

集成学习

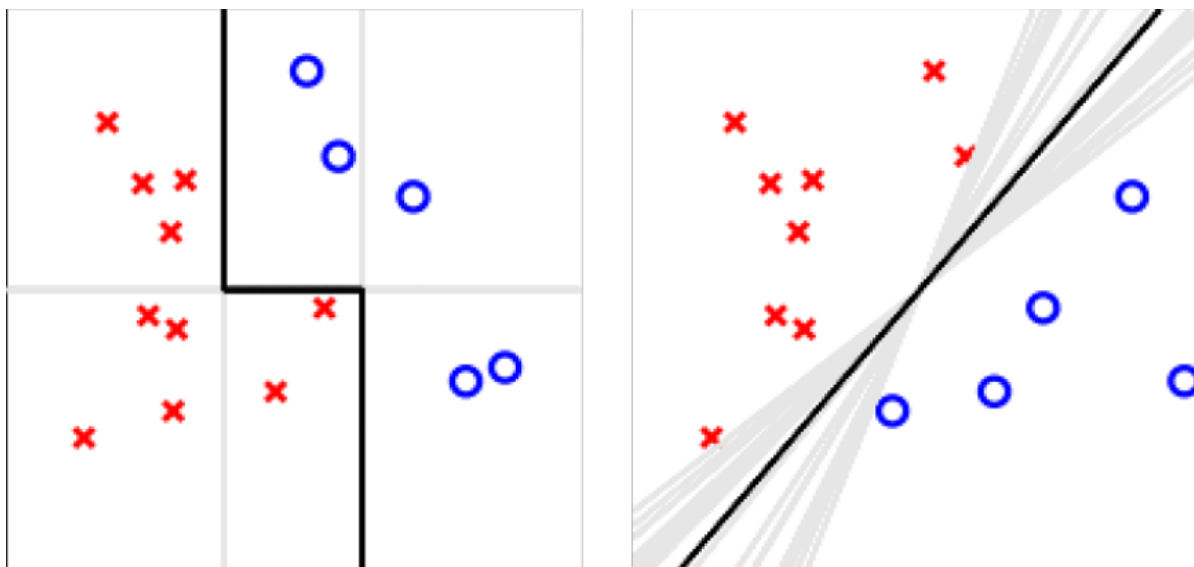
Ensemble Learning

集成学习基础

- 集成学习：构建并结合多个个体学习器来完成学习任务。
 - 基学习器：集成的学习器都是同种类型的。
 - 组件学习器：集成的学习器包含不同的类型。
- 强可学习：在概率近似正确（Probably approximately correct, PAC）学习的框架汇总，一个概念，如果存在一个多项式的学习算法能够学习它，并且正确率很高，那么就称这个概念是强可学习的
- 弱可学习：一个概念，如果存在一个多项式的学习算法能够学习它，学习的正确率仅比随机猜测略好，那么这个概念是弱可学习的
- 定理:在PAC学习的框架下，一个概念是强可学习的充分必要条件是这个概念是弱可学习的。
- 集成学习通常研究弱学习器！

- 集成学习的好处：

- 从统计的角度：学习任务的假设空间很大，可能有多假设在训练集上达到同等性能，单个学习器可能因误选导致泛化性能不佳。
- 从计算的角度：学习算法往往会陷入局部极小，多次运行之后进行结合，可降低陷入糟糕局部极小点的风险。
- 从表示的角度：某些学习任务的真实假设可能不在当前学习算法的假设空间中，通过结合多个学习器，有可能学得更好的近似。



假设集成包含 T 个学习器 h_1, h_2, \dots, h_T , 集成结果是 H . $h_t(\mathbf{x})_c$ 是学习器 $h_t(\mathbf{x})$ 在类别 c 上的输出.

学习器的结合策略:

- 均匀组合: $H(x) = \frac{1}{T} \sum_{t=1}^T h_t(\mathbf{x})$
- 加权平均: $H(\mathbf{x}) = \sum_{t=1}^T \alpha_t h_t(\mathbf{x})$, 其中 $\forall t, \alpha_t \geq 0$ 以及 $\sum_{t=1}^T \alpha_t = 1$

- 绝对多数投票:

$$H(\mathbf{x}) = \begin{cases} c & \text{若 } \sum_{t=1}^T h_t(\mathbf{x})_c > \frac{1}{2} \sum_{c=1}^C \sum_{t=1}^T h_t(\mathbf{x})_c \\ \text{拒绝预测} & \text{否则} \end{cases}$$

- 相对多数投票: $H(\mathbf{x}) = \arg \max_c \sum_{t=1}^T h_t(\mathbf{x})_c$
- 加权投票: $H(\mathbf{x}) = \arg \max_c \sum_{t=1}^T \alpha_t h_t(\mathbf{x})_c$, 其中 $\forall t, \alpha_t \geq 0$ 以及 $\sum_{t=1}^T \alpha_t = 1$.

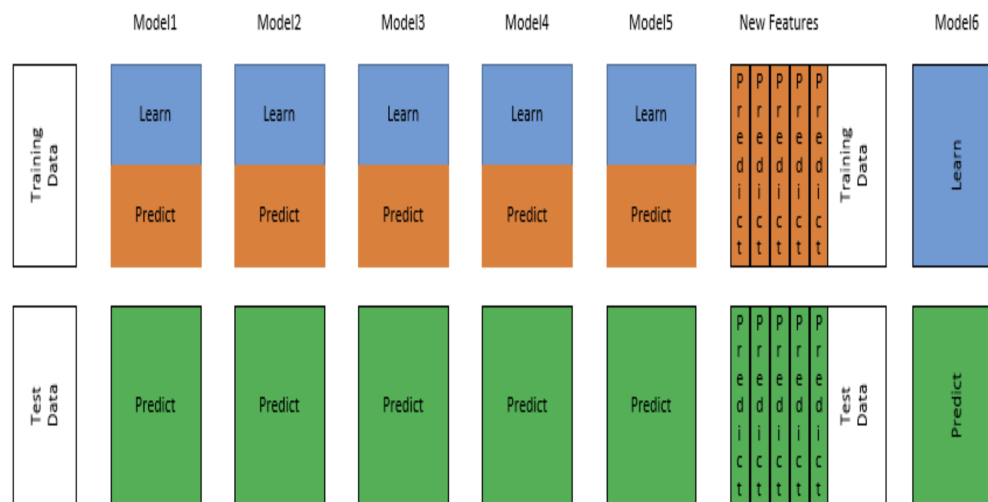
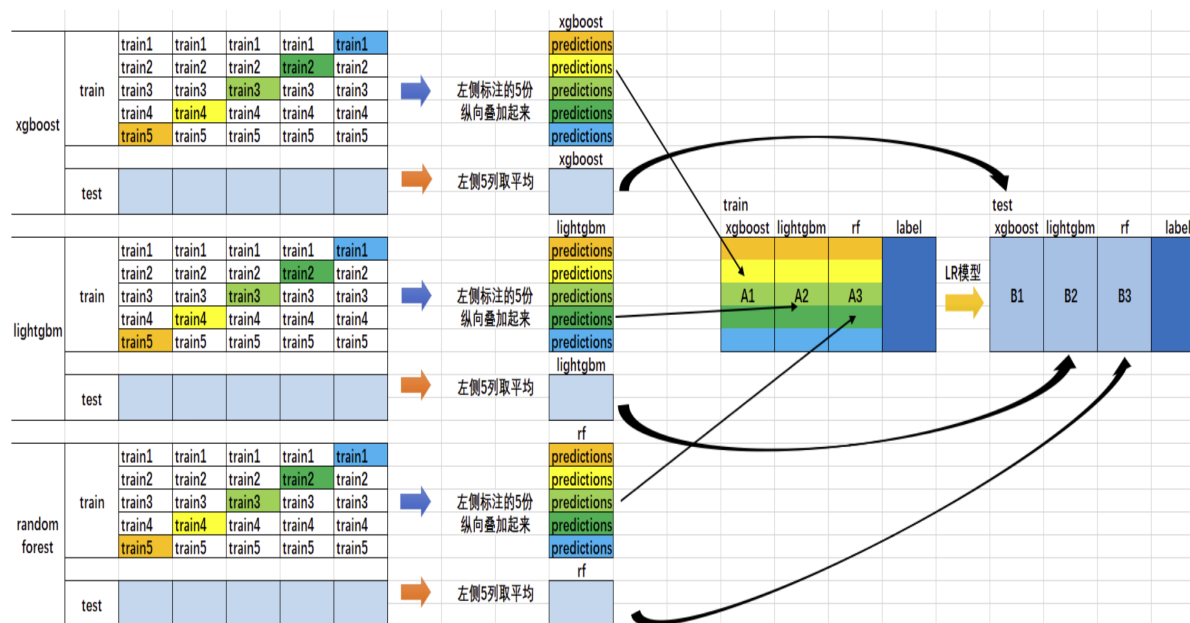
Stacking: 通过另一个学习器，称为次级学习器或元学习器（meta-learner），对个体学习器（称为初级学习器）的结果进行结合。

- 利用训练集 $D = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$ 分别训练初级学习器 h_1, h_2, \dots, h_T .
- 构造次级训练集 $\tilde{D} = \{(\tilde{\mathbf{x}}_i, y_i)\}_{i=1}^m$, 其中 $\tilde{x}_i = (h(\mathbf{x}_i)_1, h(\mathbf{x}_i)_2, \dots, h(\mathbf{x}_i)_T) \in \mathbb{R}^T$
- 利用次级训练集 \tilde{D} 训练次级学习器 H

Note: 1. 一般使用交叉验证或者留一法，在每一折，用训练初级学习器未使用的样本来产生次级学习器的训练样本。

2. 初级学习器的输出类概率作为次级学习器的输入属性。

3. 用多响应回归（Multi-response linear regression, MLR）作为次级学习算法。



多样性

- 定理 假设基分类器的错误率都为 e 且相互独立, 对二分类问题, 简单平均集成的错误率随个体分类器数目 T 指数下降, 即:

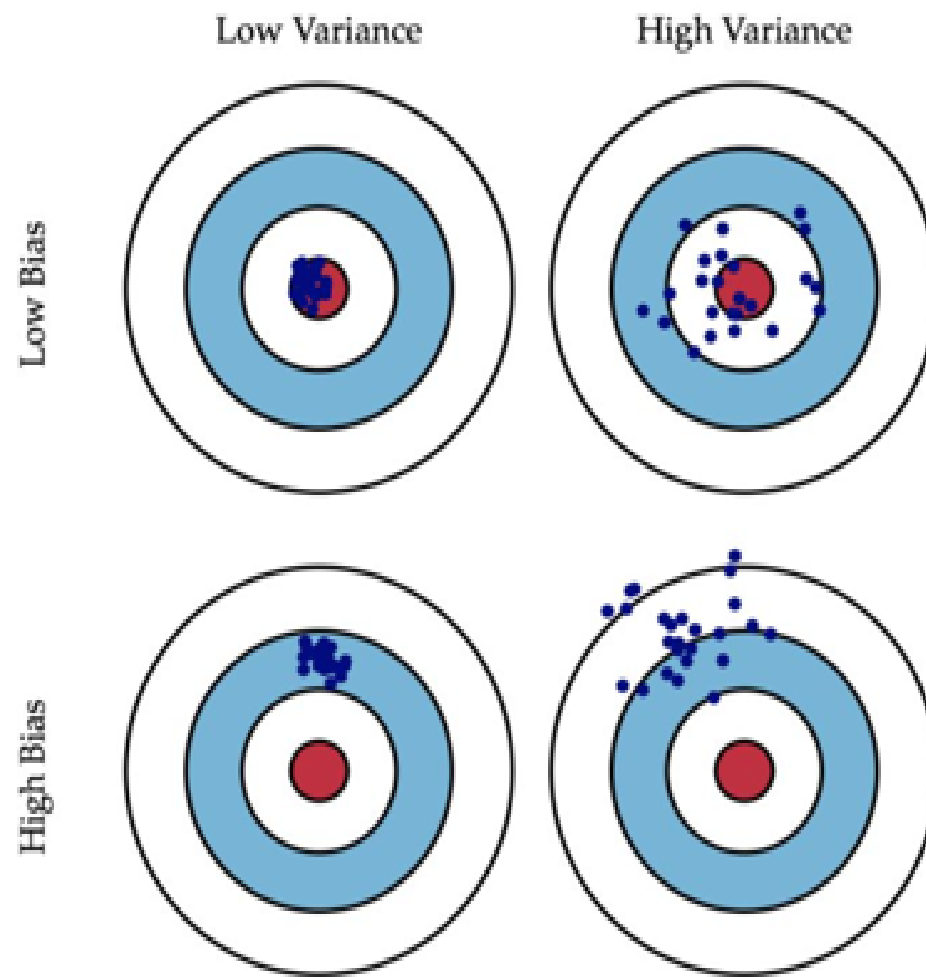
$$E[\mathbb{I}(H(\mathbf{x}) \neq y)] \leq \exp(-\frac{1}{2}T(1 - 2e^2))$$

该定理假设基学习器误差相互独立, 但在现实任务中, 基学习器是为解决同一个问题训练出来的, 它们显然不独立. 通过在学习过程引入随机性, 可以获得依赖程度没有那么高的学习器.

- 偏差-方差分解(Bias-Variance Decomposition): 假设使用加权平均法完成回归任务, 则:

$$(H(\mathbf{x}) - y)^2 = \sum_{t=1}^T \alpha_t (h_t(\mathbf{x}) - y)^2 - \sum_{t=1}^T \alpha_t (h_t(\mathbf{x}) - H(\mathbf{x}))^2$$

可以看出个体学习器准确性越高、多样性越大, 则集成越好.



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

(a) 集成提升性能 (b) 集成不起作用 (c) 集成起负作用

多样性增强方法

- 数据样本扰动：从给定数据集 D 中采样产生不同的数据子集 $\sim D$, 再利用不同的数据子集训练不同的个体学习器. 例如Bagging使用自助采样, AdaBoost 使用序列采样.
- 输入属性扰动：从初始的高维属性空间投影产生低维属性空间, 不同的子空间提供了观察数据的不同视角, 再利用不同的子空间训练不同的个体学习器.
- 算法参数扰动：例如改变神经网络的隐层神经元数、初始连接权值等.
- 输出表示扰动：翻转法(flipping output)随机改变一些训练样本的标记.

Bagging: 从训练集 D 采样相互有交叠的采样子集, 并用不同的子集训练不同的个体学习器.

- 自助法采样(bootstrapsampling): 从 D 中有放回地采样 m 次得到数据集 \tilde{D}
- 引理: D 中的某个样本在自助法采样中不被采到的概率约是36.8%. 也就是说, 在 D 中大约有36.8%的样本未出现在采样数据集 \tilde{D} 中.
- 包外验证(out-of-bag estimation): 给定数据集 D , 我们可以把自助法采样得到的数据集 \tilde{D} 作为训练集, 用 $D - \tilde{D}$ 作为验证集.

Bagging 的具体过程包括如下三步.

1. 从训练集 D 中由自助法采样得到 T 个采样集 $\tilde{D}_1, \tilde{D}_2, \dots, \tilde{D}_T$.
2. 基于自助采样集 \tilde{D}_t 训练基学习器 h_t .
3. 使用相对多数投票(分类任务) 或简单平均(回归任务)得到 H .

在利用Bagging 进行包外估计时, 我们考虑那些未使用 \mathbf{x} 训练的基学习器在 \mathbf{x} 上的预测:

$$H_{oob}(\mathbf{x}): = \arg \max_c \sum_{t=1}^T \mathbb{I}(\mathbf{x} \notin \tilde{D}_t) \mathbb{I}(h_t(\mathbf{x}) = c)$$

那么Bagging的包外误差估计为

$$E_{oob} := \frac{1}{m} \sum_{i=1}^m I(H_{oob}(\mathbf{x}_i) \neq y_i)$$

Boosting: 先从初始训练集 D 训练出一个基学习器, 再根据基学习器的表现对训练样本分布进行调整. 使得先前做错的训练样本在后续受到更多关注, 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.

AdaBoost(adaptive boosting): 通过给样本加权实现来对训练样本的分布进行调整.

考虑数据集 $D = (x_1, y_1), (x_2, y_2), (x_3, y_3), (x_4, y_4)$

经过bootstrap可得 $D_t = (x_1, y_1), (x_1, y_1), (x_2, y_2), (x_4, y_4)$

损失函数为:
$$E(h) = \frac{1}{4} \sum_{n=1}^4 u_n^t \cdot \mathbb{I}[y_n \neq h(x_n)]$$

u_n^t 表示数据出现的次数, $u_1^t = 2, u_2^t = u_4^t = 1, u_3^t = 0$.

$$g_t = \arg \min_h \sum_{n=1}^N u_n^t \cdot \llbracket y_n \neq h(x_n) \rrbracket$$

$$g_{t+1} = \arg \min_h \sum_{n=1}^N u_n^{t+1} \cdot \llbracket y_n \neq h(x_n) \rrbracket$$

g_t 是由 u_n^t 得到的, g_{t+1} 是由 u_n^{t+1} 得到的, 一般来讲我们希望所有的 g_t 之间具有较大的差异性。

Idea: 在 g_t 的预测下, u_n^{t+1} 中错误率为0.5时, g_t 和 g_{t+1} 有较大的差异性。即:

$$\frac{\sum_{n=1}^N u_n^{t+1} \cdot \llbracket y_n \neq g_t(x_n) \rrbracket}{\sum_{n=1}^N u_n^{t+1}} = 1/2$$

$$\text{令 } A_{t+1} = \sum_{n=1}^N u_n^{t+1} \cdot \llbracket y_n \neq g_t(x_n) \rrbracket$$

$$B_{t+1} = \sum_{n=1}^N u_n^{t+1} \cdot \llbracket y_n = g_t(x_n) \rrbracket$$

$$\text{希望: } A_{t+1} = B_{t+1}$$

方法: 令

$$\epsilon_t = \frac{\sum_{n=1}^N u_n^t \cdot \llbracket y_n \neq g_t(x_n) \rrbracket}{\sum_{n=1}^N u_n^t}$$

$$\lambda_t = \sqrt{\frac{1 - \epsilon_t}{\epsilon_t}}$$

则对于在 g_t 下分类正确的 x_n , $u_n^{t+1} = \frac{u_n^t}{\lambda_t}$

对于在 g_t 下分类不正确的 x_n , $u_n^{t+1} = u_n^t \cdot \lambda_t$

$\lambda_t \geq 1$ 当且仅当 $\epsilon_t \leq \frac{1}{2}$

Boosting (提升法)

$u^1 = [1/N, 1/N, \dots, 1/N]$

For $t = 1, 2, \dots, T$

(1) 获得 $g_t = \mathcal{A}(D, u^t)$

(2) 根据下面方法更新 u_t 到 u_{t+1} ,

If $\llbracket y_n \neq g_t(x_n) \rrbracket$, 则 $u_{t+1} = u_t \cdot \lambda_t$,

If $\llbracket y_n = g_t(x_n) \rrbracket$, 则 $u_{t+1} = \frac{u_t}{\lambda_t}$,

(3) 计算 $a_t = \ln \lambda_t$

Return $G(x) = \text{sign}(\sum_{t=1}^T a_t g_t(x))$