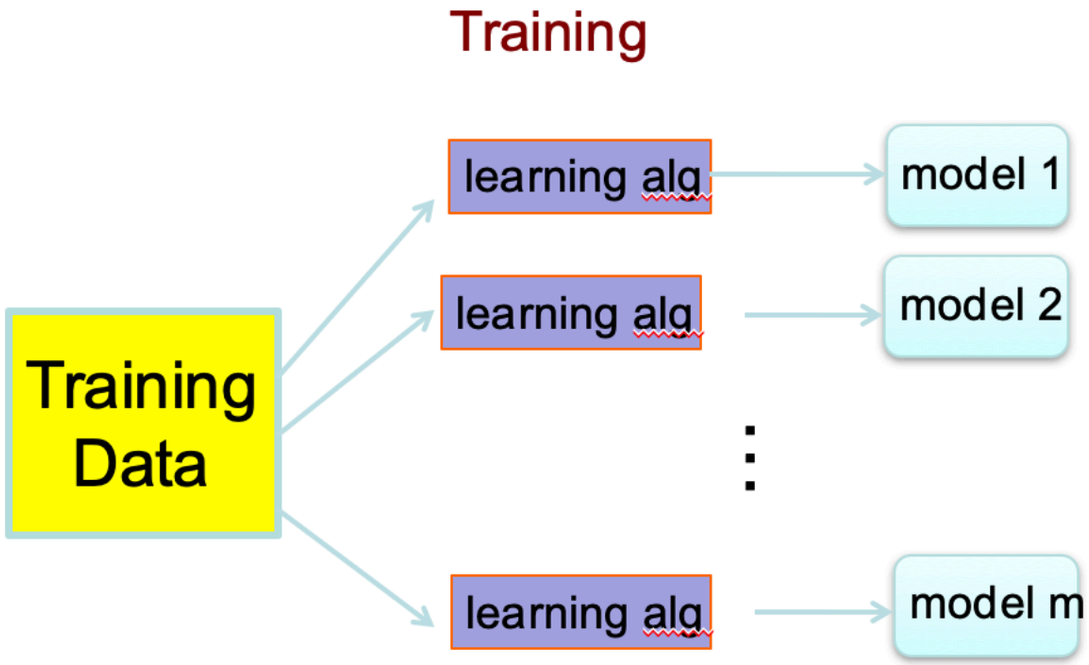
集成学习

Ensemble Learning

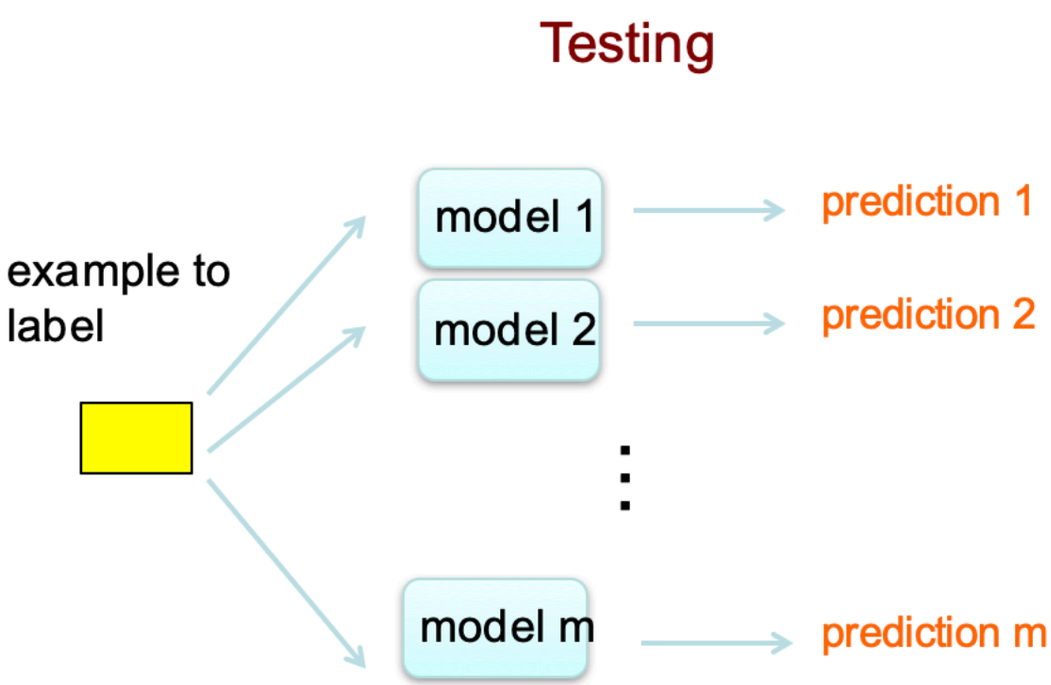
集成学习

- 最基本的思想：如果一个分类器工作的还不错，那么为什么不使用多个分类器一起解决问题呢



集成学习

- 对多个分类器的结果进行投票，获得最终结果



集成学习

- 考虑一个例子：三个不同分类器，在三个样本上的表现

	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

- 集成的个体：应该是好而不同

集成学习

- 考虑一个二分类器 $y \in \{+1, -1\}$ 和真实函数f，假定基分类器的错误率为e，对于每个基分类器hi(x)有：

$$P(h_i(x) \neq f(x)) = e$$

- 假定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

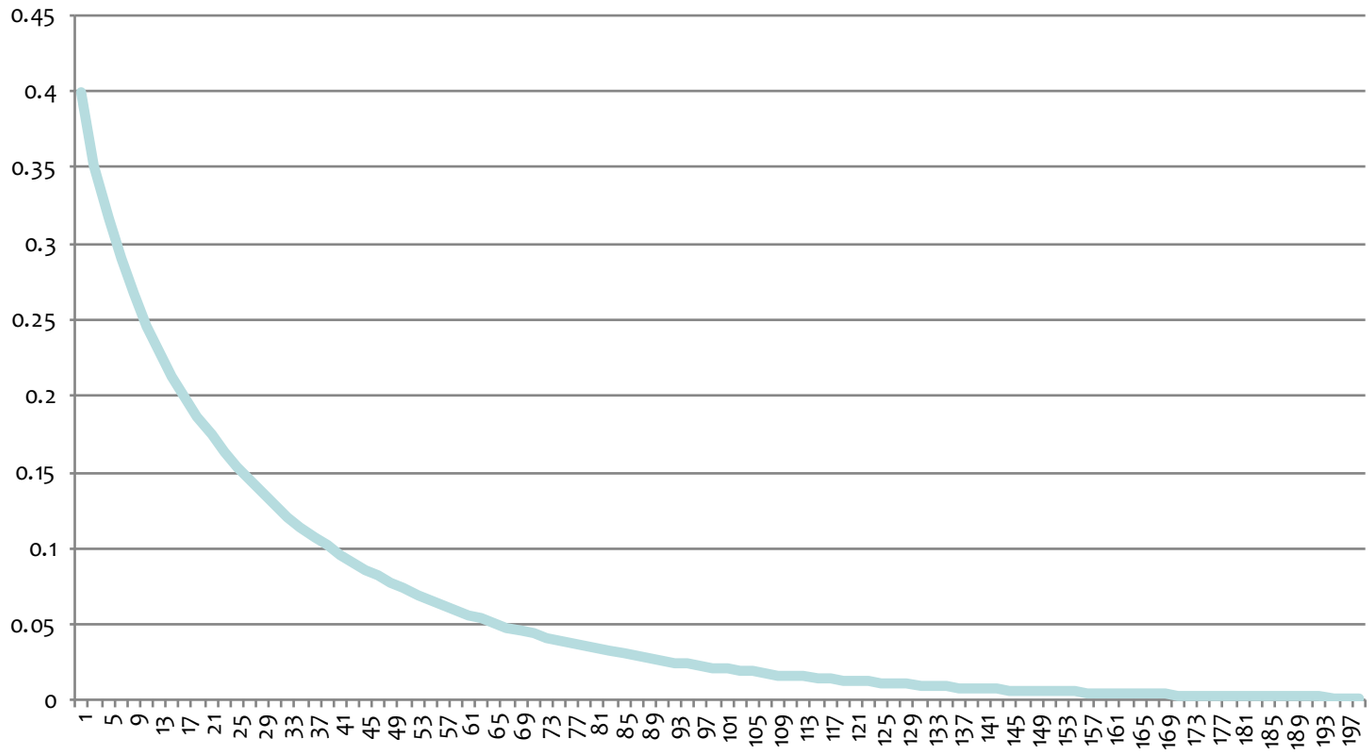
$$H(x) = sign(\sum_{i=1}^{T} h_i(x))$$

- 假定基分类器错误率相互独立，由霍丁夫不等式克制，集成的错误率为

$$P(h_i(x) \neq f(x)) = \sum_{k=1}^{|T/2|} \binom{T}{k} (1-e)^k e^{T-k} \leqslant exp(-\frac{1}{2}T(1-2e)^2)$$

集成学习

- 随着个体分类器的增多，集成的错误率将指数级下降，最终趋近于0

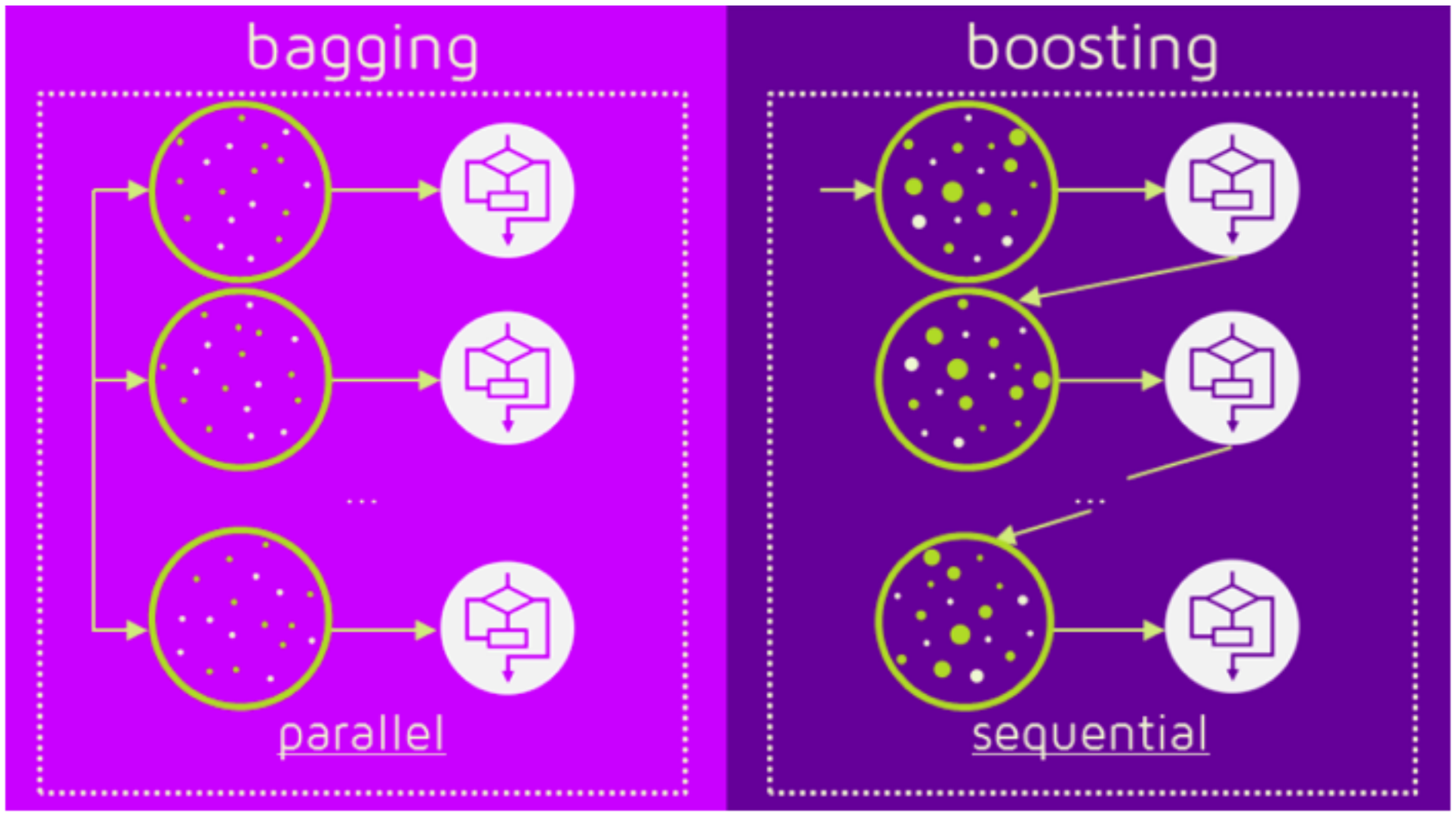


集成学习

- 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

- 主要分两种



Bagging

bagging

- bagging的策略
  - 从样本中重复采样（有放回）选出n个样本
  - 对这n个样本建立分类器
  - 重复以上两步知道m次，生成m个分类器
  - 预测数据时，通过m个分类器投票选举进行分类
- bagging的代表是随机森林(Random Forest)

随机森林

- 从样本数据集中有放回的采样出n个样本
- 从所有属性中随机选择k个属性。使用n个样本与随机选择的k个样本构建决策树
- 重复上述两步m次，生成m棵决策树
- 预测时，采用投票的方式决定将数据分为哪一类

随机森林

- k的参考值log2(d)
- 优点
    - 在很多数据集上，他的表现都优于其他模型
    - 他可以处理很高维度的数据
        - 因为特征是随机选择的
    - 随机森林算法可以给出哪些特征比较重要
    - 因为每棵树之间时独立的，所以可以并行训练
    - 对于不平衡数据来说，可以平衡误差

随机森林-属性的重要度

- 袋外误差（Out of Bag，OOB）：每次随机从训练集中选择n个样本用来训练，剩下的数据用来做测试，产生的误差叫做袋外误差
- 训练完一棵决策树后，计算袋外误差err_x_oob1
  - 随机对于特征X加入噪声后，在计算袋外误差err_x_oob2
  - 有m棵树，则X的重要度为 $\frac{\sum(err\_x\_oob2 - err\_x\_oob1)}{m}$ 如果说加入噪声后，误差明显变大后，说明该特征很重要

随机森林

```
class sklearn.ensemble.RandomForestClassifier(
    n_estimators=10, # 生成树的数目
    crite-rion='gini', # "gini" or "entropy"(default="gini")。使用gini还是entropy，来选择最合适的节点。
    max_depth=None,  # 每棵树的最大深度。None的时候为每个叶子结点只有一个类别，或者达到min_samples_split条件
    min_samples_split=2, # 每个划分最少的样本数
    min_samples_leaf=1,  # 叶子结点最少的样本数
    max_features='auto', # 选择特征的数目
                         # None: n_features (所有特征)
                         # auto: sqrt(n_features)
                         # sqrt: sqrt(n_features)
                         # log2: log2(n_features)
    max_leaf_nodes=None, # 最大叶子数
    bootstrap=True, # 是否使用bootstrap方式采样，如果不是则使用全部数据集
    oob_score=False, # 是否使用oob_score来评估
    )
```

实践-使用随机森林对人脸数据进行分类

Boosting

boosting

- 能够将预测精度仅比随机猜度略高的弱学习器增强为预测精度高的强学习器

- 代表算法是adaboost与gdbt

adaboost

- adaboost是Adaptive Boosting的缩写，意思为自适应增强
- 算法流程：

1. 初始化训练数据的权重D1，有N个训练样本，则每个训练样本被赋予1/N的权重
2. 训练若分类器hi，如果训练样本中能被hi分类正确的数据，则减小他们的权重，如果分错则增大他的权重，更新权重，用于下一次分类
3. 全部弱分类器训练完成后，对每个弱分类器同样赋予权重，使错误率低的模型的权重高，错误率低的模型权重低

adaboost算法流程

- 给定训练集 $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\},\ y \in \{-1,1\},\ i = 1,2,...,N$
- 初始化训练集样本的权值

$$D_1(i) = (w_1, w_2, \ldots, w_N) = (\frac{1}{N}, \frac{1}{N}, \ldots, \frac{1}{N})$$

- 进行T次迭代

1. 选取当前误差最低的分类器h作为当前迭代的分类器，计算分类器h在当前分布D上的误差

$$e_t = P(h_t(x_i) \neq y_i) = \sum_{i=1}^{N} w_{ti} I(h_t(x_i) \neq y_i)$$

其实就是没有分对权重的合

adaboost算法流程

2. 计算该弱分类器在最终分类器中所占的权重 $\alpha_t = \frac{1}{2} ln(\frac{1-e_t}{e_t})$

3. 更新训练样本的权值分布Dt+1, Z为归一化常数

$$D_{t+1} = \frac{D_t(i) exp(-\alpha_t y_i h_t(x_i))}{Z_t}, \; Z_t = 2\sqrt{e_t(1-e_t)}$$

4. 最终按照弱分类器的权重，组合各个弱分类器

$$h(x) = sign(f(x)) = sing(\sum_{t=1}^{T} \alpha_t h_t(x))$$

GBDT

- GBDT 是gradient boosting decision tree的缩写，梯度提升数

- Facebook使用GBDT来提高CTR（Click-through Ratio Prediction）

    - 预估《Practical Lessons from Predicting Clicks on Ads at Facebook》

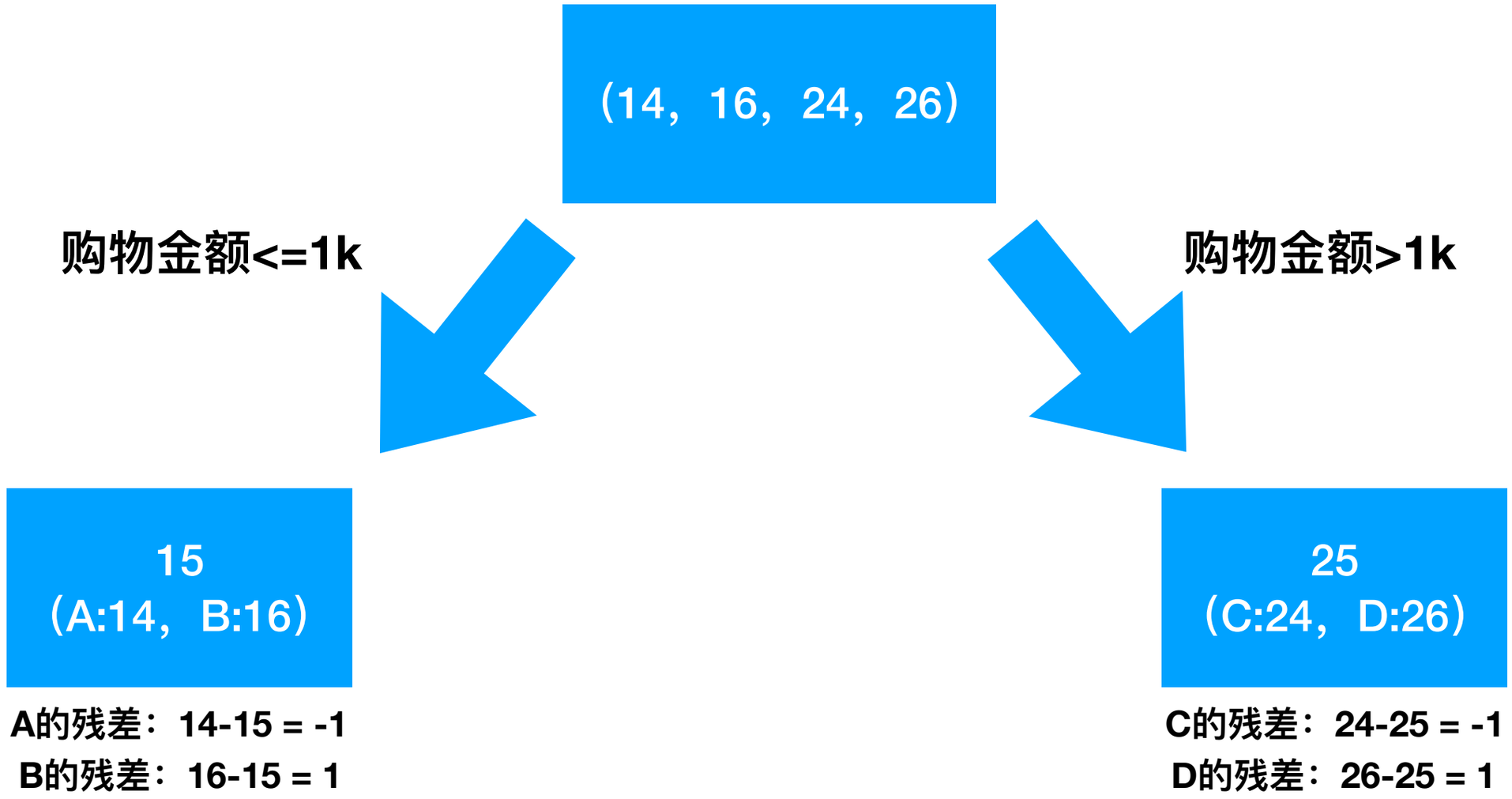
- 与RandomForest同样广泛被使用

GBDT

- GBDT核心
  - 每一棵树学的是之前所有树结论和的残差，这个残差就是一个加预测值后能得真实值的累加量
- 举例：我们要对年龄进行预测，训练集中有A，B，C，D四个人，年龄分别为14，16，24，26

GBDT

- 首先限定每棵子树的深度为2，创建第一棵子树

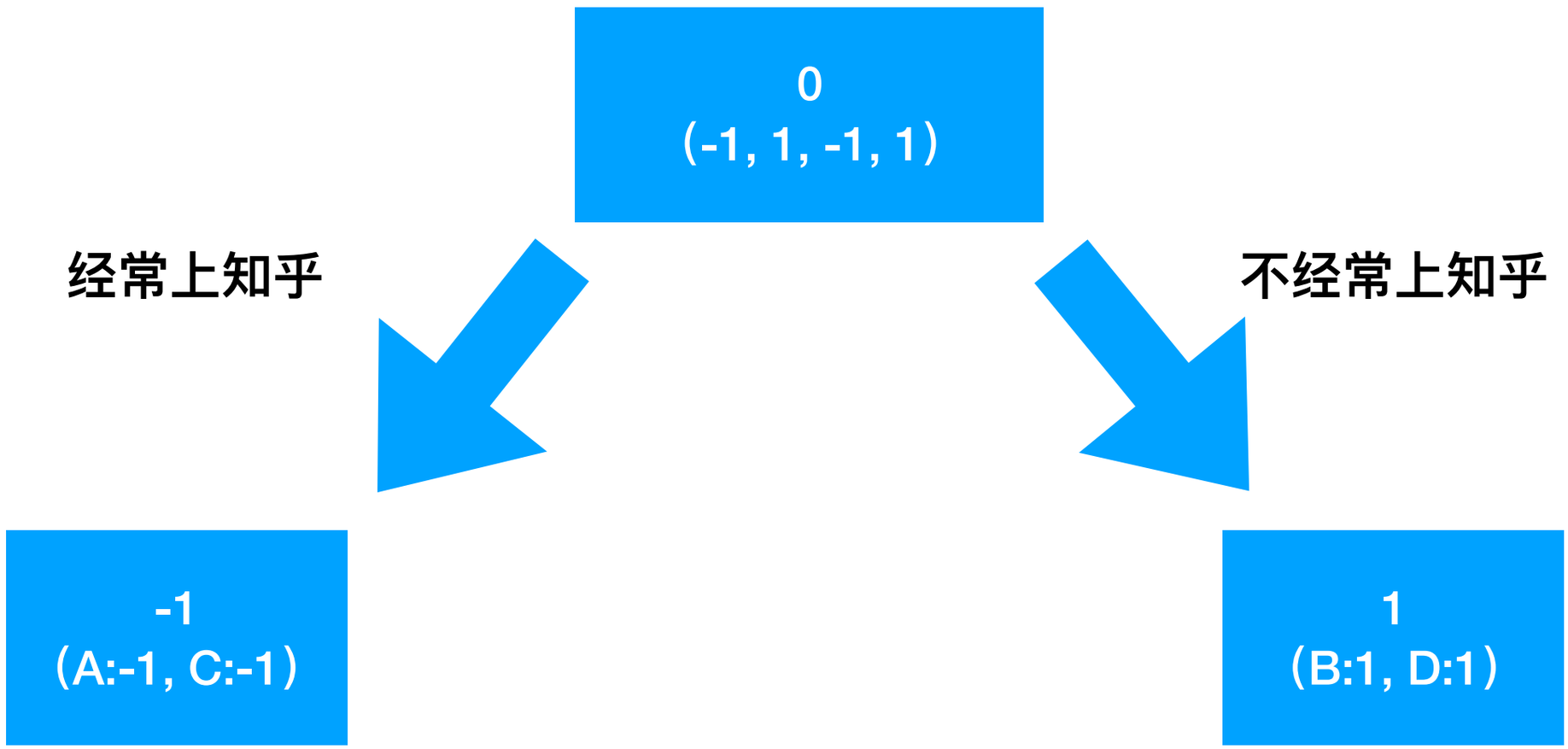


GBDT

- 计算残差：残差=预测值-真实值
  - A的残差 = 14 - 15 = -1
  - B的残差 = 16 - 15 = 1
  - C的残差 = 24 - 25 = -1
  - D的残差 = 26 - 25 = 1

GBDT

- 根据残差建立第二棵树



GBDT

- 推断：年龄 = 第一棵树的结果 + 第二棵树的结果
    - A的年龄 = 15 - 1 = 14
    - B的年龄 = 15 + 1 = 16
    - C的年龄 = 25 - 1 = 24
    - D的年龄 = 25 + 1 = 26