明理精工 笃学致远

第8章 集成学习

Ensemble Learning

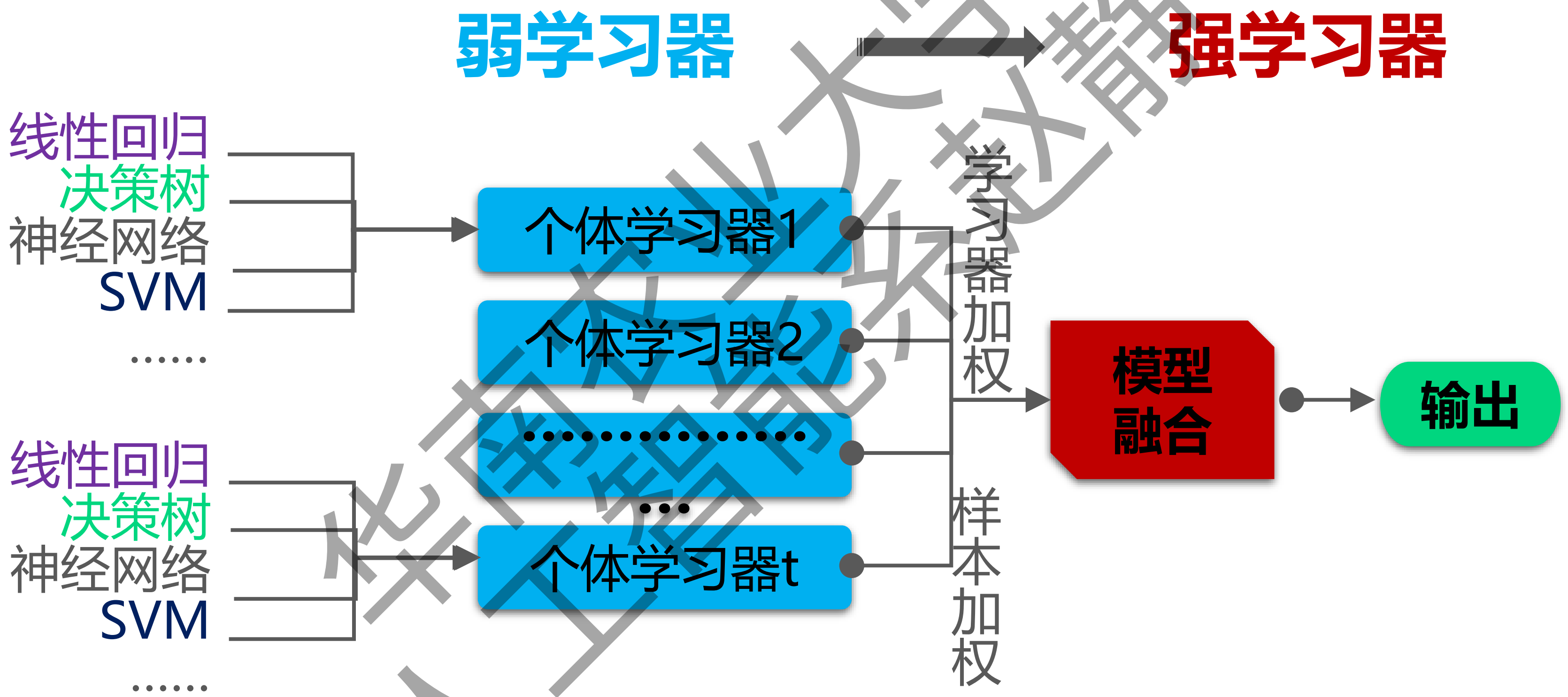
电子工程学院、人工智能学院

college of Electronic Engineering , college of Artificial Intelligence

讲授内容

- 个体与集成
- Bagging
- Boosting

1. 个体与集成


弱学习器
强学习器
线性回归
决策树
神经网络
SVM
……
线性回归
决策树
神经网络
SVM
……
个体学习器1
个体学习器2
……………
…
个体学习器t
学习器加权
样本加权
模型融合
输出

	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

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

#周志华

- 个体与集成
- Bagging
  - Bagging
  - 随机森林
- Boosting

2.1 Bagging

- 对给定有N个样本的数据集$\mathcal{D}$进行**B**ootstrap采样，得到$\mathcal{D}^1$，在$\mathcal{D}^1$上训练模型f_1
- 上述过程重复M次，得到M个模型，则M个模型的平均（回归）/投票（分类）为：

$$f_{avg}(\mathbf{x}) = \frac{1}{M}\sum_{m=1}^{M} f_m(\mathbf{x})$$

- 可以证明：Bagging可以降低模型的方差。

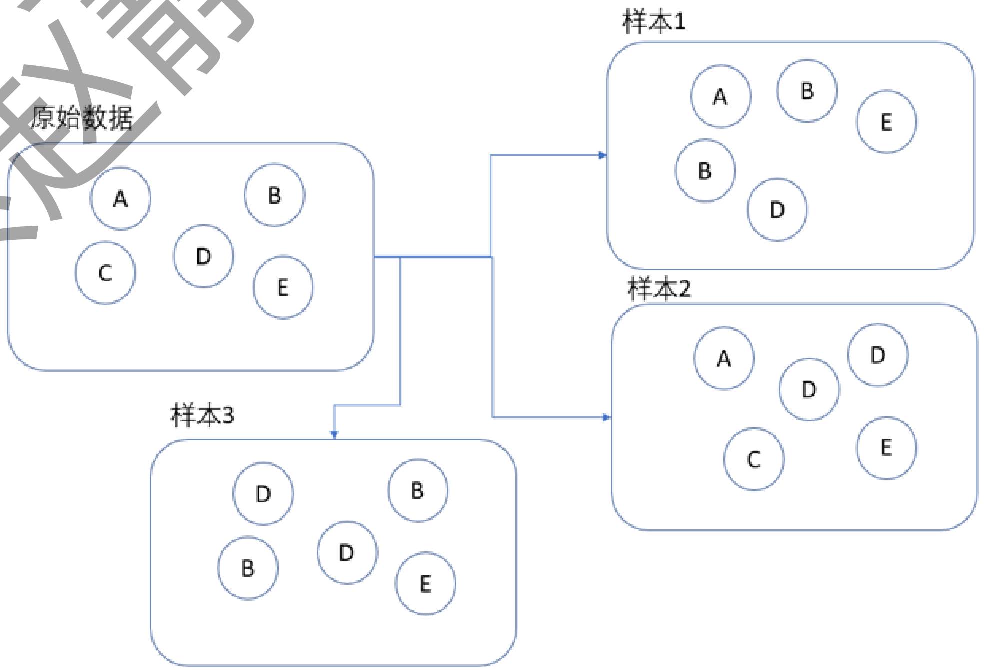
➢ Bootstrap采样

- 通过从原始的N个样本数据$\mathcal{D}=\{\mathbf{x}_1,\dots,\mathbf{x}_N\}$进行$N$次有放回采样$N$个数据$\mathcal{D}'$，称为一个bootstrap样本。

如：若原始样本为$\mathcal{D}=\{A,B,C,D,E\}$

则bootstrap样本可能为：$\mathcal{D}^1=\{A,B,B,D,E\}$

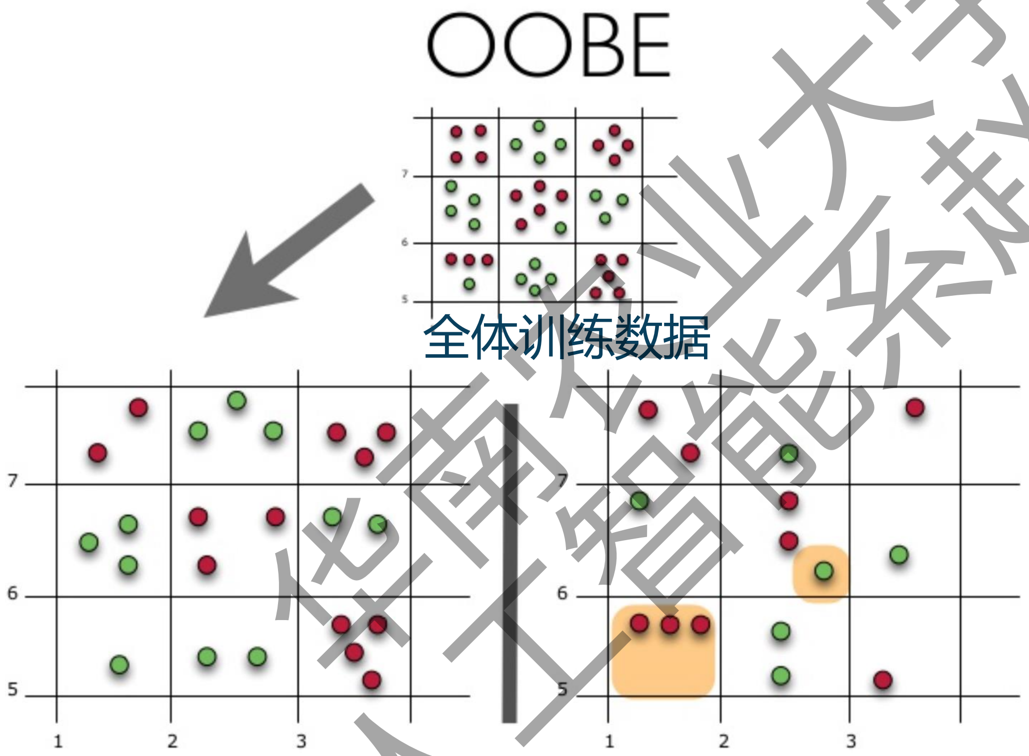
$$\mathcal{D}^2=\{A,C,D,D,E\}$$



一个样本不在采样集中出现的概率：$\left(1-\frac{1}{N}\right)^N$。（$lim_{N\to\infty}\left(1-\frac{1}{N}\right)^N=0.368$）

原始训练集中约有：$1-0.368=63.2\%$的样本出现在采样集中。

➢ Out-of-bag error （OOBE)



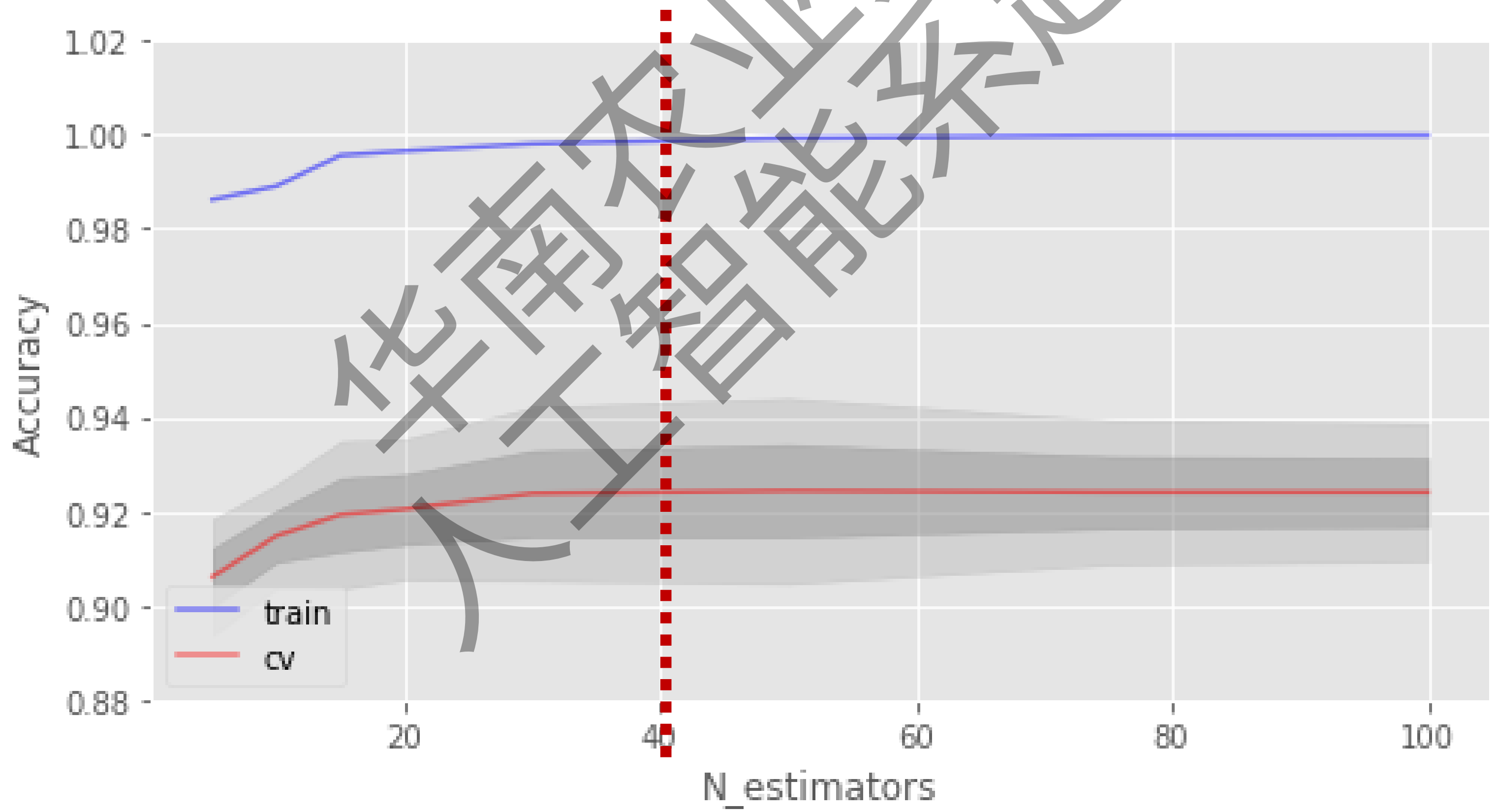
全体训练数据

单次训练数据

➤ 基学习器数目

参数值建议：

对分类问题，可设置基学习器数目为$\sqrt{D}$，其中D为特征数目；
对回归问题，可设置基学习器数目为$D/3$。



➢ Bagging特点

- Bagging可降低模型方差，不改变模型偏差，适合对偏差低、方差高的模型进行融合
  如决策树、神经网络

决策树很容易过拟合 → 偏差低、方差高

如果每个训练样本为一个叶子结点，训练误差为0

推荐阅读：
1. 《为什么说bagging是减少variance，而boosting是减少bias?》
2. 使用sklearn进行集成学习——理论https://www.cnblogs.com/jasonfreak/p/5657196.html

2.2 随机森林（Random Forest）

➢ 随机森林的基本原理

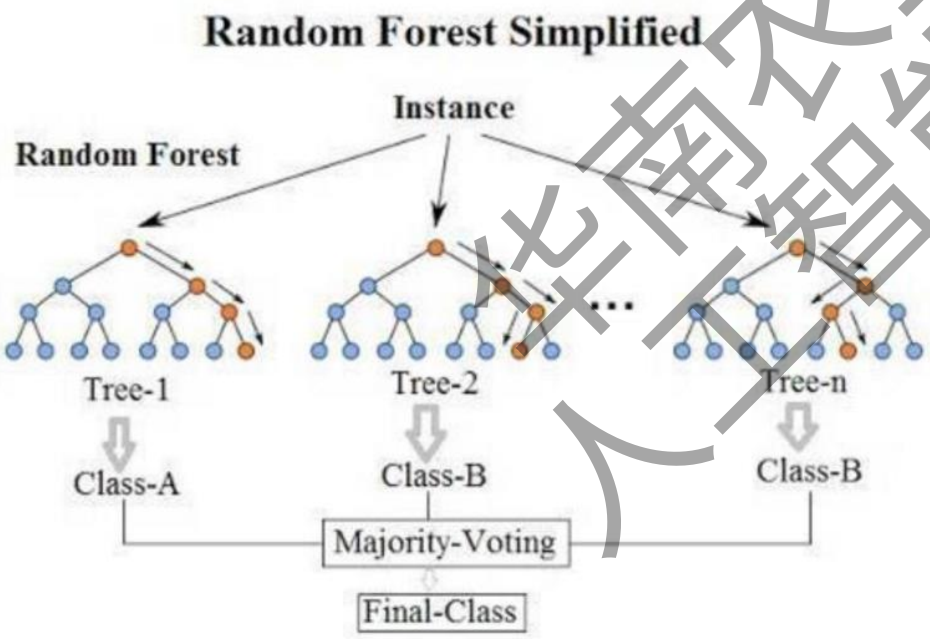
- 随机选择一部分特征
- 随机选择一部分样本

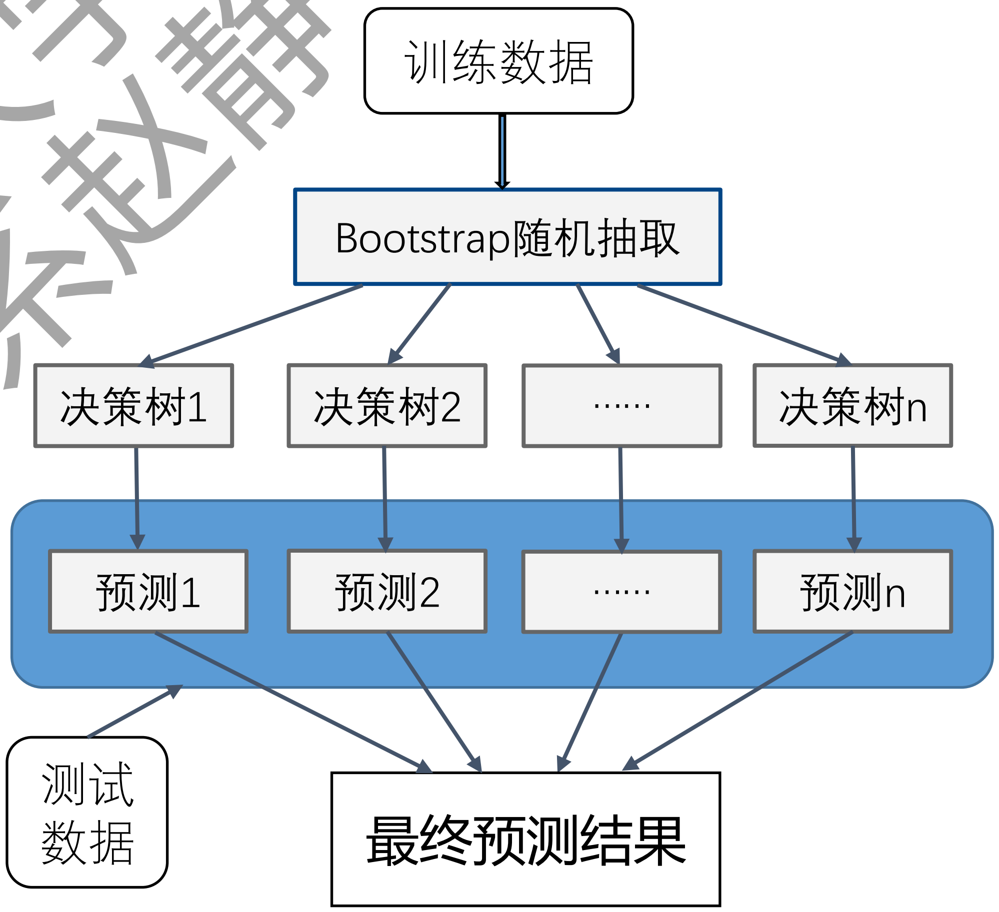
随机森林降低树的相关性

- 森林：多棵树
- 随机：对样本和特征进行随机抽取

➤ 随机森林步骤

1. 随机选择样本（放回抽样）；
2. 随机选择特征;
3. 构建决策树;
4. 随机森林投票（平均）。





- 个体与集成
- Bagging
- Boosting
  - AdaBoost
  - GBDT
  - XGBoost

- Boosting: 将弱学习器组合成强分类器
- Boosting学习框架

  ① 学习第一个弱学习器ϕ_1

  ② 学习第二个弱学习器ϕ_2，ϕ_2要能帮助ϕ_1（ϕ_2和ϕ_1互补）

  …

  最后，组合所有的弱学习器：$f(\mathbf{x}) = \sum_{m=1}^{M} \alpha_m \phi_m(\mathbf{x})$

弱学习器是按顺序学习的!

- 怎样得到互补的学习器?

$(\mathbf{x}_1, y_1, w_1)$

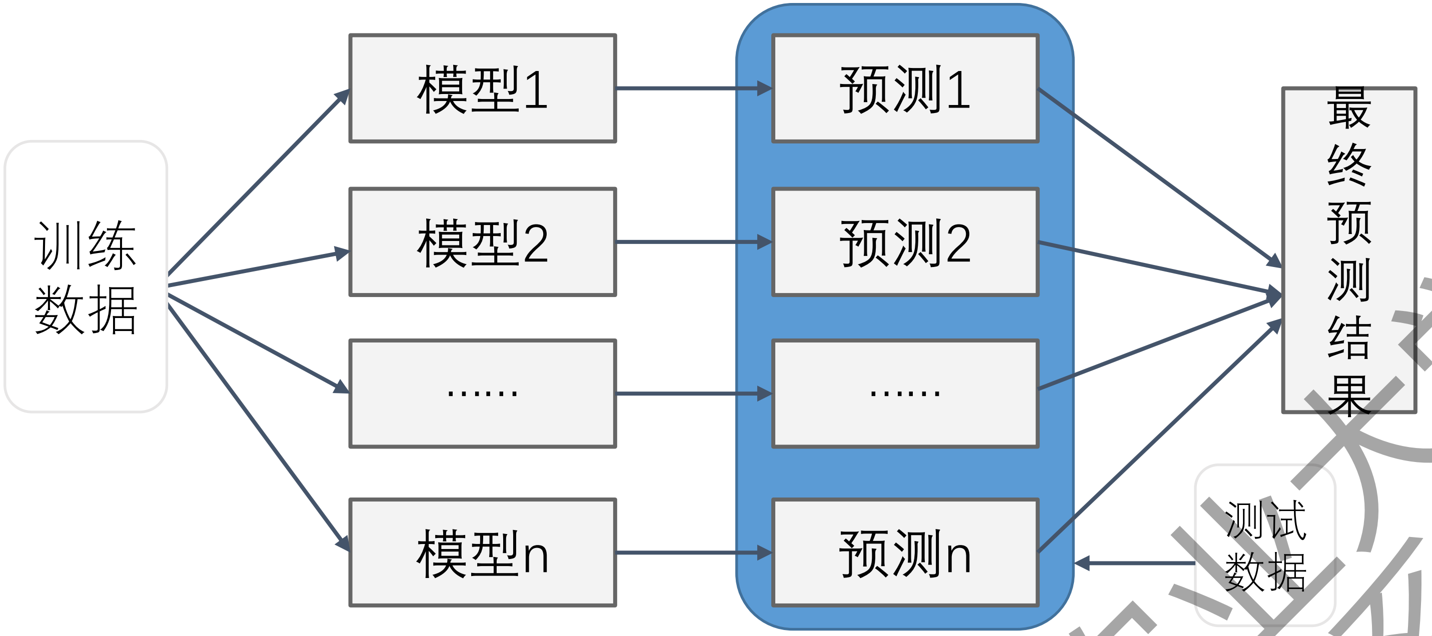
…

$(\mathbf{x}_N, y_N, w_N)$

$$\boxed{J(f,\lambda) = \sum_{i=1}^{N} L(y_i, f(\mathbf{x}_i)) + \lambda R(f)}$$

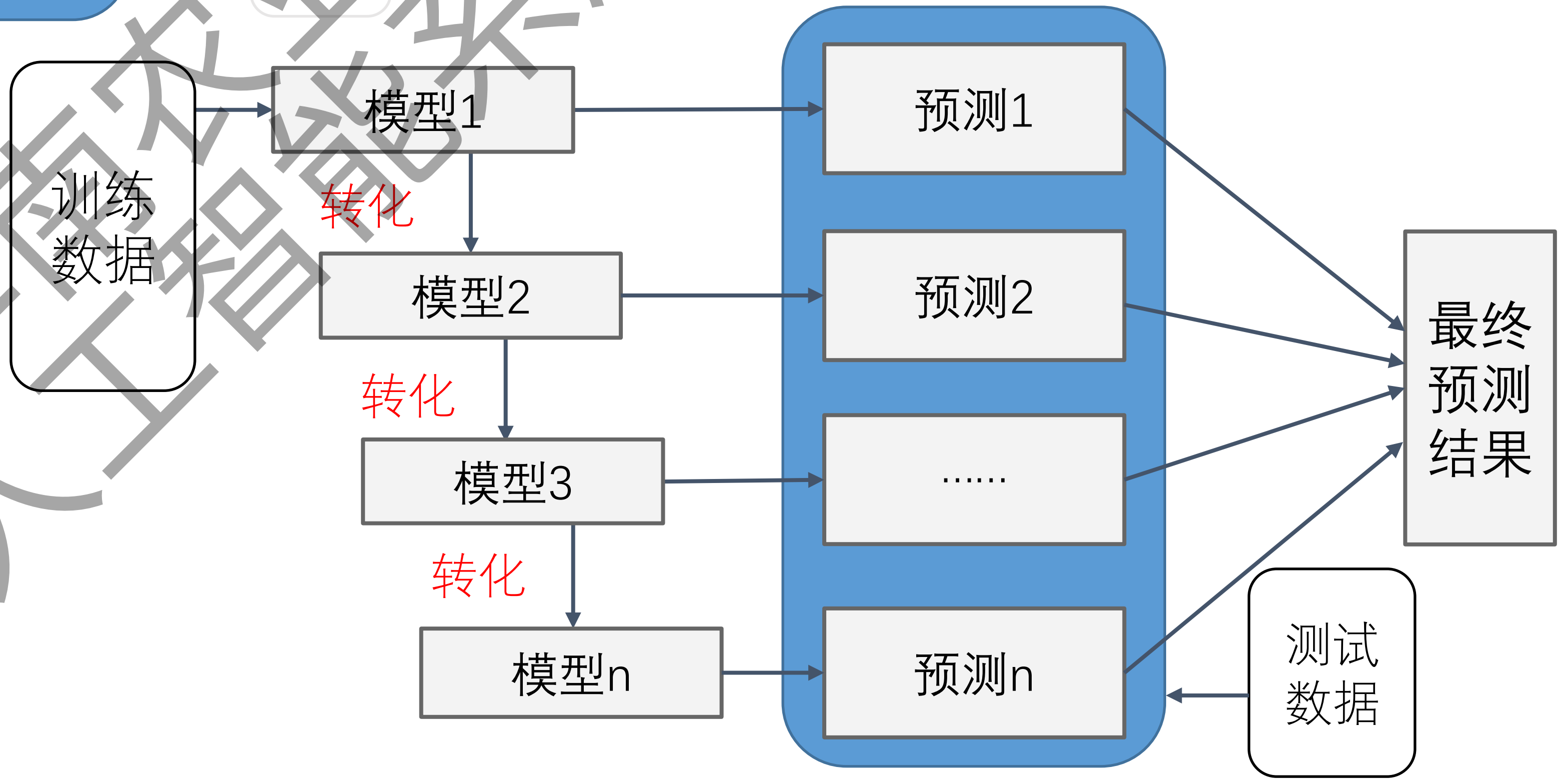
$$\Downarrow$$

$$\boxed{J(f,\lambda) = \sum_{i=1}^{N} w_i L(y_i, f(\mathbf{x}_i)) + \lambda R(f)}$$



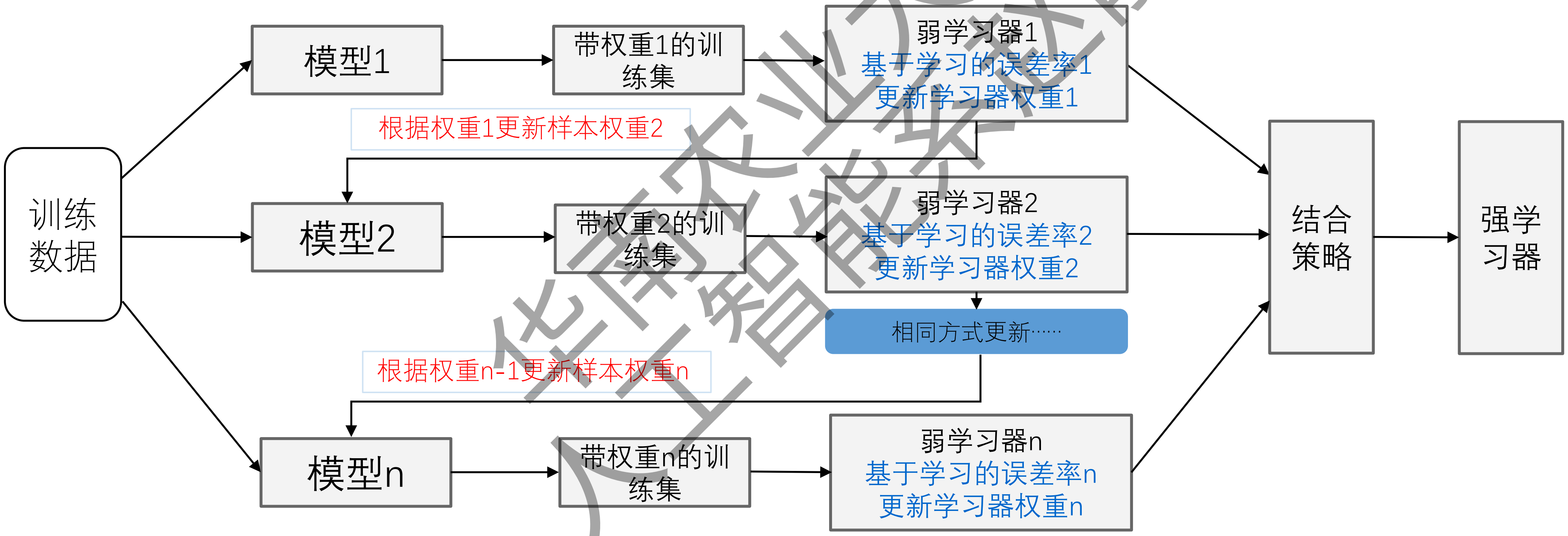
Bagging

Boosting



➢ Boosting算法思想：

后一个模型的训练永远是在前一个模型的基础上完成



➢ Boosting的一般框架

1. 初始化 $f_0(\mathbf{x})$

2. for $m = 1{:}\,M$ do

- 找一个弱学习器$\phi_m(\mathbf{x})$，使得$\phi_m(\mathbf{x})$能改进$f_{m-1}(\mathbf{x})$；
- 更新 $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \alpha_m \phi_m(\mathbf{x})$

3. return $f(\mathbf{x}) = f_M(\mathbf{x})$

3.1 AdaBoost（Adaptive Boosting）

➢ AdaBoost的基本思想

- 在弱学习器ϕ_1失败的样本上学习第二个弱学习器ϕ_2
- 令弱学习器ϕ_1在其训练集上的误差为：

$$\varepsilon_1 = \sum_{i=1}^{N} w_{1,i}\mathbb{I}\big(y_i \neq \phi_1(\mathbf{x}_i)\big) < \frac{1}{2},$$

- 将权重由$\mathbf{w}_1$变成$\mathbf{w}_2$，使得

$$\sum_{i=1}^{N} w_{2,i}\mathbb{I}\big(y_i \neq \phi_1(\mathbf{x}_i)\big) = \frac{1}{2}$$

- 根据权重$\mathbf{w}_2$训练弱学习器ϕ_2

➢ 样本重新加权的方法

- 分对的样本的权重：$w_{2,i} = \dfrac{w_{1,i}/d_1}{Z_1}$　　权重减小
- 分错的样本的权重：$w_{2,i} = \dfrac{w_{1,i}d_1}{Z_1}$　　权重增大

归一化因子：$Z_1 = \sum_{i=1}^{N} w_{1,i} d_1 \mathbb{I}(y_i \neq \phi_1(\mathbf{x}_i)) + \sum_{i=1}^{N} w_{1,i}/d_1 \, \mathbb{I}(y_i = \phi_1(\mathbf{x}_i))$

- 求解 d_1

$$\sum_{i=1}^{N} w_{2,i} \mathbb{I}(y_i \neq \phi_1(\mathbf{x}_i)) = \sum_{i=1}^{N} \frac{w_{1,i} d_1}{Z_1} \mathbb{I}(y_i \neq \phi_1(\mathbf{x}_i))$$

$$= \frac{\sum_{i=1}^{N} w_{1,i} d_1 \mathbb{I}(y_i \neq \phi_1(\mathbf{x}_i))}{\sum_{i=1}^{N} w_{1,i} d_1 \mathbb{I}(y_i \neq \phi_1(\mathbf{x}_i)) + \sum_{i=1}^{N} w_{1,i}/d_1 \, \mathbb{I}(y_i = \phi_1(\mathbf{x}_i))} = \frac{1}{2}$$

$$\sum_{i=1}^{N} w_{1,i} d_1 \mathbb{I}(y_i \neq \phi_1(\mathbf{x}_i)) = \sum_{i=1}^{N} w_{1,i}/d_1 \, \mathbb{I}(y_i = \phi_1(\mathbf{x}_i))$$

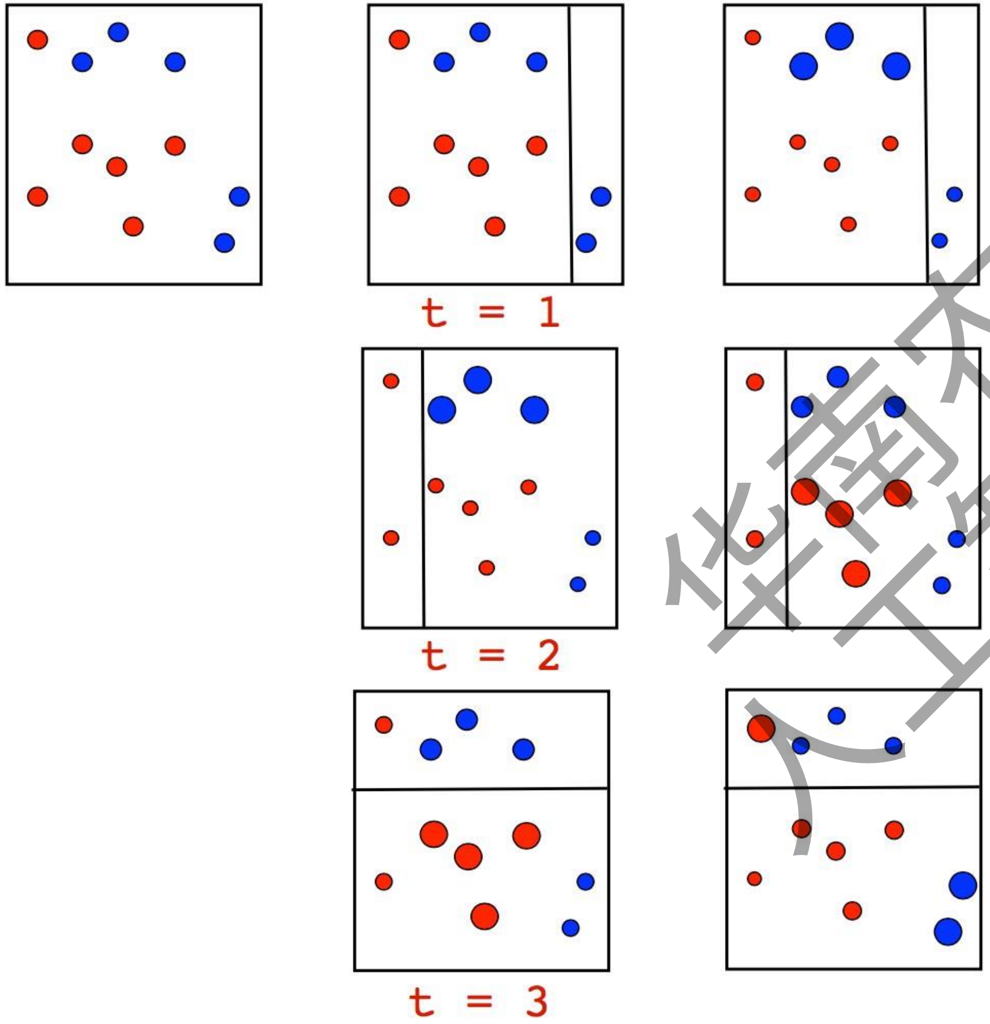
$$d_1 \underbrace{\sum_{i=1}^{N} w_{1,i} \mathbb{I}(y_i \neq \phi_1(\mathbf{x}_i))}_{\varepsilon_1} = \frac{1}{d_1} \underbrace{\sum_{i=1}^{N} w_{1,i} \mathbb{I}(y_i = \phi_1(\mathbf{x}_i))}_{(1-\varepsilon_1)} \qquad d_1 \varepsilon_1 = \frac{1}{d_1}(1-\varepsilon_1) \longrightarrow \boxed{d_1 = \sqrt{(1-\varepsilon_1)/\varepsilon_1} > 1}$$

➢ 举例

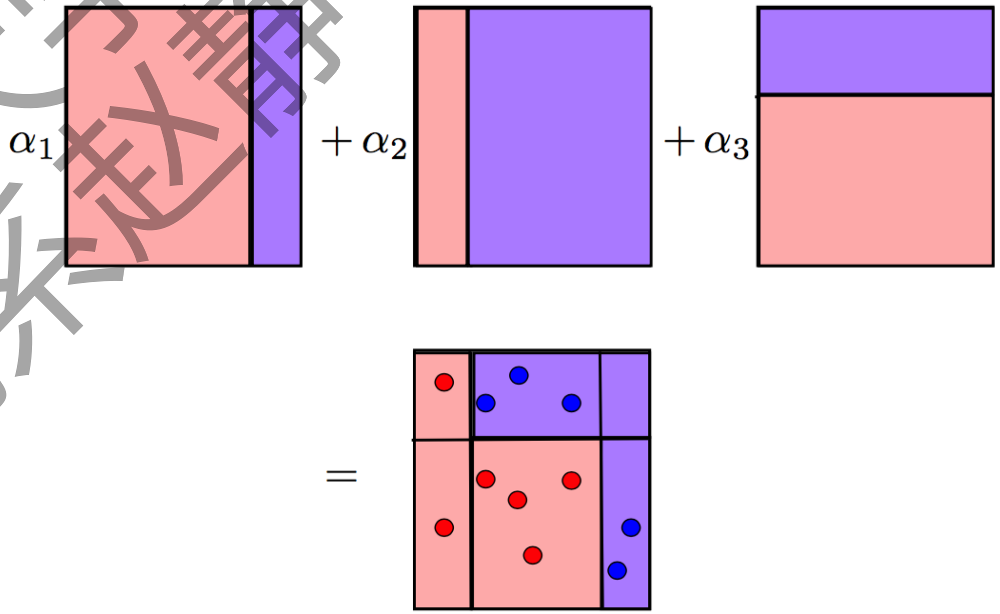
初始样本权重

增加弱学习器之前的样本权重

增加弱学习器之后的样本权重



弱学习器：深度为1的决策树（树桩，stumps）



强学习器

最后的强分类器为：

$$f(\mathbf{x}) = \operatorname{sgn}\left(\sum_{m=1}^{M} \alpha_m \phi_m(\mathbf{x})\right)$$

3.2 Gradient Boosting

- Gradient Boosting：目标函数$J(f)=\sum_{i=1}^{N} L(f(\mathbf{x}_i), y_i)$最小

梯度下降：$f_m(\mathbf{x})=f_{m-1}(\mathbf{x})-\eta\left[\frac{\partial J(f)}{\partial f(\mathbf{x})}\right]_{f(\mathbf{x})=f_{m-1}(\mathbf{x})}$

$f_m(\mathbf{x})=f_{m-1}(\mathbf{x})+\alpha_m\phi_m(\mathbf{x})$

用弱学习器来拟合目标函数的负梯度

➢ 算法

1. Initialize $f_0(\mathbf{x}) = \underset{f}{\operatorname{argmin}} \frac{1}{N} \sum_{i=1}^{N} L\big(y_i, f(\mathbf{x}_i)\big)$
2. for $m = 1{:}\,M$ do

   ①Compute the gradient residual using $r_{m,i} = -\left[\frac{\partial L(y_i, f(\mathbf{x}_i))}{\partial f(\mathbf{x}_i)}\right]_{f=f_{m-1}}$

   ②Use the weak learner which minimizes $\sum_{i=1}^{N} \big(r_{m,i} - \phi_m(\mathbf{x}_i)\big)^2$

   ③Update $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \eta\phi_m(\mathbf{x})$

3. return $f(\mathbf{x}) = f_M(\mathbf{x})$

例：L2Boosting

- 对L2损失：$L(f(\mathbf{x}), y) = \frac{1}{2}(f(\mathbf{x}) - y)^2$
- $J(f) = \sum_{i=1}^{\mathrm{N}} L(f(\mathbf{x}_i), y_i) = \frac{1}{2}\sum_{i=1}^{\mathrm{N}}(f(\mathbf{x}_i) - y_i)^2$
- $\frac{\partial J(f)}{\partial f} = \sum_{i=1}^{\mathrm{N}}(f(\mathbf{x}_i) - y_i)$

（1）初始化：$f_0(\mathbf{x}) = \bar{y}$

（2）①计算负梯度：$-\left[\frac{\partial J(f)}{\partial f}\right]_{f=f_{m-1}} = -\sum_{i=1}^{\mathrm{N}}(f_{m-1}(\mathbf{x}_i) - y_i) = r_{m,i}$为预测残差。

②找一个弱学习器$\phi_m(\mathbf{x})$，使$\sum_{i=1}^{\mathrm{N}}\left(r_{m,i} - \phi_m(\mathbf{x}_i)\right)^2$最小

③更新 $f_m(\mathbf{x}) = f_{m-1}(\mathbf{x}) + \alpha_m\phi_m(\mathbf{x})$

（3）return $f_M(\mathbf{x})$

从Gradient Descent 到 Gradient Boosting

参数空间

$$\theta^t = \theta^{t-1} + \theta_t$$

第t次迭代后的参数　第t-1次迭代后的参数　第t次迭代的参数增量

$$\theta_t = -\alpha_t g_t$$

参数更新方向为负梯度方向

$$\theta = \sum_{t=0}^{T} \theta_t$$

最终参数等于每次迭代的增量的累加和，

θ_0 为初值

函数空间

$$f^t(x) = f^{t-1}(x) + f_t(x)$$

第t次迭代后的函数　第t-1次迭代后的函数　第t次迭代的函数增量

$$f_t(x) = -\alpha_t g_t(x)$$

同样地，拟合负梯度

$$F(x) = \sum_{t=0}^{T} f_t(x)$$

最终函数等于每次迭代的增量的累加和，

$f_0(x)$ 为模型初始值，通常为常数

3.3 XGBoost

XGBoost：eXtreme Gradient Boosting

➢ 特点：

- **损失函数**：采用二阶Taylor展开近似损失函数
- **正则项**：叶子节点数目、叶子结点的分数
- **建树**：支持分裂点近似搜索，稀疏特征处理，缺失值处理
- 并行计算
- 内存优化

推荐阅读：深入理解XGBoost（https://blog.csdn.net/fengdu78/article/details/104284924）

➢ **损失函数的二阶近似**

$$\boxed{f(x+\Delta x) \cong f(x) + f'(x)\Delta x + \frac{1}{2} f''(x)\Delta x^2}$$

- 在第m步时，令 $g_{m,i} = \left[\frac{\partial L(f(\mathbf{x}_i), y_i)}{\partial f(\mathbf{x}_i)}\right]_{f=f_{\mathrm{m}-1}}$，$h_{m,i} = \left[\frac{\partial^2 L(f(\mathbf{x}_i), y_i)}{\partial^2 f(\mathbf{x}_i)}\right]_{f=f_{\mathrm{m}-1}}$

- $L\big(y_i, f_{m-1}(\mathbf{x}_i) + \phi(\mathbf{x}_i)\big) = \underbrace{L(f_{m-1}(\mathbf{x}_i), y_i)}_{\text{与未知量}\phi(\mathbf{x}_i)\text{无关}} + g_{m,i}\ \phi(\mathbf{x}_i) + \frac{1}{2} h_{m,i}\phi(\mathbf{x}_i)^2$

- 所以$L(f_{m-1}(\mathbf{x}_i) + \phi(\mathbf{x}_i), y_i) = g_{m,i}\ \phi(\mathbf{x}_i) + \frac{1}{2} h_{m,i}\phi(\mathbf{x}_i)^2$

对L2损失，$L(f(\mathbf{x}), y) = \frac{1}{2}(f(\mathbf{x}) - y)^2$，$\nabla_f L(\boldsymbol{\theta}) = f(\mathbf{x}) - y$，$\nabla_f^2 L(\boldsymbol{\theta}) = 1$。
所以$g_{m,i} = f_{m-1}(\mathbf{x}_i) - y_i$，$h_{m,i}=1$。

➢ 正则项

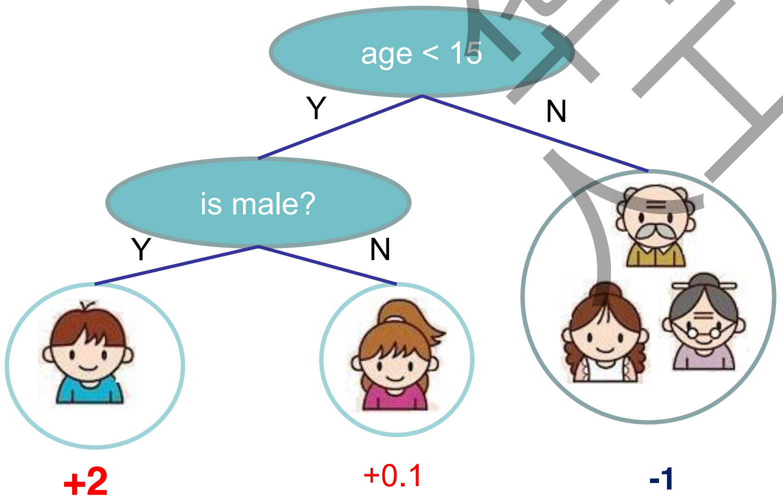
- 基学习器为**二叉决策树**

$$\phi(\mathbf{x}) = w_{q(\mathbf{x})},\quad \mathbf{w} \in R^T, q\colon R^D \to \{1, \dots, T\}$$

结构函数q：把输入$\mathbf{x}$映射到叶子的索引号

w：每个索引号对应的叶子的分数

T为树中叶子结点的数目，D为特征维数



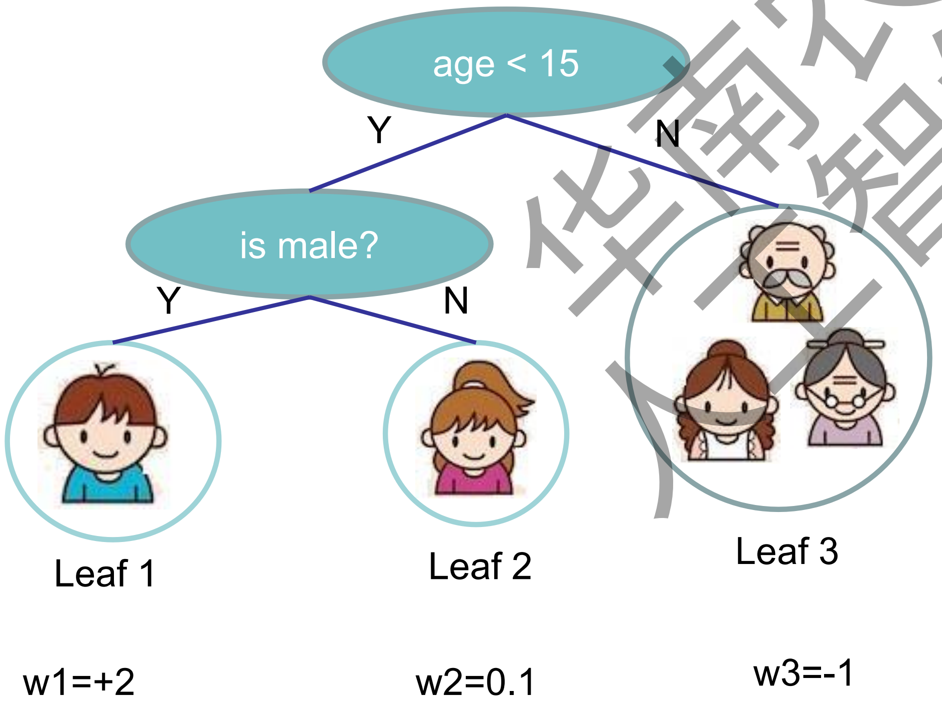
q() = 1

q() = 3

- 正则项:

$$R(\phi(\mathbf{x})) = \gamma T + \frac{1}{2}\lambda \sum_{t=1}^{T} w_t^2$$

叶子节点的数目T（L1正则）、叶子节点分数的平方和（L2正则）



$$\Omega = \gamma 3 + \frac{1}{2}\lambda(4 + 0.01 + 1)$$

$$2^2+0.1^2+(-1)^2$$

目标函数

令每个叶子t上的样本集合为$I_t = \{i | q(\mathbf{x}_i) = t\}$

$$
\begin{aligned}
J(f) &= \sum_{i=1}^{N} L(f(\mathbf{x}_i;\boldsymbol{\theta}), y_i) + R(f) \\
&\cong \sum_{i=1}^{N} \left(g_{m,i}\phi(\mathbf{x}_i) + \frac{1}{2} h_{m,i}\phi(\mathbf{x}_i)^2 \right) + \gamma T + \frac{1}{2}\lambda \sum_{t=1}^{T} w_t^2 \\
&= \sum_{i=1}^{N} \left(g_{m,i} w_{q(\mathbf{x}_i)} + \frac{1}{2} h_{m,i} w_{q(\mathbf{x}_i)}^{\ 2} \right) + \gamma T + \frac{1}{2}\lambda \sum_{t=1}^{T} w_t^2 \\
&= \sum_{t=1}^{T} \left[\sum_{i\in I_t} g_{m,i}\, \mathrm{w}_t + \frac{1}{2} \sum_{i\in I_t} h_{m,i}\, \mathrm{w}_t^{\ 2} \right] + \frac{1}{2}\lambda \sum_{t=1}^{T} w_t^2 + \gamma T \\
&= \sum_{t=1}^{T} \left[\underbrace{\sum_{i\in I_t} g_{m,i}}_{G_t} \mathrm{w}_t + \frac{1}{2} \left(\underbrace{\sum_{i\in I_t} h_{m,i}}_{H_t} + \lambda \right) w_t^2 \right] + \gamma T \\
&= \sum_{t=1}^{T} \left[G_t \mathrm{w}_t + \frac{1}{2} (H_t + \lambda) w_t^2 \right] + \gamma T
\end{aligned}
$$

T个独立的二次函数之和

假设我们已经知道树的结构q，

$$J(f)=\sum_{t=1}^{T}\left[G_t w_t+\frac{1}{2}(H_t+\lambda)w_t^2\right]+\gamma T$$

则由$\frac{\partial J(f)}{\partial w_t}=G_t+(H_t+\lambda)w_t=0$

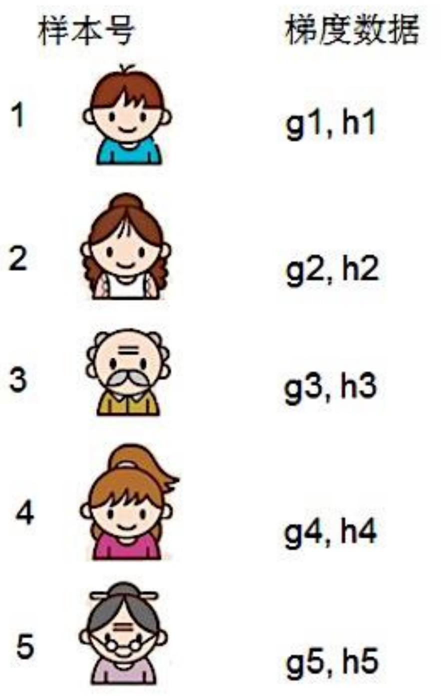
得到最佳的模型参数**w**：$w_t=-\frac{G_t}{H_t+\lambda}$

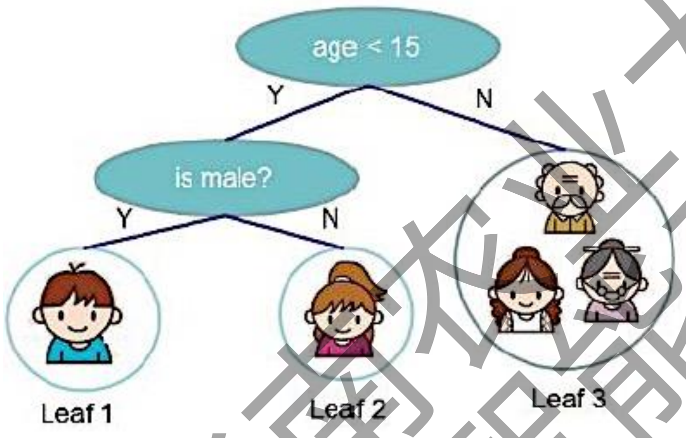
以及最佳的**w**对应的目标函数：

$$J(f)=-\frac{1}{2}\sum_{t=1}^{T}\left[\frac{G_t^2}{H_t+\lambda}\right]+\gamma T$$

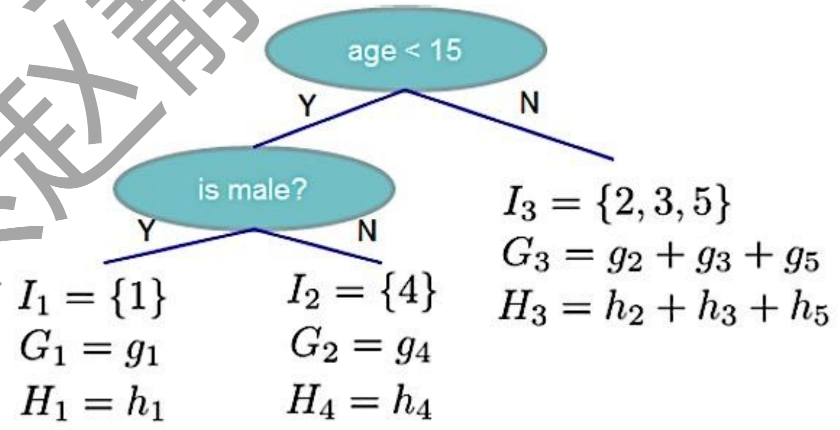
分数越小的树越好！

例：树的分数





$$J = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j{}^2}{H_j+\lambda} + \gamma T$$



$$J = -\frac{1}{2}\sum_{j=1}^{T}\frac{G_j{}^2}{H_j+\lambda} + 3\gamma$$

越小，代表这个树的结构越好

➢ 回归树的学习策略

✓ 贪心算法：每次尝试分裂一个叶节点，计算分裂前后的增益，选择增益最大的分裂。

✓ 分裂的增益度量：

- ID3算法采用信息增益
- C4.5算法采用信息增益比
- CART采用Gini系数
- XGBoost： $J(f)=-\frac{1}{2}\sum_{t=1}^{T}\left[\frac{G_t^2}{H_t+\lambda}\right]+\gamma T$

$$J(f) = -\frac{1}{2}\sum_{t=1}^{T}\left[\frac{G_t^2}{H_t+\lambda}\right] + \gamma T$$

(1) 从深度为0的树开始

(2) 对于树的每个叶子节点，尝试增加一个分裂点：

令I_L和I_R分别表示加入分裂点后左右叶子结点的样本集合，则该分裂的增益为增加分裂点后目标函数的变化：

$$Gain = gain(before) - gain(after)$$

$$= \frac{1}{2}\left(\underbrace{\frac{G_L^2}{H_L+\lambda}}_{\text{分裂后左子树分数}} + \underbrace{\frac{G_R^2}{H_R+\lambda}}_{\text{分裂后右子树分数}} - \underbrace{\frac{G_L^2+G_R^2}{H_L+H_R+\lambda}}_{\text{分裂前的分数}}\right) - \gamma$$

- Bagging
  - Bagging
  - 随机森林
- Boosting
  - AdaBoost
  - GBDT
  - XGBoost