第十一章 特征选择与稀疏学习

11.1 子集搜索与评价

11.1.1 特征选择

对一个学习任务来说, 给定属性集, 其中有些属性可能很关键、很有用, 另一些属性则可能没什么用.

我们将属性称为特征 (feature).

- **相关特征 (relevant feature)**: 对当前学习任务有用的属性称为"相关特征"
- **无关特征 (irrelevant feature):** 对当前学习任务没什么用的属性称为"无关特征"
- **特征选择 (feature selection给)**: 从给定的特征集合中选择出相关特征子集的过程, 称为"特征选择".

11.1.2 特征选择的意义

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**.

为何要进行特征选择, 特征选择的意义.

- **缓解维数灾难:** 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部分特征上构建模型, 则维数灾难问题会大为减轻. 在这个意义上, 特征选择与降维具有相似的动机. 实际上, **特征选择和降维是处理高维数据的两大主流技术.**
- **降低学习任务的难度:** 去除不相关特征往往会降低学习任务的难度

同时还需要注意两点:

- 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**与当前辈习任务无关**的特征.

11.1.3 冗余特征的概念

冗余特征 (redundant feature), 是指**他们所包含的信息能从其他特征中推演出来**. 比如, 特征"底面长", "底面宽", 那么"底面积"就是冗余特征, 因为可以从长和宽推断得到.

- **冗余特征在很多时候不起作用**, 去除它们会减轻学习过程的负担.

- 但有时侯, **冗余特征会降低学习任务的难度**. 更确切地说, 若**某个冗余特征恰好对应了完成学习任务所需的"中间概念", 则该冗余特征是有益的.**

11.1.4 特征选择的具体过程

先产生一个"候选子集", 评价出它的好坏, 基于评价结果产生下一个候选子集, 再对其进行评价, ……这个过程持续进行下去, 直至**无法找到更好的候选子集为止**.

1 第一个环节--"子集搜索" (subset search)

- **前向搜索:** 将每个特征看成一个候选子集, 选出其中最优的, 再增加一个特征,构成两个特征的子集, 选出最优, ……直至无法在下一轮选出更优, 那么停止生成候选子集, 并将上一轮作为特征选择结果
- **后向搜索:** 从完整的特征集合开始, 每次尝试去掉一个无关特征, 这样逐渐减少特征的策略称为"后向" 搜索.
- **双向搜索:** 还可将前向与后向搜索结合起来, 每一轮逐渐增加选定相关特征、同时减少无关特征.

注意的是, **上述策略是贪心策略, 因为仅仅考虑了使本轮选定集最优**, 有缺陷, 但也无法避免.

2 第二个环节--"子集评价" (subset evaluation)

给定数据集 D, 假定 D 中第 i 类样本所占比例为 $p_i (i = 1, 2, \ldots, |\mathcal{Y}|)$. 对属性子集 A , 假定根据其取值将 D 分成了 V 个子集 $\left\{D^1, D^2, \ldots, D^V\right\}$, 每个子集中的样本在 A 上取值相同, 那么就可以计算属性子集 A 的信息增益

$$\operatorname{Gain}(A) = \operatorname{Ent}(D) - \sum_{v=1}^{V} \frac{|D^v|}{|D|} \operatorname{Ent}(D^v) \tag{11.1}$$

其中, 信息熵定义为:

$$\operatorname{Ent}(D) = -\sum_{i=1}^{|\mathcal{Y}|} p_k \log_2 p_k \tag{11.2}$$

信息增益 $\operatorname{Gain}(A)$ 越大, 意味着特征子集 A 包含的有助于分类的信息越多.

因此, 对每个候选子集, 可基于训练数据集 D 来计算其信息增益, 以此作为评价准则.

更一般的, 特征子集 A 实际上确定了对数据集 D 的一个划分, 每个划分区域对应着 A 上的一个取值, 而样本标记信息 Y 则对应着对 D 的**真实划分**, 通过估算这**两个划分的差异**, 就能对 A 进行评价. **与 Y 对应的划分的差异越小, 则说明 A 越好**.

而信息熵仅仅使判断这个差异的其中一种途径, 其他能判断两个划分差异的机制都能用于特征子集评价.

3 特征选择方法

将特征**子集搜索机制**与**子集评价机制**相**结合**, 即可得到**特征选择方法**

常见的特征选择方法大致可以分为**三大类**:

- **过滤式(filter)**
- **包裹式(wrapper)**

- **嵌入式(embedding)**

11.2 过滤式选择

11.2.1 过滤式选择的基本概念

过滤式方法**先对数据集进行特征选择, 然后再训练学习器, 特征选择过程**与后**续学习器无关**.

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.

11.2.2 Relief 特征选择方法

1 Relief 的基本概念

Relief (Relevant Features) 是一种著名的过滤式特征选择方法, 该方法设计了一个**"相关统计量"**来**度量特征的重要性**.

该统计量是一个向量, 其**每个分量**分别对应于**一个初始特征**, 而**特征子集**的重要性则是由子集中**每个特征所对应的相关统计量分量之和**来决定.

因此, 最终只需指定一个阈值 τ , 然后选择比 τ 大的相关统计量分量所对应的特征即可; 也可指定欲选取的特征个数 k , 然后选择相关统计量分量最大的 k 个特征.

2 Relief 的计算过程

给定训练集 $\{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$, 对每个示例 $\boldsymbol{x}_i$, Relief 现在 $\boldsymbol{x}_i$ 的同类样本中寻找其最近邻 $\boldsymbol{x}_{i,nh}$, 称为"猜中近邻"(near-hit), 再从 $\boldsymbol{x}_i$ 的异类样本中寻找其最近邻 $\boldsymbol{x}_{i,nm}$ 称为"猜错近邻"(near-miss), 然后, 相关统计量对应于属性 j 的分量为:

$$\delta^j = \sum_i -\operatorname{diff}\left(x_i^j, x_{i,\mathrm{nh}}^j\right)^2 + \operatorname{diff}\left(x_i^j, x_{i,\mathrm{nm}}^j\right)^2 \tag{11.3}$$

其中, x_a^j 表示样本 $\boldsymbol{x}_i$ 在属性 j 上的取值, $\operatorname{diff}\left(x_a^j, x_b^j\right)$ 取决于属性 j 的类型:

- 若属性 j 为离散型, 则 $x_a^j = x_b^j$ 时 $\operatorname{diff}\left(x_a^j, x_b^j\right) = 0$, 否则为1
- 若属性 j 为连续性, 则 $\operatorname{diff}\left(x_a^j, x_b^j\right) = \left|x_a^j - x_b^j\right|$, 同时注意, x_a^j, x_b^j 已规范化到 $[0, 1]$ 区间.

11.2.3 Relief-F 特征选择方法

Relief 是为**二分类问题**设计的, 其扩展变体 **Relief-F** 能处理**多分类问题**.

假定数据集 D 中的样本来自 $|\mathcal{Y}|$ 个类别. 对示例 $\boldsymbol{x}_i$, 若它属于第 k 类 ($k \in \{1, 2, \ldots, |\mathcal{Y}|\}$), 则 Relief-F 先在第 k 类的样本中寻找 x_i 的最近邻示例 $\boldsymbol{x}_{i,nh}$, 并将其作为猫中近邻. 然后在第 k 类之外的**每个类**中找到一个$\boldsymbol{x}_i$的最近邻示例作为猜错近邻, 记为 $\boldsymbol{x}_{i,l,\mathrm{nm}} (l = 1, 2, \ldots, |\mathcal{Y}|; l \neq k)$. 那么, 相关统计量对应于属性 j 的分量为:

$$\delta^j = \sum_i -\operatorname{diff}\left(x_i^j, x_{i,\mathrm{nh}}^j\right)^2 + \sum_{l \neq k}\left(p_l \times \operatorname{diff}\left(x_i^j, x_{i,l,\mathrm{nm}}^j\right)^2\right) \tag{11.4}$$

其中, p_l 为第 l 类样本在数据集 D 中所占的比例.

11.3 包裹式选择

11.3.1 包裹式选择的概念

与过滤式特征选择不考虑后续学习器不同, 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于集的评价准则. 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、 "量身走做"的特征子集.

一般而言, 因为包裹式特征选择方法直接针对给定学习器进行优化, 因此, 从最终学习器的性能来看, 包裹式比过滤式更好, 但另一方面, 包裹式特征选择的计算开销比过滤式大得多.

11.3.2 LVW 方法

LVW (Las Vegas Wrapper) 是一个典型的包裹式特征选择方法.

它在拉斯维加斯方法 (Las Vegas method) 框架下使用**随机策略**来进行子集搜索, 并以**最终分类器的误差**为**特征子集评价准则**.

算法描述如图 11.1 所示

输入: 数据集 D;
特征集 A;
学习算法 $\mathfrak{L}$;
停止条件控制参数 T.

过程:

$E = \infty$;
$d = |A|$;
$A^* = A$;
$t = 0$;
while $t < T$ **do**
  随机产生特征子集 A';
  $d' = |A'|$;
  $E' = \text{CrossValidation}(\mathfrak{L}(D^{A'}))$;
  if $(E' < E) \vee ((E' = E) \wedge (d' < d))$ **then**
    $t = 0$;
    $E = E'$;
    $d = d'$;
    $A^* = A'$
  else
    $t = t + 1$
  end if
end while

输出: 特征子集 A^*

图 11.1 LVW 算法描述

11.4 嵌入式选择与 L_1 正则化(推导略)

11.4.1 嵌入式选择的概念

嵌入式特征选择是将特征选择过程与学习器训练过程融为一体, 两者在同一个优化过程中完成, 即在学习器训练过程中自动地进行了特征选择.

11.4.2 数学过程

给定数据集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$, 其中, $\boldsymbol{x} \in \mathbb{R}^d, y \in \mathbb{R}$. 考虑最简单的线性回归模型, 以平方误差为损失函数, 则优化目标为

$$\min_{\boldsymbol{w}} \sum_{i=1}^{m} \left(y_i - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i\right)^2 \tag{11.5}$$

当样本特征很多, 而样本数相对较少时, 式 (11.5) 容易陷入过拟合.

为此, 引入正则化项. 若使用 L_2 范数正则化, 有

$$\min_{\boldsymbol{w}} \sum_{i=1}^{m} \left(y_i - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i\right)^2 + \lambda \|\boldsymbol{w}\|_2^2 \tag{11.6}$$

其中正则化参数 $\lambda > 0$. 式 (11.6) 称为"岭回归"(ridge regression).

若采用 L_1 范数, 有

$$\min_{\boldsymbol{w}} \sum_{i=1}^{m} \left(y_i - \boldsymbol{w}^{\mathrm{T}} \boldsymbol{x}_i\right)^2 + \lambda \|\boldsymbol{w}\|_1 \tag{11.7}$$

其中正则化参数 $\lambda > 0$. 式 (11.7) 称为 LASSO

L_1 范数和 L_2 范数正则化都有助于降低过拟合风险. 但 L_1 范数正则化还会带来一个额外的好处:

它比 L_2 范数正则化更容易获得"稀疏"解, 也即是, 它求得的 $\boldsymbol{w}$ 会有更少的非零分量.

如图 11.2

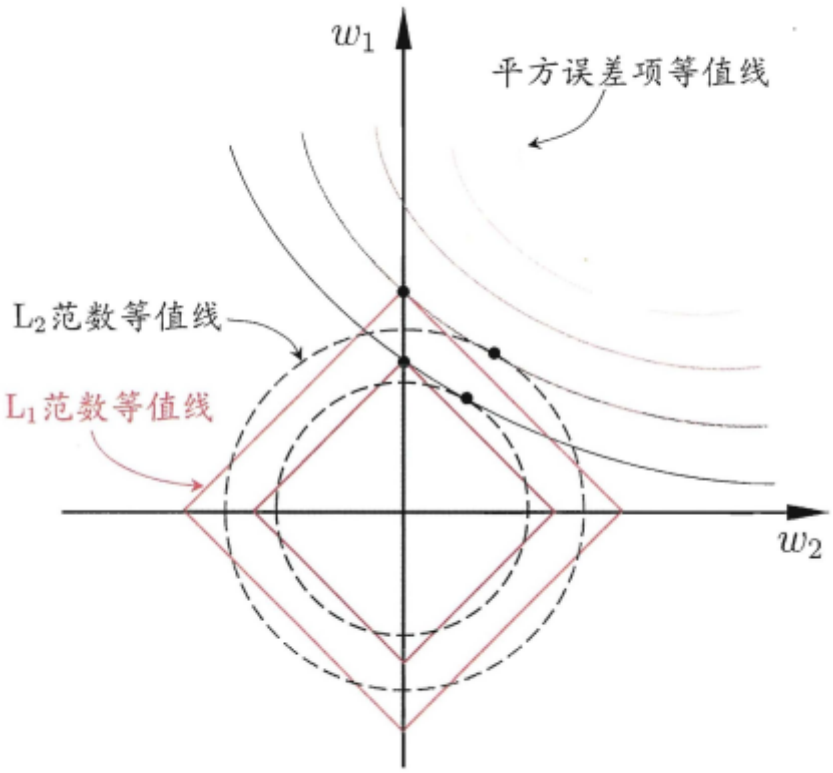

图 11.2 L_1 正则化比 L_2 正则化更易于得到稀疏解

L_1 正则化求解过程暂放.