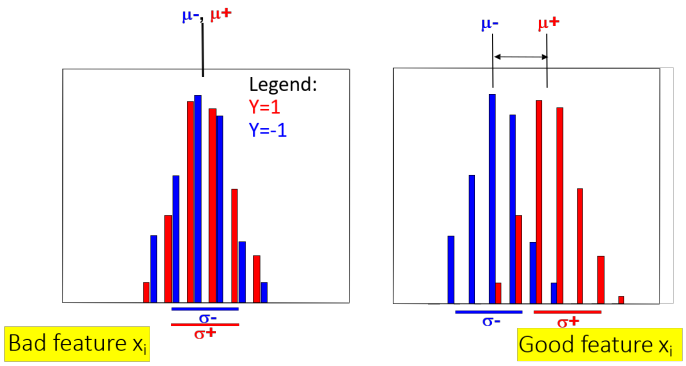

Figure 1: 使用 T 检验验证特征 x_i

那么需要验证的是这两个样本群体的均值是否相等。因为样本之间互相独立，因此使用双侧检验和双总体检验。首先建立假设和确定检验水准。零假设为 $H_0:\mu_+=\mu_-$，其中 μ_+ 和 μ_- 分别是两个样本群体的均值，检验水准 α 一般取值 0.05。然后计算检验统计量

$$t=\frac{\mu_+-\mu_-}{s_p\cdot\sqrt{\frac{1}{n_1}+\frac{1}{n_2}}} \tag{2}$$

$$s_p=\sqrt{\frac{(n_1-1)s_+^2+(n_2-1)s_-^2}{n_1+n_2-2}} \tag{3}$$

其中 n_1 为 $+$ 类样本的个数，n_2 为 $-$ 类样本的个数，s_+^2 和 s_-^2 分别为两类样本的样本方差，则 t 满足 t 分布 $t\sim t(n_1+n_2-2)$。根据样本计算出统计量 t 的值，最后查表判断是否满足检验水准。若接受零假设，则可以认为这个特征与标签不相关，可以去除，反之则是好的特征，如图1所示。

单变量法比较简单，但在一些特殊情况下会失效，如图2所示。圆圈和五角星分别代表两个类别的样本，可以看出特征 x_1 的两个样本群体的均值相同，x_2 也是如此，用单变量法判断出 x_1 和 x_2 都是不相关的特征，然而实际上却并非如此。标签需要由两个特征共同决定，这是单变量法无法体现出来的，需要多变量法。

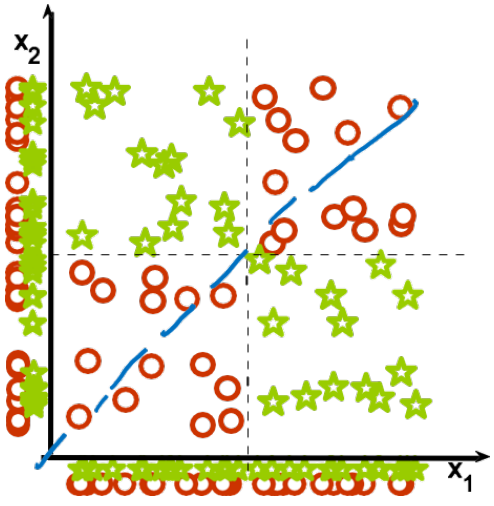

Figure 2: 单变量法失效的特殊情形

Algorithm 1: 过滤法的单变量法

输入：**样本矩阵** $x \in \mathbb{R}^{n \times p}$，**样本对应的标签** $y \in \mathbb{R}^n$，**要保留的特征个数** m;
for $i \leftarrow 1$ *to* p **do**
  评估第 i **个特征** x_i **与标签** y **的相关性;**
end
对 p **个评估值排序;**
选出 m **个有最大评估值的特征;**
$x' \leftarrow$ **去掉** x **中其他特征;**
输出：**过滤后的样本矩阵** x'

1.2 多变量法

多变量法需要每一次选择一个特征子集进行评估，与学习算法无关。等过滤完特征后，再来训练模型。多变量法采用的评估方法有组相关性，信息论过滤法里的马尔科夫毯等方法，在这里不做深入探讨。选择完评估方法后，我们还需要一个搜索策略来选择特征子集进行评估。如果有 p 个特征，那么就有 2^p 个特征子集，数量非常庞大。寻找到一个使评估值最优的特征子集是一个 NP-hard 问题，因此需要一个好的启发式算法。

1.3 过滤法总结

对过滤法的总结有以下几条：

- 通常比较快速
- 可以提供通用的特征选择，不需要学习模型进行调参。
- 但是也经常需要检查一下，因为选出的特征子集对于学习模型并不一定是最优的
- 通常被用作其他方法的预处理步骤

编程实现

实现使用皮尔森相关系数的过滤法

```python
import numpy as np

n = 100 #训练样本个数
p = 10  #特征数
X_train = 10 * np.random.uniform(size=(n, p))
theta = np.array([1, -4, 0.3, -2, 0.001, 7, -0.04, 7, -6, 0])
Y_train = X_train @ theta + 0.1 * np.random.normal(size=(n,))

# 计算每个特征与标签的皮尔森相关系数
coef = np.empty(p)
for i in range(p):
  coef[i] = np.corrcoef(X_train[:, i], Y_train)[0, 1]

# 保留特征的个数
m = 5
argsort = np.argsort(-np.abs(coef))
filtering = np.argsort(argsort) < m
X_filtered = X_train[:, filtering]

# coef为每个特征与标签的相关系数，系数较小的特征对应的相关系
# 数也会比较小，因此从中选取m个相关系数绝对值最大的特征保留
print(coef)
# Output:
# [ 0.19687654 -0.41520055 -0.11133188 -0.20355642
#   0.03838866  0.55792475 -0.10055488  0.58765207
#  -0.52146651 -0.02813482]

# filtering为判断特征是否要保留的向量，False表示舍去，True表
# 示保留
print(filtering)
# Output:
# [False  True False  True False
#   True False  True  True False]

# X_filtered中仅有保留的m个特征
print(X_filtered.shape)
# Output:
# (100, 5)
```

2 包裹法

与过滤法不同，包裹法采用的是根据学习效果来搜索特征子集的方法，也就是从特征集合中不断选择特征子集，根据学习器的表现来对子集进行评估，选择出最佳的子集。学习器可以看作一个黑盒，黑盒的接口根据学习器使用某特征子集的预测能力为该特征子集评分，所以不同的学习器可能会选出不同的特征子集。使用包裹法需要解决两个重要的问题：

- **搜索**：如何搜索特征子集？
- **评估**：对于一个特定的特征子集如何评价学习器的性能？

2.1 搜索

正如过滤法中提到的，搜索全局最优的特征子集是一个 NP-hard 问题。许多启发式算法可以使用，它们主要分为两大类：

- 前向选择：从空的特征集开始搜索，每一步增加新的特征
- 后向排除：从满的特征集开始搜索，每一步去掉特征

预测效果则通过验证集或者交叉验证来评估。因为学习器可以看作一个黑盒，包裹法比较通用和简单，但是需要大量的运算。

最简单的启发式搜索是序列前向选择法（SFS）和序列后向选择法（SBS）。序列前向选择法从空集开始，首先选择出一个最好的特征加入特征集合，然后从剩下的特征再选择出一个最好的特征加入集合，这样集合中就有两个最好的特征了，此后每一步都从剩下的特征中选择一个最好的特征加入集合，直到选择了设定个数个特征。序列前向选择法适合最优特征子集比较小的情况。

Algorithm 2: 序列前向选择法

输入：**样本矩阵** $x \in \mathbb{R}^{n \times p}$，**样本对应的标签** $y \in \mathbb{R}^n$，**要保留的特征个数** m，**评估函数** J;
设置空集 $Y_0 = \varnothing$;
for $i \leftarrow 1$ *to* m **do**
  $x^* = \arg\max_{x \notin Y_{i-1}} J(Y_{i-1} + x)$;
  $Y_i \leftarrow Y_{i-1} + x$
end
输出：**特征子集** Y_m

序列后向选择法与序列前向选择法类似，但是是从全集开始，每次去掉一个最差的特征，直到剩下 m 个特征。我们再简单介绍几个搜索算法：

- 集束搜索（Beam search）：选择得分最高的 k 个特征作为加入一个有限长度的特征子集队列，每一步选择 k 个最佳路径
- 广义序列前向选择法（GSFS）：当剩下 $(p-k)$ 个特征时，每次加入包含 g 个特征的特征子集，在其中选择一个最优的特征子集。每步会有更多训练，但步数更少

- 增 L 去 R 选择算法：在每一步使用 l 次序列前向选择法（SFS），再使用 r 次序列后向选择法（SBS）。

- 浮动选择法（Floating search）：每次使用 SFS（SBS），然后再使用 SBS（SFS）只要能够找到更好的子集即可。

2.2 评估

为了评估已选择出的特征子集，我们可以将数据分成训练集，验证集和测试集。

(1) 对于每一个特征子集，用训练集训练出一个预测器。

(2) 用验证集测试预测器，选出表现最好的特征子集。如果想要减少方差可以重复这一步并取平均。

(3) 最后再用测试集测试选出的预测器。

2.3 编程实现

本节展示一种前向选择法的实现。

```python
import numpy as np

def forward_select(data, feature_assess):
  '''参数：
    data - 原数据集，每行形式为(x1, x2, ..., xp, y)，n行p+1列矩阵。
    feature_assess - 特征评估函数，输入为子数据集，返回一个非负实数（特征得分），其越高表示该特征子集效果越好。

  返回：
    best_features - 非负整数列表，其中每一项索引了一个原数据集中的特征。
  '''

  p = data.shape[1] - 1   # 特征维数
  best_features = []    # 初始化特征子集
  current_score = 0     # 初始化当前特征得分
  for _ in range(p):
    scores = [None]*p # 初始化一个数列以记录下一步搜索得特征子集的得分
    for new_feature in range(p):
      if new_feature not in best_features:
        test_features = best_features + [new_feature,]
        scores[new_feature] = feature_assess(
                        data[:, test_features + [p,]])
      else:
        scores[new_feature] = -1    # -1表示该特征已经在当前特征子集里
    next_best_score = np.max(scores)    # 获得加入新特征后的最高得分
    if next_best_score <= current_score:  # 添加任何一个特征都无法得到更高的分，说明已搜索到最优子集
      break
    else:                   # 将最好的新特征加入当前子集中
      next_feature = np.argmax(scores)
      best_features.append(next_feature)
  return best_features

```

3 嵌入法

定义：嵌入式特征选择

嵌入式特征选择是指，在训练模型的同时，自动地在模型内部进行特征选择。也即我们将特征选择的任务交给模型和训练去完成。

3.1 例子

事实上，在线性回归问题中提到的正则化就是一种嵌入式特征选择的方法。

- 岭回归：$\theta = \underset{\theta}{\operatorname{argmin}}\ L(X,y) + \lambda \sum \theta_i^2$
- 套索回归：$\theta = \underset{\theta}{\operatorname{argmin}}\ L(X,y) + \lambda \sum |\theta_i|$

特征选择在这里具体体现为，对于不重要的特征其对应 θ_i 较小，而较大的 θ_i 则意味着该特征影响较大。改变 λ 的值，可以调节特征选择的强度。

3.2 特点

- 只需要训练好一个模型，即可完成特征选择及预测数据两个任务。
- 对于不同的假设和模型，其形式可能不相同。即，嵌入式特征选择是针对一个特定模型而设计的。
- 特征选择的过程被隐含在模型训练的过程中。

4 降维

没有一种特征选择方法是普适的，特征选择的本质实际上就是将高维的原始数据降维为低维数据。实际上，我们可以考虑将一些原始特征加权处理为新的特征，同样也能达到降低维度的目标。

定义：降维法

若原始维度空间为 $\mathcal{X} = \mathcal{X}_1 \times \mathcal{X}_2 \times \cdots \times \mathcal{X}_p$，定义变换 $g_i : \mathcal{X} \mapsto \mathcal{X}_i'$，则变换 $g = (g_1, g_2, \cdots, g_{p'})^\top$ 定义了一个从原始维度空间 $\mathcal{X}$（维度为 p）到低维空间 $\mathcal{X}'$（维度为 p'）的映射。该变换操作即为降维。

在实际应用中，需要通过一些方法和限制找到合适的降维变换，从而获得更好效果。

4.1 PCA

一个简单的降维想法是，认为数据的方差是最有用的信息。为了使降维后尽可能保留方差信息，我们可以将数据投影到方差最大的方向上。

定义：PCA

PCA 中，变换 $g(x) = \beta^\top x$，其中 $\beta = \underset{\beta}{\operatorname{argmax}} \operatorname{Var}(X\beta)$。

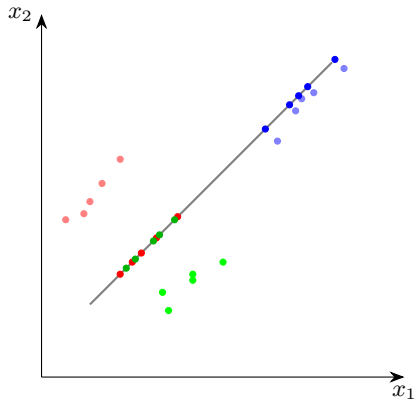

Figure 3: PCA 示意图

PCA 如上图所示，数据点投影到灰色直线上后，方差信息得以保留。

5 模型选择与评估

当我们正在解决一个机器学习问题时，我们通常想知道选用哪个模型更好，通过比较多个模型的性能，选定一个模型应用，这个过程被称为模型选择。特别地，对模型中的超参数进行取舍也可称为模型选择。其中定量表达模型性能的方法即为模型评估。均方误差（MSE）是一个经常使用的定量评估方法，MSE 越大，代表模型的预测能力越差。对于这两个步骤，在不同场合下有不同解决方法。

- 数据量充足情况：我们可以将数据集分为 3 部分，训练集、验证集、测试集。使用训练集、验证集进行模型选择，选择出一个合适的模型后利用训练集 + 验证集进行训练，最后在测试集上测试以评估模型。

- 数据量缺乏情况：模型选择时，可以借助 AIC、BIC、MDL、SRM 等方法代替验证集，也可以采用交叉验证以充分利用数据集。

6 模型复杂性

模型的复杂性，一般而言指模型中待学习参数的数量，能够影响到它在测试集上的表现。概括而言，简单的模型不能很好地利用特征，复杂的模型也有可能过多使用了特征中的噪音而导致效果变差。在这里我们引入偏差和方差的概念。

定义：模型中的偏差与方差

假设总体服从的模型为 $y = f(x) + \epsilon$，其中 $\epsilon \sim \mathcal{N}(0, \sigma^2)$。对于样本集 (X, y)，我们拟合得到的模型为 $\hat{f}$。考虑期望预测均方误差 $E\left[(y - \hat{f})^2\right] = \sigma^2 + E\left[(f - E\hat{f})^2\right] + E\left[(\hat{f} - E\hat{f})^2\right]$。定义偏差 $\text{Bias} = f - E\hat{f}$、方差 $\text{Variance} = E\left[(\hat{f} - E\hat{f})^2\right]$。则 $E\left[(y - \hat{f})^2\right] = \sigma^2 + \text{Bias}^2 + \text{Variance}$。

偏差和方差共同决定了模型预测值与真实值的期望均方误差。下图清楚地展示了不同偏差、方差所呈现的样子。

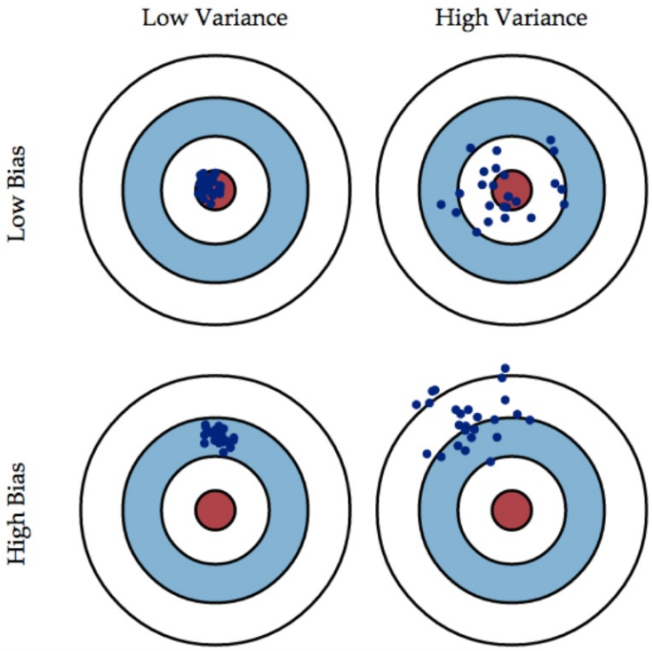

Figure 4: 偏差-方差示意图

- 偏差：描述了拟合模型与真实模型接近的程度。通俗地讲，偏差越低，拟合得到的参数就越接近真实参数，同时预测值也就越靠近真实值。

- 方差：描述了在不同的训练集下，拟合模型预测值的离散程度。一般而言，改变训练集会影响到拟合出的参数，方差越高，则改变训练集所造成的参数变化就越剧烈。

偏差和方差与之前提到过的欠拟合、过拟合有很大的联系。下面我们将讨论它们之间的对应关系。

6.1 欠拟合

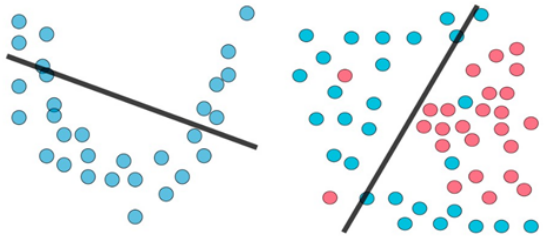

Figure 5: 欠拟合模型

大多情况下欠拟合是由于模型过于简单导致的，也即模型参数较少从而无法利用更多的特征信息。

- 由于参数少，从而拟合得到的模型很难接近真实模型，其表现即为高偏差。

- 同时也因为模型简单，所以对于不同的训练集，拟合模型不会变化太多——基本上都能落在所能达到的最优解附近，表现为低方差。

欠拟合的学习曲线图通常如图6，虽然在训练集和测试集上的 MSE 差别不大（意味着低方差），但它们的数值都较大，准确性低（意味着高偏差）。

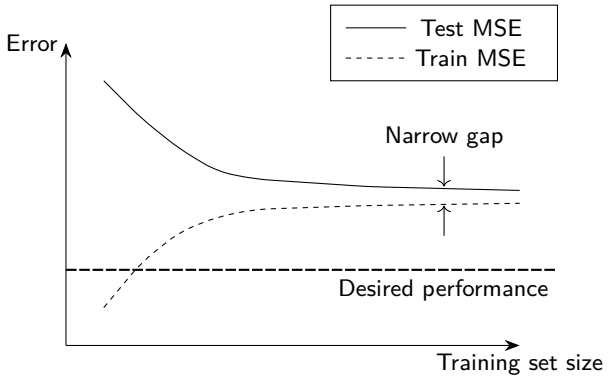

Figure 6: 欠拟合模型的学习曲线

欠拟合/高偏差的解决办法：

- 选用更复杂的模型。
- 有时特征数较少也会导致欠拟合，此时可以考虑加入更多特征。

6.2 过拟合

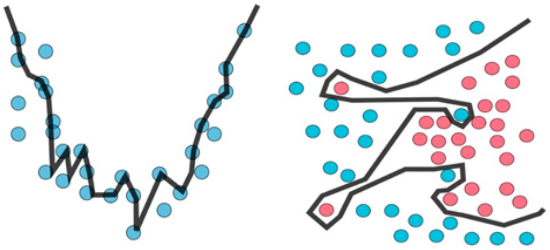

Figure 7: 过拟合模型

过拟合则通常源于复杂模型，有时过于复杂的模型对某一训练集甚至有多解。

- 正因为如此，模型对训练集的变化很敏感，从而导致高方差。
- 尽管零星的预测值与真实值偏差很大，但如果投入大量的数据，总能拟合出一个贴合所有数据点的模型，此时模型的偏差便很小了。

过拟合的学习曲线如图8，训练集上的表现明显地优秀，而测试集上的表现则不尽如人意。

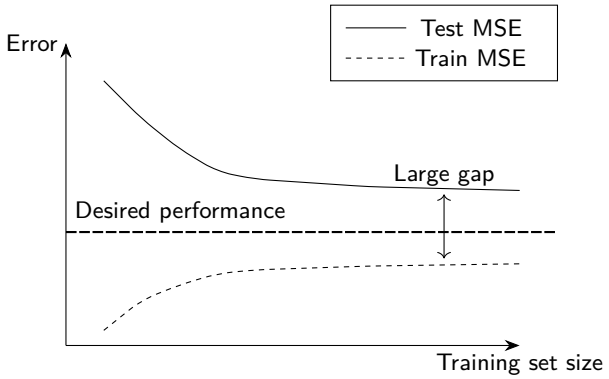

Figure 8: 过拟合模型的学习曲线

过拟合/高方差的解决办法：

- 选用更简单的模型。
- 采用正则化方法。
- 给特征降维。
- 采集更多样本数据。

引用

[1] https://blog.csdn.net/yimenglin/article/details/102796748