第六讲 集成学习与随机森林

授课老师：郭迟 教授

guochi@whu.edu.cn

武汉大学测绘学院

2021.11

集成学习与随机森林

1 个体与集成

2 模型组合策略

3 提升（Boosting）

4 装袋（Bagging）

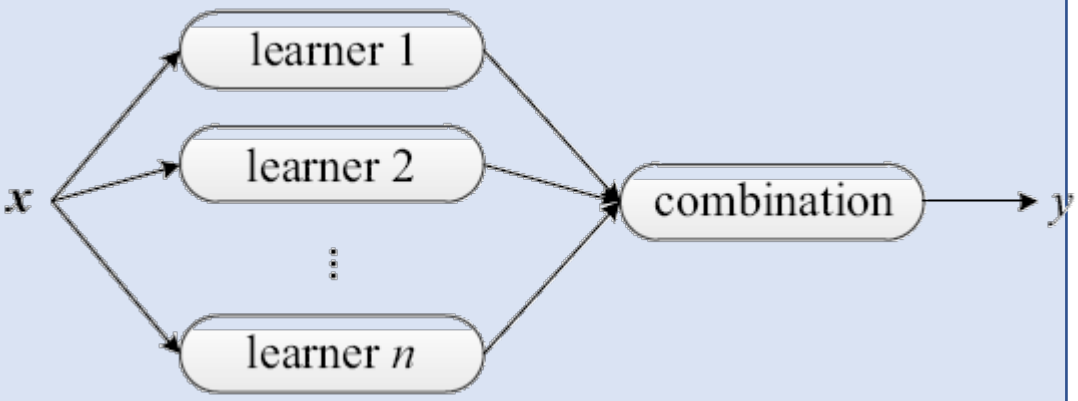
5 随机森林

第六讲 集成学习与随机森林

个体与集成

- 集成学习（Ensemble Learning）通常是指利用多个同种类的学习器来对一个问题进行学习。这里的同种类是指所使用的学习器属于同一个类型，生成同质集成（Homogeneous Ensemble），如所有的学习器都是决策树、支持向量机、神经网络等
- 广义地说，只要是使用多个学习器来解决问题，就可以认为符合集成学习的广义形式。这种集成称为异质集成 (Heterogeneous Ensemble)

同质集成学习通用框架：一个集成由多个基学习器(base learner)构成，基学习器由基学习算法在训练数据上训练获得，它们可以是决策树、神经网络或其他算法



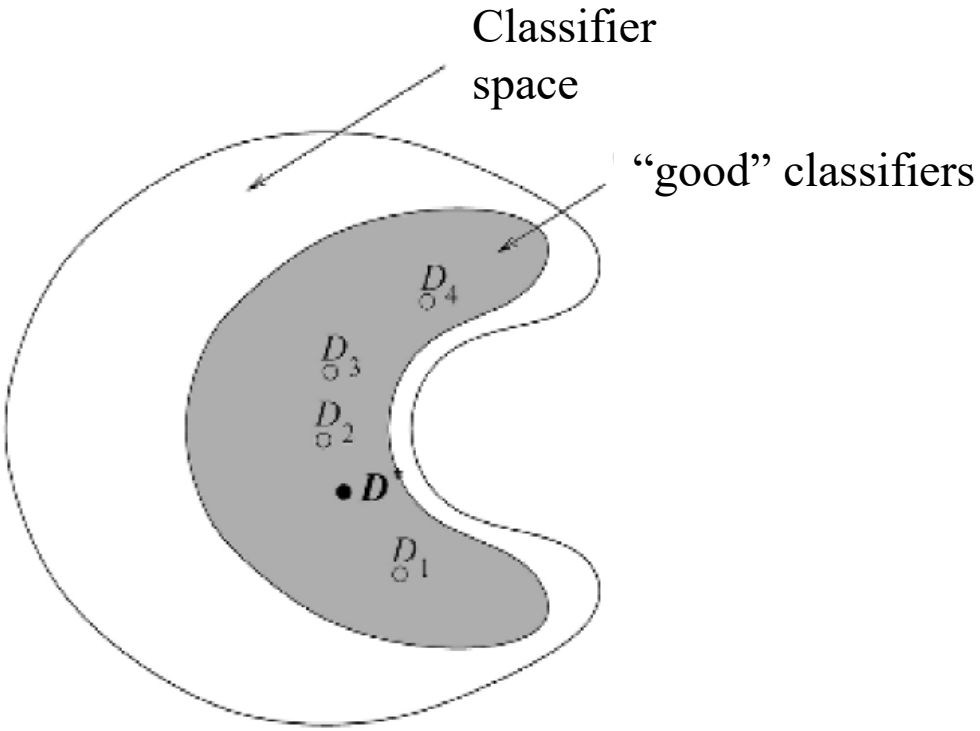
第六讲 集成学习与随机森林

个体与集成

- 同质集成 (Homogeneous Ensemble)：集成中只包含同种类型的个体学习器，个体学习器也被称为“基学习器”，相应的学习算法称为“基学习算法”
- 异质集成 (Heterogeneous Ensemble)：个体学习器由不同的学习算法生成，因此不存在“基学习算法”。相应的，个体学习器一般不称为“基学习器”

个体与集成

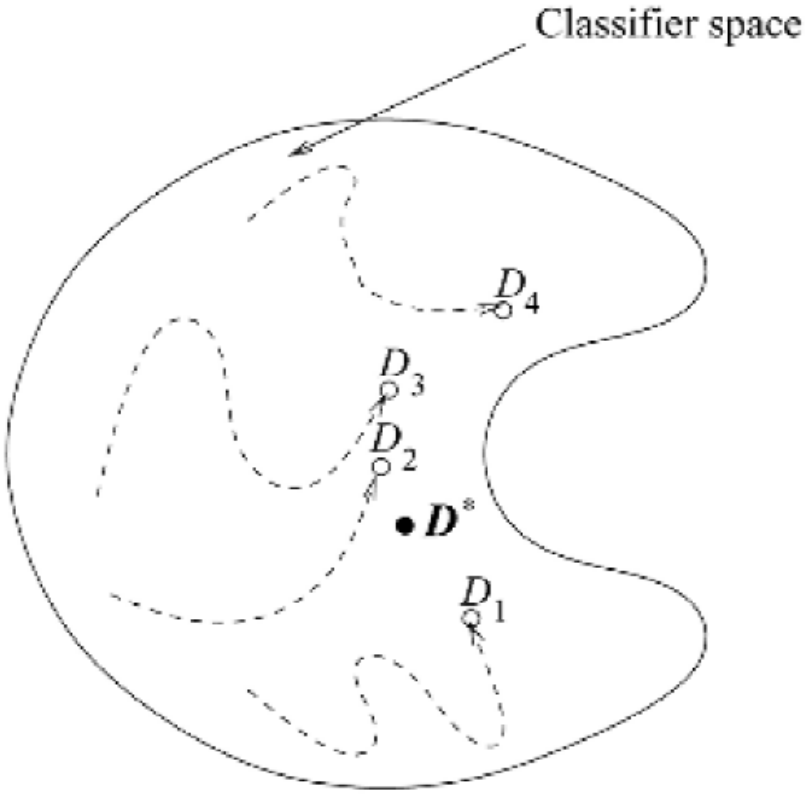
- 从统计分析方面看，虽然在同一训练集上可以学习训练性能最好的分类器，但是这些分类器在未来数据中的性能则是很难保证的。为了降低这种单一选择的风险，结合多学习器就是一种合理的选择



第六讲 集成学习与随机森林

个体与集成

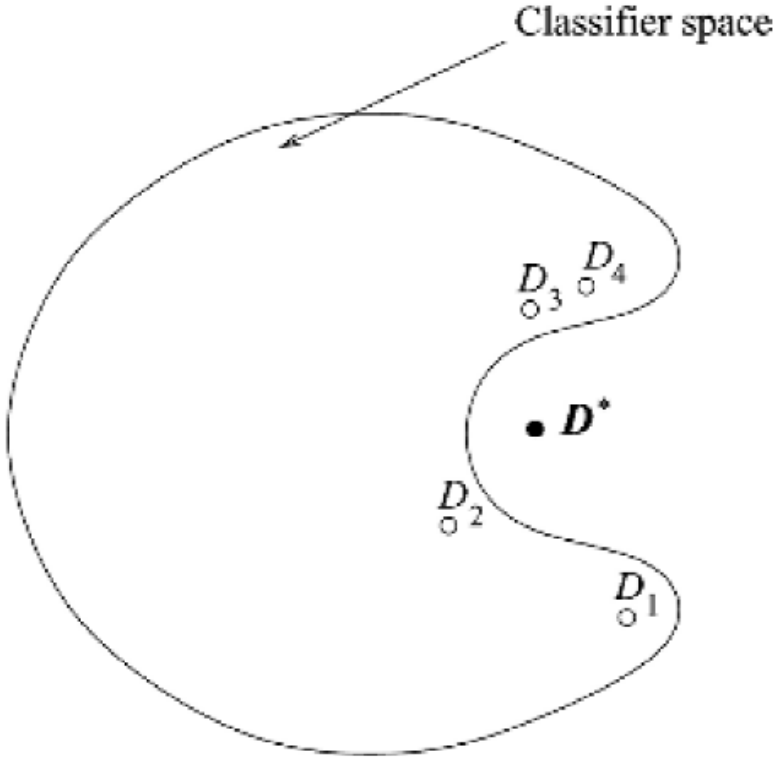
- 从算法求解方面看，很多学习算法的使用梯度下降法，往往会陷入到局部极小点，其泛化性能可能很低。通过分类器集成，可以降低陷入糟糕的局部极小点的风险



第六讲 集成学习与随机森林

个体与集成

- 从模型表达方面看，某些学习任务真实的最佳分类器可能不在当前学习算法所考虑的模型空间中，此时使用单一学习器则肯定无效。通过集成多个学习器，可以让假设空间所有扩大，比如分类器的线性组合，就有可能学出更接近真实的分类器



第六讲 集成学习与随机森林

个体与集成

- 如果你随机向几千个人询问一个复杂问题，然后汇总他们的回答。在许多情况下，你会发现， 这个汇总的回答比专家的回答还要好。这被称为群体智慧 (Crowd Intelligence)
- 同样，如果你聚合一组预测器 (学习器/估计器：分类器或回归器) 的预测结果，得到的预测结果会比最好的单个预测器要好

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

(a) 集成提升性能

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

(b) 集成不起作用

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

(c) 集成起负作用

注意：并不是集成就会得到好结果 ⟹ **“好而不同”是关键！**

集成学习与随机森林

1 个体与集成

2 模型组合策略

3 提升（Boosting）

4 装袋（Bagging）

5 随机森林

模型组合策略

1.平均法（averaging）

- 对数值型输出$h_i(x) \in R$，最常见的结合策略是使用平均法
  - 简单平均法（simple averaging）

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

  - 加权平均法（weighted averaging）

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

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

显然，简单平均法是加权平均法的一个特例（令$w_i = 1/T$）

2. 投票法（voting）

一个计算：假设创建了一个包含1000 个分类器的集成，每个分类器都只有51% 的几率是正确的 (几乎没比随机猜测强多少)。如果以简单多数投票的类别作为预测结果，你可以期待的准确率高达75%

$$\sum_{k=501}^{1000} C_{1000}^{k} p^{k} (1-p)^{1000-k} \approx 0.75$$

- 当然，能完全达到这个数字的**前提**是：**所有的分类器**都是**完全独立**的，彼此的错误毫不相关。显然这是不可能的，因为它们都是在相同的数据集下进行的训练。它们很可能会犯相同的错误，所以也会有很多次大多数投给了错误的类别， 导致集成的准确率有所降低

2. 投票法（voting）

- 对分类任务来说，学习器h_i从类标记集合$\{c_1, c_2, \cdots, c_N\}$中预测出一个标记，最常见的集成策略是投票法。为便于讨论，我们将h_i在样本x上的预测输出表示为一个N维向量$\{h_i^1(x), h_i^2(x), \cdots, h_i^N(x)\}$，其中$h_i^i(x)$是$h_i$在类别标记$c_j$上的输出。
  - 绝对多数投票法（majority voting）

$$H(x) = \begin{cases} c_i, & if \sum_{i=1}^{T} h_i^j(x) > 0.5 \sum_{k=1}^{N} \sum_{i=1}^{T} h_i^k(x) \\ reject, & otherwise \end{cases}$$

即若某标记得票过半数，则预测为该标记；否则拒绝预测

模型组合策略

2. 投票法（voting）

- 相对多数投票法（plurality voting）

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

即预测为票最多的标记，若同时存在多个票最多标记，则随机选取一个

- 加权投票法（ weighted voting）

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

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

现实任务中可能产生，不同类型个体学习器可能产生不同类型的h_i^j值：

- 类标记：$h_i^j \in \{0,1\}$，输出0或1，亦被称为“硬投票”。
- 类概率：$h_i^j \in [0,1]$，输出一个后验概率值，亦被称为“软投票”。

不同类型的h_i^j值不能混用！

模型组合策略

3.堆叠法（Stacking）

- Stacking首先从初始数据中训练出初级学习器，然后“生成”一个新数据集用于训练次级学习器。在这个新数据集中，初级学习器的输出被当作样例输入特征，而初始样本的标记仍被当作样例标记
- Stacking不是把模型的结果组织起来，而把模型组织起来

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

过程：

```
for t = 1,2,...,T do
    h_t = L_t(D);
end for
D' = ∅;
for i = 1,2,...,m do
    for t = 1,2,...,T do
        z_it = h_t(x_i);
    end for
    D' = D' ∪ ((z_i1, z_i2, ..., z_iT), y_i);
end for
h' = L(D');
```

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

使用初级学习算法ζ_t产生初级学习器h_t

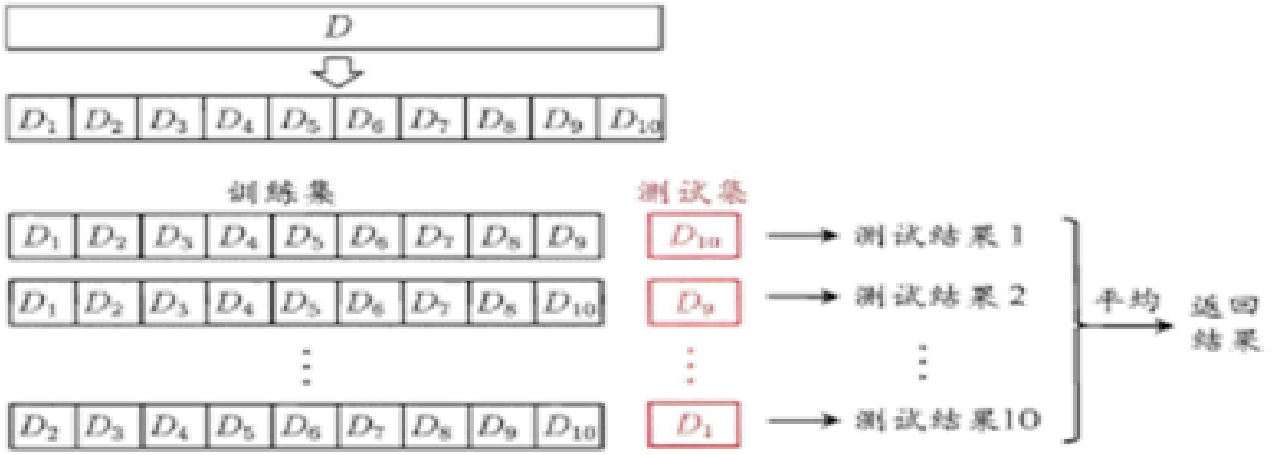
生成次级训练集

使用次级学习算法ζ产生次级学习器h'

模型组合策略

3.堆叠法（Stacking）

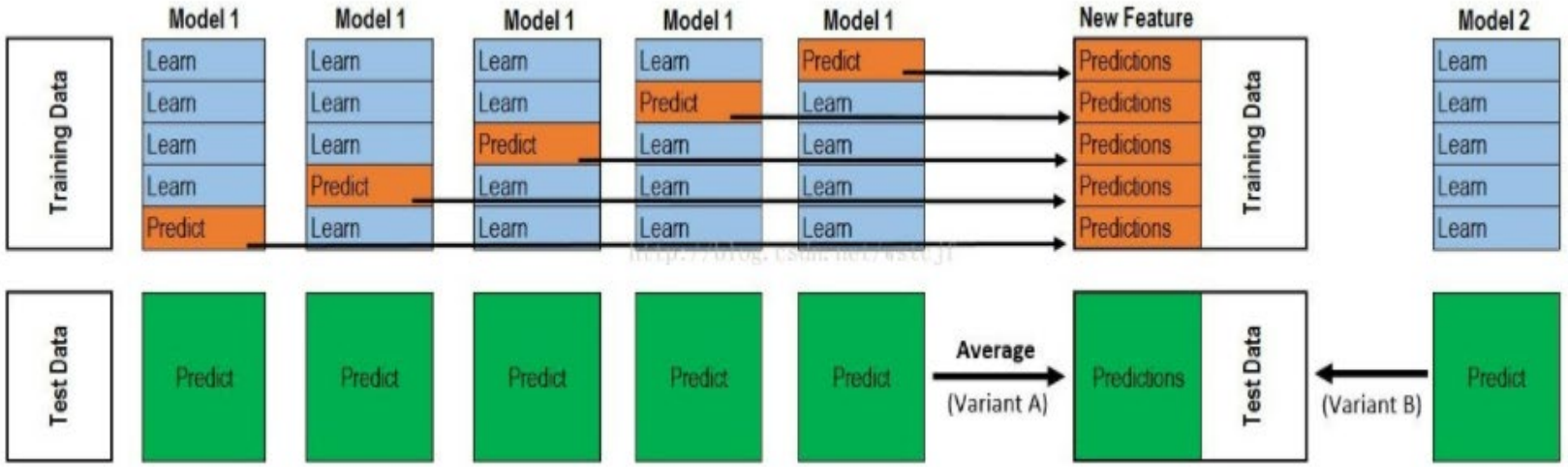
复习：交叉验证



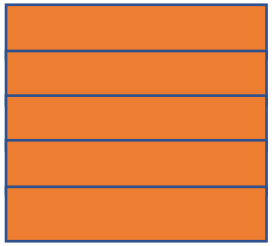
- 将数据集D划分为k个大小相同的互斥子集，满足$D=D_1 \cup D_2 \cup ... \cup D_k$，$D_i \cap D_j = \emptyset$（$i \neq j$），保持数据分布的一致性，采用分层抽样的方法获得这些子集。每次用$k-1$个子集的并集作为训练集，余下的那个子集作为测试集。这样就有k种训练集/测试集划分的情况，从而可进行k次训练和测试，最终返回k次测试结果的均值

模型组合策略

3.堆叠法（Stacking）



- 上半部分用一个初级学习器在训练集上进行5折交叉验证，最终输出5个验证集的预测，把它们按行拼接起来。下半部分用训练好的模型对测试集进行预测，最终输出5个预测值，对它们求和取平均

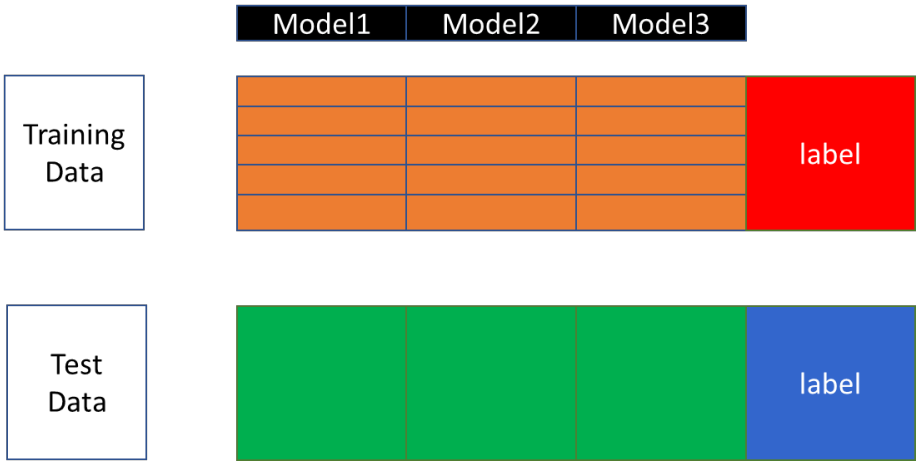


称其为New Feature，一个初级学习器对应一个New Feature

称其为Predictions，一个初级学习器对应一个Predictions

模型组合策略

3.堆叠法（Stacking）



- 假设有三个初级学习器，那么经过上面的操作后，我们得到每个学习器对应的New Feature（橘色框）和求和平均后的Predictions（绿色框）。将三个初级学习器在训练集上的预测值作为3个“新特征”A1,A2,A3，使用LR模型进行训练
- 使用训练好的LR模型，在三个基模型之前在测试集上的预测值所构建的三个"特征"的值(B1,B2,B3)上，进行预测

集成学习与随机森林

1 个体与集成

2 模型组合策略

3 提升（Boosting）

4 装袋（Bagging）

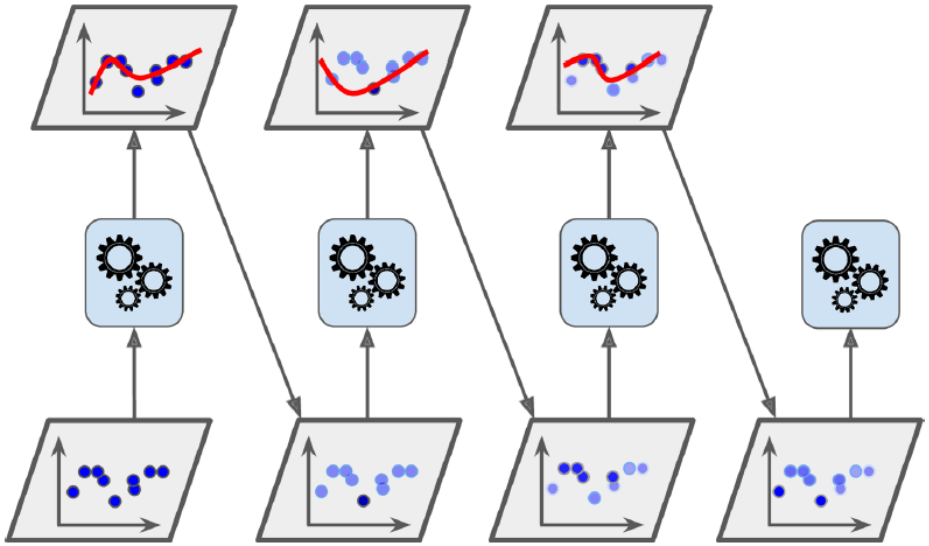
5 随机森林

第六讲 集成学习与随机森林

提升

- Boosting是一族可将弱学习器提升为强学习器的算法。这类算法工作机制如下：现先从初始训练集训练出一个基学习器，再根据学习器的表现对训练样本分布进行调整，使得先前基学习器做错的训练样本在后续获得更多关注，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，直到基学习器的数目达到预先指定值T，最终将在T个基学习器进行加权求和

教材P287



AdaBoost中的循环训练，每轮训练都对样例权重进行更新

第六讲 集成学习与随机森林

提升

- Boosting算法是串行式集成学习方法。它的基学习器之间存在强依赖关系
- Boosting算法要求基学习器能对特定的数据分布进行学习，可通过“重采样法”和“重赋权法”来实施。Boosting算法在训练的每一轮都要检查当前生成的基学习器是否满足基本条件 ，若不满足则抛弃当前基学习器，学习过程停止
- 代表方法为自适应提升法（Adaptive Boosting，AdaBoost）

- **重加权方法** (re-weighting)。在训练过程的每一轮中，根据样本分布为每个训练样本重新赋予一个权重。
- **重采样方法** (re-sampling)。对无法接受带权样本的基学习算法，可通过在每一轮学习中，根据样本分布对训练集重新采样，再用重采样获得的样本集对基学习器进行训练

提升

AdaBoost

《统计机器学习》P156

输入：训练集$T=\{(x_1,y_1),(x_2,y_2),\cdots,(x_N,y_N)\}$；弱学习算法；

输出：最终分类器$G(x)$

（1）初始化训练数据的权值分布

$$D_1=(w_{11},\ldots,w_{1i},\ldots,w_{1N}),\quad w_{1\mathrm{i}}=\frac{1}{N},\quad i=1,2,\cdots,N$$

（2）对$\mathrm{m}=1,2,\cdots,M$

（a）使用具有权值分布D_{m}的训练数据集学习，得到基本分类器

$$G_m(x):\chi\rightarrow\{-1,+1\}$$

（b）计算$G_m(x)$在训练数据集上的分类误差率

$$e_m=\sum_{i=1}^{N}P(G_m(x_i)\neq y_i)=\sum_{i=1}^{N}w_{mi}I(G_m(x_i)\neq y_i)$$

（c）计算$G_m(x)$的系数

提升

AdaBoost

$$\alpha_m = \frac{1}{2}\log\frac{1-e_m}{e_m}$$

这里的对数是自然对数。

（d）更新训练集的权值分布

$$D_{\mathrm{m+1}} = (w_{m+1,1}, \ldots, w_{m+1,i}, \ldots, w_{\mathrm{m+1},N})$$

$$w_{m+1,i} = \frac{w_{mi}}{Z_M}\exp(-\alpha_m y_i G_m(x_i)), \quad i = 1,2,\cdots,N$$

这里，Z_M是规范化因子

$$Z_M = \sum_{i=1}^{N} w_{mi}\exp(-\alpha_m y_i G_m(x_i))$$

它使D_{m+1}成为一个概率分布。

提升

AdaBoost

（3）构建基本分类器的线性组合

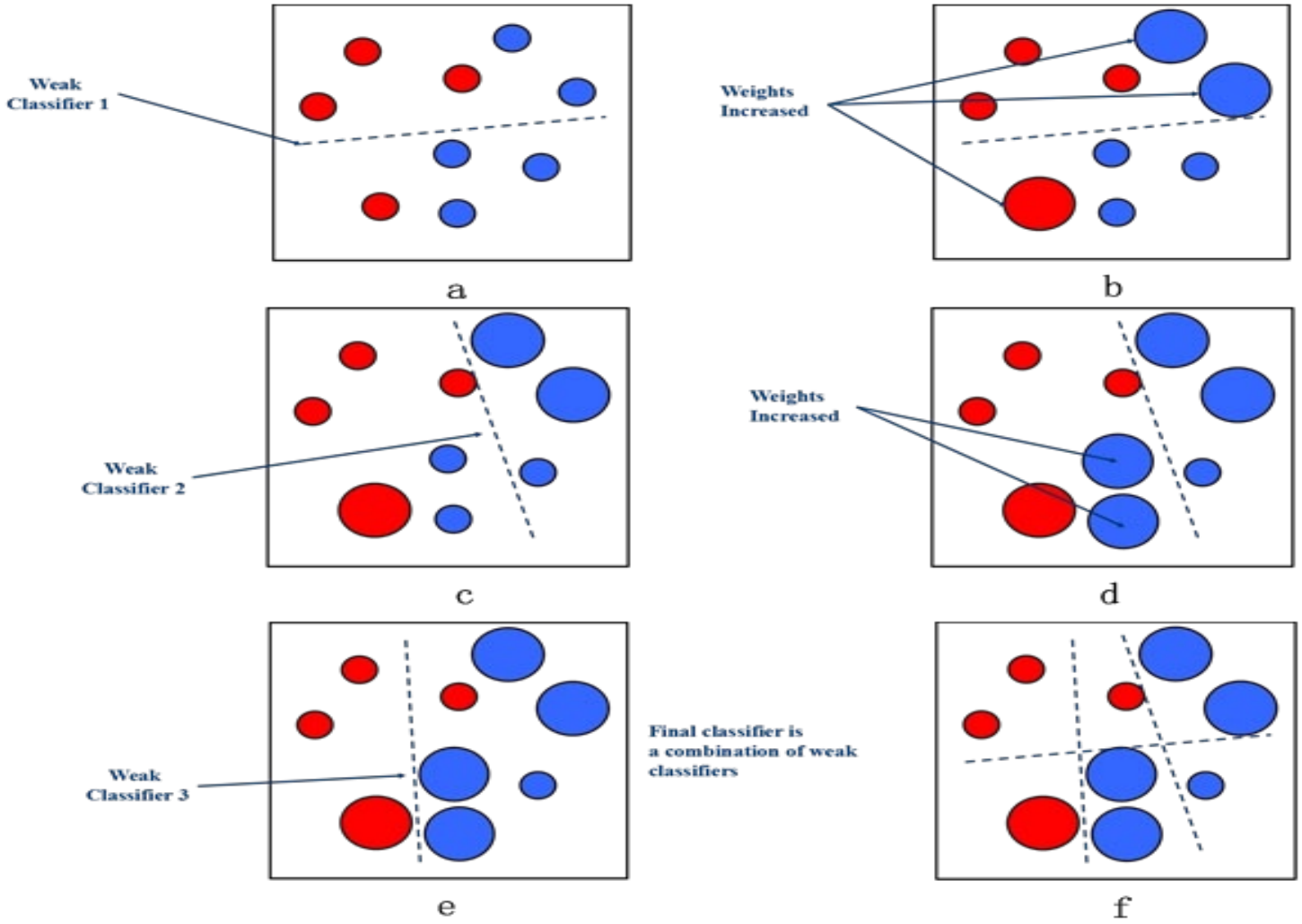
$$f(x) = \sum_{m=1}^{M} \alpha_m G_m(x)$$

得到最终分类器

$$G(x) = sign\big(f(x)\big)$$

$$= sign\left(\sum_{m=1}^{M} \alpha_m G_m(x)\right)$$

提升



Boosting算法示意图

提升

例题：根据下列样本，获得一个强分类器

X	0	1	2	3	4	5
Y	1	1	-1	-1	1	-1

初始权重0.167

第一轮迭代

1、选择最优弱分类器

若按1.5切分数据，得弱分类器g1(x): x < 1.5,则 y = 1; x > 1.5, 则 y = -1

此时错误率e_m为1 x 0.167 = 0.167

2、计算最优弱分类器的权重 $a_m = 0.5 \times \ln((1 - 0.167) / 0.167) = 0.8047$

3、更新样本权重

x = 0, 1, 2, 3, 5时，y分类正确，则样本权重为：0.167 x exp(-0.8047) = 0.075

x = 4时，y分类错误，则样本权重为：0.167 x exp(0.8047) = 0.373

规范化：0.075 / 0.748 = 0.10，0.373 / 0.748 = 0.50

新的样本权重为(0.1, 0.1, 0.1, 0.1, 0.5, 0.1)

提升

第一轮迭代后构造的分类器： $G(x) = 0.8047 \times g1(x)$

分类器的错误率为 $1 / 6 = 0.167$

第二轮迭代

1、选择最优弱分类器

若按4.5切分数据，得弱分类器 $g2(x)$: $x < 4.5$, 则 $y = 1$; $x > 4.5$, 则 $y = -1$

此时错误率 e_m 为 $2 \times 0.1 = 0.2$

2、计算最优弱分类器的权重 $a_m = 0.5 \times \ln((1 - 0.2) / 0.2) = 0.6931$

3、更新样本权重

$x = 0, 1, 5$ 时，y分类正确，则样本权重为：$0.1 \times \exp(-0.6931) = 0.05$

$x = 4$ 时，y分类正确，则样本权重为：$0.5 \times \exp(-0.6931) = 0.25$

$x = 2, 3$ 时，y分类错误，则样本权重为：$0.1 \times \exp(0.6931) = 0.2$

规范化后新的样本权重为(0.0625, 0.0625, 0.250, 0.250, 0.3125, 0.0625)

提升

第二轮迭代后构造的分类器： $G(x) = 0.8047 \cdot g1(x) + 0.6931 \cdot g2(x)$

分类器的错误率为 $1 / 6 = 0.167$

第三轮迭代

1、选择最优弱分类器

若按3.5切分数据，得弱分类器 $g3(x)$: $x > 3.5$, 则 $y = 1$; $x < 3.5$, 则 $y = -1$

此时错误率 e_m 为 $3 \times 0.0625 = 0.1875$

2、计算最优弱分类器的权重 $a_m = 0.5 \times \ln((1 - 0.1875) / 0.1875) = 0.7332$

3、更新样本权重

$x = 2$，3时，y分类正确，则样本权重为：$0.25 \times \exp(-0.7332) = 0.1201$

$x = 4$ 时，y分类正确，则样本权重为：$0.3125 \times \exp(-0.7332) = 0.1501$

$x = 2, 3$时，y分类错误，则样本权重为：$0.0625 \times \exp(0.7332) = 0.1301$

规范化后新的样本权重为(0.1667, 0.1667, 0.1539, 0.1539, 0.1923, 0.1667)

提升

第三轮迭代后构造的分类器： $G(x) = 0.8047 \cdot g1(x) + 0.6931 \cdot g2(x) + 0.7332 \cdot g3(x)$

按G(x)分类所有样本均分类正确，强分类器的错误率为0 / 6 = 0，停止迭代

- g1(x): x < 1.5,则 y = 1; x > 1.5, 则 y = -1
- g2(x): x < 4.5,则 y = 1; x > 4.5, 则 y = -1
- g3(x): x > 3.5,则 y = 1; x < 3.5, 则 y = -1

X	0	1	2	3	4	5	
Y	1	1	-1	-1	1	-1	
g1	1	1	-1	-1	-1	-1	x 0.8047
g2	1	1	1	1	1	-1	x 0.6931
g3	-1	-1	-1	-1	1	1	x 0.7332
G(x)	1	1	-1	-1	1	-1	

提问：AdaBoost算法中的Adaptive自适应体现在哪？

第六讲 集成学习与随机森林

提升

偏差：度量学习算法的期望预测与真是结果的偏离程度，刻画学习算法本身的拟合能力

方差：度量了同样大小的训练集的变动所导致的学习性能的变化，即刻画数据扰动造成的影响

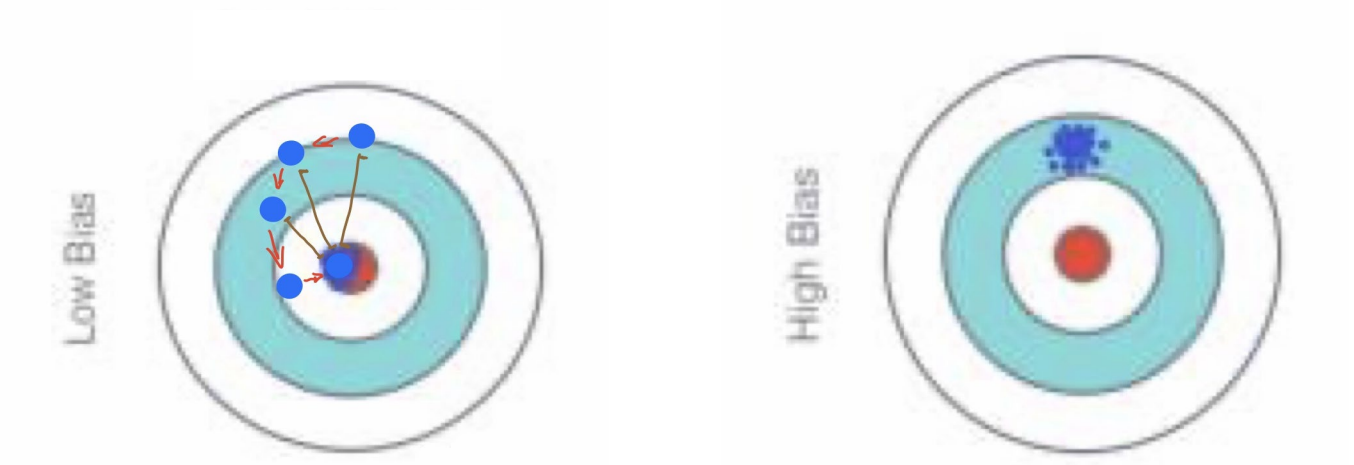
$$E(f;D) = bias^2(\boldsymbol{x}) + var(\boldsymbol{x}) + \varepsilon^2$$

$$bias^2(\boldsymbol{x}) = \left(\bar{f}(\boldsymbol{x}) - y\right)^2$$

$$var(\boldsymbol{x}) = \mathbb{E}_D\left[\left(f(\boldsymbol{x};D) - \bar{f}(\boldsymbol{x})\right)^2\right]$$

$$\varepsilon^2 = \mathbb{E}_D\left[(y_D - y)^2\right]$$

- 从偏差-方差角度来看，Boosting主要关注降低偏差，因此Boosting能基于泛化性能相当弱的学习器构建出很强的集成



集成学习与随机森林

1 个体与集成

2 模型组合策略

3 提升（Boosting）

4 装袋（Bagging）

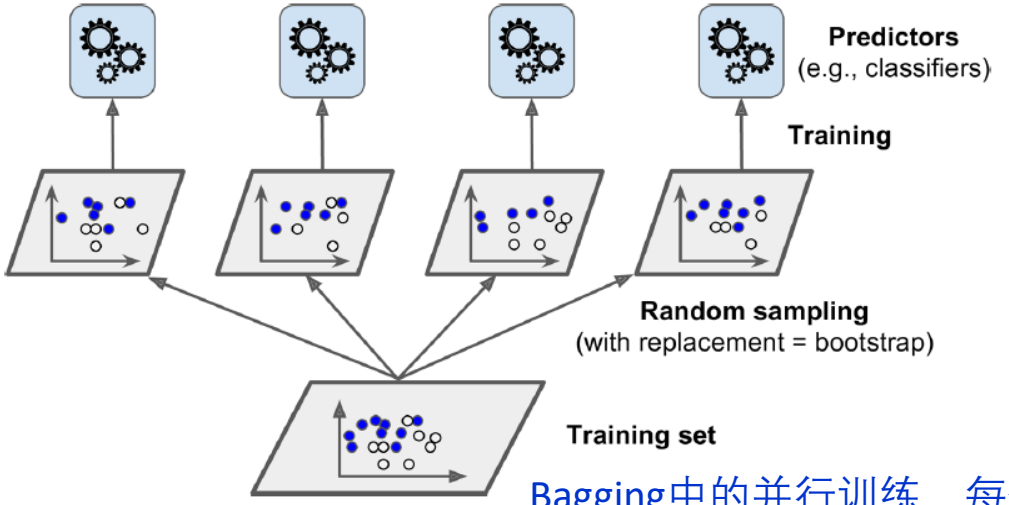
5 随机森林

第六讲 集成学习与随机森林

装袋

- Bagging是并行式集成学习方法最著名的代表，其基于自主采样法（bootstrap sampling）。给定m个样本，采取有放回的采样，经过m次随机采样操作，最终得到含m个样本的采样集。初始集的样本有的在采样集多次出现，有的未出现。计算$\lim_{n\to\infty}(1-\frac{1}{n})^n=\frac{1}{e}\approx 0{,}368$可得，初始集大约有63.2%的样本出现在采样集中。照此来得到T个采样集，然后基于采样集来训练基学习器，最终将这些学习器结合。

教材P286



Bagging中的并行训练，每个训练都使用不同的自助样本集

第六讲 集成学习与随机森林

装袋

- Bagging算法高效（计算复杂度低）且能直接应用与分类（简单投票）和回归任务（简单平均）
- 自主采样法剩下大约36.8% 的样本可用作验证集来对泛化性能进行“包外估计”（out-of-bag estimate）

- 其中$\mathcal{D}_{bs}$表示自主分布（Bootstrap）
- 基于多个样本自助集，训练出多个基学习器，再将这些基学习器集成

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

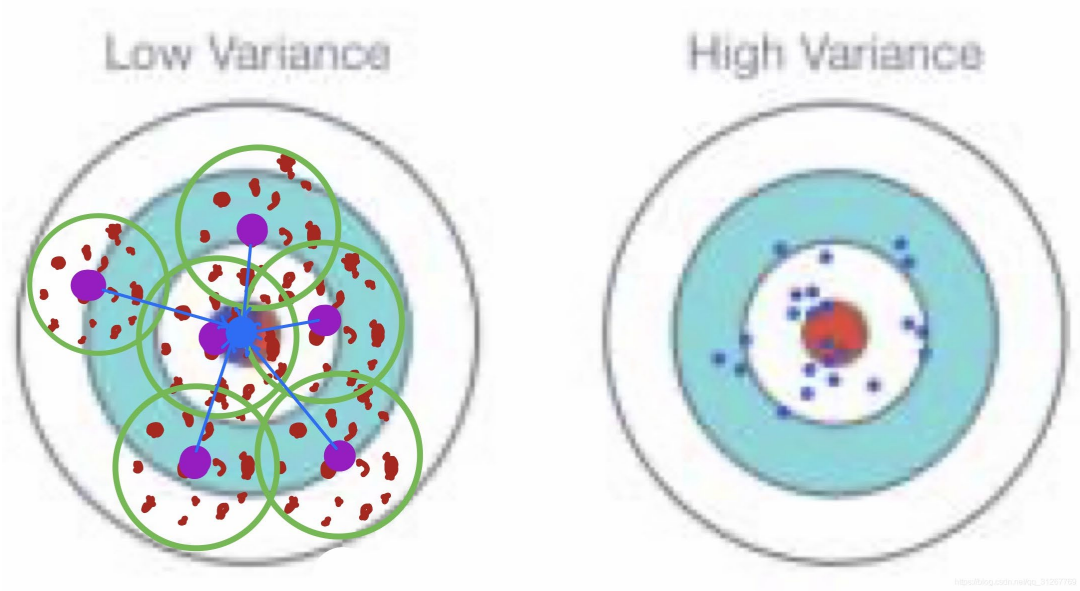
Bagging算法

西瓜书P178

第六讲 集成学习与随机森林

装袋

- 从偏差-方差角度来看，Bagging主要关注降低方差，因此它在不剪枝决策、神经网络等易受样本扰动的学习器上效用更为明显



集成学习与随机森林

1 个体与集成

2 模型组合策略

3 提升（Boosting）

4 装袋（Bagging）

5 随机森林

第六讲 集成学习与随机森林

随机森林

- 随机森林（Randon Forrest，RF）是Bagging方法的一个扩展和变体。在以决策树为基学习器构建Bagging集成的基础上，在决策树的训练过程中引入了随机属性选择。具体来说，传统决策树在选择划分属性时是在当前结点的属性集合（假设d个属性）中选择一个最优属性；而在随机森林方法中，对基决策树的每个结点，先从该结点的属性集合中随机选择一个包含k个属性的子集，然后再从这个子集中选择一个最优属性用于划分。这里的参数k控制了随机性的引入程度：若令$k = d$，则基决策树的构建与传统决策树相同；若令$k = 1$，则是随机选择一个属性用于划分；一般情况下，推荐值 $k = \log_2 d + 1$
- 随机森林方法简单、易实现、计算开销小，在很多实际任务中展现出强大的性能，被誉为“代表集成学习技术水平的方法”

第六讲 集成学习与随机森林

随机森林

- 随机森林对Bagging做了一点小改动：与一般Bagging方法中基学习器的多样性主要来自样本扰动（自助采样）不同，随机森林中基学习器的多样性不仅来自样本扰动，还来自特征扰动，这就使得集成学习器的泛化性能，通过个体学习器之间差异性的增加得以提升
- 随机森林的解释性不如决策树，但是随机森林有一个大的优势，那就是不必为如何选择好的超参数值而担心。通常不需要修剪随机森林，因为集合的模型对来自于单个决策树的噪声抵抗力强。实际上，唯一需要关心的参数是随机森林中决策树的个数 。通常情况下，树越多，随机森林分类器的性能就越好，伴随着计算成本就越大

第六讲 集成学习与随机森林

随机森林

- 随机森林方法背后的逻辑是对分别受较大方差影响的多个决策树取平均值，以建立一个具有更好的泛化性能和不易过拟合的强算法。算法可概括为以下几个简单步骤：

1. 从原始训练集中有放回地随机抽取规模为m的自助样本

2. 基于自助样本，按以下要求划分每个结点，完成一棵决策树的学习：

    2.1. 随机选择k个特征子集 (无放回)

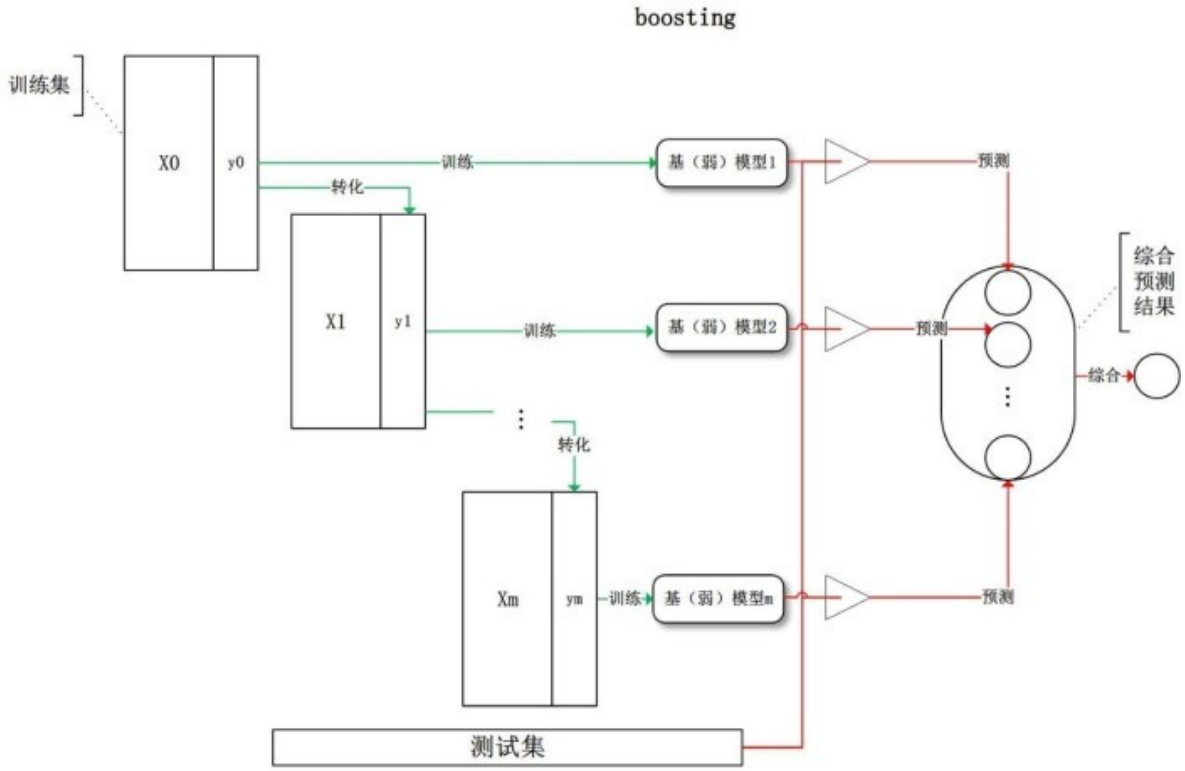
    2.2. 根据设定的最佳划分准则，选择最佳特征划分当前结点

3. 重复k次步骤1和2

第六讲 集成学习与随机森林

总结

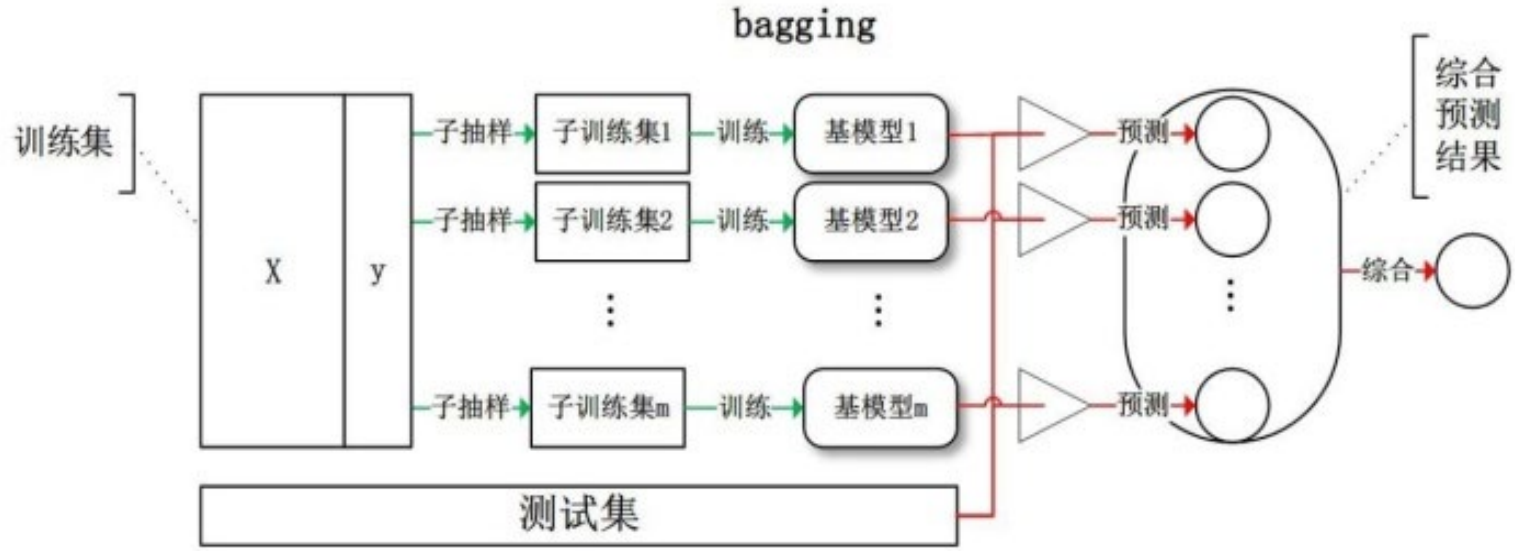
- 迭代提升（Boosting）：循环训练个体学习器，每个后继学习器都对其前驱学习器做出一些改进。因此个体学习器之间存在强依赖关系，必须串行生成



第六讲 集成学习与随机森林

总结

- 自助聚合 (Bagging)：每个个体学习器使用的算法是相同的，但是在不同的自助样本集上进行训练。因此个体学习器之间不存在强的计算依赖关系，可并行学习，最后再进行结果聚合。由于可并行学习，自助聚合法尤其在随机森林法中得到了广泛使用



第六讲 集成学习与随机森林

总结

1. 训练集

- Bagging：每个训练集都是从原始训练集中有放回的选取出来的，每个训练集各不相同且相互独立
- Boosting：每一轮的训练集都是原始选练集，只是每次训练后会根据本轮的训练结果调整训练集中的各个样本的权重，调整完权重的训练集用于下一轮的训练

2. 样本权重

- Bagging：使用Bootstraping的方式均匀抽样
- Boosting：根据每一轮的训练不断调整权值，分类错误的样本拥有更高的权值

总结

3. 弱分类器权重

- Bagging：所有弱分类器权重相同，使用voting的方式（或均值）决定最终结果
- Boosting：每个弱分类器都有相应的权重，对于分类误差小的分类器会有更大的权重

4. 并行计算

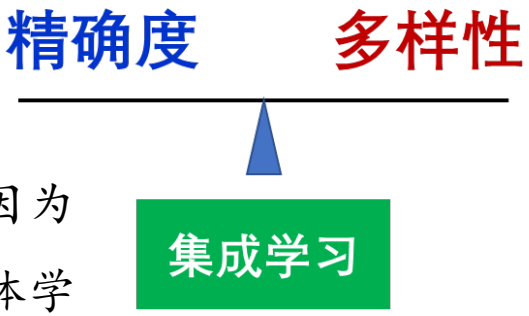
- Bagging：各个预测函数可以并行生成，因为数据集相互独立，每个模型之间也独立，没有序列关系
- Boosting：各个预测函数只能顺序生成，因为下一个模型的产生依赖于之前模型的计算结果

第六讲 集成学习与随机森林

总结

- 集成多样性 (diversity)：集成多样性，即个体学习器之间的差异，是集成学习方法的重要议题。
  - 若结合完全相同的个体学习器，显然不会有性能提升
  - 要想在结合之后有性能提升，个体学习器之间必须有差异
- 多样性至关重要，却难以刻画：目前还没有一个广泛接受的定义

理解多样性→集成学习领域的圣杯问题



个体学习器应当兼具精确性与多样性。正是因为“互补性”比单纯的精度更重要，结合高精度的个体学习器经常比结合相对稍弱的个体学习器性能更差。集成学习的成功依赖于个体学习器精度和多样性之间的权衡

第六讲 集成学习与随机森林

总结

7.11 sklearn.ensemble: Ensemble Methods

The *sklearn.ensemble* module includes ensemble-based methods for classification, regression and anomaly detection.

User guide: See the *Ensemble methods* section for further details.

ensemble.AdaBoostClassifier([...])	An AdaBoost classifier.
ensemble.AdaBoostRegressor([base_estimator, ...])	An AdaBoost regressor.
ensemble.BaggingClassifier([base_estimator, ...])	A Bagging classifier.
ensemble.BaggingRegressor([base_estimator, ...])	A Bagging regressor.
ensemble.ExtraTreesClassifier([...])	An extra-trees classifier.
ensemble.ExtraTreesRegressor([n_estimators, ...])	An extra-trees regressor.
ensemble.GradientBoostingClassifier(*[, ...])	Gradient Boosting for classification.
ensemble.GradientBoostingRegressor(*[, ...])	Gradient Boosting for regression.
ensemble.IsolationForest(*[, n_estimators, ...])	Isolation Forest Algorithm.
ensemble.RandomForestClassifier([...])	A random forest classifier.
ensemble.RandomForestRegressor([...])	A random forest regressor.
ensemble.RandomTreesEmbedding([...])	An ensemble of totally random trees.
ensemble.StackingClassifier(estimators[, ...])	Stack of estimators with a final classifier.
ensemble.StackingRegressor(estimators[, ...])	Stack of estimators with a final regressor.
ensemble.VotingClassifier(estimators, *[, ...])	Soft Voting/Majority Rule classifier for unfitted estimators.
ensemble.VotingRegressor(estimators, *[, ...])	Prediction voting regressor for unfitted estimators.
ensemble.HistGradientBoostingRegressor([...])	Histogram-based Gradient Boosting Regression Tree.
ensemble.HistGradientBoostingClassifier([...])	Histogram-based Gradient Boosting Classification Tree.

总结

思考题1：如果已经在完全相同的训练集上训练了五个不同的学习器，并且它们都达到了95%的准确率，是否还有机会通过组合这些学习器来获得更好的结果？如果可以，该怎么做？如果不行，为什么？

答：如果已经训练了五个不同的学习器，并且都达到了95% 的精度，你可以尝试将它们组合成一个投票集成，这通常会带来更好的结果。如果学习器模型之间非常不同（例如，一个SVM 分类器，一个决策树分类器，以及一个Logistic 回归分类器等），则效果更优。如果它们是在不同的训练样本集（这是bagging 集成的关键点）上完成训练，那就更好了；如果不是，只要学习器模型非常不同，这个集成仍然有效

总结

思考题2： 请根据下列数据，运用AdaBoost构造一个强分类器

x	0	1	2	3	4	5	6	7	8	9
y	1	1	1	-1	-1	-1	1	1	1	-1

	x	0	1	2	3	4	5	6	7	8	9	阈值	误判个数	epsilon=sum(wi*错误位置)	备注
	y	1	1	1	-1	-1	-1	1	1	1	-1				
fx1	g1	1	1	1	-1	-1	-1	-1	-1	-1	-1	2.5	3	0.3	
fx2	g2	1	1	1	1	1	1	-1	-1	-1	-1	5.5	6	0.6	超过0.5，修改判别规则
fx3	g3	1	1	1	1	1	1	1	1	1	-1	8.5	3	0.3	
	w	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1				
													误判个数	epsilon=sum(wi*错误位置)	
	g1	1	1	1	-1	-1	-1	-1	-1	-1	-1	2.5	3	0.3	选择误差最小的分类器，以g1为例
	g2	-1	-1	-1	-1	-1	-1	1	1	1	1	5.5	4	0.4	
	g3	1	1	1	1	1	1	1	1	1	-1	8.5	3	0.3	alpha1=0.4236
	w	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1				fx1=sign(alpha1*g1)
	第1次更新w	0.0714	0.0714	0.0714	0.0714	0.0714	0.0714	0.1667	0.1667	0.1667	0.0714				
	alpha1*g1	0.4236	0.4236	0.4236	-0.4236	-0.4236	-0.4236	-0.4236	-0.4236	-0.4236	-0.4236				
	fx1=sign(alpha1*g1)	1	1	1	-1	-1	-1	-1	-1	-1	-1	2.5	3	0.1667*3=0.5	
	g2	-1	-1	-1	-1	-1	-1	1	1	1	1	5.5	4	0.0714*3=0.2857	
	g3	1	1	1	1	1	1	1	1	1	-1	8.5	3	0.0714*3=0.2143	选择最小的误差分类器g3
	第2次更新w	0.0455	0.0455	0.0455	0.1667	0.1667	0.1667	0.1061	0.1061	0.1061	0.0455				alpha3=0.6496
															fx3=sign(alpha1*g1+alpha3*g3)
	fx1=sign(alpha1*g1)	1	1	1	-1	-1	-1	-1	-1	-1	-1	2.5	3	0.1061*3=0.3182	
	g2	-1	-1	-1	-1	-1	-1	1	1	1	1	5.5	4	0.0455*4=0.1818	选择最小的误差分类器g2
	fx3=sign(alpha1*g1+alpha3*g	1	1	1	1	1	1	1	1	1	-1	8.5	3	0.1667*3=0.5	alpha2=0.7520
	(alpha1*g1+alpha3*g3)	1.0733	1.0733	1.0733	0.2260	0.2260	0.2260	0.2260	0.2260	0.2260	-1.0733				fx2=sign(alpha1*g1+alpha2*g2+alpha3*g3)
	第3次更新w	0.1250	0.1250	0.1250	0.1019	0.1019	0.1019	0.0648	0.0648	0.0648	0.1250				
	fx1=sign(alpha1*g1)	1	1	1	-1	-1	-1	-1	-1	-1	-1	2.5	3	0.0648*3=0.1944	
2=sign(alpha1*g1+alpha2*g2+alp		1	1	1	-1	-1	-1	1	1	1	-1	5.5	0	0	最小误差为0，则fx3即所求组合分类器
	fx3=sign(alpha1*g1+alpha3*g	1	1	1	1	1	1	1	1	1	-1	8.5	3	0.1019*3=0.3056	