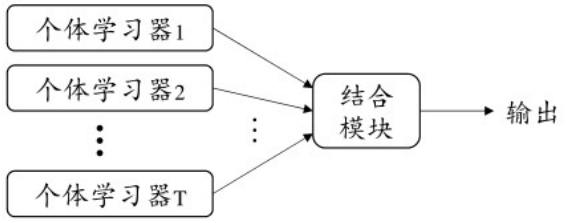
第八章：集成学习

集成学习

- 个体与集成
- Boosting
  - Adaboost
- Bagging与随机森林
- 结合策略
  - 平均法
  - 投票法
  - 学习法
- 多样性
  - 误差-分歧分解
  - 多样性度量
  - 多样性扰动

个体与集成

- 集成学习(ensemble learning)通过构建并结合多个学习器来提升性能



个体与集成

- 考虑一个简单的例子，在二分类问题中，假定3个分类器在三个样本中的表现如下图所示，其中√ 表示分类正确，X 号表示分类错误，集成的结果通过投票产生。

	测试例1	测试例2	测试例3
h_1	√	√	×
h_2	×	√	√
h_3	√	×	√
集群	√	√	√

(a) 集群提升性能

	测试例1	测试例2	测试例3
h_1	√	√	×
h_2	√	√	×
h_3	√	√	×
集群	√	√	×

(b) 集群不起作用

	测试例1	测试例2	测试例3
h_1	√	×	×
h_2	×	√	×
h_3	×	×	√
集群	×	×	×

(c) 集群起负作用

- 集成个体应：好而不同

个体与集成 – 简单分析

- 考虑二分类问题，假设基分类器的错误率为：

$$P\left(h_i\left(\boldsymbol{x}\right) \neq f\left(\boldsymbol{x}\right)\right) = \epsilon$$

- 假设集成通过简单投票法结合T个分类器，若有超过半数的基分类器正确则分类就正确

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

个体与集成 – 简单分析

- 假设基分类器的错误率相互独立，则由Hoeffding不等式可得集成的错误率为：

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

- 上式显示，在一定条件下，随着集成分类器数目的增加，集成的错误率将指数级下降，最终趋向于0

个体与集成 – 简单分析

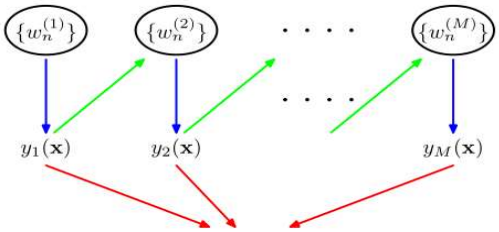
- 上面的分析有一个关键假设：基学习器的误差相互独立
- 现实任务中，个体学习器是为解决同一个问题训练出来的，显然不可能互相独立
- 事实上，个体学习器的“准确性”和“多样性”本身就存在冲突
- 如何产生“好而不同”的个体学习器是集成学习研究的核心
- 集成学习大致可分为两大类

集成学习

- 个体与集成
- Boosting
  - Adaboost
- Bagging与随机森林
- 结合策略
  - 平均法
  - 投票法
  - 学习法
- 多样性
  - 误差-分歧分解
  - 多样性度量
  - 多样性扰动

Boosting

- 个体学习器存在强依赖关系，
- 串行生成
- 每次调整训练数据的样本分布



$$Y_M(\mathbf{x}) = \text{sign}\left(\sum_m^M \alpha_m y_m(\mathbf{x})\right)$$

Boosting - Boosting算法

Input: Sample distribution $\mathcal{D}$;
Base learning algorithm $\mathfrak{L}$;
Number of learning rounds T.

Process:
1. $\mathcal{D}_1 = \mathcal{D}$. % Initialize distribution
2. **for** $t = 1, \ldots, T$:
3. $h_t = \mathfrak{L}(\mathcal{D}_t)$; % Train a weak learner from distribution $\mathcal{D}_t$
4. $\epsilon_t = P_{\boldsymbol{x} \sim D_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$; % Evaluate the error of h_t
5. $\mathcal{D}_{t+1} = Adjust\_Distribution(\mathcal{D}_t, \epsilon_t)$
6. **end**

Output: $H(\boldsymbol{x}) = Combine\_Outputs(\{h_1(\boldsymbol{x}), \ldots, h_t(\boldsymbol{x})\})$

- Boosting族算法最著名的代表是AdaBoost

Boosting – AdaBoost算法

输入: 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
基学习算法 $\mathfrak{L}$;
训练轮数 T.

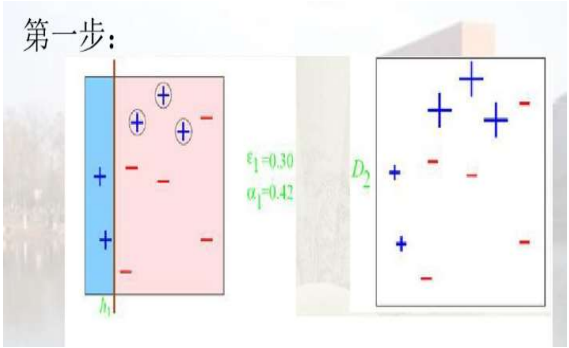
过程:
1: $\mathcal{D}_1(\boldsymbol{x}) = 1/m$.
2: **for** $t = 1, 2, \ldots, T$ **do**
3: $h_t = \mathfrak{L}(D, \mathcal{D}_t)$;
4: $\epsilon_t = P_{\boldsymbol{x} \sim \mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))$;
5: **if** $\epsilon_t > 0.5$ **then break**
6: $\alpha_t = \frac{1}{2} \ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$;
7: $\mathcal{D}_{t+1}(\boldsymbol{x}) = \frac{\mathcal{D}_t(\boldsymbol{x})}{Z_t} \times \begin{cases} \exp(-\alpha_t), & \text{if } h_t(\boldsymbol{x}) = f(\boldsymbol{x}) \\ \exp(\alpha_t), & \text{if } h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}) \end{cases}$
$= \frac{\mathcal{D}_t(\boldsymbol{x}) \exp(-\alpha_t f(\boldsymbol{x}) h_t(\boldsymbol{x}))}{Z_t}$
8: **end for**
输出: $H(\boldsymbol{x}) = \text{sign}\left(\sum_{t=1}^T \alpha_t h_t(\boldsymbol{x})\right)$

Boosting

- **AdaBoost**（Adaptive boosting）算法：
  - 初始化训练数据的权值分布。如果有N个样本，则每一个训练样本最开始时都被赋予相同的权值：1/N。
  - 训练弱分类器。具体训练过程中，如果某个样本点已经被准确地分类，那么在构造下一个训练集中，它的权值就被降低；相反，如果某个样本点没有被准确地分类，那么它的权值就得到提高。然后，权值更新过的样本集被用于训练下一个分类器，整个训练过程如此迭代地进行下去。
  - 将各个训练得到的弱分类器组合成强分类器。各个弱分类器的训练过程结束后，加大分类误差率小的弱分类器的权重，使其在最终的分类函数中起着较大的决定作用，而降低分类误差率大的弱分类器的权重，使其在最终的分类函数中起着较小的决定作用。换言之，误差率低的弱分类器在最终分类器中占的权重较大，否则较小。

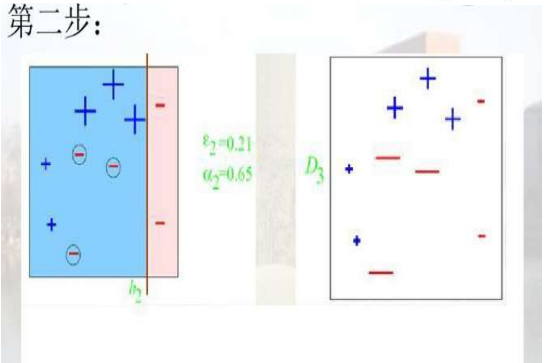
$$\alpha_t = \frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t}$$

第一步：

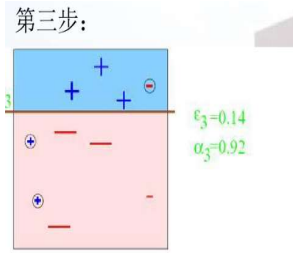


$$\alpha_t = \frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t}$$

第二步：



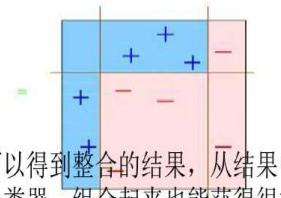
第三步：



$$\alpha_t = \frac{1}{2}\ln\frac{1-\varepsilon_t}{\varepsilon_t}$$

得到一个子分类器h_3

$$H_{final} = \text{sign}\left(0.42 \quad + 0.65 \quad + 0.92\right)$$



因此可以得到整合的结果，从结果中看，即使简单的分类器，组合起来也能获得很好的分类效果。!

Boosting – AdaBoost推导

- 基学习器的线性组合

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

- 最小化指数损失函数

$$\ell_{\exp}(H \mid \mathcal{D}) = \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H(\boldsymbol{x})}]$$

Boosting – AdaBoost推导

- 若$H(x)$能令指数损失函数最小化，则上式对$H(x)$的偏导值为0，即

$$\frac{\partial \ell_{\exp}(H \mid \mathcal{D})}{\partial H(\boldsymbol{x})} = -e^{-H(\boldsymbol{x})} P(f(\boldsymbol{x}) = 1 \mid \boldsymbol{x}) + e^{H(\boldsymbol{x})} P(f(\boldsymbol{x}) = -1 \mid \boldsymbol{x})$$

$$\operatorname{sign}(H(\boldsymbol{x})) = \operatorname{sign}\left(\frac{1}{2}\ln\frac{P(f(x)=1 \mid \boldsymbol{x})}{P(f(x)=-1 \mid \boldsymbol{x})}\right) \qquad H(\boldsymbol{x}) = \frac{1}{2}\ln\frac{P(f(x)=1 \mid \boldsymbol{x})}{P(f(x)=-1 \mid \boldsymbol{x})}$$

$$= \begin{cases} 1, & P(f(x)=1 \mid \boldsymbol{x}) > P(f(x)=-1 \mid \boldsymbol{x}) \\ -1, & P(f(x)=1 \mid \boldsymbol{x}) < P(f(x)=-1 \mid \boldsymbol{x}) \end{cases}$$

$$= \underset{y\in\{-1,1\}}{\arg\max} P(f(x) = y \mid \boldsymbol{x}),$$

$sign(H(x))$达到了贝叶斯最优错误率，说明指数损失函数是分类任务原来0/1损失函数的一致的替代函数。

Boosting – AdaBoost推导

- 当基分类器h_t基于分布D_t产生后，该基分类器的权重α_t应使得$\alpha_t h_t$最小化指数损失函数

$$\begin{aligned}
\ell_{\exp}(\alpha_t h_t \mid \mathcal{D}_t) &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[e^{-f(\boldsymbol{x})\alpha_t h_t(\boldsymbol{x})}\right] \\
&= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[e^{-\alpha_t}\mathbb{I}(f(\boldsymbol{x}) = h_t(\boldsymbol{x})) + e^{\alpha_t}\mathbb{I}(f(\boldsymbol{x}) \neq h_t(\boldsymbol{x}))\right] \\
&= e^{-\alpha_t} P_{\boldsymbol{x}\sim\mathcal{D}_t}(f(\boldsymbol{x}) = h_t(\boldsymbol{x})) + e^{\alpha_t} P_{\boldsymbol{x}\sim\mathcal{D}_t}(f(\boldsymbol{x}) \neq h_t(\boldsymbol{x})) \\
&= e^{-\alpha_t}(1-\epsilon_t) + e^{\alpha_t}\epsilon_t \qquad \epsilon_t = P_{\boldsymbol{x}\sim\mathcal{D}_t}(h_t(\boldsymbol{x}) \neq f(\boldsymbol{x}))
\end{aligned}$$

- 令指数损失函数的导数为0，即

$$\frac{\partial \ell_{\exp}(\alpha_t h_t \mid \mathcal{D}_t)}{\partial \alpha_t} = -e^{-\alpha_t}(1-\epsilon_t) + e^{\alpha_t}\epsilon_t \qquad \alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$$

Boosting – AdaBoost推导

- 在获得H_{t-1}之后的样本分布进行调整，使得下一轮的基学习器h_t能纠正H_{t-1}的一些错误。理想的h_t能纠正全部错误

$$\begin{aligned}
\ell_{\exp}(H_{t-1} + h_t \mid \mathcal{D}) &= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})(H_{t-1}(\boldsymbol{x})+h_t(\boldsymbol{x}))}] \\
&= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})} e^{-f(\boldsymbol{x})h_t(\boldsymbol{x})}]
\end{aligned}$$

- 泰勒展开近似为

$$\begin{aligned}
\ell_{\exp}(H_{t-1} + h_t \mid \mathcal{D}) &\simeq \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\left(1 - f(\boldsymbol{x})h_t(\boldsymbol{x}) + \frac{f^2(\boldsymbol{x})h_t^2(\boldsymbol{x})}{2}\right)\right] \\
&= \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\left(1 - f(\boldsymbol{x})h_t(\boldsymbol{x}) + \frac{1}{2}\right)\right]
\end{aligned}$$

Boosting – AdaBoost推导

- 于是，理想的基学习器：

$$\begin{aligned} h_t(\boldsymbol{x}) &= \arg\min_h \ell_{\exp}(H_{t-1}+h \mid \mathcal{D}) \\ &= \arg\min_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\left(1-f(\boldsymbol{x})h(\boldsymbol{x})+\frac{1}{2}\right)\right] \\ &= \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}f(\boldsymbol{x})h(\boldsymbol{x})\right] \\ &= \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\frac{e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}]}f(\boldsymbol{x})h(\boldsymbol{x})\right], \end{aligned}$$

- 注意到$\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}]$是一个常数，令$D_t$表示一个分布：

$$\mathcal{D}_t(\boldsymbol{x}) = \frac{\mathcal{D}(\boldsymbol{x})e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}]}$$

Boosting – AdaBoost推导

- 根据数学期望的定义，这等价于令：

$$\begin{aligned} h_t(\boldsymbol{x}) &= \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[\frac{e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}]}f(\boldsymbol{x})h(\boldsymbol{x})\right] \\ &= \arg\max_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[f(\boldsymbol{x})h(\boldsymbol{x})\right] . \end{aligned}$$

- 由$f(x), h(x) \in \{-1, +1\}$有：

$$f(\boldsymbol{x})h(\boldsymbol{x}) = 1 - 2\,\mathbb{I}\big(f(\boldsymbol{x}) \neq h(\boldsymbol{x})\big)$$

Boosting – AdaBoost推导

- 则理想的基学习器

$$h_t(\boldsymbol{x}) = \arg\min_h \mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}_t}\left[\mathbb{I}\big(f(\boldsymbol{x}) \neq h(\boldsymbol{x})\big)\right]$$

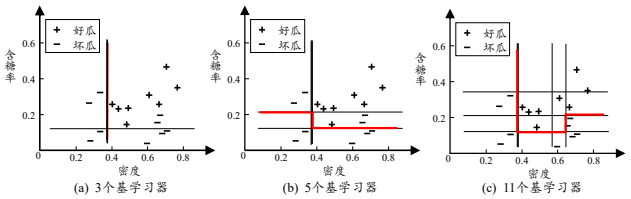
- 最终的样本分布更新公式

$$\begin{aligned} \mathcal{D}_{t+1}(\boldsymbol{x}) &= \frac{\mathcal{D}(\boldsymbol{x})\,e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}\right]} \\ &= \frac{\mathcal{D}(\boldsymbol{x})\,e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}e^{-f(\boldsymbol{x})\alpha_t h_t(\boldsymbol{x})}}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}\right]} \\ &= \mathcal{D}_t(\boldsymbol{x})\cdot e^{-f(\boldsymbol{x})\alpha_t h_t(\boldsymbol{x})}\frac{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_{t-1}(\boldsymbol{x})}\right]}{\mathbb{E}_{\boldsymbol{x}\sim\mathcal{D}}\left[e^{-f(\boldsymbol{x})H_t(\boldsymbol{x})}\right]} \end{aligned}$$

Boosting – AdaBoost注意事项

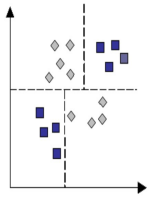
- 数据分布的学习
  - 重赋权法
  - 重采样法
- 重启动，避免训练过程过早停止

Boosting – AdaBoost实验



- 从偏差-方差的角度：降低偏差，可对泛化性能相当弱的学习器构造出很强的集成

- Boosting+CART+任意损失函数=GBDT
- CART假设决策树是二叉树，内部结点特征的取值为“是”和“否”，左分支是取值为“是”的分支，右分支是取值为“否”的分支。这样的决策树等价于递归地二分每个特征，将输入空间即特征空间划分为有限个单元，并在这些单元上确定预测的概率分布，也就是在输入给定的条件下输出的条件概率分布。
- CART算法由以下两步组成：
  - （1）决策树的生成：基于训练数据集生成决策树，生成的决策树要尽量大（大是为了更好地泛化）；
  - （2）决策树剪枝：用验证数据集对已生成的树进行剪枝并选择最优子树，这时损失函数最小作为剪枝的标准。



- GBDT模型表示为决策树的加法模型

$$f_M(x)=\sum_{m=1}^{M}T(x;\Theta_m)$$

其中,T (x; θm)表示决策树; m为决策树的参数; M为树的个数。

采用 $f_m(x)=f_{m-1}(x)+T(x;\Theta_m)$

要优化的损失函数

经验风险极小化确定下一棵树的参数

$$\hat{\Theta}_m=\arg\min_{\Theta_m}\sum_{i=1}^{N}L(y_i,f_{m-1}(x_i)+T(x_i;\Theta_m))$$

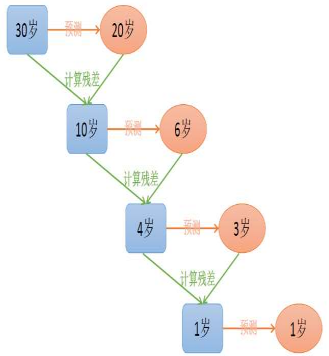
$$L(y,f(x))=(y-f(x))^2$$

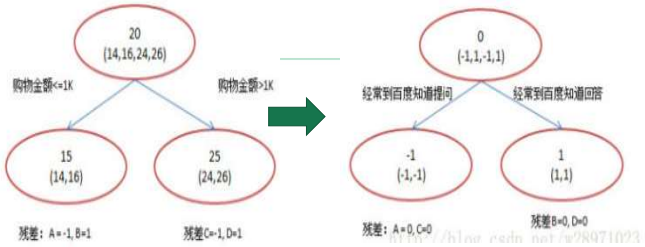
则上面损失函数变为：

$$L(y,f_{m-1}(x)+T(x;\Theta_m))=[y-f_{m-1}(x)-T(x;\Theta_m)]^2=[r-T(x;\Theta_m)]^2$$

残差

- GBDT 在模型训练的时候，是要求模型预测的样本损失尽可能的小。
- 残差的意思就是：
  - A的实际值 - A的预测值 = A的残差
- **GBDT每一次的计算是都为了减少上一次的残差，进而在残差减少（负梯度）的方向上建立一个新的模型**





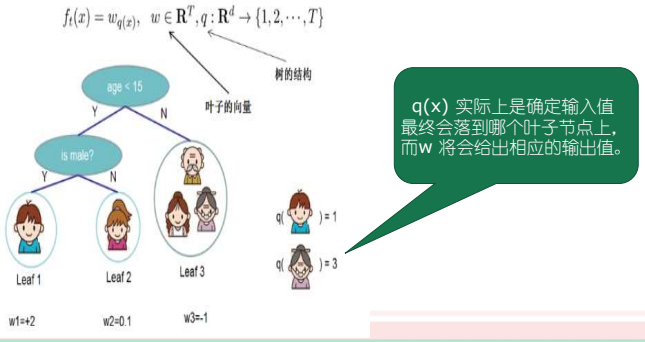
- 残差值都为0，相当于第二棵树的预测值和它们的实际值相等，则只需把第二棵树的结论累加到第一棵树上就能得到真实年龄了，即每个人都得到了真实的预测值。
- 预测年龄A = 15 – 1 = 14　预测年龄B = 15 + 1 = 16
- 预测年龄C = 25 – 1 = 24

GBDT优缺点

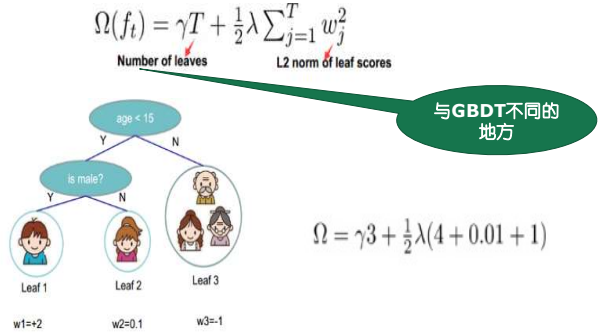
- GBDT主要的优点有：
  - 可以灵活处理各种类型的数据，包括连续值和离散值。
  - 在相对少的调参时间情况下，预测的准确率也可以比较高。这个是相对SVM来说的。
  - 使用一些健壮的损失函数，对异常值的鲁棒性非常强。比如 Huber 损失函数和Quantile损失函数。
- GBDT的主要缺点有：
  - 由于弱学习器之间存在依赖关系，难以并行训练数据。不过可以通过自采样的SGBT来达到部分并行。

Xgboost

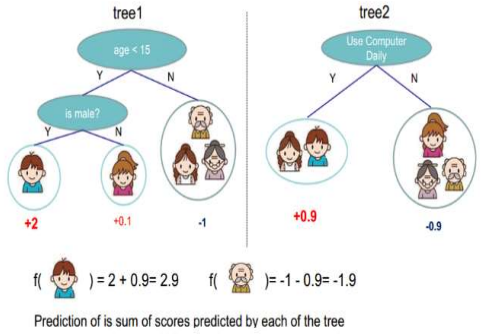
- 首先把树拆分成结构部分 q 和叶子节点输出值 w ，在这里 w 是一个向量，表示各叶子节点中的输出值。在这里就囊括了上面提到的两点，确定树结构q 和叶子结点的输出值 w 。



- 引入正则化项 $\Omega(f_t)$ 来控制树的复杂度，从而实现有效的控制模型的过拟合，



- **XGBoost中的Boosting Tree模型**



集成学习

- 个体与集成
- Boosting
  - Adaboost
- Bagging与随机森林
- 结合策略
  - 平均法
  - 投票法
  - 学习法
- 多样性
  - 误差-分歧分解
  - 多样性度量
  - 多样性扰动

Bagging与随机森林

- 个体学习器不存在强依赖关系
- 并行化生成
- 自助采样法

Bagging与随机森林 - Bagging算法

输入： 训练集 $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
基学习算法 $\mathfrak{L}$;
训练轮数 T.

过程：

1: **for** $t = 1, 2, \ldots, T$ **do**
2: $\quad h_t = \mathfrak{L}(D, \mathcal{D}_{bs})$
3: **end for**

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

Bagging与随机森林 – Bagging算法特点

- 时间复杂度低
  - 假定基学习器的计算复杂度为O(m)，采样与投票/平均过程的复杂度为O(s)，则bagging的复杂度大致为T(O(m)+O(s))
  - 由于O(s)很小且T是一个不大的常数
  - 因此训练一个bagging集成与直接使用基学习器的复杂度同阶
- 可使用包外估计

Bagging与随机森林 – 包外估计

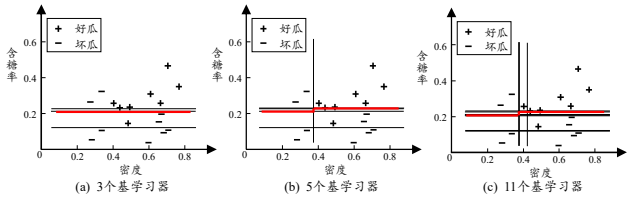
- $H^{oob}(x)$表示对样本x的包外预测，即仅考虑那些未使用样本x训练的基学习器在x上的预测

$$H^{oob}(\boldsymbol{x}) = \arg\max_{y \in \mathcal{Y}} \sum_{t=1}^{T} \mathbb{I}(h_t(\boldsymbol{x}) = y) \cdot \mathbb{I}(\boldsymbol{x} \notin D_t)$$

- Bagging泛化误差的包外估计为：

$$\epsilon^{oob} = \frac{1}{|D|} \sum_{(\boldsymbol{x},y) \in D} \mathbb{I}(H^{oob}(\boldsymbol{x}) \neq y)$$

Bagging与随机森林– Bagging实验

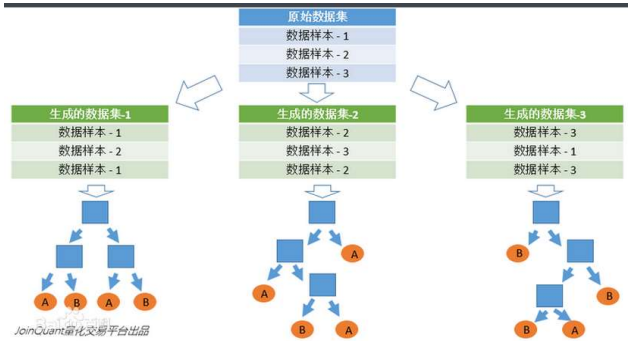


- 从偏差-方差的角度：降低方差，在不剪枝的决策树、神经网络等易受样本影响的学习器上效果更好

Bagging与随机森林–随机森林

- 随机森林(Random Forest，简称RF)是bagging的一个扩展变种
- 采样的随机性
- 属性选择的随机性

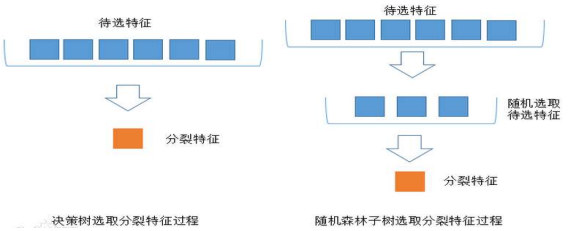
数据随机抽取



41

特征随机抽取

- 与数据集的随机选取类似，随机森林中的子树的每一个分裂过程并未用到所有的待选特征，而是从所有的待选特征中随机选取一定的特征，之后再在随机选取的特征中选取最优的特征。这样能够使得随机森林中的决策树都能够彼此不同，提升系统的多样性，从而提升分类性能。



42

- 每棵树的按照如下规则生成：
  - 如果训练集大小为N，对于每棵树而言，随机且有放回地从训练集中的抽取N个训练样本（这种采样方式称为bootstrap sample方法），作为该树的训练集；
  - 如果每个样本的特征维度为M，指定一个常数m<<M，随机地从M个特征中选取m个特征子集，每次树进行分裂时，从这m个特征中选择最优的；
  - 每棵树都尽最大程度的生长，并且没有剪枝过程。

43

- **随机森林分类效果（错误率）与两个因素有关：**
  - 森林中任意两棵树的相关性：相关性越大，错误率越大；
  - 森林中每棵树的分类能力：每棵树的分类能力越强，整个森林的错误率越低。

减小特征选择个数m，树的相关性和分类能力也会相应的降低；增大m，两者也会随之增大。所以关键问题是如何选择最优的m（或者是范围），这也是随机森林唯一的一个参数。

44

- from sklearn.ensemble import RandomForestClassifier
- #开始利用机器学习算法计算，括号里面的n_estimators就是森林中包含的树的数目啦
- clf = RandomForestClassifier(n_estimators=10)
- #训练的代码clf.fit(x_train, y_train)
- #
- 得到测试结果的代码prediction = clf.predict(x_test)
- # 看看预测对了没
- print(prediction == y_test)
- print('all done')
-
- 输出：
- [True]all done

45

Bagging与随机森林 – 随机森林算法

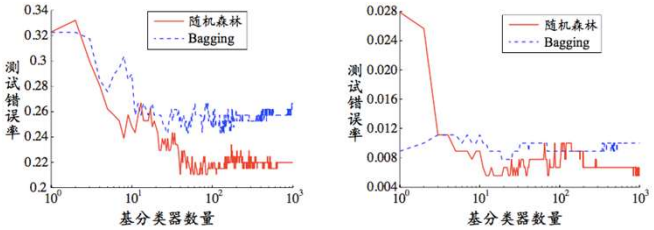
- 随机森林算法

Input: Data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
Feature subset size K.

Process:
1. $N \leftarrow$ create a tree node based on D;
2. **if** *all instances in the same class* **then return** N
3. $\mathcal{F} \leftarrow$ the set of features that can be split further;
4. **if** $\mathcal{F}$ *is empty* **then return** N
5. $\tilde{\mathcal{F}} \leftarrow$ select K features from $\mathcal{F}$ randomly;
6. $N.f \leftarrow$ the feature which has the best split point in $\tilde{\mathcal{F}}$;
7. $N.p \leftarrow$ the best split point on $N.f$;
8. $D_l \leftarrow$ subset of D with values on $N.f$ smaller than $N.p$;
9. $D_r \leftarrow$ subset of D with values on $N.f$ no smaller than $N.p$;
10. $N_l \leftarrow$ call the process with parameters (D_l, K);
11. $N_r \leftarrow$ call the process with parameters (D_r, K);
12. **return** N

Output: A random decision tree

Bagging与随机森林 – 随机森林实验



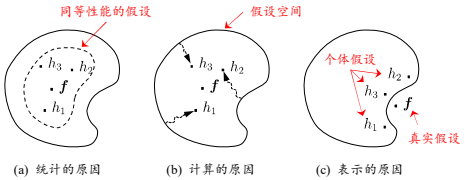
(a) glass 数据集　　(b) auto-mpg 数据集

集成学习

- 个体与集成
- Boosting
  - Adaboost
- Bagging与随机森林
- 结合策略
  - 平均法
  - 投票法
  - 学习法
- 多样性
  - 误差-分歧分解
  - 多样性度量
  - 多样性扰动

结合策略

- 学习器的组合可以从三个方面带来好处



(a) 统计的原因　(b) 计算的原因　(c) 表示的原因

结合策略 – 平均法

- 简单平均法

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

- 加权平均法

$$H(\boldsymbol{x}) = \sum_{i=1}^{T} w_i h_i(\boldsymbol{x}), \qquad w_i \geq 0 \ \text{ and } \ \sum_{i=1}^{T} w_i = 1.$$

结合策略 – 平均法

- 简单平均法是加权平均法的特例
- 加权平均法在二十世纪五十年代被广泛使用
- 集成学习中的各种结合方法都可以看成是加权平均法的变种或特例
- 加权平均法可认为是集成学习研究的基本出发点
- 加权平均法未必一定优于简单平均法

结合策略 – 投票法

- 绝对多数投票法（majority voting）

$$H(\boldsymbol{x}) = \begin{cases} c_j & \text{if } \sum_{i=1}^{T} h_i^j(\boldsymbol{x}) > \frac{1}{2}\sum_{k=1}^{l}\sum_{i=1}^{T} h_i^k(\boldsymbol{x}) \\ \text{rejection} & \text{otherwise}. \end{cases}$$

- 相对多数投票法（plurality voting）

$$H(\boldsymbol{x}) = c_{\arg\max_j \sum_{i=1}^{T} h_i^j(\boldsymbol{x})}$$

- 加权投票法（weighted voting）

$$H(\boldsymbol{x}) = c_{\arg\max_j \sum_{i=1}^{T} w_i h_i^j(\boldsymbol{x})}$$

结合策略 – 学习法

- Stacking是学习法的典型代表

Input: Data set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), \ldots, (\boldsymbol{x}_m, y_m)\}$;
First-level learning algorithms $\mathfrak{L}_1, \ldots, \mathfrak{L}_T$;
Second-level learning algorithm $\mathfrak{L}$.

Process:

```
for t = 1,...,T:    % Train a first-level learner by applying the
  h_t = L_t(D);     % first-level learning algorithm L_t
end
D' = ∅;             % Generate a new data set
for i = 1,...,m:
  for t = 1,...,T:
    z_it = h_t(x_i);
  end
  D' = D' ∪ ((z_i1,...,z_iT), y_i);
end
h' = L(D');         % Train the second-level learner h' by applying
                    % the second-level learning algorithm L to the
                    % new data set D'.
```

Output: $H(\boldsymbol{x}) = h'(h_1(\boldsymbol{x}), \ldots, h_T(\boldsymbol{x}))$

多响应线性回归(MLR)作为次级学习器的学习算法 效果较好

- 贝叶斯模型平均(BMA)