第7章　集成学习

刘家锋

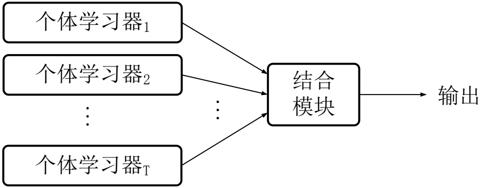
哈尔滨工业大学

第7章 集成学习

7.1 集成学习

集成学习

- **个体与集成**
  - 个体学习器：由一个现有的学习算法从训练数据产生，也称为“基学习器”；
  - 集成学习：组合多个个体学习器，取得比个体学习器更好的性能，也称为“委员会学习”，“多分类器系统”等；



集成学习的性能

- **集成学习如何能够获得更好的性能？**
  - 要求个体学习器有一定的准确性和多样性；
  - 例如：集成三个不同的分类器h_1, h_2, h_3，在三个不同测试样例$\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3$上的分类结果

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$
h_1	✓	✓	×
h_2	×	✓	✓
h_3	✓	×	✓
集成	✓	✓	✓

(a) 性能提升

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$
h_1	✓	✓	×
h_2	✓	✓	×
h_3	✓	✓	×
集成	✓	✓	×

(b) 性能不变

	$\mathbf{x}_1$	$\mathbf{x}_2$	$\mathbf{x}_3$
h_1	✓	×	×
h_2	×	✓	×
h_3	×	×	✓
集成	×	×	×

(c) 性能下降

集成学习的性能

- **集成学习性能的简单分析**
  - 考虑二分类问题，假设T个基分类器的错误率均为ϵ，并且相互独立；
  - 集成采用简单投票法，只要超过半数的基分类器正确，集成分类器就能判别正确；
  - 集成分类器的错误率：

$$P(\text{error}) = \sum_{k=0}^{\lfloor T/2 \rfloor} \binom{T}{k} (1-\epsilon)^k \epsilon^{T-k} \le \exp\left(-\frac{1}{2} T (1-2\epsilon)^2\right)$$

  - 集成错误率随数量T的增大指数下降；

结合的策略

- **回归器的结合**
  - T个基回归器$\{h_1, \cdots, h_T\}$，$h_i(\mathbf{x}) \in \mathbb{R}$表示回归器$h_i$在示例$\mathbf{x}$上的输出，$H(\mathbf{x})$表示集成回归器的输出；
  - **简单平均法：**

$$H(\mathbf{x}) = \frac{1}{T}\sum_{i=1}^{T} h_i(\mathbf{x})$$

  - **加权平均法：**

$$H(\mathbf{x}) = \frac{1}{T}\sum_{i=1}^{T} w_i h_i(\mathbf{x}), \quad w_i \geq 0, \sum_{i=1}^{T} w_i = 1$$

权重$w_1, \cdots, w_T$是需要学习的参数；

结合的策略

- 分类器的结合
  - 用向量$(h_i^1(\mathbf{x}), \cdots, h_i^N(\mathbf{x}))^t$表示基分类器$h_i$在示例$\mathbf{x}$上的输出，$N$为类别标记数；
  - **绝对多数投票法**：只有当某个类别得票过半时，判断为该类别标记
$$H(\mathbf{x}) = \begin{cases} c_j, & \text{if } \sum_{i=1}^T h_i^j(\mathbf{x}) > \frac{1}{2}T \\ \text{reject}, & \text{otherwise} \end{cases}$$
  - **相对多数投票法**：预测为得票最多的类别标记
$$H(\mathbf{x}) = c_{\arg\max_j \sum_{i=1}^T h_i^j(\mathbf{x})}$$
  - **加权投票法**：
$$H(\mathbf{x}) = c_{\arg\max_j \sum_{i=1}^T w_i h_i^j(\mathbf{x})}, \quad w_i \geq 0, \sum_{i=1}^T w_i = 1$$

硬投票与软投票

- **硬投票**
  - 类别标记：$h_i^j(\mathbf{x}) \in \{0,1\}$

$$h_i^j(\mathbf{x}) = \begin{cases} 1, & h_i\text{预测}\mathbf{x}\text{的类别标记为}c_j \\ 0, & \text{otherwise} \end{cases}$$

- **软投票**
  - 类别标记：$h_i^j(\mathbf{x}) \in [0,1]$，一般为$h_i$估计的后验概率

$$h_i^j(\mathbf{x}) = P_i(y = c_j|\mathbf{x})$$

学习法

Algorithm 1 Stacking算法

Input: 训练集$D = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_m, y_m)\}$，基学习算法$\mathcal{L}_1, \cdots, \mathcal{L}_T$，结合学习算法$\mathcal{L}$

Output: 集成学习器$H(\mathbf{x})$

procedure STACKING
    for $t = 1, \cdots, T$ **do**
        $h_t = \mathcal{L}_t(D)$ ▷ 训练基学习器
    end for
    $D' = \varnothing$ ▷ 初始化结合训练集
    for $i = 1, \cdots, m$ **do**
        $\mathbf{z}_i = (h_1(\mathbf{x}_i), \cdots, h_T(\mathbf{x}))^t$ ▷ 计算基学习器在训练集上的输出
        $D' \leftarrow D' \cup \{(\mathbf{z}_i, y_i)\}$ ▷ 产生结合器学习样本集
    end for
    $h' = \mathcal{L}(D')$ ▷ 学习结合器
return $H(\mathbf{x}) = h'(h_1(\mathbf{x}), \cdots, h_T(\mathbf{x}))$ ▷ 输出集成学习预测结果
end procedure

7.2 Bagging和随机森林

训练集的重采样

- **自助法重采样(bootstrap sampling)**
  - 训练集$D=\{(\mathbf{x}_1,y_1),\cdots,(\mathbf{x}_m,y_m)\}$包含$m$个样本；
  - 每次从训练集D中随机抽取一个样本放入采样集D_{bs}，抽取的样本放回D中，下一次采样仍有可能被选中；
  - 重复抽样m次，得到重采样训练集$D_{bs}=bootstrap(D)$；
- **Bagging算法**
  - 初始训练集D中约有63.2%的样本出现在重采样集D_{bs}中；
  - 使用同样的学习算法$\mathcal{L}$，在不同的采样集D_{bs}上学习，可以得到具有“多样性”的学习器；

Bagging算法

Algorithm 2 Bagging算法

Input: 训练集$D=\{(\mathbf{x}_1,y_1),\cdots,(\mathbf{x}_m,y_m)\}$，基学习算法$\mathcal{L}$，基学习器数量$T$

Output: 集成学习器$H(\mathbf{x})$

```
procedure BAGGING
    for t = 1, ··· , T do
        D_bs = bootstrap(D)                                  ▷ bootstrap抽样
        h_t = L(D_bs)                                        ▷ 训练基学习器
    end for
return H(x) = arg max_{y∈Y} Σ_{t=1}^{T} I(h_t(x) = y)        ▷ 集成预测
end procedure
```

$$D_{bs}=bootstrap(D)$$

$$h_t=\mathcal{L}(D_{bs})$$

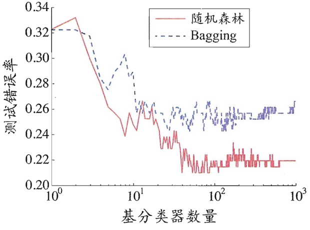
$$H(\mathbf{x})=\arg\max_{y\in\mathcal{Y}}\sum_{t=1}^{T}\mathbb{I}(h_t(\mathbf{x})=y)$$

随机森林

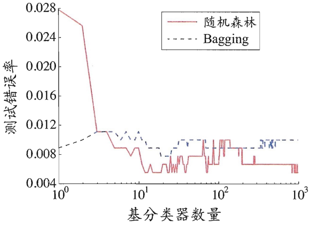
- **随机森林(Random Forrest)**
  - 随机森林是Bagging的一个扩展变体；
  - 以决策树作为基学习器，采用bagging算法集成学习；
  - 为了增加基学习器的多样性，除了训练集上的随机性之外，增加了划分属性的随机性；
  - 对于每个节点，先从属性集合中随机选择包含k个元素的子集，在子集中选择最优属性用于划分；
  - 决策树学习过程中，一般不做剪枝处理；

随机森林

- **随机森林与Bagging的对比**
  - 分别用随机森林算法，和使用标准决策树作为基学习器的Bagging算法学习UCI的两个数据集；
  - 随着基学习器的数量增多，随机森林的性能提升明显，并优于Bagging算法；



(a) glass 数据集



(b) auto-mpg 数据集

7.3 Boosting

序列化集成学习方法

- **Boosting**
  - Bagging算法属于并行化的集成学习方法，基学习器之间互无关联；
  - Boosting算法是序列化的集成学习，下一个基学习器的学习需要根据之前的学习结果来调整；
  - 这其中最有代表性的是AdaBoost算法；
- **AdaBoost集成分类器**
  - AdaBoost一般用于二分类问题，类别标记$y_i \in \{-1, +1\}$；
  - 学习T个基分类器$\{h_t(\mathbf{x})\}$，加权线性组合：

$$H(\mathbf{x}) = \sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})$$

  - 基分类器的权重α_t是算法学习得到的参数；

AdaBoost算法

- **样本集的加权重采样**
  - AdaBoost算法中基分类器的训练集也是由样本集D重采样得到的；
  - 样本按照一个不断变化的分布，而不是均匀分布来抽样的；
  - 简单地说，为每个样本赋予一个权重w_i，以w_i为概率决定是否抽样相应样本；
  - 学习得到一个新的基分类器$h_t(\mathbf{x})$之后，对权重调整，被正确分类的样本权重降低，错误分类的权重增加：

$$w_i \leftarrow \frac{w_i}{z_t} \times \begin{cases} e^{-\alpha_t}, & h_t\text{正确分类}\mathbf{x}_i \\ e^{\alpha_t}, & h_t\text{错误分类}\mathbf{x}_i \end{cases}$$

其中，z_t为归一化因子，保证$\sum_{i=1}^{m} w_i = 1$；

AdaBoost算法

- **基分类器的权重**
  - 基分类器$h_t(\mathbf{x})$的权重α_t体现了分类性能，与错误率ϵ_t有关；
  - 错误率ϵ_t同样是在一个加权重采样样本集上统计的，而不是初始训练集D；
  - 令依据权重$\{w_i\}$抽样的数据集为D_ϵ，错误率为：

$$\epsilon_t = \frac{1}{m} \sum_{\mathbf{x} \in D_\epsilon} \mathbb{I}(h_t(\mathbf{x}) \neq y)$$

  - 基分类器$h_t(\mathbf{x})$的权重：

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right)$$

AdaBoost算法

Algorithm 3 AdaBoost算法

Input: 训练集$D = \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_m, y_m)\}$，基分类算法$\mathcal{L}$，基分类器数量$T$

Output: 集成分类器$H(\mathbf{x})$

procedure AdaBoost

  $w_i = 1/m, i = 1, \cdots, m$ ▷ 初始化样本权重

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

    $D_w = bootstrap(D, \{w_i\})$ ▷ 加权抽样

    $D_\epsilon = bootstrap(D, \{w_i\})$

    $h_t = \mathcal{L}(D_w)$ ▷ 训练基分类器

    $\epsilon_t = \frac{1}{m} \sum_{\mathbf{x} \in D_\epsilon} \mathbb{I}(h_t(\mathbf{x}) \neq y)$ ▷ 计算分类误差

    if $\epsilon_t > 0.5$ **then break**

    $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right), w_i \leftarrow \frac{w_i}{z_t} \times \begin{cases} e^{-\alpha_t}, & h_t\text{正确分类}\mathbf{x}_i \\ e^{\alpha_t}, & h_t\text{错误分类}\mathbf{x}_i \end{cases}$

  end for

return $H(\mathbf{x}) = \text{sign}\left(\sum_{t=1}^{T} \alpha_t h_t(\mathbf{x})\right)$ ▷ 集成预测

end procedure

AdaBoost算法

- **AdaBoost示例**
  - AdaBoost算法学习3个基分类器的集成，基分类器采用线性分类器；