目录

第八章　集成学习

集成学习 (ensemble learning) 通过构建整合多个学习器来完成学习任务。

0.1　个体与集成

当所有个体学习器都由同样的学习算法生成时，也即集成中只包含同种类型的个体学习器时，称为**同质**（homogeneous）**集成**，这些个体学习器又被称为基学习器（base learner），相应的学习算法称为基学习算法（base learning algorithm）；

当个体学习器由不同的学习算法生成时，称为**异质**（heterogenous）**集成**，这些个体学习器称为组件学习器（component learner）或直接称为个体学习器。

集成学习通过结合多个学习器，通常能获得比单一学习器更优越的泛化性能，对弱学习器（weak learner）（略优于随机猜测的学习器，例如二分类任务中精度略高于 50%）的提升尤为明显。要获得好的集成效果，个体学习器应该“好而不同”（准确性与多样性）。

关于 Hoeffding 不等式：假设抛硬币正面朝上的概率为 p，反面朝上的概率为 $1-p$。令 $H(n)$ 代表抛 n 次硬币所得正面朝上的次数，则最多 k 次正面朝上的概率为

$$P(H(n) \leq k) = \sum_{i=0}^{k} \binom{n}{i} p^i (1-p)^{n-i}$$

对 $\delta > 0, \quad k = (p-\delta)n$，有 Hoeffding 不等式

$$P(H(n) \leq (p-\delta)n) \leq e^{-2\delta^2 n}$$

假设基学习器的错误率相互独立，则随着集成中个体分类器数目 T 的增大，集成的错误率将指数级下降，最终趋向于零。

证明：考虑二分类问题 $y \in \{-1, +1\}$ 和真实函数 f，假定基分类器的错误率为 ϵ，即对每个基分类器 h_i 都有：

$$P(h_i(\mathbf{x}) \neq f(\mathbf{x})) = \epsilon \tag{1}$$

假设集成通过**简单投票法**结合 T 个基分类器，若有超过半数的基分类器分类正确，则集成分类就正确：

$$H(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{T} h_i(\mathbf{x})\right) \tag{2}$$

假设基分类器的错误率相互独立，结合上面的 Hoeffding 不等式：令 $p = 1-\epsilon$，$k = \lfloor \frac{T}{2} \rfloor = (p-\delta) \times T$，

可得 $\delta = p - \frac{k}{T} = 1 - \epsilon - \frac{1}{T}\lfloor\frac{T}{2}\rfloor$，所以集成的错误率为：

$$\begin{aligned} P(H(\mathbf{x}) \neq f(\mathbf{x})) &= \sum_{k=0}^{\lfloor T/2 \rfloor} \binom{T}{k} (1-\epsilon)^k \epsilon^{T-k} \\ &\leq \exp\left(-2(1-\epsilon-\frac{1}{T}\lfloor\frac{T}{2}\rfloor)\right) \\ &= \exp\left(-\frac{1}{2}T(2-2\epsilon-1)\right) \\ &= \exp\left(-\frac{1}{2}T(1-2\epsilon)^2\right) \end{aligned} \tag{3}$$

得证。

上面的分析中有一个关键假设：**基学习器的误差相互独立**。然而，现实任务中，个体学习器均是为解决同一个问题训练出来的，它们显然不可能相互独立。因此，**集成学习研究的核心就是：如何产生的那个并结合“好而不同”的学习器。**

根据个体学习器的生成方式，集成学习方法大致可分为两个大类：

- 个体学习器间存在强依赖关系，必须串行生成的序列化方法，例如：Boosting 算法；
- 个体学习器间不存在强依赖关系，可同时生成的并行化方法，例如：Bagging，随机森林。

0.2 Boosting

Boosting 族算法的工作机制：先从初始训练集训练出一个基学习器，再根据基学习器的表现对**样本分布**进行调整，修正先前基学习器做错的训练样本；然后基于调整后的样本分布来训练下一个基学习器；如此重复进行，直至基学习器的数目达到指定的值 T，最终将这 T 个基学习器进行加权结合。

Boosting 族算法最著名的代表是 AdaBoost 算法，其中 $y_i \in \{-1, +1\}, f$ 是真实函数。

AdaBoost 算法

输入： 训练集 $D = \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \cdots, (\boldsymbol{x_m}, y_m)\}$；基学习算法 $\mathfrak{L}$；训练轮数 T.

1: $\mathcal{D}_1(\boldsymbol{x}) = \frac{1}{m}$ [$\mathcal{D}_t$ 为第 t 轮的分布]
2: **for** $t = 1, 2, \cdots, T$ **do**
3: $h_t = \mathfrak{L}(D, \mathcal{D}_t)$; [从数据集 D 中基于分布 $\mathcal{D}_t$ 训练得到分类器 h_t]
4: $\epsilon_t = P_{\boldsymbol{x}\sim\mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$ [估计 h_t 的误差]
5: **if** $\epsilon_t > 0.5$ **then**
6: ***break*** [检查当前基分类器是否比随机猜测要好]
7: **end if**
8: $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$ [分类器权重更新公式]
9: $\mathcal{D}_{t+1}(\boldsymbol{x}) = \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} \times \exp\left(-\alpha_t f(\boldsymbol{x}) h_t(\boldsymbol{x})\right)$ [更新样本分布，其中 Z_t 是规范化因子，以确保 $\mathcal{D}_{t+1}$ 是一个分布]
10: **end for**

输出： $H(\boldsymbol{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\boldsymbol{x})\right)$

从偏差-方差分解的角度看，**Boosting 主要关注降低偏差**，因此 Boosting 能基于泛华性能相当弱的学习器构建出很强的集成。

下面介绍两种产生彼此有较大差异的基学习器的算法：Bagging 和随机森林。

0.3 Bagging

Bagging（由 Bootstrap AGGregatING 缩写而来）通过对训练样本采样，产生若干个不同的子集，从每个子集中训练出基学习器。

Bagging 直接基于**自助采样法**（bootstrap sampling）：给定包含 m 个样本的数据集，先随机取出一个样本放入采样集中，再把样本**放回**初始数据集，使得下次采样时该样本仍有可能被选中，这样经过 m 次随机采样后，得到含有 m 个样本的采样集。初始训练集中的样本在采样集里有的多次出现，有的从未出现，约有 63.2% 的样本出现在采样集里。

Bagging 算法的基本流程：先采样出 T 个含有 m 个训练样本的采样集，然后基于每个采样集训练出一个基学习器，再将这些基学习器进行结合。在结合时，Bagging 通常对分类任务使用**简单投票法**，对回归任务使用**简单平均法**。

Bagging 算法

输入： 训练集 $D = \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \cdots, (\boldsymbol{x_m}, y_m)\}$；基学习算法 $\mathfrak{L}$；训练轮数 T.

1: **for** $t = 1, 2, \cdots, T$ **do**

2: $\quad h_t = \mathfrak{L}(D, \mathcal{D}_{bs})$； $\quad$ [$\mathcal{D}_{bs}$ 是自主采样产生的样本分布]

3: **end for**

输出： $H(\boldsymbol{x}) = \arg\max\limits_{y \in \mathcal{Y}} \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y)$

与标准 AdaBoost 只适用于二分类任务不同的是，Bagging 能不经修改地用于多分类、回归等任务。

从偏差-方差分解的角度看，Bagging 主要关注**降低方差**，因此它在不剪枝决策树、神经网络等易受样本扰动的学习器上效果更明显。

0.4 随机森林

随机森林（(Random Forest, 简称 RF) 是 Bagging 的一个扩展体：在以决策树为基学习器构建 Bagging 集成的基础上，进一步在决策树的训练过程中引入了**随机属性选择**。因此，随机森林中基学习器的多样性不仅来自样本扰动，还来自属性扰动，这就使得最终集成的泛化性能可通过个体学习器之间差异度的增加而进一步提升。

随机森林的训练效率通常优于 Bagging。

0.5 结合策略

学习器结合的好处：

- 从统计方面看：由于学习任务的假设空间往往很大，可能有多个假设在训练集上达到同等性能，结合多个学习器可以降低使用单学习器可能因误选而导致泛化性能不佳的风险；
- 从计算方面看：结合可以降低陷入糟糕局部极小点的风险
- 从表示方面看：某些学习任务的真实假设可能不在当前学习算法所考虑的假设空间中，使用多个学习器结合，由于相应的假设空间有所扩大，有可能学得更好的近似。

假定集成包含 T 个基学习器 $\{h_1, h_2, ..., h_T\}$，其中 h_i 在示例 $\boldsymbol{x}$ 上的输出为 $h_i(\boldsymbol{x})$。下面是几种对 h_i 进行结合的常见策略。

0.5.1 平均法

- 简单平均法 (simple averaging)： $H(\boldsymbol{x}) = \frac{1}{T} \sum_{i=1}^{T} h_i(\boldsymbol{x})$
- 加权平均法 (weighted averaging)： $H(\boldsymbol{x}) = \frac{1}{T} \sum_{i=1}^{T} w_i h_i(\boldsymbol{x})$

其中 w_i 是个体学习器 h_i 的权重，通常要求 $w_i \geq 0, \sum_{i=1}^{T} w_i = 1$。

0.5.2 投票法

- 绝对大多数投票法：所有分类器在某标记上的得票超过半数，则预测为该标记，否则拒绝预测。
- 相对多数投票法：预测为得票最多的标记。
- 加权投票法

0.5.3 学习法

学习法即通过另一个学习器来结合基学习器，一个典型代表是 Stacking。把个体学习器称为初级学习器，把用于结合的学习器称为次级学习器或元学习器 (meta-learner)。

Stacking 算法

输入： 训练集 $D = \{(\boldsymbol{x_1}, y_1), (\boldsymbol{x_2}, y_2), \cdots, (\boldsymbol{x_m}, y_m)\}$；初级学习算法 $\mathfrak{L}_1, \mathfrak{L}_2, \cdots, \mathfrak{L}_T$；次级学习算法 $\mathfrak{L}$.

for $t = 1, 2, \cdots, T$ **do**
  $h_t = \mathfrak{L}_t(D)$; [使用初级学习算法 $\mathfrak{L}_t$ 产生初级学习器 h_t]
end for
$D' = \varnothing$ [D' 是次级训练集]
for $i = 1, 2, \cdots, m$ **do**
  for $t = 1, 2, \cdots, T$ **do**
    $z_{it} = h_t(\boldsymbol{x}_i)$
  end for
  $D' = D' \cup ((z_{i1}, z_{i2}, \cdots, z_{iT}), y_i)$
end for
$h' = \mathfrak{L}(D')$

输出： $H(\boldsymbol{x}) = h'(h_1(\boldsymbol{x}), h_2(\boldsymbol{x}), \cdots, h_T(\boldsymbol{x}))$

次级训练集是由初级学习器产生的，如果直接用初级学习器的训练集来产生次级训练集，则很有可能产生过拟合。

将初级学习器的输出类概率作为次级学习器的输入属性，用多响应线性回归 (Multi-response Linear Regression，简称 MLR) 作为次级学习算法效果较好。

MLR 是基于线性回归的分类器，它对每个类分别进行线性回归，属于该类的训练样例所对应的输出被置位 1，其他类置为 0；测试样例将被分给输出值最大的类。