决策树与集成方法

主讲人： 林倞

决策树课程大纲

- 介绍
- 选择扩展属性的准则
- 决策树学习

什么是决策树？

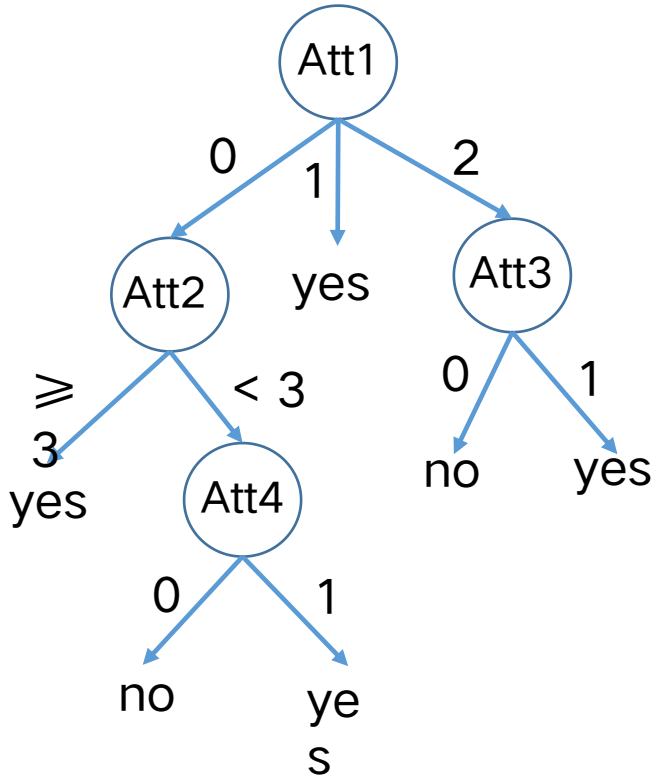
- 给定下面的数据集，我们希望基于属性来预测其结果

属性　　结果

	Type	Length	Director	Famous actors	Liked?
1	Comedy	Short	Adamson	No	Yes
2	Animated	Short	Lasseter	No	No
3	Drama	Medium	Adamson	No	Yes
4	Animated	Long	Lasseter	Yes	No
5	Comedy	Long	Lasseter	Yes	No
6	Drama	Medium	Singer	Yes	Yes
7	Animated	Short	Singer	No	Yes
8	Comedy	Long	Adamson	Yes	Yes
9	Drama	Medium	Lasseter	No	Yes

- 与之前的分类器不同，决策树通过构建属性树将数据分到不同的类别中

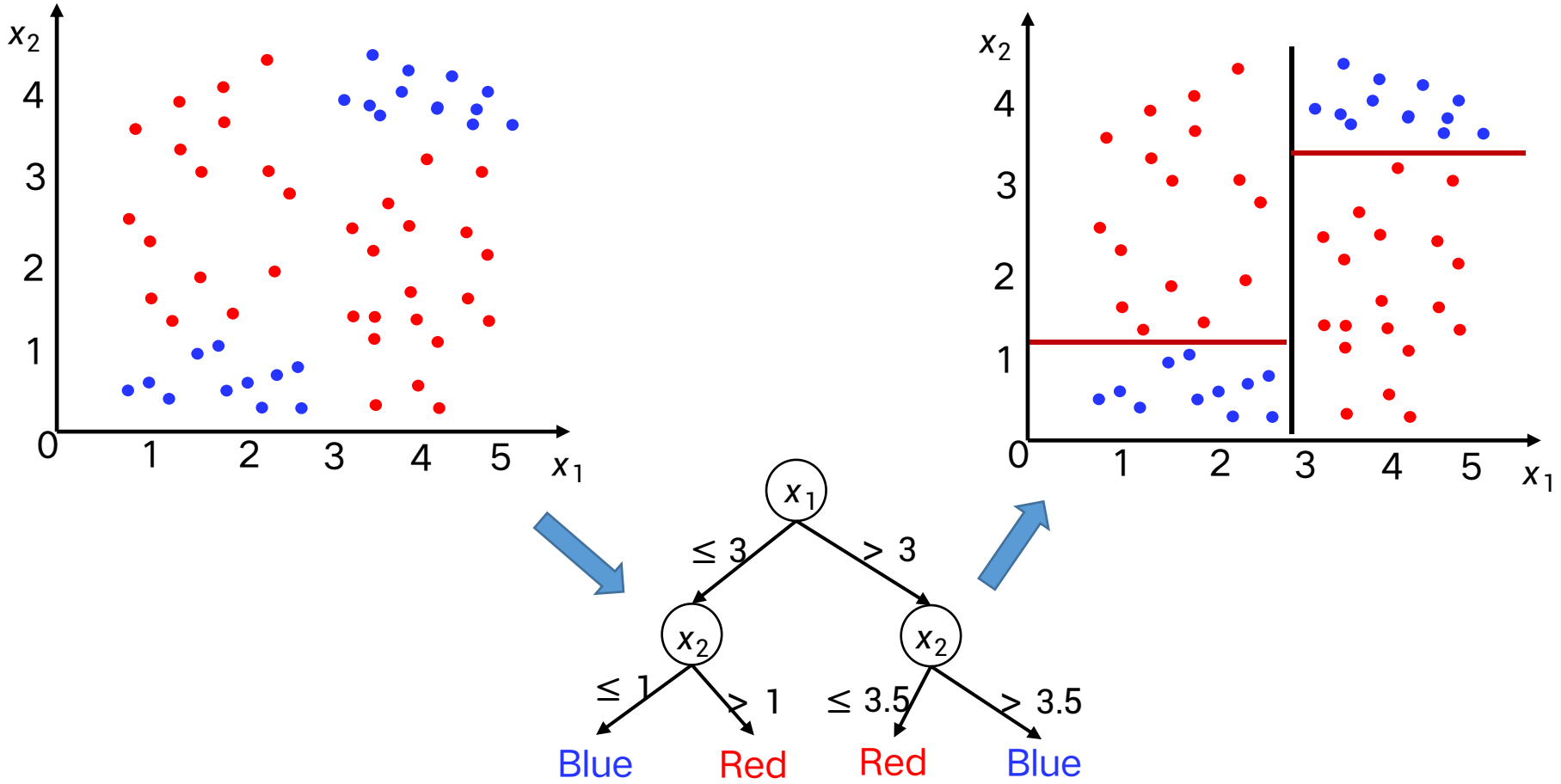
- 决策树（DT） 的结构
  - 内部节点对应于属性
  - 叶子节点对应于结果
  - 边对应于属性值



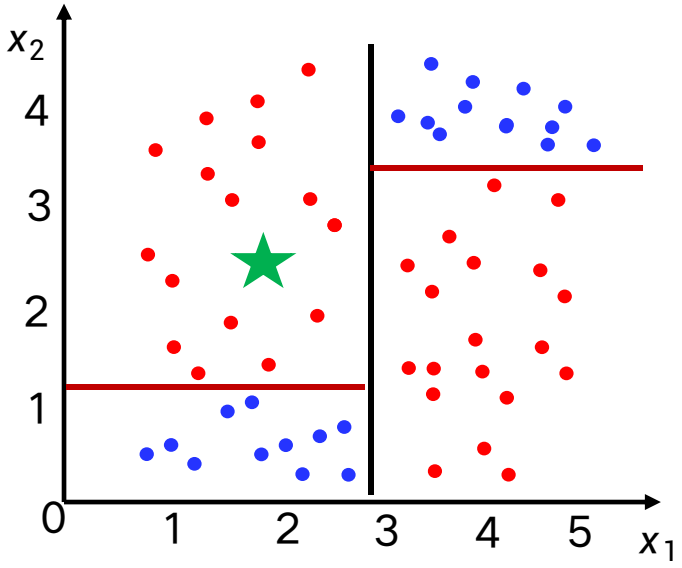
- 学习决策树
  - 在每个节点使用哪个属性？
  - 如何划分属性值？
  - 何时停止扩展树？

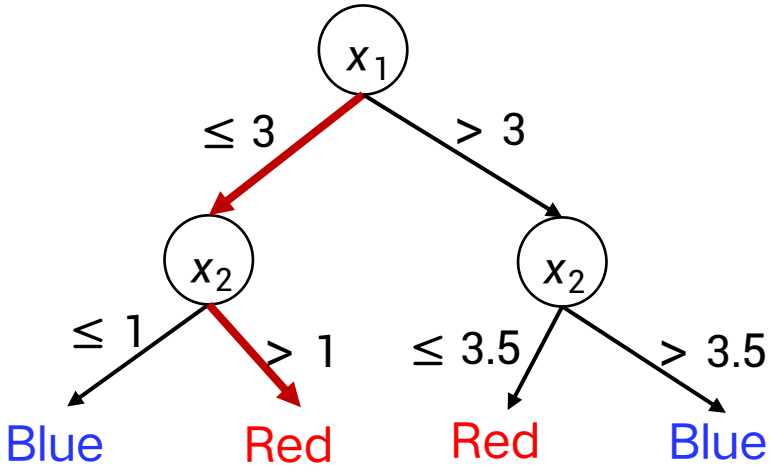
决策树学习：示例1

- 给定一个包含两个属性 x_1 和 x_2 的数据集，如下图所示，我们希望构建一个决策树来对实例进行分类



- 根据这个例子，为了构建这棵树，我们需要决定：
  - 在每一步使用哪个属性
  - 如何划分属性值
- 当树已经构建好之后，对于一个新的输入数据，可以根据其属性值找到一条从根节点到叶子节点的路径，并从中读取其类别

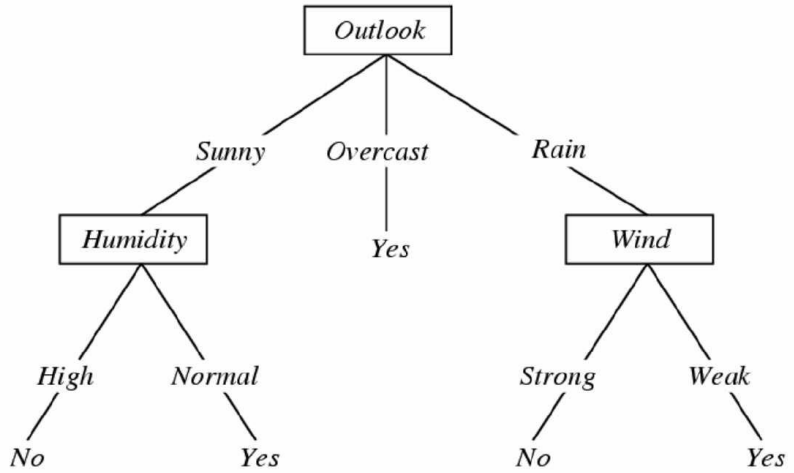




决策树学习：示例2

- 给定下面的数据集，构建一个决策树，根据四个属性值来决定是否去打球

day	outlook	temperature	humidity	wind	play
1	sunny	hot	high	weak	no
2	sunny	hot	high	strong	no
3	overcast	hot	high	weak	yes
4	rain	mild	high	weak	yes
5	rain	cool	normal	weak	yes
6	rain	cool	normal	strong	no
7	overcast	cool	normal	strong	yes
8	sunny	mild	high	weak	no
9	sunny	cool	normal	weak	yes
10	rain	mild	normal	weak	yes
11	sunny	mild	normal	strong	yes
12	overcast	mild	high	strong	yes
13	overcast	hot	normal	weak	yes
14	rain	mild	high	strong	no



- *问题*
  a) 为什么首先选择 Outlook，然后是 Humidity 和 Wind？
  b) 为什么在 Humidity 和 Wind 之后不再进一步扩展？
  c) 为什么 Temperature 属性没有出现在树中？

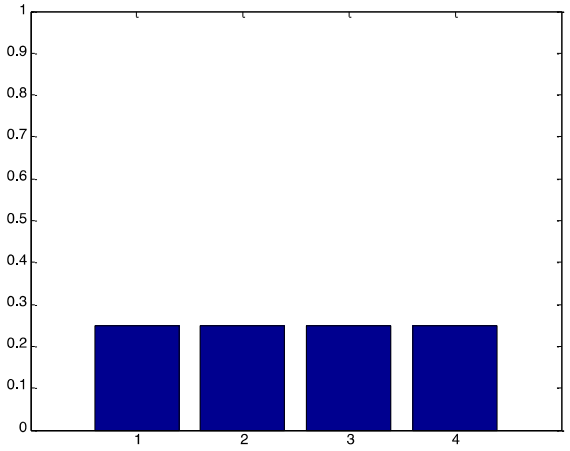
课程大纲

- 介绍
- 选择扩展属性的准则
- 决策树学习

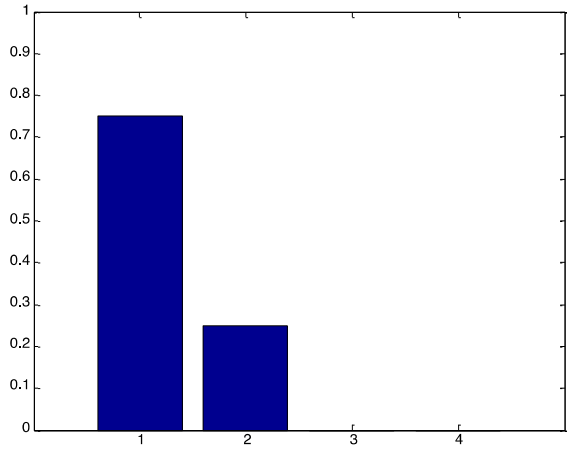
熵

- 在所有属性中，选择包含*信息量最多*的那个来扩展
- 如何衡量一个属性所包含的信息量？

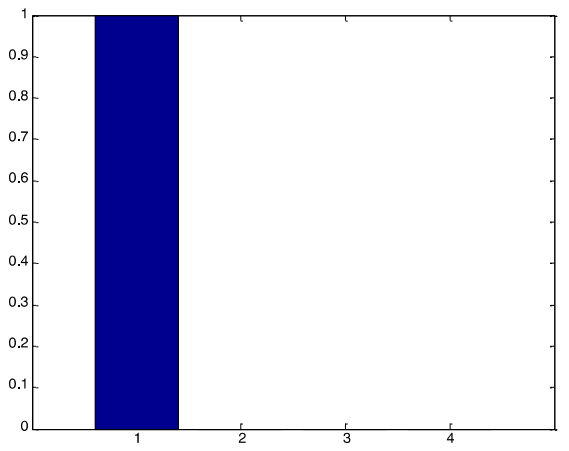
我们首先需要衡量一个随机变量的*不确定性*



(a)



(b)



(c)

- 在数学中，一个随机变量的不确定性是通过*熵* 来衡量的

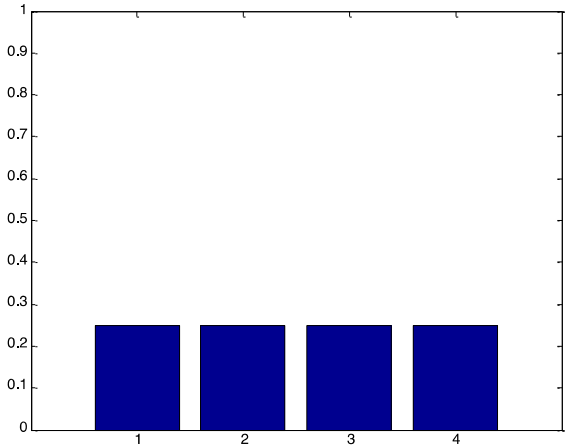
- 定义：给定一个服从分布 $p(z)$ 的随机变量 Z，其熵被定义为：

$$H(Z) = -\sum_{z \in \mathbb{C}} p(z) \log_2 p(z)$$

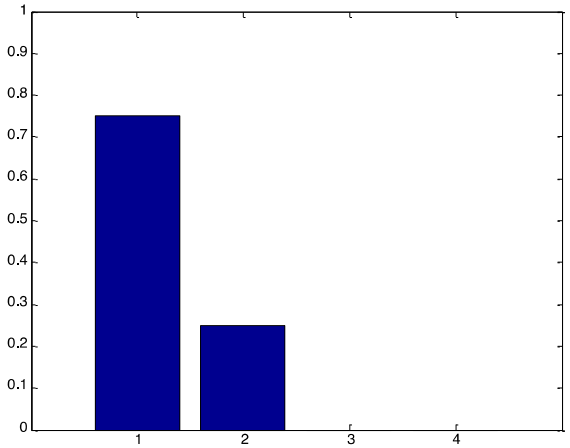
  - $\mathbb{C}$ 是随机变量 Z 的所有可能取值的集合

- 示例

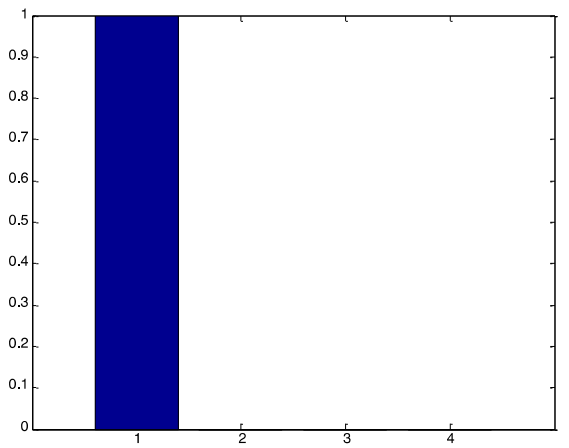
下面哪个分布的熵最大？



(a)



(b)



(c)

a) $H(Z) = -4 \times 0.25 \log_2 0.25 = 2$ bits

b) $H(Z) = -0.75 \log_2 0.75 - 0.25 \log_2 0.25 \approx 0.8133$ bits

c) $H(Z) = -1 \log_2 1 = 0$ bits

分布 a) 的熵最大，而 c) 的熵最小

- 示例：二分类问题（伯努利熵）
- 伯努利分布：

$$P(Z=1)=p;\ P(Z=0)=1-p$$

- 根据熵的定义：

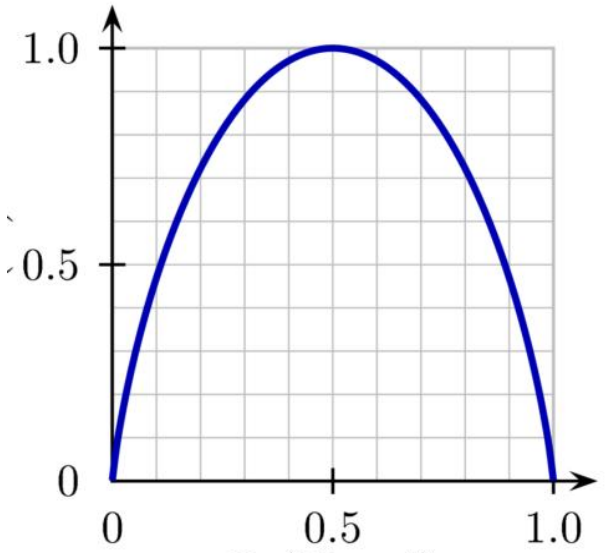
$$H(Z)=-\sum_{z\in\mathbb{C}} p(z)\log_2 p(z)$$

- 伯努利熵：

$$H(Z)=-\left[p\log_2 p+(1-p)\log_2(1-p)\right]$$

- 伯努利随机变量的熵作为概率 $P(Z=1)$ 的函数

$$H(Z) = -[p\log_2 p + (1-p)\log_2(1-p)]$$



当 $p=0.5$ 时，熵最大，不确定性最大；当 $p=0$ 或者 $p=1$ 时，不确定性为0

- 熵与我们的直觉是一致的，即：

分布越平坦，不确定性就越大

条件熵

- 条件熵 $H(Z|Y)$：在已知随机变量 Y 的值后，随机变量 Z 的熵

$$H(Z|Y=y) = -\sum_{z\in\mathbb{C}} p(z|y)\log p(z|y)$$

$$H(Z|Y) = \sum_{y\in\mathbb{T}} P(Y=y)H(Z|Y=y)$$

Y	Z
t	t
t	t
t	t
t	t
t	t
f	t
f	f

  - 示例

$$p(Z=t|Y=t)=1 \text{ 且 } p(Z=f|Y=t)=0 \quad \Longrightarrow \quad H(Z|Y=t)=0$$

$$p(Z=t|Y=f)=0.5 \text{ 且 } p(Z=f|Y=f)=0.5 \quad \Longrightarrow \quad H(Z|Y=f)=1$$

$$p(Y=t)=4/6 \text{ 且 } p(Y=f)=2/6$$

$$\Longrightarrow \quad H(Z|Y)=\frac{4}{6}\times 0 + \frac{2}{6}\times 1 = \frac{2}{6}$$

- 条件熵 $H(Z|Y)$ 与熵 $H(Z)$ 是不同的

对于给定的例子，可以看出：

$$H(Z) = -p(z=t)\log p(z=t) - p(z=f)\log p(z=f)$$

$$= -\frac{5}{6}\log_2\frac{5}{6} - \frac{1}{6}\log_2\frac{1}{6}$$

$$= -\frac{5}{6}\log_2\frac{5}{6} - \frac{1}{6}\log_2\frac{1}{6}$$

$$\approx 0.65$$

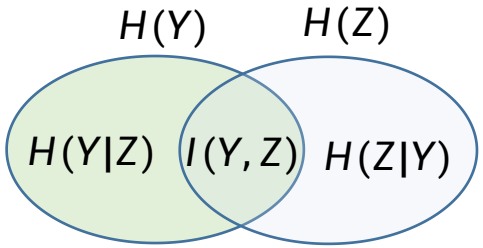
显然，它大于条件熵 $H(Z|Y) \approx 0.33$

实际上，不等式 $H(Z) \geq H(Z|Y)$ 总是成立的

信息增益

- 随机变量 Y 的信息增益是在已知其值之后，随机变量 Z 的*熵减少的量*

$$IG(Y) = H(Z) - H(Z|Y)$$



$$IG(Y) = I(Y, Z)$$

- 对于上面给出的例子，随机变量 Y 的信息增益为：

$$IG(Y) = 0.65 - 0.33 = 0.32$$

- Y 的信息增益意味着，如果已知其值，则*平均可以减少的不确定性*

课程大纲

- 介绍
- 选择扩展属性的准则
- 决策树学习

选择根节点

- 结果变量“Liked?”的*熵*

$$P(Like = yes) = 2/3 \text{ 且 } P(Like = no) = 1/3 \quad \Longrightarrow \quad H(Like) = 0.91$$

- 在给定属性（Type, Length, Director 和 Actors）后，结果的*条件熵*

$$H(Like|Type) = 0.61$$

$$H(Like|Length) = 0.61$$

$$H(Like|Director) = 0.36$$

$$H(Like|Actor) = 0.85$$

	Type	Length	Director	Famous actors	Liked?
1	Comedy	Short	Adamson	No	Yes
2	Animated	Short	Lasseter	No	No
3	Drama	Medium	Adamson	No	Yes
4	Animated	Long	Lasseter	Yes	No
5	Comedy	Long	Lasseter	Yes	No
6	Drama	Medium	Singer	Yes	Yes
7	Animated	Short	Singer	No	Yes
8	Comedy	Long	Adamson	Yes	Yes
9	Drama	Medium	Lasseter	No	Yes

- 信息增益

$$IG(Type) = H(Like) - H(Like|Type) = 0.3$$

$$IG(Length) = H(Like) - H(Like|Length) = 0.3$$

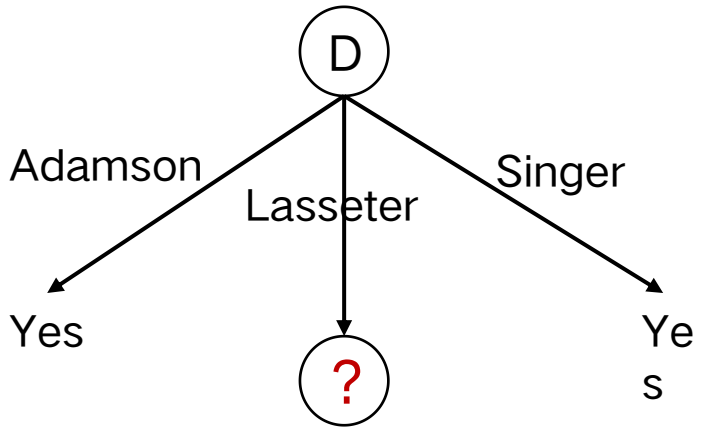
$$IG(Actor) = H(Like) - H(Like|Actor) = 0.06$$

$$IG(Director) = H(Like) - H(Like|Director) = 0.55$$

$\Rightarrow$ Director 应该作为根节点

	Type	Length	Director	Famous actors	Liked?
1	Comedy	Short	Adamson	No	Yes
2	Animated	Short	Lasseter	No	No
3	Drama	Medium	Adamson	No	Yes
4	Animated	Long	Lasseter	Yes	No
5	Comedy	Long	Lasseter	Yes	No
6	Drama	Medium	Singer	Yes	Yes
7	Animated	Short	Singer	No	Yes
8	Comedy	Long	Adamson	Yes	Yes
9	Drama	Medium	Lasseter	No	Yes

- 构建树



	Type	Length	Director	Famous actors	Liked?
1	Comedy	Short	Adamson	No	Yes
2	Animated	Short	Lasseter	No	No
3	Drama	Medium	Adamson	No	Yes
4	Animated	Long	Lasseter	Yes	No
5	Comedy	Long	Lasseter	Yes	No
6	Drama	Medium	Singer	Yes	Yes
7	Animated	Short	Singer	No	Yes
8	Comedy	Long	Adamson	Yes	Yes
9	Drama	Medium	Lasseter	No	Yes

- 由于来自“Adamson”和“Singer”分支的所有结果都是“Yes”，因此我们无需进一步扩展这两个分支
- 问题是如何为“Lasseter”分支选择属性

继续扩展

- 选择导演“Lasseter”后，剩余的数据是

	Type	Length	Director	Famous actors	Liked?
2	Animated	Short	Lasseter	No	No
4	Animated	Long	Lasseter	Yes	No
5	Comedy	Long	Lasseter	Yes	No
9	Drama	Medium	Lasseter	No	Yes

- 重新计算熵和条件熵得到：

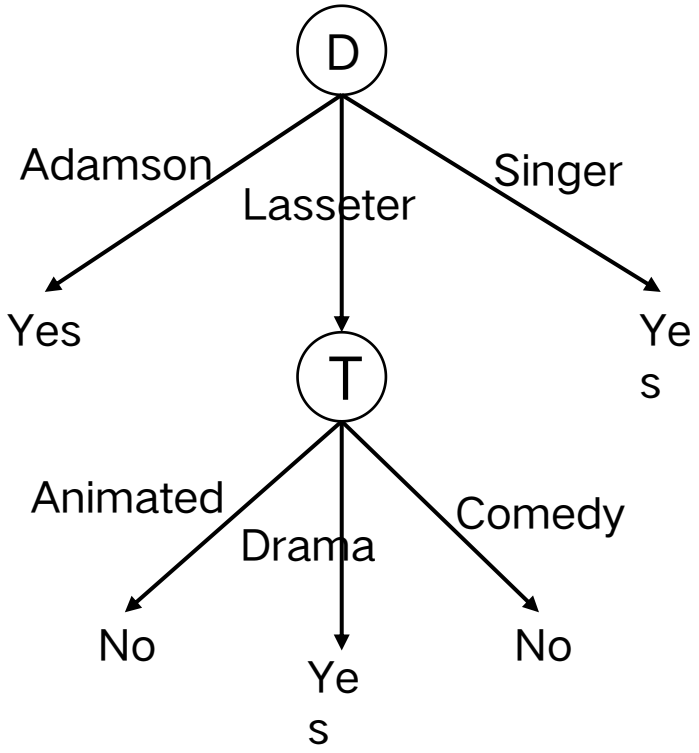
$$H(Like) = 0.81 \quad H(Like|Type) = 0 \quad H(Like|Length) = 0 \quad H(Like|Actor) = 0.5$$

- 因此，信息增益为

$$IG(Type) = 0.81 \quad IG(Length) = 0.81 \quad IG(Actor) = 0.31$$

因此，我们应该选择属性“Type”或“Length”来扩展

- 构建树



这是最终的决策树！！

	Type	Length	Director	Famous actors	Liked?
2	Animated	Short	Lasseter	No	No
4	Animated	Long	Lasseter	Yes	No
5	Comedy	Long	Lasseter	Yes	No
9	Drama	Medium	Lasseter	No	Yes

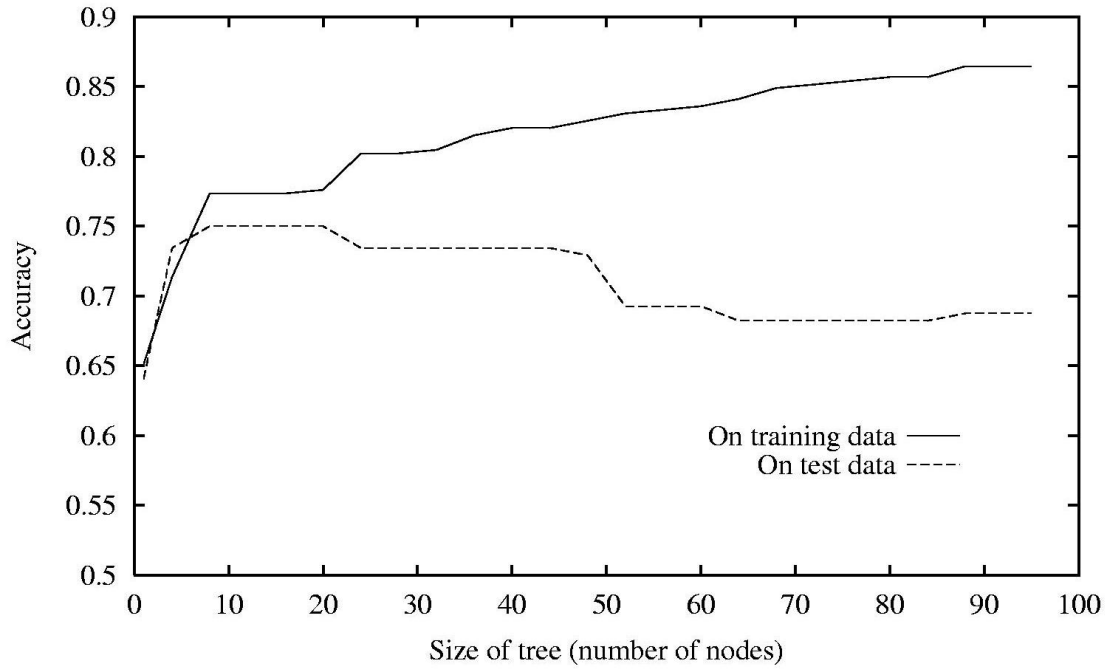
- 我们停止进一步扩展这棵树，因为在训练数据中，对于每条从根节点到叶节点的路径，都只有一种可能的结果

停止扩展的准则

- 树不能无限扩展，应该在某个点停止。以下是阐述的一些停止准则：
  - 所有剩余的实例都具有相同的标签
  - 我们用完了所有的属性
  - 树的深度达到最大限制
  - 信息增益小于某个阈值
  - ……。

过拟合问题

- 如果树太大或过于复杂，它在训练数据上会表现得很好，但在测试数据上表现可能很差

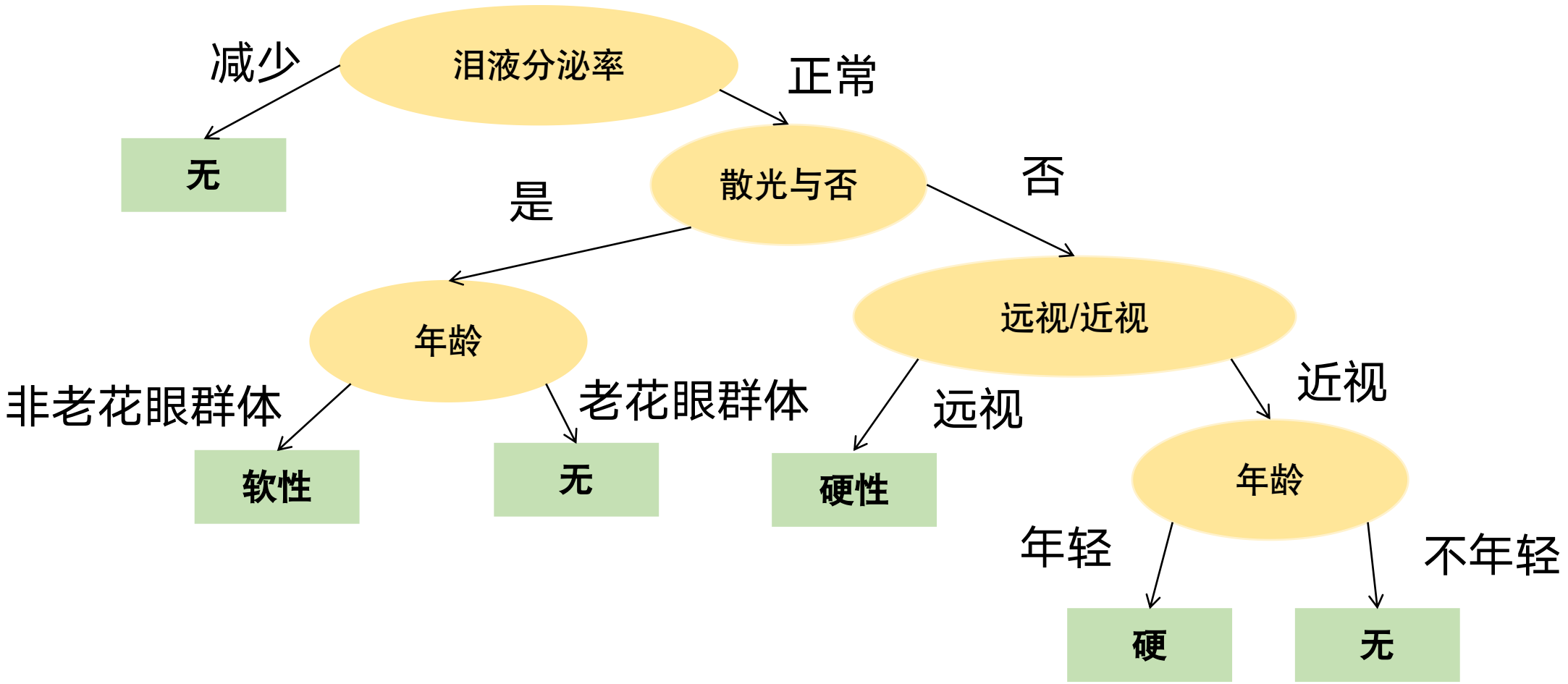


树剪枝

- 为了控制决策树的复杂度，我们可以：
  - 在学习树时进行剪枝
  - 在学习树后进行剪枝
- 基于一个验证数据集，我们可以：
  - 剪掉那些不损害验证集准确率的节点
  - 贪婪地移除对验证准确率提升最小的节点
  - 当验证集准确率开始下降时停止

树剪枝

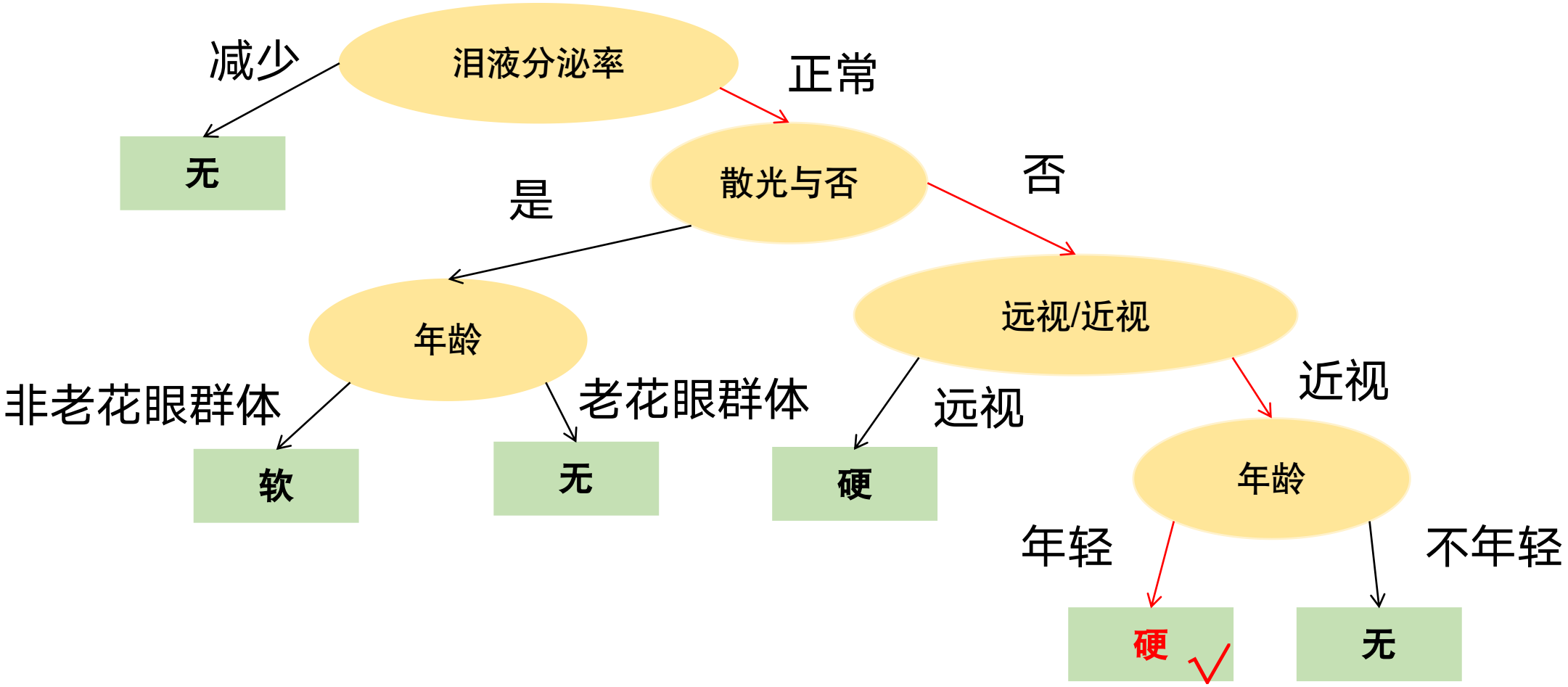
- 以下是一个例子，我们构建了一颗确定一个人应该戴何种隐形眼镜的决策树：



树剪枝

- 接下来，我们用这颗树对一个实例进行评估：

小明	泪液分泌正常	不散光	近视眼	年轻群体

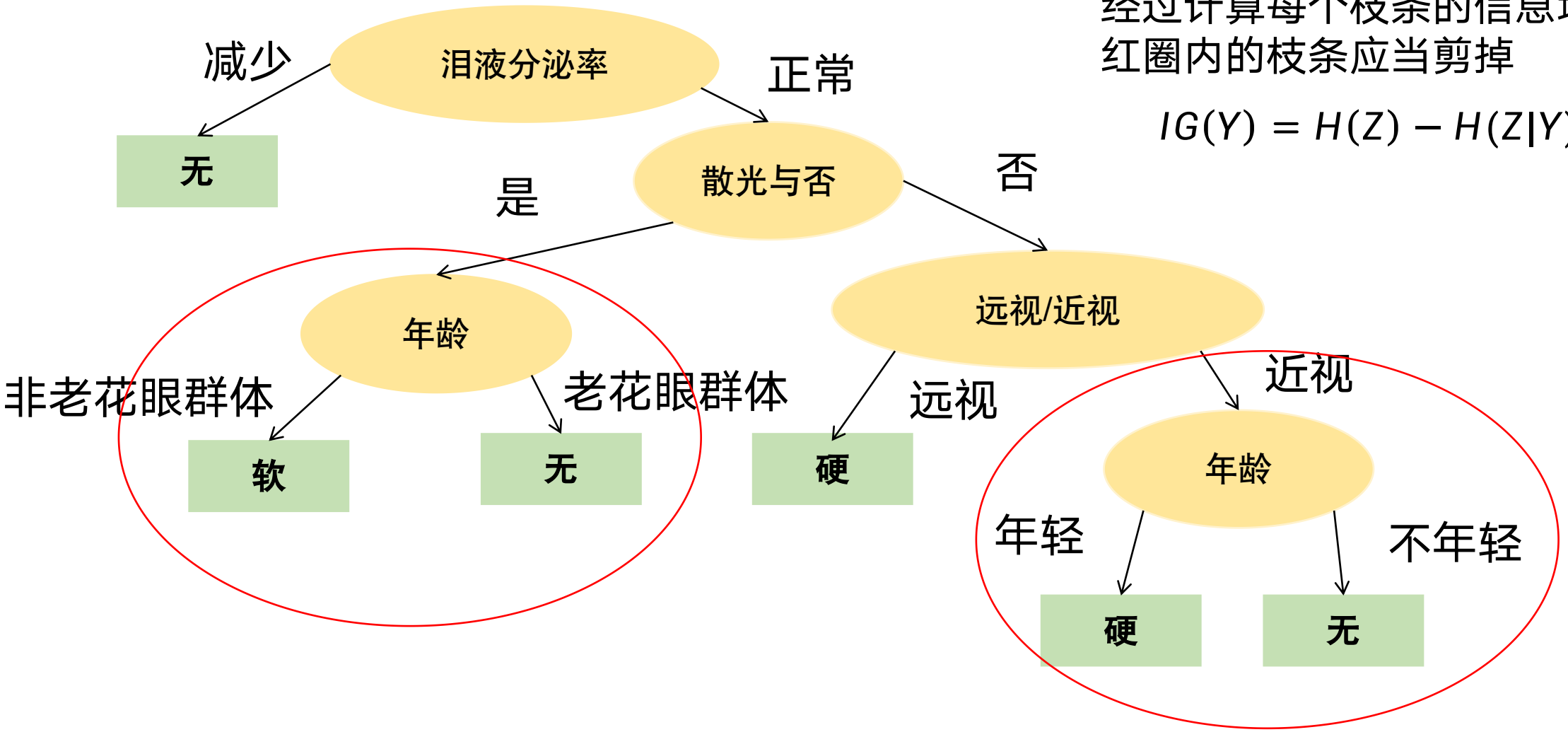


树剪枝

- 这棵树被分得太细，有过拟合风险，我们使用基于信息增益的剪枝：

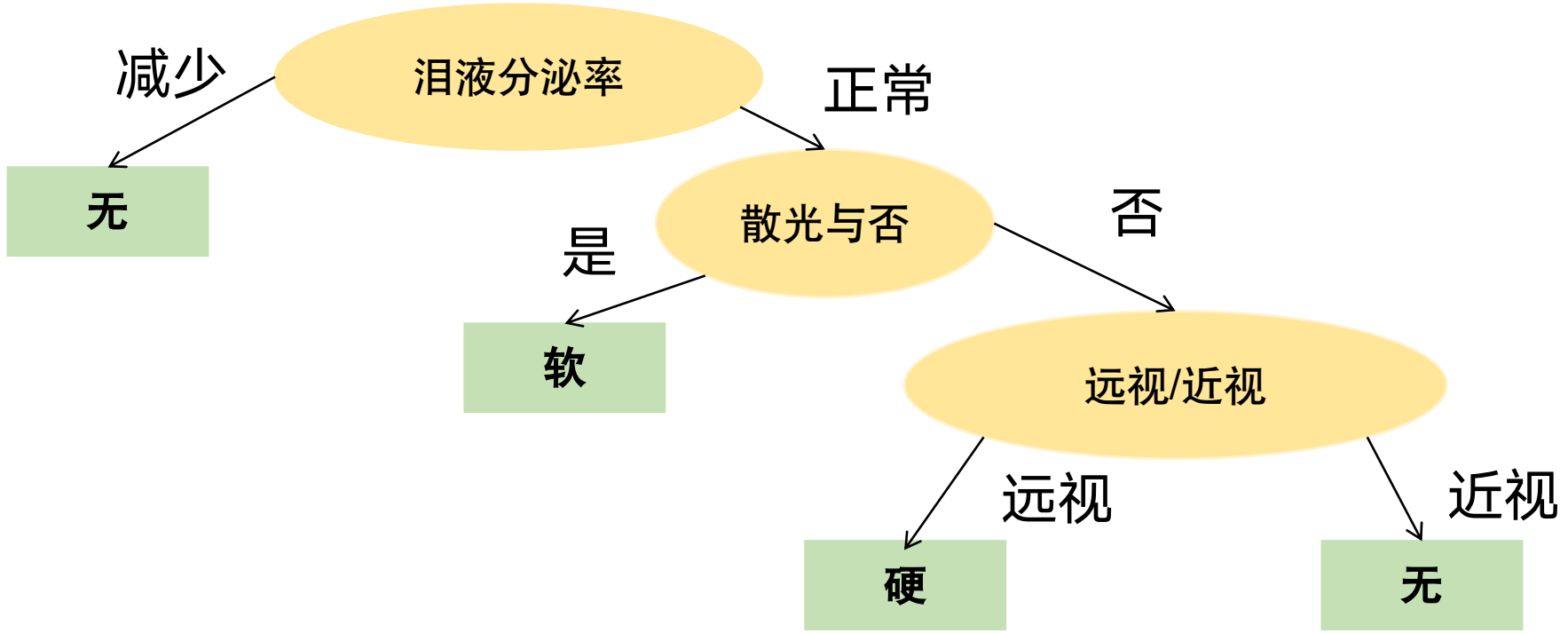
经过计算每个枝条的信息增益，
红圈内的枝条应当剪掉

$$IG(Y) = H(Z) - H(Z|Y)$$



树剪枝

- 最后的决策树：



小明	泪液分泌正常	不散光	近视眼	年轻群体

课堂小问：剪枝后小明还是戴硬性眼镜吗？剪枝会带来什么样的结果？

决策树小结

- 什么是决策树
- 决策树构建相关概念：熵、条件熵、互信息、信息增益
- 决策树的过拟合、剪枝问题

集成方法概述

- 很难学习到一个总是能正确分类实例的*强分类器*
- 但很容易学习到很多*弱分类器*

  一个弱分类器可能在整个数据集上表现不佳，但可能在部分样本上表现良好，例如，有些可能善于识别“猫”，有些可能善于识别“狗”
- 如果弱分类器在不同的样本子集上表现良好，就有可能*通过以适当的方式组合这些弱分类器来得到一个强分类器*
- 两个问题
  1) 如何生成这些弱分类器？
  2) 如何组合这些弱分类器？

两种组合方法

1) 无加权平均

- 多数投票法

2) 加权平均

- 给予更好的分类器更大的权重

对于一个二分类问题，例如分类标签为 {-1，1} 的问题，考虑两个基本分类器：

两个基本分类器 $\hat{y}_1 = sign(f_1(\boldsymbol{x}))$ $\hat{y}_2 = sign(f_2(\boldsymbol{x}))$

最终分类器 $\hat{y}_e = sign(a_1 f_1(\boldsymbol{x}) + a_2 f_2(\boldsymbol{x}))$

*备注：*这些弱分类器可以是任何类型，例如决策树、支持向量机（SVM）、神经网络、逻辑回归等

集成方法课程大纲

- 集成方法
  - Bagging (多数投票法)
  - Boosting (加权平均法)

得到弱分类器

- 如何得到在不同样本子集上表现良好的分类器是未知的

- 我们可以尝试获得预测误差*相互独立的*分类器。例如：

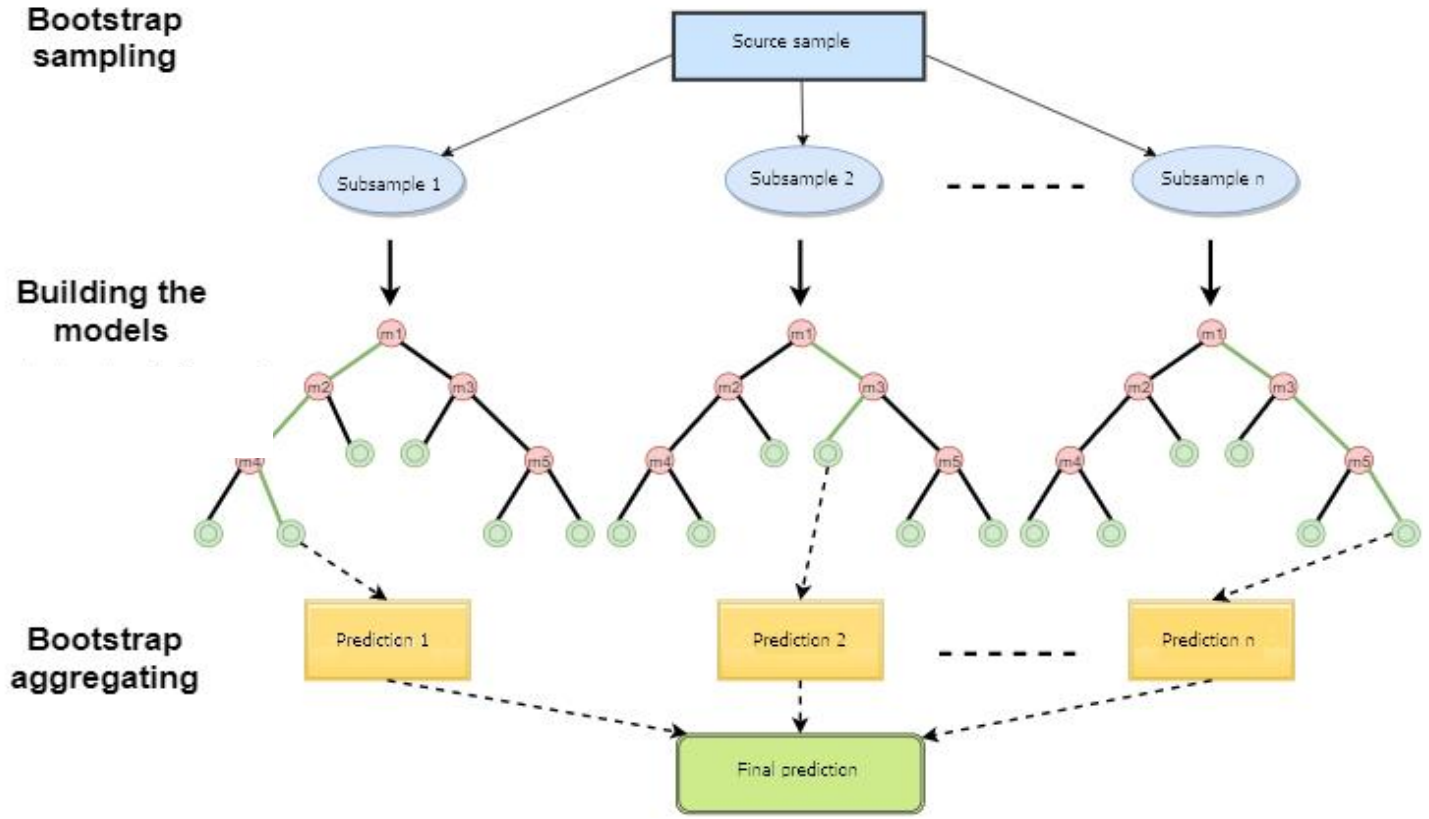
1) 通过自举法（bootstrapping）创建训练数据集的子集

   - 从 N 个样本的训练数据集中*有放回地*随机抽取 N' 个样本

   - 重复上述过程 K 次，生成子集 $S_1, S_2, \cdots, S_K$

2) 在每个子集 S_k 上训练一个决策树

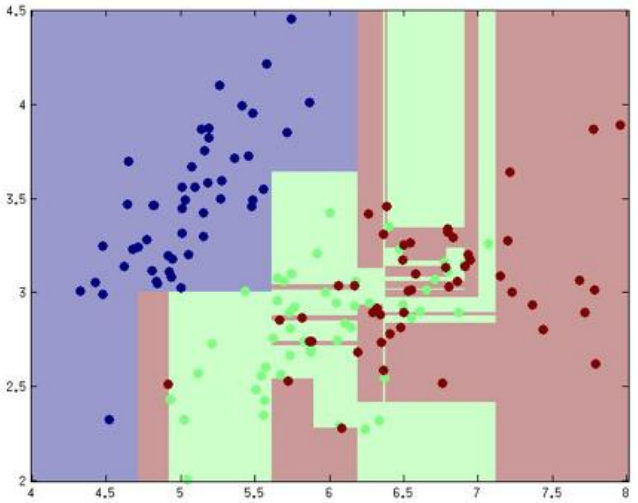
3) 通过多数投票法将 K 个决策树组合成一个



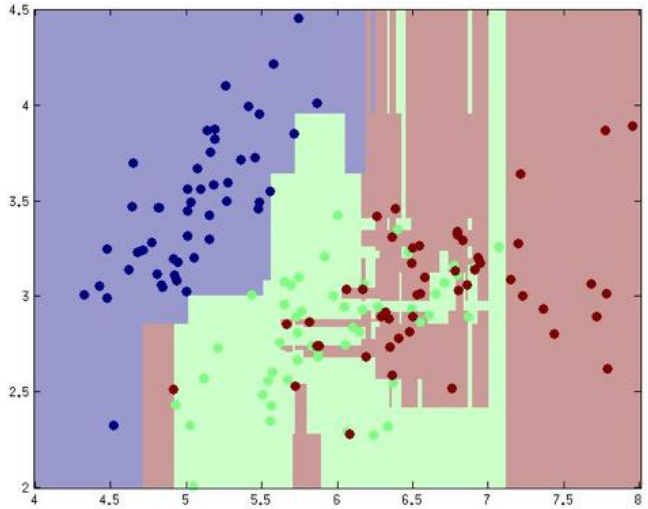
在测试时，将测试数据通过所有 K 个分类器，并使用多数投票的结果作为最终预测

示例：Bagged 决策树

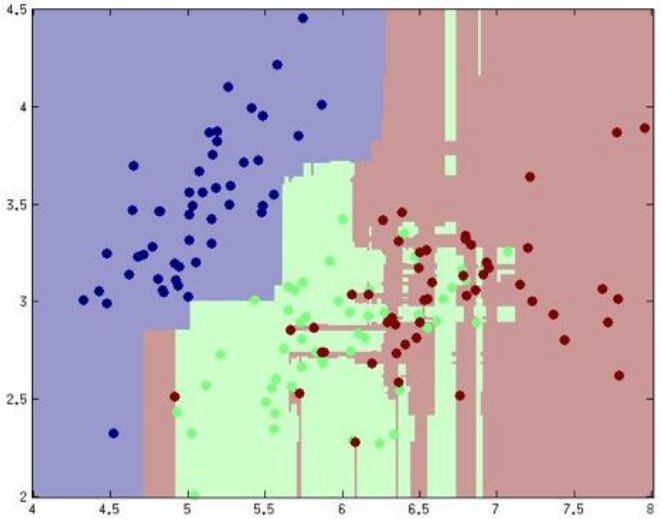
单棵树



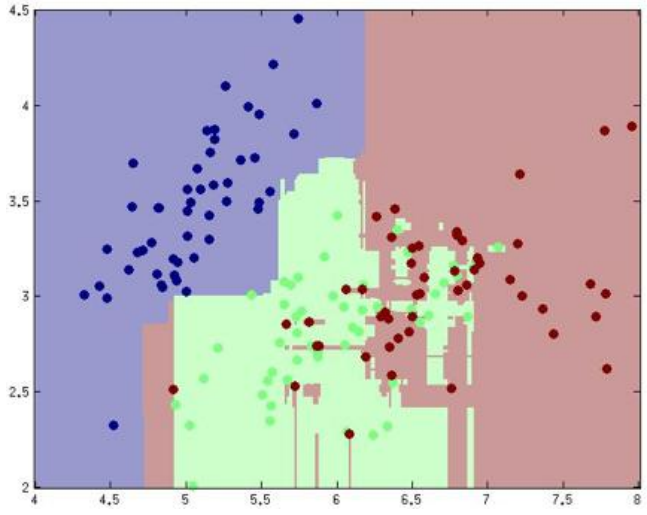
5棵树平均



25棵树平均



100棵树平均



随机森林

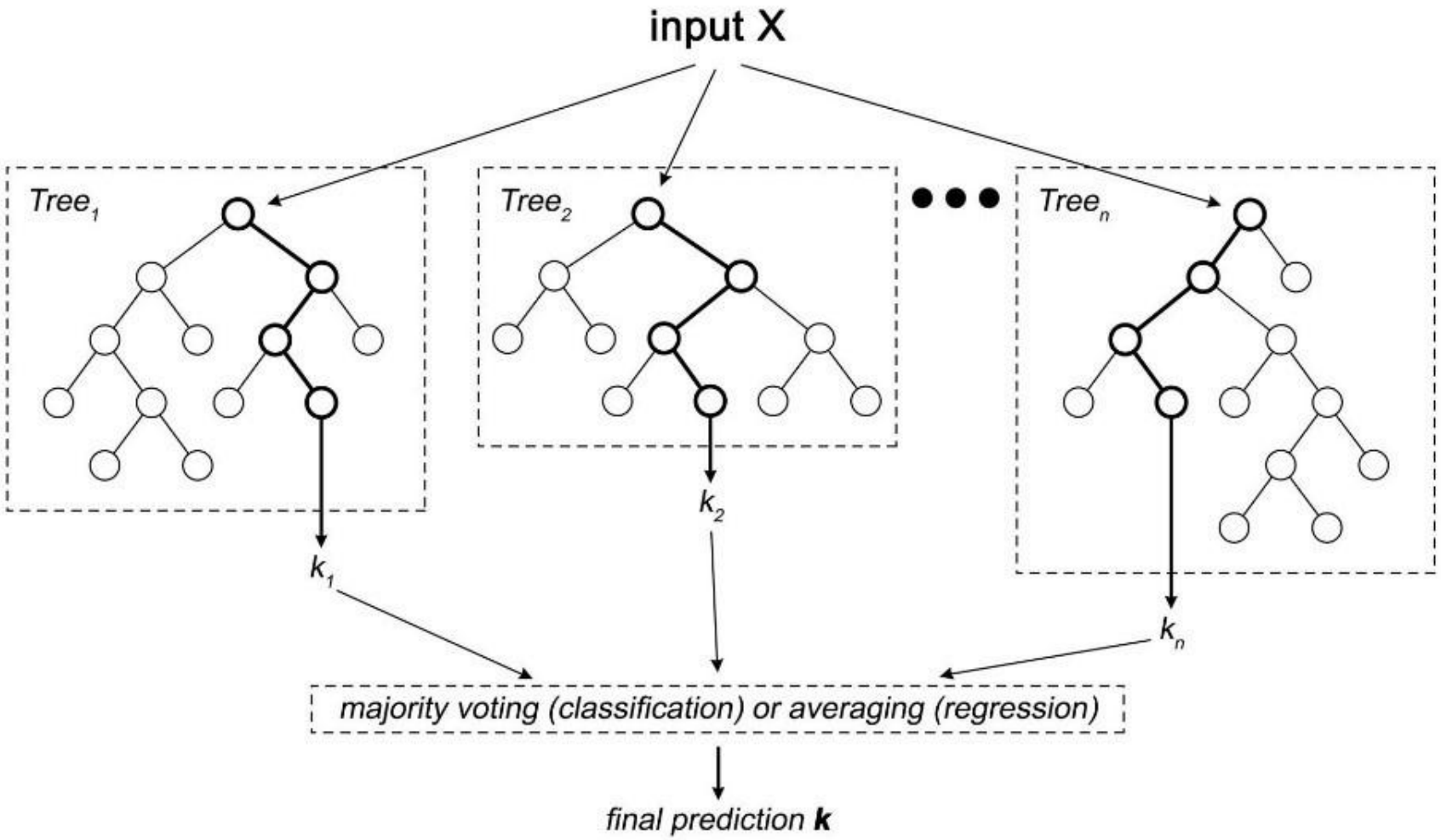
- 如果前面得到的分类器的预测误差*仍然高度相关*，会发生什么？

  ⇒ 仅仅通过多数投票法组合它们可能对最终性能没有太大帮助

- *补救措施：*在决策树的学习过程中引入额外的随机性

  仅使用一个随机选择的属性子集来构建决策树

- 随机森林图解


input X
Tree1
Tree2
Treen
k1
k2
kn
majority voting (classification) or averaging (regression)
final prediction k

课程大纲

- 集成方法
  - Bagging (多数投票法)
  - Boosting (加权平均法)

提升法（Boosting）概述

- *弱分类器来源*

  重复以下步骤数次：

  1) 识别被错误分类的样本

  2) 通过*对错误分类的样本赋予更大的权重* 来重新训练分类器

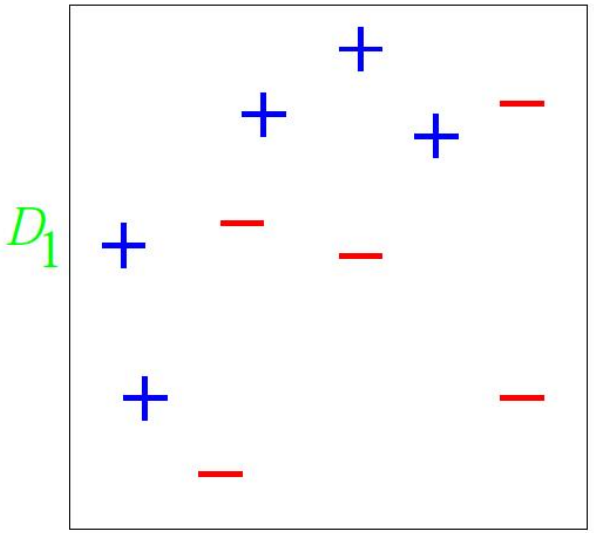
- *组合*

  通过*加权平均* 组合每个分类器的预测结果

如何*对样本和预测结果进行加权* 是关键所在

Adaboost

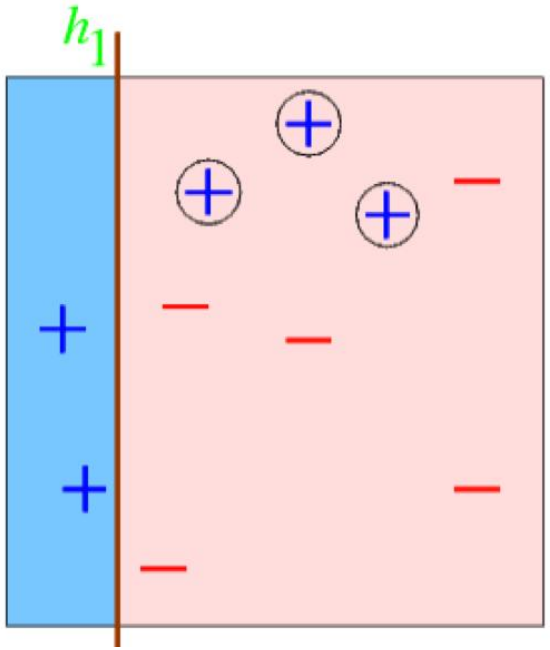
- 考虑一个包含10个训练样本的二分类问题



- 为简单起见，仅考虑决策边界与坐标轴平行的弱分类器，即：

$$\hat{y} = sign(x_1 + b) \quad \text{or } \hat{y} = sign(x_2 + b)$$

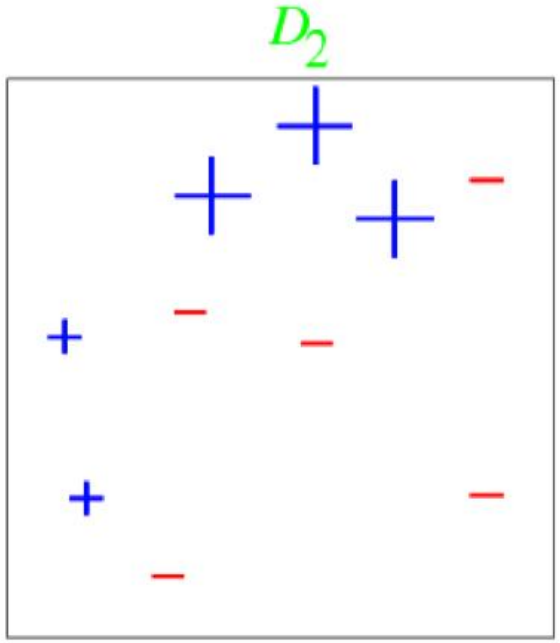
- 第一次迭代



- 第一个分类器 h_1 的错误率：$\epsilon_1 = 0.3$
- 分类器 h_1 的权重：$\alpha_1 = \frac{1}{2}\ln\left(\frac{1-\epsilon_1}{\epsilon_1}\right) = 0.42$

$$\frac{1-\epsilon_1}{\epsilon_1} = \frac{\text{正确率}}{\text{错误率}}$$

$\Rightarrow$ 权重与分类器的性能呈*正比*



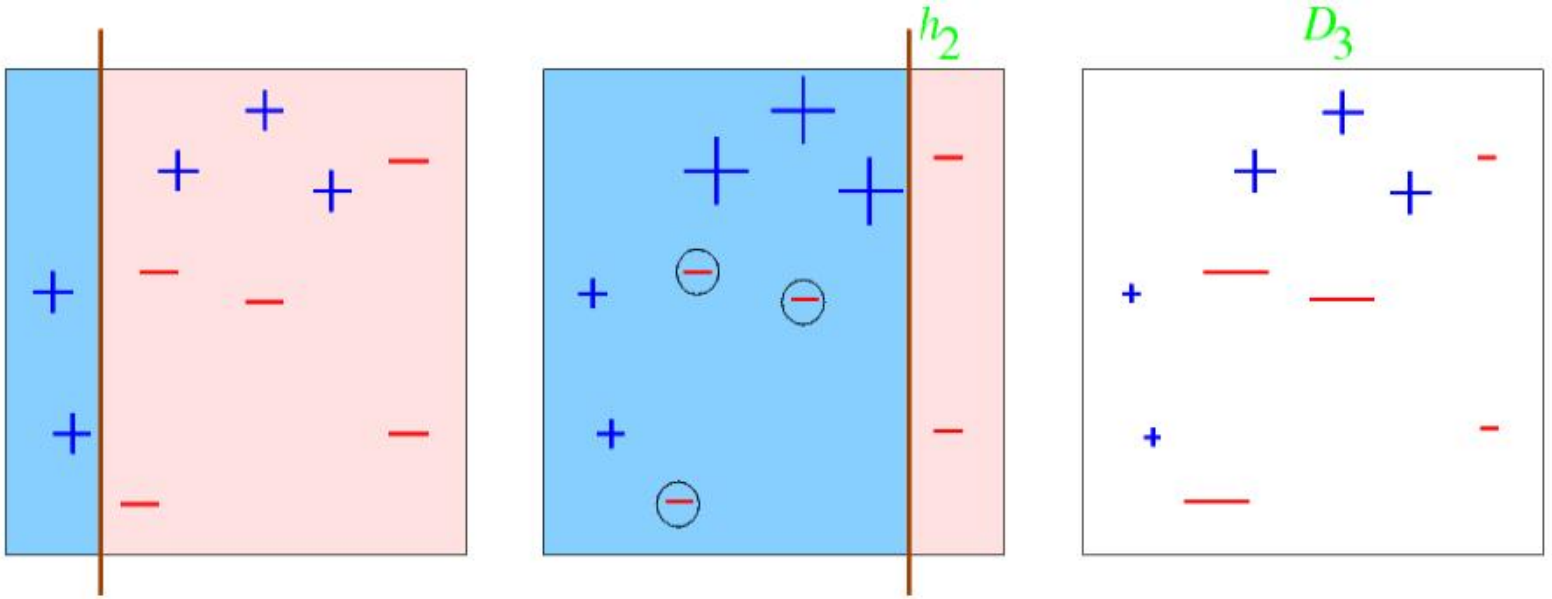
- *被错误分类* 样本的权重被 e^{α_1} 放大

$$e^{\alpha_1} = \sqrt{\frac{1-\epsilon_1}{\epsilon_1}} = \sqrt{\frac{\text{正确率}}{\text{错误率}}}$$

- *被正确分类* 样本的权重被 $e^{-\alpha_1}$ 削弱

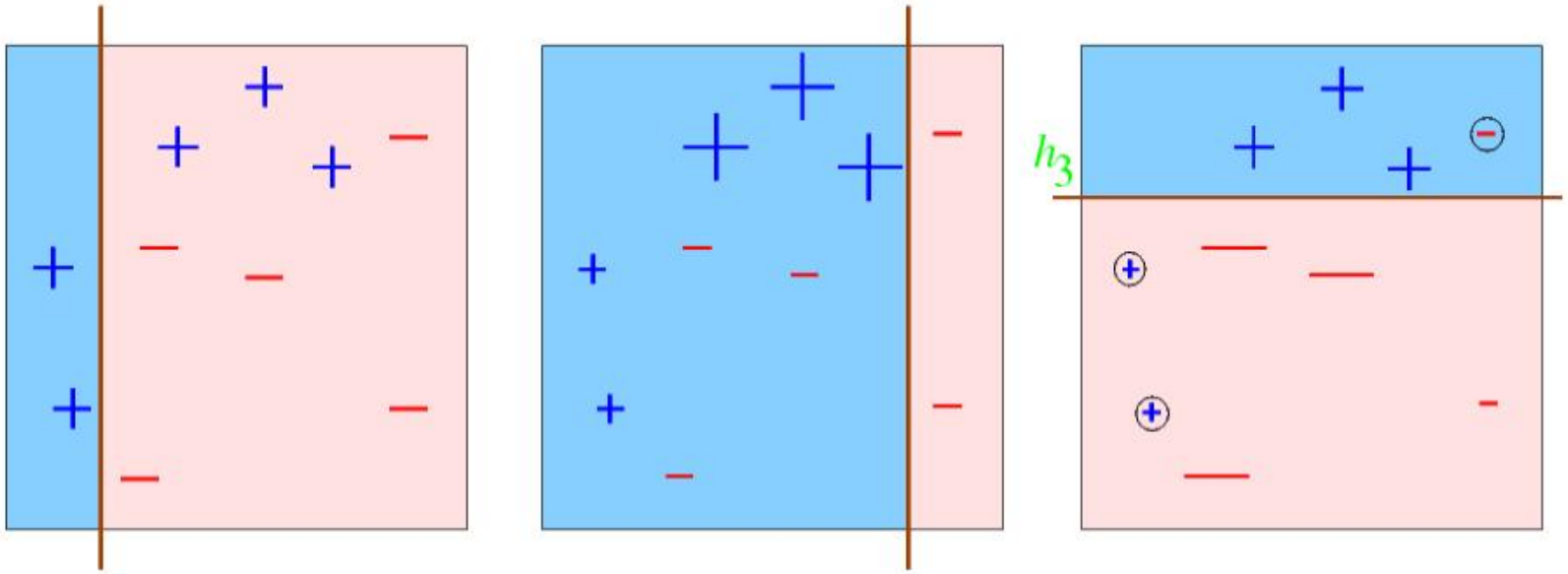
$\Rightarrow$ 分类器权重：$\ln\left(\sqrt{\frac{\text{正确率}}{\text{错误率}}}\right)$，样本权重：$\sqrt{\frac{\text{正确率}}{\text{错误率}}}$

- 第二次迭代



- 第二个分类器 h_2 的错误率：$\epsilon_2 = 0.21$
- 分类器 h_2 的权重：$\alpha_2 = \frac{1}{2}\ln\left(\frac{1-\epsilon_2}{\epsilon_2}\right) = 0.65$
- 被错误分类样本的权重被 $e^{\alpha_2} = \sqrt{\frac{1-\epsilon_2}{\epsilon_2}}$ 放大
- 被正确分类样本的权重被 $e^{-\alpha_2} = \sqrt{\frac{\epsilon_2}{1-\epsilon_2}}$ 削弱

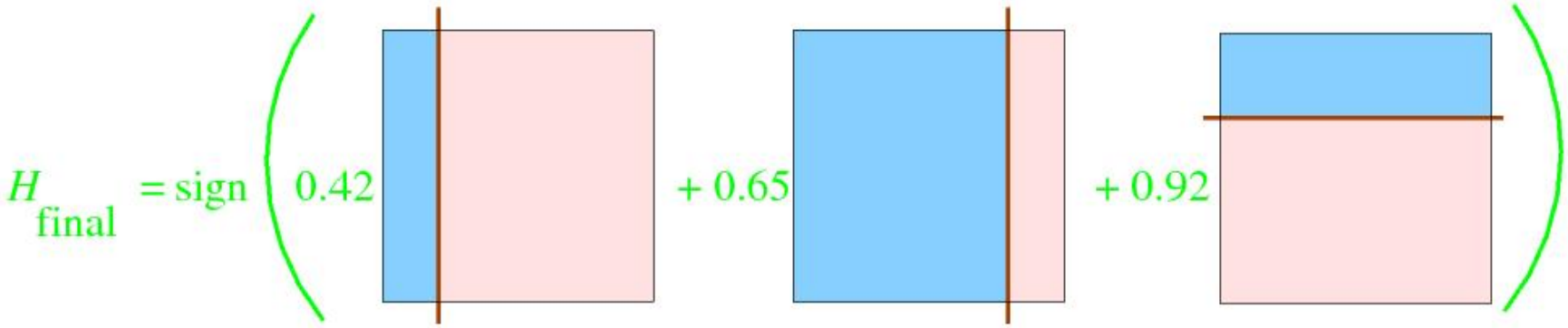
- 第三次迭代

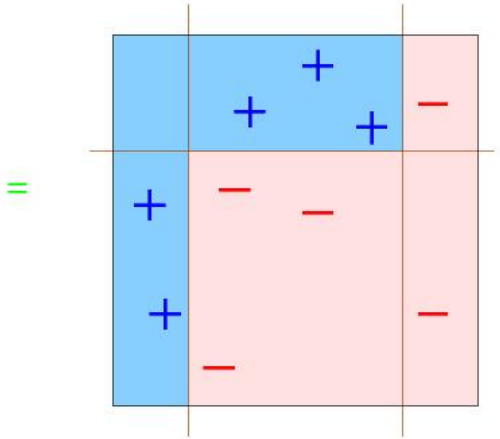


- 第二个分类器 h_3 的错误率：$\epsilon_3 = 0.14$
- 分类器 h_3 的权重：$\alpha_3 = \frac{1}{2}\ln\left(\frac{1-\epsilon_3}{\epsilon_3}\right) = 0.92$
- 停止迭代

- 最终分类器

最终分类器通过线性组合来组合这三个分类器





AdaBoost 算法

1) 初始化样本权重，$\omega_0^{(n)} = \frac{1}{N}$，其中 $n = 1, \cdots, N$

2) 对于第 k 次迭代，使用按 $w_{k-1}^{(n)}$ 加权的训练样本训练一个分类器 $\mathrm{h}_k(\boldsymbol{x})$

3) 评估加权分类错误：

$$\epsilon_k = \frac{\sum_{n=1}^{N} \omega_{k-1}^{(n)} I(y_i \neq \mathrm{h}_k(\boldsymbol{x}_n))}{\sum_{n=1}^{N} \omega_{k-1}^{(n)}}$$

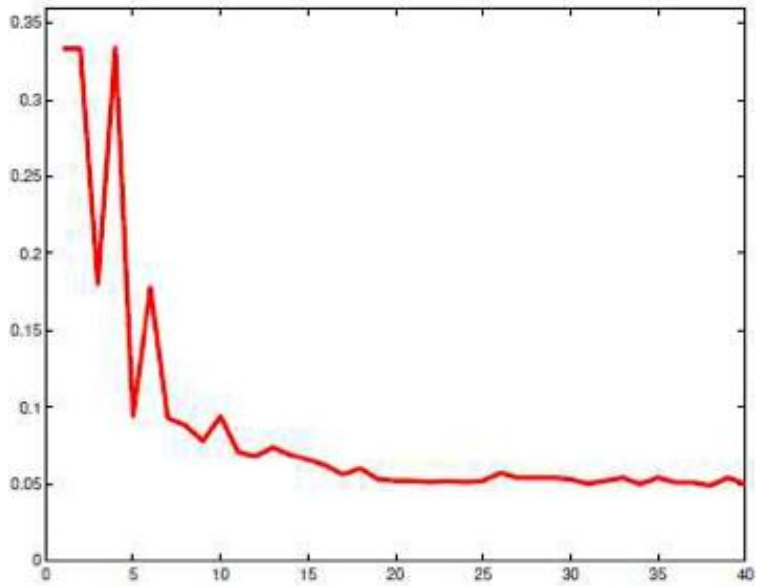
4) 确定第 k 个分类器的投票权重：

$$a_k = \frac{1}{2} \ln\left(\frac{1-\epsilon_k}{\epsilon_k}\right)$$

5) 更新样本权重：

$$\omega_k^{(n)} = \omega_{k-1}^{(n)} \exp\{-y_i \mathrm{h}_k(\boldsymbol{x}_i) a_k\}$$

- 典型的错误率曲线，以弱分类器数量为函数



- 典型的弱分类器：
  - 决策树
  - 逻辑回归
  - 神经网络

AdaBoost 背后的目标函数

- 定义以下*指数损失*:

$$\mathfrak{l} = \sum_{n=1}^{N} \exp\left\{-y^{(n)} \mathrm{h}_{combine}(\boldsymbol{x}^{(n)})\right\}$$

其中 $\boldsymbol{x}^{(n)}$ 是输入，$y^{(n)} \in \{-1, 1\}$ 是标签，$\mathrm{h}_{combine}(\cdot)$ 是组合分类器:

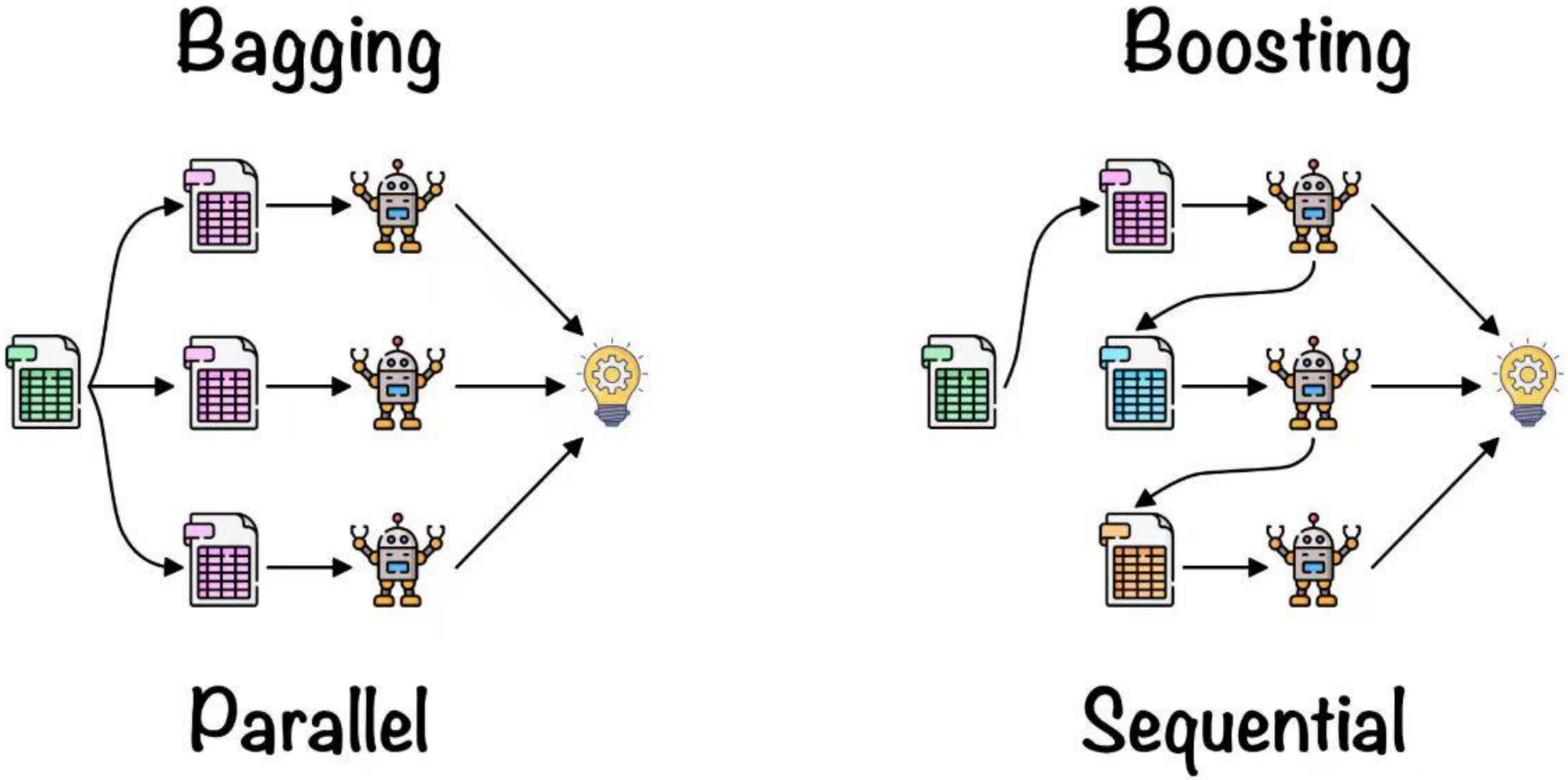
$$\mathrm{h}_{combine}(\boldsymbol{x}) = a_1 \mathrm{h}_1(\boldsymbol{x}) + \cdots + a_K \mathrm{h}_K(\boldsymbol{x})$$

$\mathrm{h}_k(\boldsymbol{x})$ 代表第 k 个组成分类器，例如 $\mathrm{h}_k(\boldsymbol{x}) = sign(\boldsymbol{w}_k^T \boldsymbol{x} + b_k)$

- 可以证明，AdaBoost 算法*等价于* 以序列式的方式*最小化指数损失*

Bagging和Boosting的本质区别

- 本质区别是并行和串行：



集成方法小结

- 什么是集成学习？
- Bagging（多数投票法）
- Boosting（加权平均法）