数据挖掘
第3章 分类-过拟合

教师：王东京
学院：计算机学院
邮箱：dongjing.wang@hdu.edu.cn

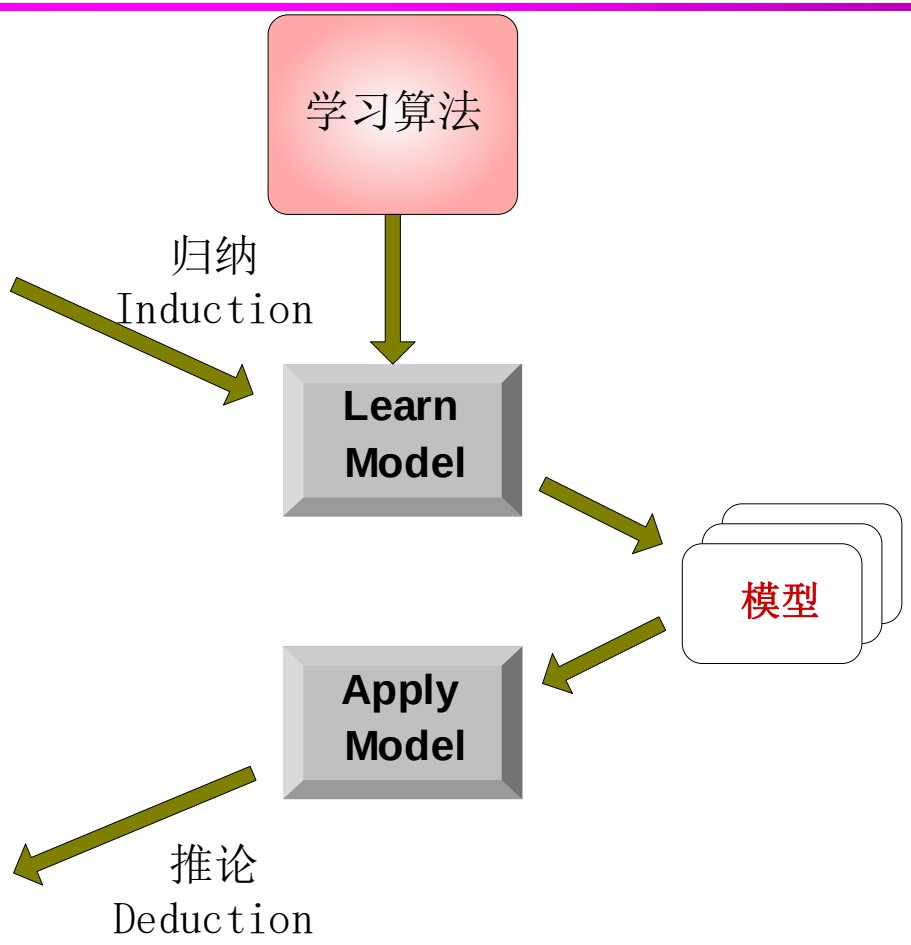
构建分类模型的通用手段

Tid	Attrib1	Attrib2	Attrib3	Class
1	Yes	Large	125K	No
2	No	Medium	100K	No
3	No	Small	70K	No
4	Yes	Medium	120K	No
5	No	Large	95K	Yes
6	No	Medium	60K	No
7	Yes	Large	220K	No
8	No	Small	85K	Yes
9	No	Medium	75K	No
10	No	Small	90K	Yes

训练集
Training Set

Tid	Attrib1	Attrib2	Attrib3	Class
11	No	Small	55K	?
12	Yes	Medium	80K	?
13	Yes	Large	110K	?
14	No	Small	95K	?
15	No	Large	67K	?

测试集
Test Set



分类误差 Classification Errors

训练误差 Training errors

- 训练集上的误差 Errors committed on the training set
- 也称为再代入误差（resubstitution error）或者表现误差（apparent errors）

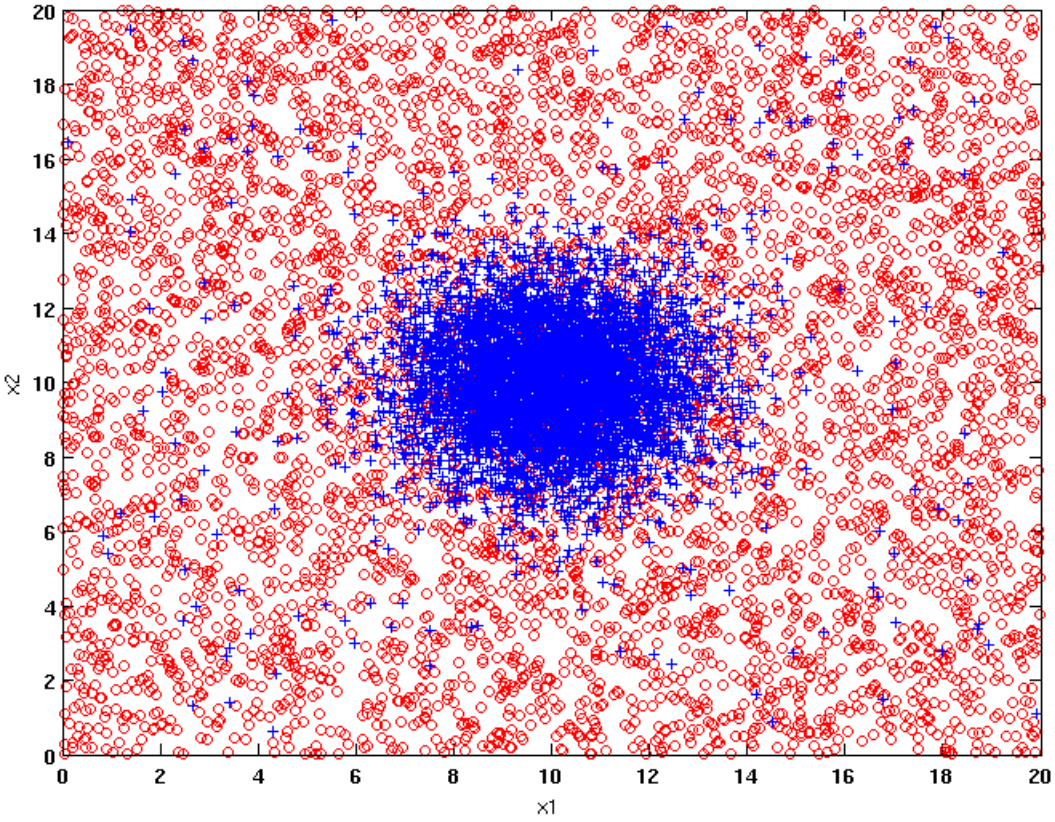
泛化误差 Generalization errors

- 模型在未知记录上的期望误差
- Expected error of a model over random selection of records from same distribution

测试误差 Test errors

- 测试集上的误差 Errors committed on the test set

Example Data Set



二分类（Two class）问题：

+ : 5400 个正样本实例（instances）

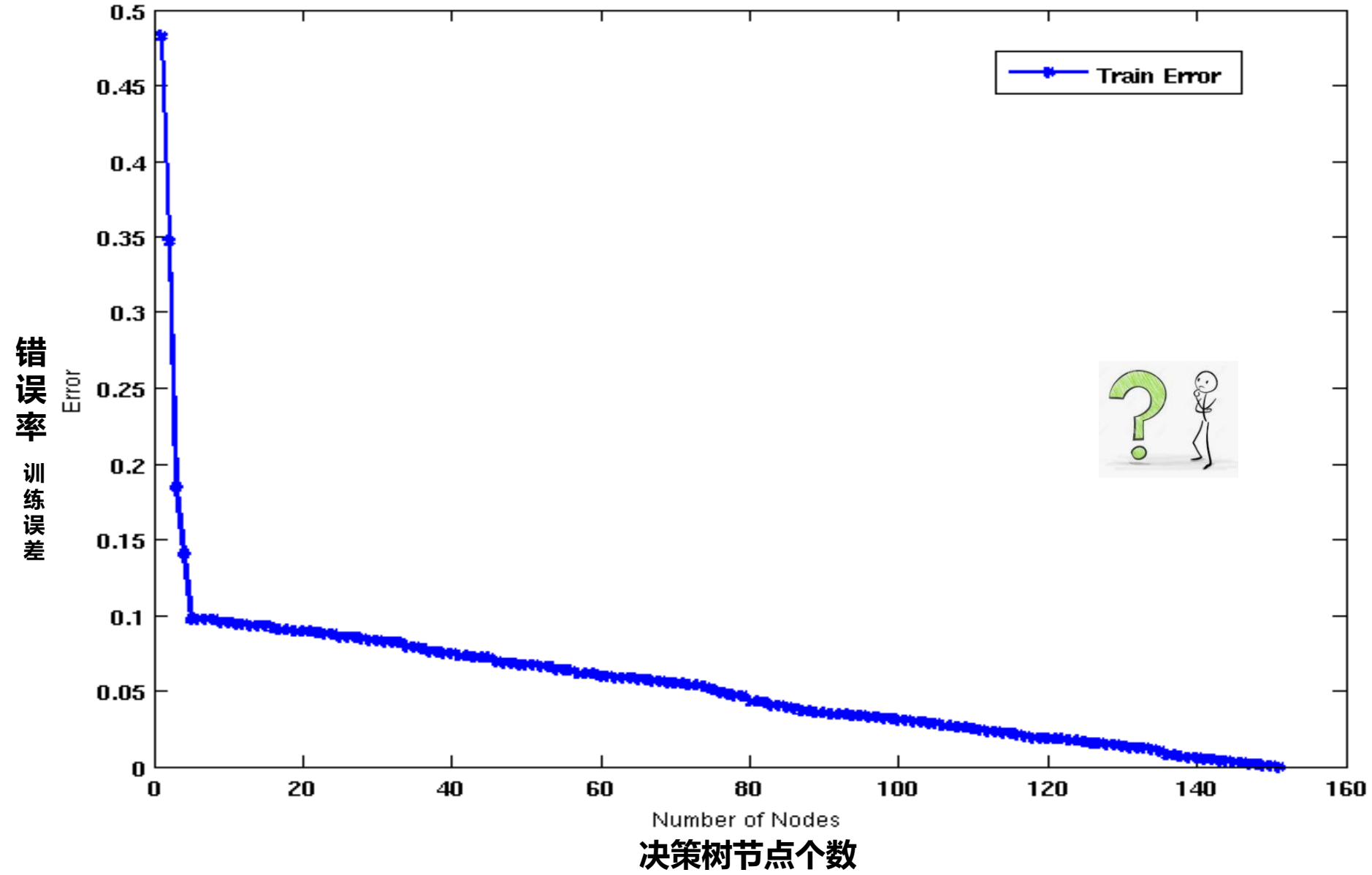
- **高斯分布**
- **5000 instances generated from a Gaussian centered at (10,10)**
- ***400 noisy instances added***

o : 5400 个负样本实例

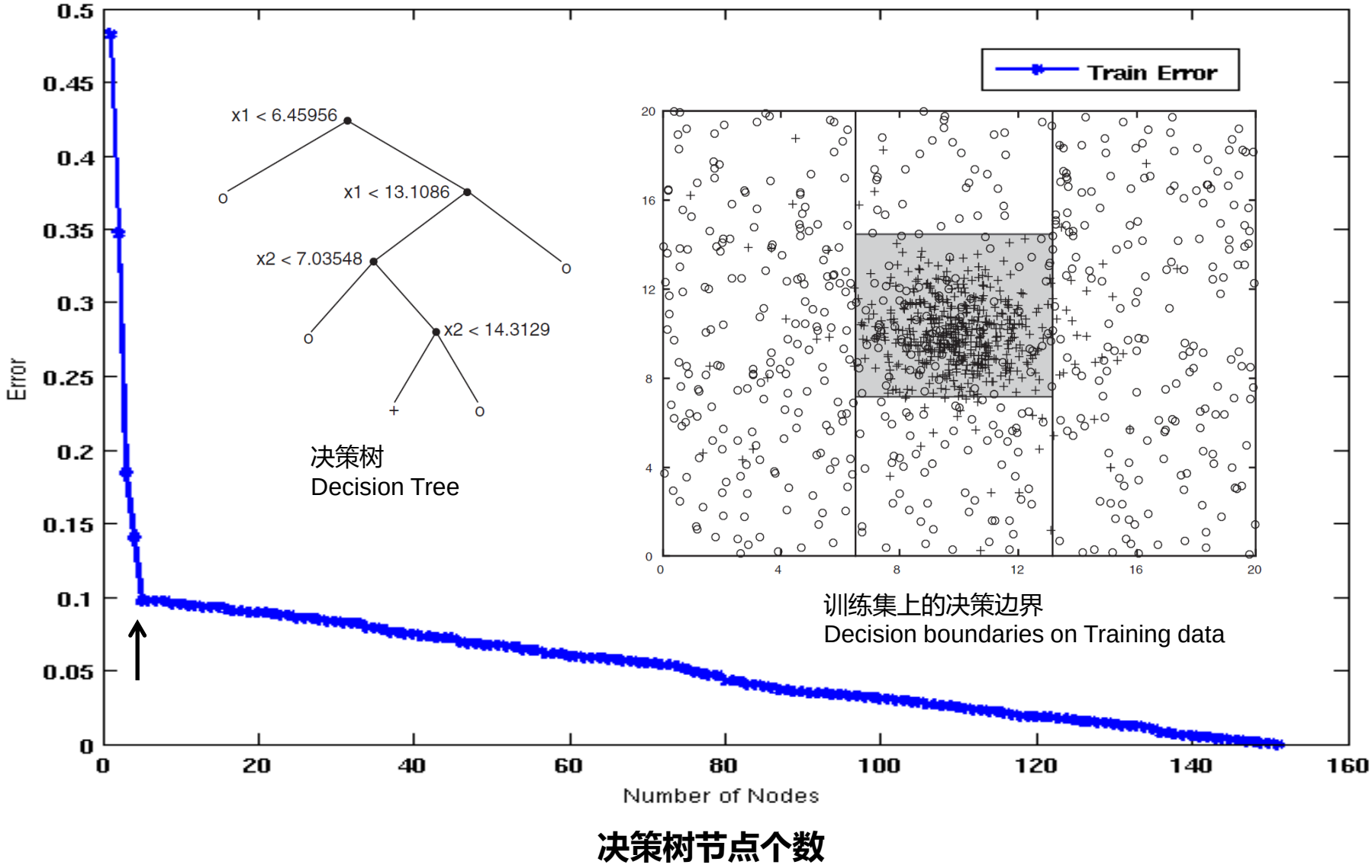
- **均匀分布**
- **Generated from a uniform distribution**

10 % of the data used for training and 90% of the data used for testing

增加决策树节点个数
Increasing number of nodes in Decision Trees



Decision Tree with 4 nodes



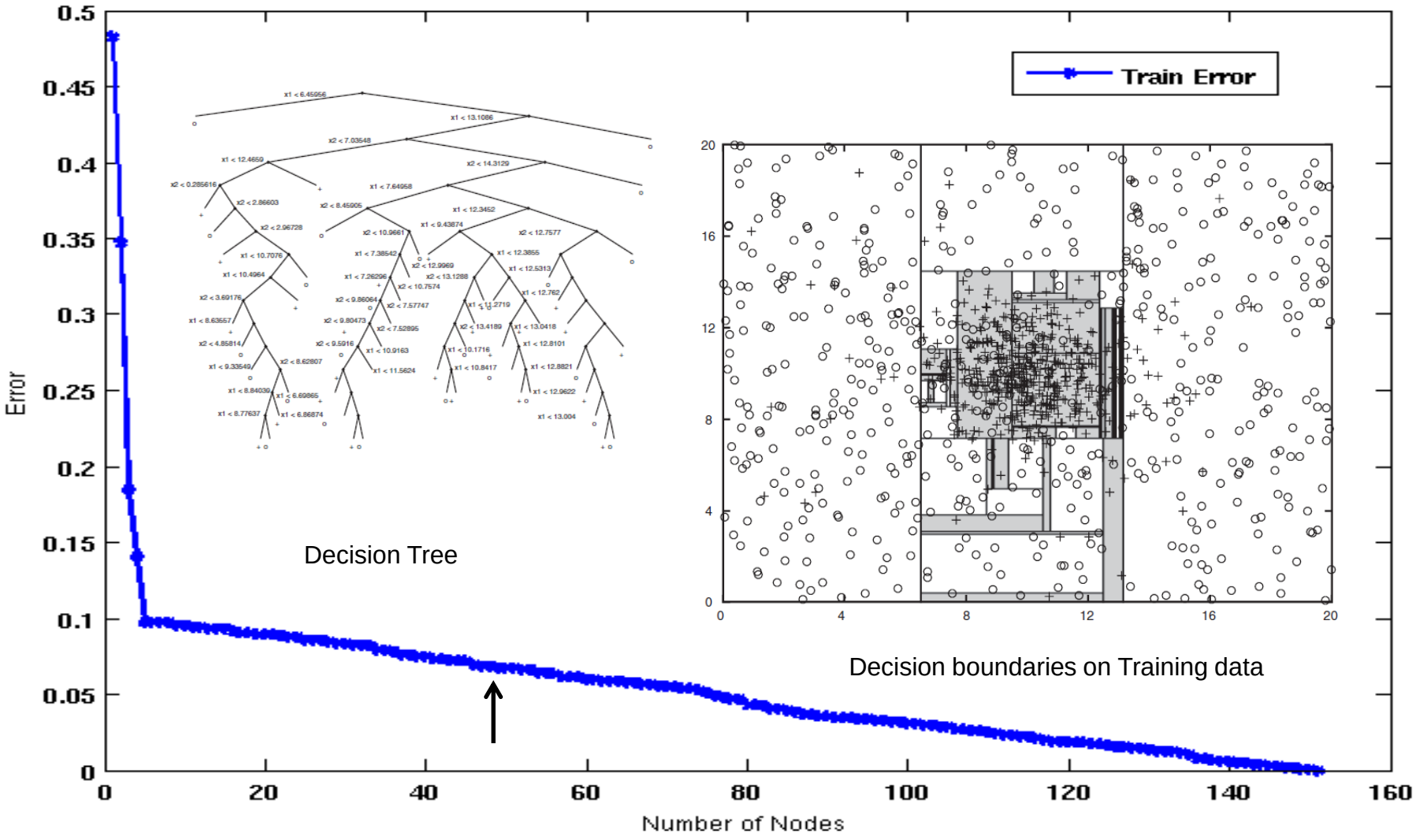
决策树
Decision Tree

训练集上的决策边界
Decision boundaries on Training data

错误率 训练误差

决策树节点个数

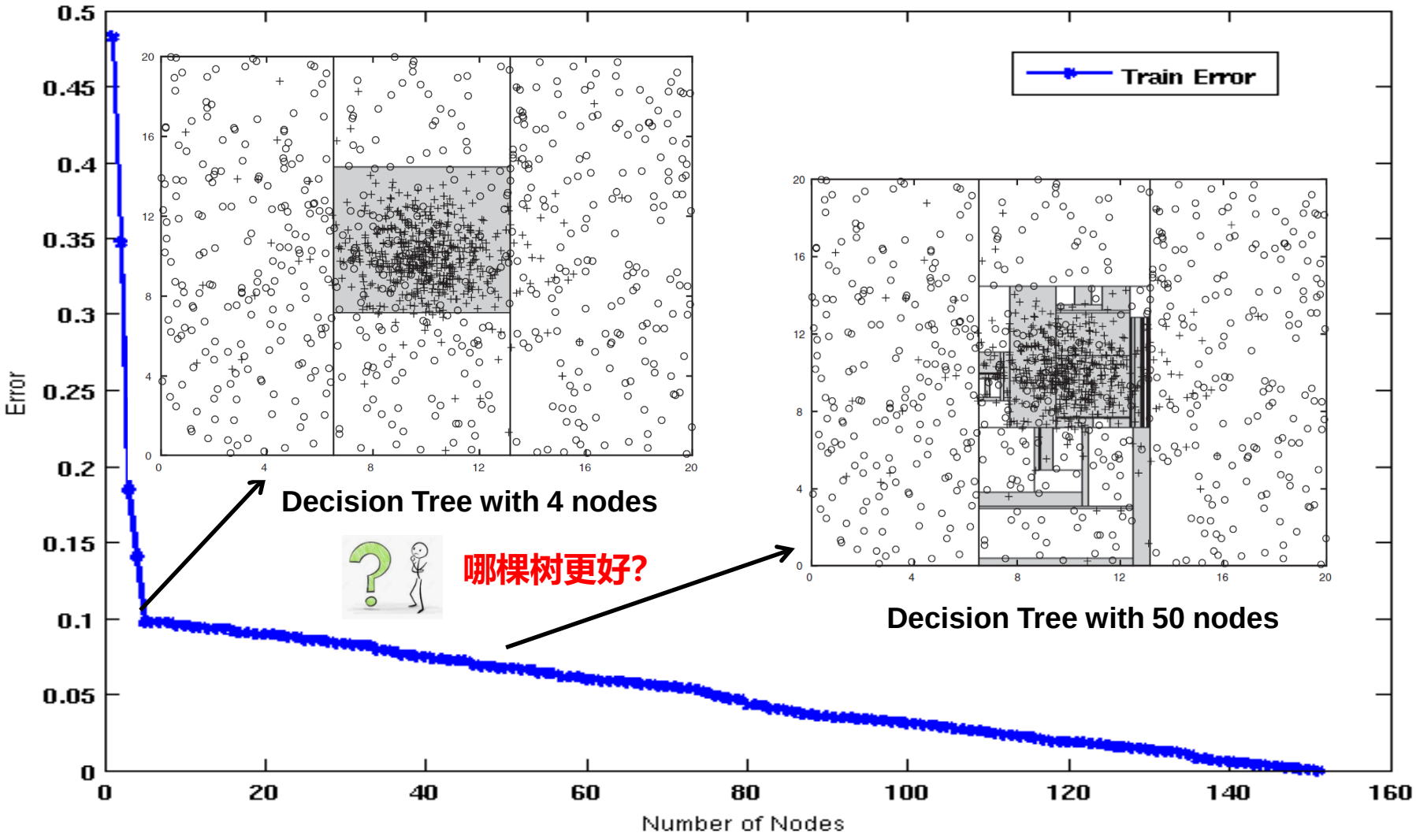
Decision Tree with 50 nodes



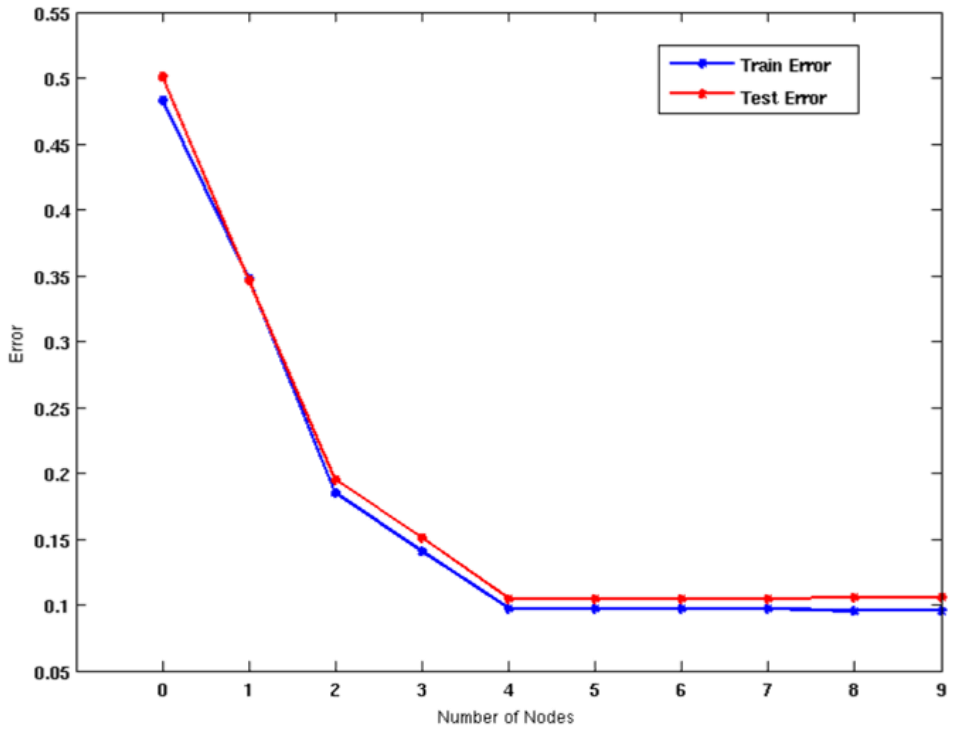
错误率 训练误差

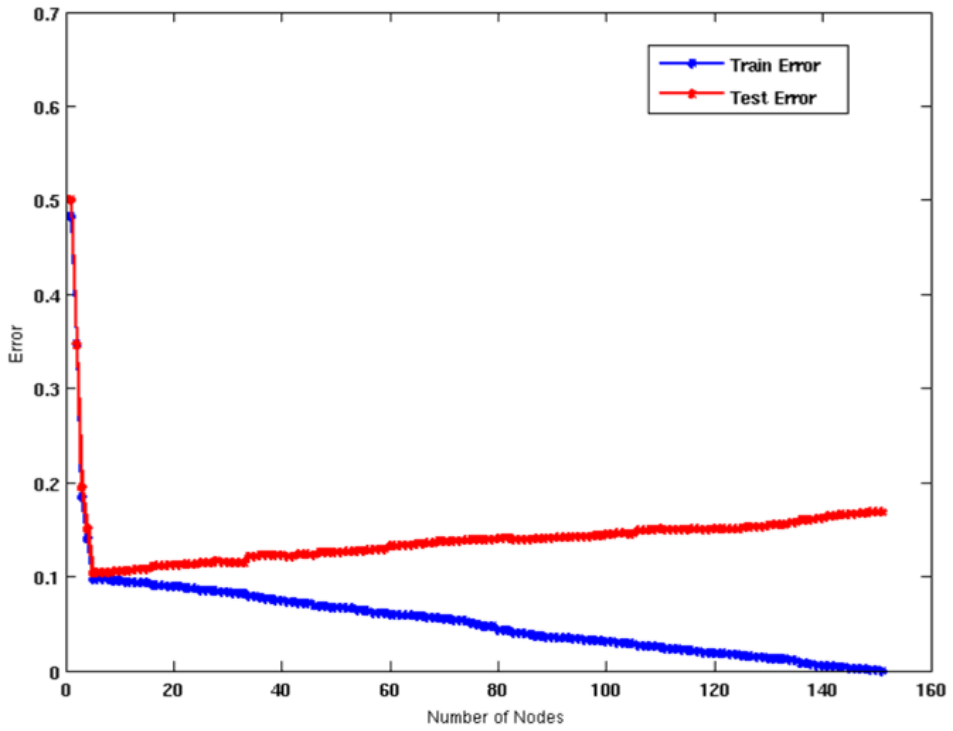
决策树节点个数

哪棵树更好?


Train Error
Error
Number of Nodes
0.5
0.45
0.4
0.35
0.3
0.25
0.2
0.15
0.1
0.05
0
0
20
40
60
80
100
120
140
160
Decision Tree with 4 nodes
哪棵树更好?
Decision Tree with 50 nodes
错误率
训练误差
决策树节点个数

模型过拟合 Model Overfitting





1. 首先，增加节点，训练误差和测试误差都减少；
2. 随着模型变得越来越复杂，即使训练误差可能在减少，测试误差也可能开始增加。

拟合 fitting

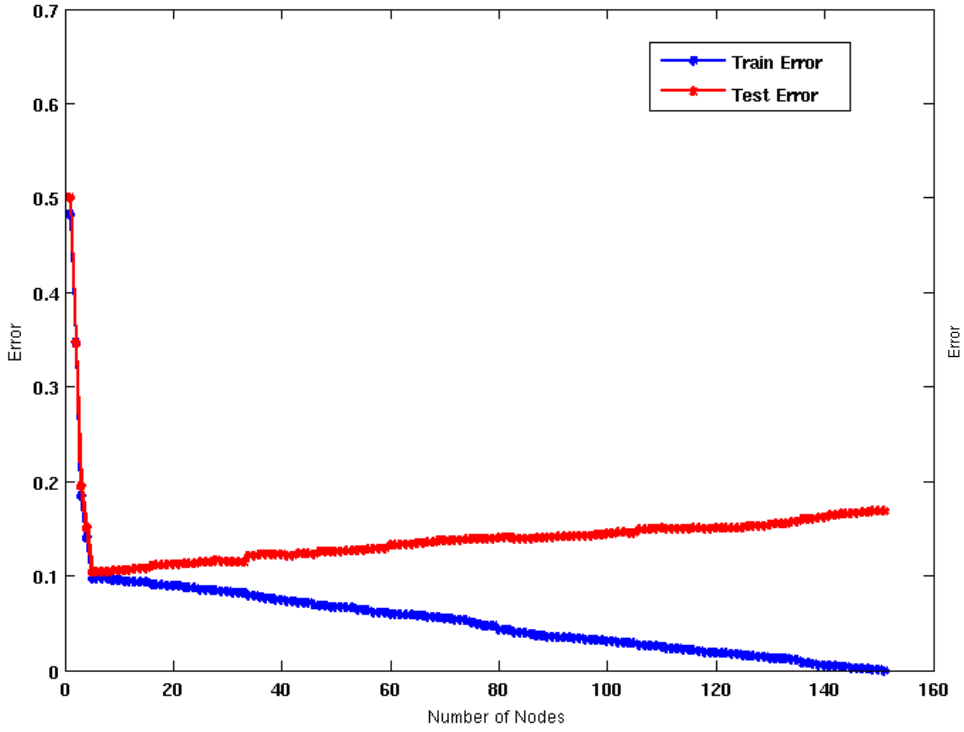
欠拟合 Underfitting:

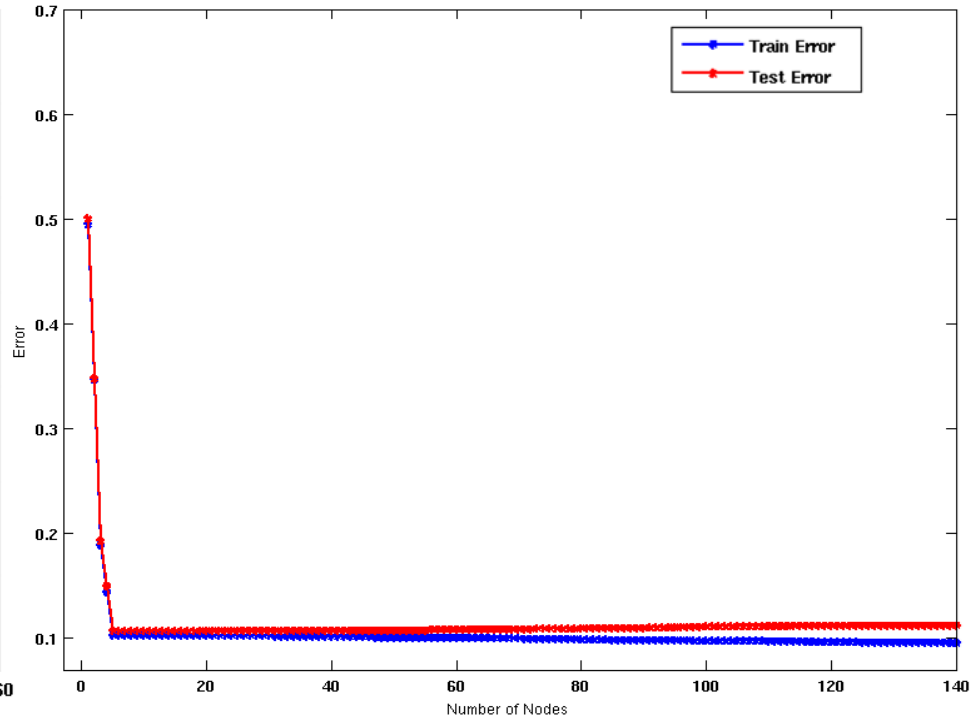
– when model is too simple, both training and test errors are large

过拟合 Overfitting:

– when model is too complex, training error is small but test error is large

Model Overfitting

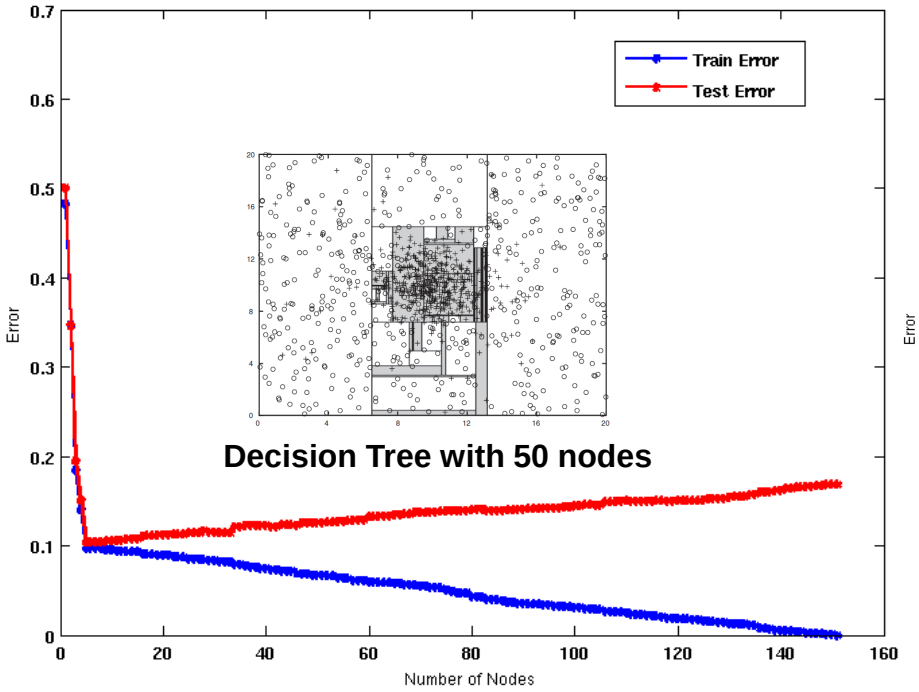


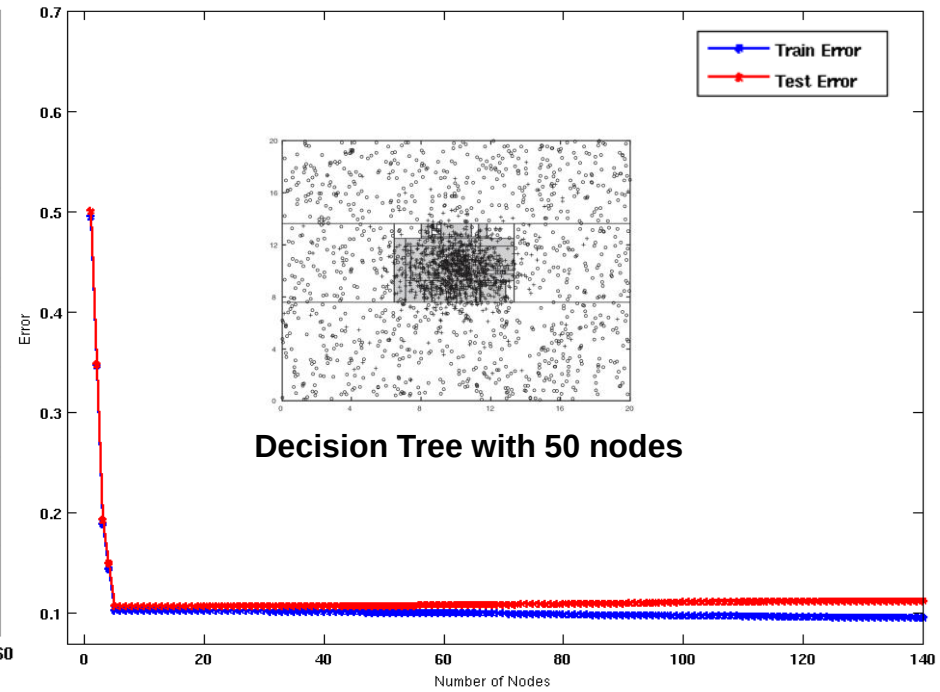


使用2倍的训练数据

在给定的模型大小下，增加训练数据的大小会减少训练和测试错误之间的差异

Model Overfitting





使用2倍的训练数据

在给定的模型大小下，增加训练数据的大小会减少训练和测试错误之间的差异

模型过拟合的原因?

噪声（noise）会导致过拟合

表 4-3　哺乳类动物分类的训练数据集样本。打星号的类标号代表错误标记的记录

名称	体温	胎生	4 条腿	冬眠	类标号
豪猪	恒温	是	是	是	是
猫	恒温	是	是	否	是
蝙蝠	恒温	是	否	是	否*
鲸	恒温	是	否	否	否*
蝾螈	冷血	否	是	是	否
科莫多巨蜥	冷血	否	是	否	否
蟒蛇	冷血	否	否	是	否
鲑鱼	冷血	否	否	否	否
鹰	恒温	否	否	否	否
虹鳉	冷血	是	否	否	否

表 4-4　哺乳类动物分类的检验数据集样本

名称	体温	胎生	4 条腿	冬眠	类标号
人	恒温	是	否	否	是
鸽子	恒温	否	否	否	否
象	恒温	是	是	否	是
豹纹鲨	冷血	是	否	否	否
海龟	冷血	否	是	否	否
企鹅	冷血	否	否	否	否
鳗	冷血	否	否	否	否
海豚	恒温	是	否	否	是
针鼹	恒温	否	是	是	是
希拉毒蜥	冷血	否	是	是	否

模型过拟合的原因?

有限的训练集 Limited Training Size

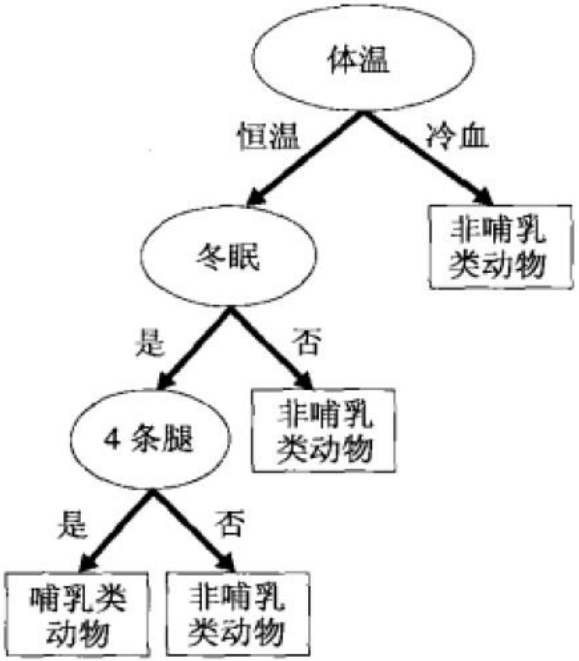
缺乏代表性样本导致的过分拟合

表 4-5　哺乳动物分类的训练集样本

名称	体温	胎生	4 条腿	冬眠	类标号
蝾螈	冷血	否	是	是	否
虹鳉	冷血	是	否	否	否
鹰	恒温	否	否	否	否
弱夜鹰	恒温	否	否	是	否
鸭嘴兽	恒温	否	是	是	是

表 4-4　哺乳类动物分类的检验数据集样本

名称	体温	胎生	4 条腿	冬眠	类标号
人	恒温	是	否	否	是
鸽子	恒温	否	否	否	否
象	恒温	是	是	否	是
豹纹鲨	冷血	是	否	否	否
海龟	冷血	否	是	否	否
企鹅	冷血	否	否	否	否
鳗	冷血	否	否	否	否
海豚	恒温	是	否	否	是
针鼹	恒温	否	是	是	是
希拉毒蜥	冷血	否	是	是	否



模型过拟合的原因?

模型复杂度过高 High Model Complexity

- 多重比较过程 Multiple Comparison Procedure

多重比较过程的作用

以 “预测股市在未来10个交易日内是否会上涨/下跌” 的任务为例

随机猜测Random guessing:

$$P(correct) = 0.5$$

连续猜10次:

- 假设涨跌随机，那么猜对超过8次的概率是多少？（包括8次）

$$P(\#correct \geq 8) = \frac{\binom{10}{8} + \binom{10}{9} + \binom{10}{10}}{2^{10}} = 0.0547$$

Day 1	Up
Day 2	Down
Day 3	Down
Day 4	Up
Day 5	Down
Day 6	Down
Day 7	Up
Day 8	Up
Day 9	Up
Day 10	Down

多重比较过程的作用

方法:

- 获得50位分析师
- 每个分析师随机进行10次猜测
- 选择做出最多正确预测的分析师

至少一名分析师做出至少8次正确预测的概率?

$$P(\#\,correct \geq 8) = 1 - (1 - 0.0547)^{50} = 0.9399$$

多重比较过程的作用

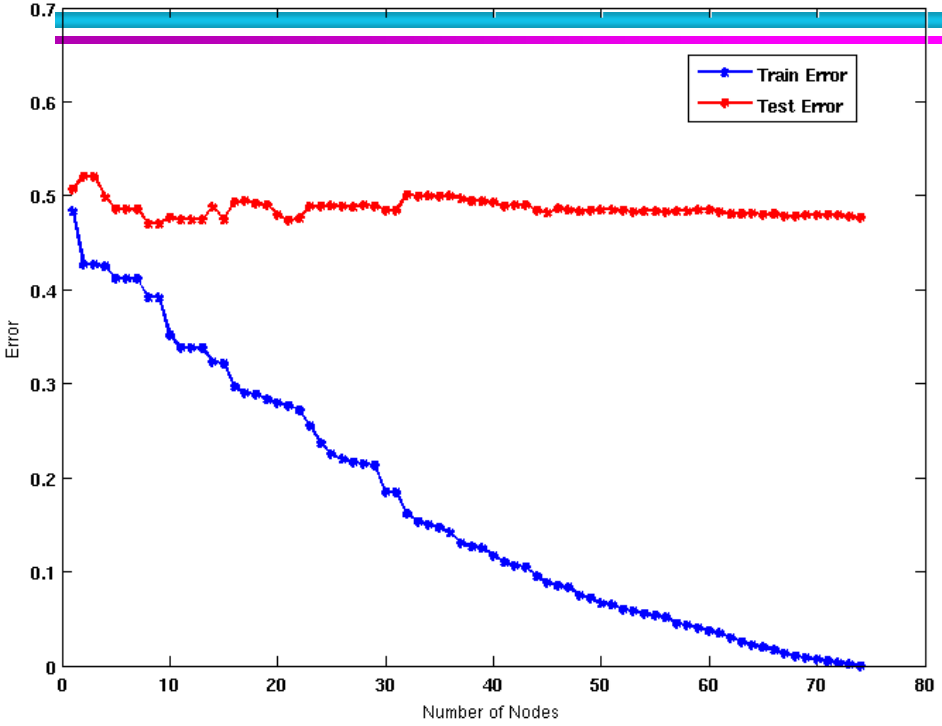
许多算法采用以下贪婪策略（greedy strategy）：

- 初始化模型 Initial model: M
- 构造可选模型 Alternative model: M' = M $\cup$ γ, 其中 γ 是添加到原模型的一个组件（component）（例如决策树中的测试条件）
- 如果效果有提升，则保留 M'
  - 观察到的改进是统计显著的：$\Delta(M,M') > \alpha$

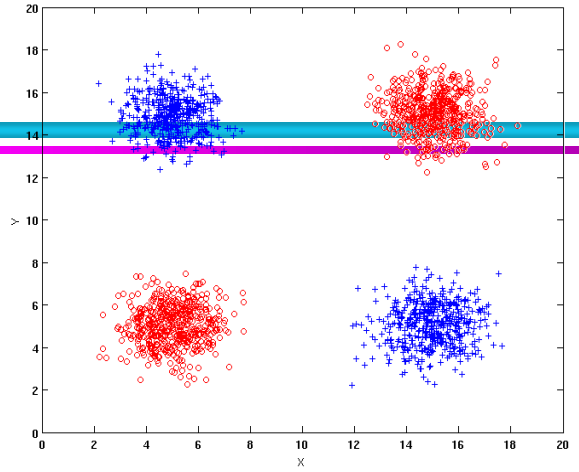
通常，γ 是从一组可选组件 $\Gamma = \{\gamma_1, \gamma_2, \ldots, \gamma_k\}$ 中选择的。

在多种可选方案中，有些则可能会无意中向模型添加无关的组件（inadvertently add irrelevant components to the model），从而导致模型过拟合（overfitting）

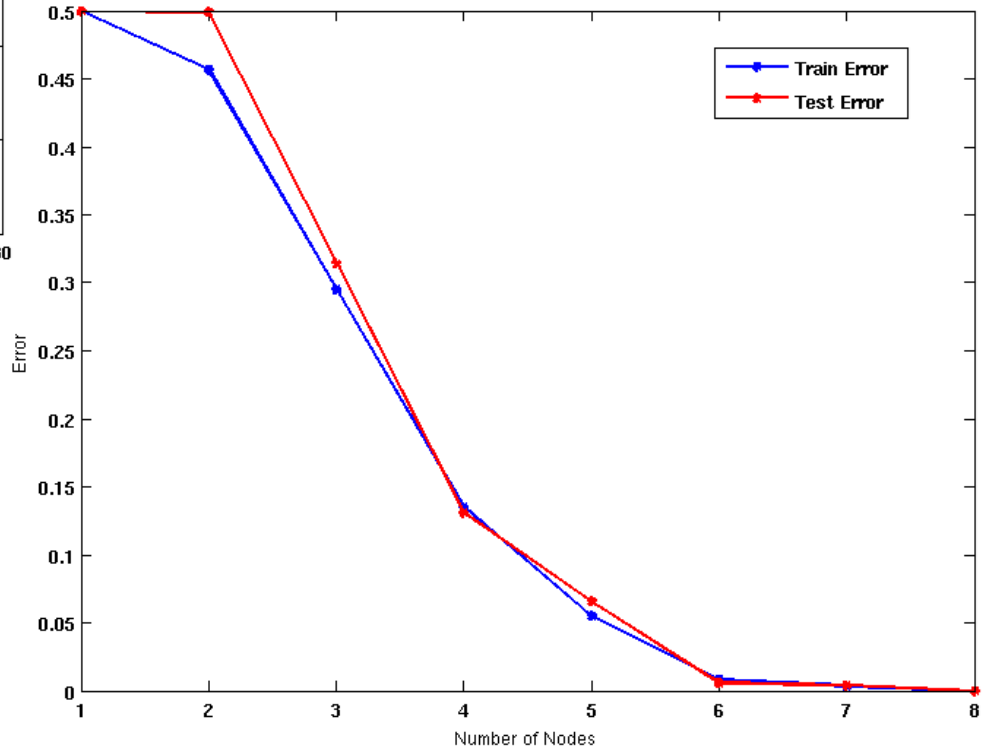
多重比较过程的作用 - 示例



使用X+Y+100个噪声变量作为属性



数据集



只使用X和Y作为属性

左上角的图使用从均匀分布生成的其它100个噪声变量和X和Y一起作为属性。
右下角的图只使用X和Y。

将30 %的数据用于训练，将70 %的数据用于测试。

过拟合总结 Notes on Overfitting

过度拟合导致决策树变得比需要的更为复杂

训练错误难以用于估计决策树在未知记录中的表现

需要一些方法来估计泛化误差

模型选择 Model Selection

在模型构建期间执行

目的是确保模型不会过于复杂（以避免过拟合）

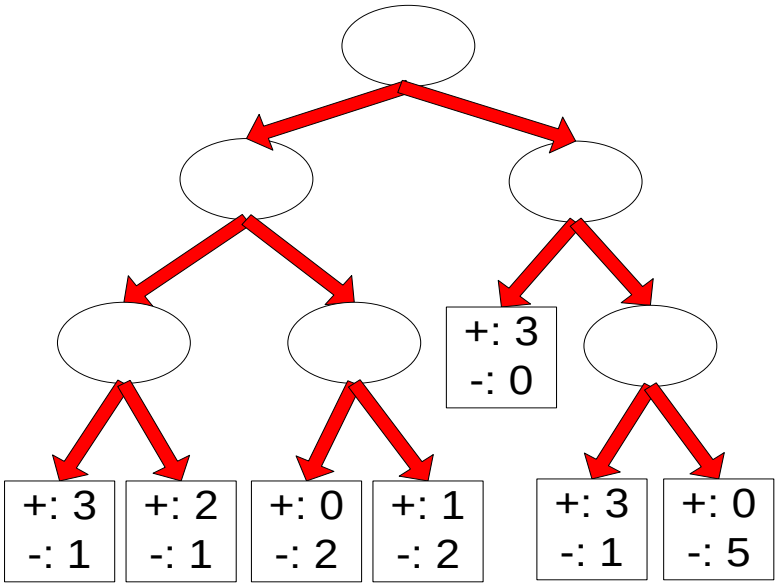
需要估计泛化误差（generalization error）

- 使用验证集（Validation Set）
- 结合模型复杂性（Model Complexity）
- 估计统计上界（Statistical Bounds）

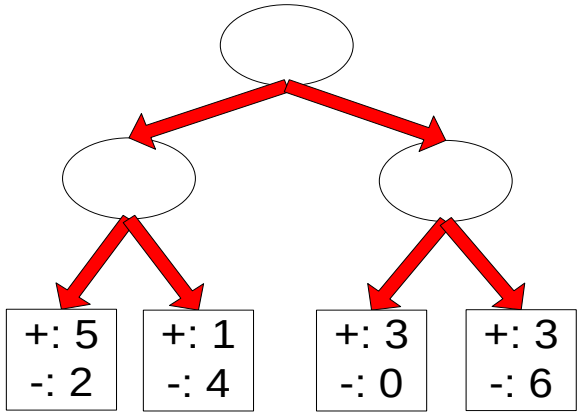
评估决策树的复杂度：实例
Estimating the Complexity of Decision Trees: Example

再代入估计方法（Resubstitution Estimate）：

- 使用训练误差（training error）作为泛化误差（generalization error）的乐观估计
- 称为乐观（optimistic）误差估计



Decision Tree, T_L



Decision Tree, T_R

训练误差

$e(T_L) = 4/24$

$e(T_R) = 6/24$

Model Selection:

使用验证集 Using Validation Set

将训练数据（training data）分为两部分：

- 训练集：
  - 用于构建模型
- 验证集：
  - 用于估计泛化误差
  - 注意：验证集与测试集（test set）不同

缺点：

- 可用于训练的数据变少

Model Selection:
结合模型复杂度 Incorporating Model Complexity

原理：奥卡姆剃刀（Occam's Razor）

- 给定两种具有相似泛化误差的模型，我们应该优先选择简单的模型
- 复杂的模型更容易被意外拟合（fitted accidentally）
- 因此，评估模型时应结合**模型的复杂度（complexity）**

模型的泛化误差 = 模型的训练误差 + 模型复杂度（加权）

Gen. Error(Model) = Train. Error(Model, Train. Data) +

α x Complexity(Model)

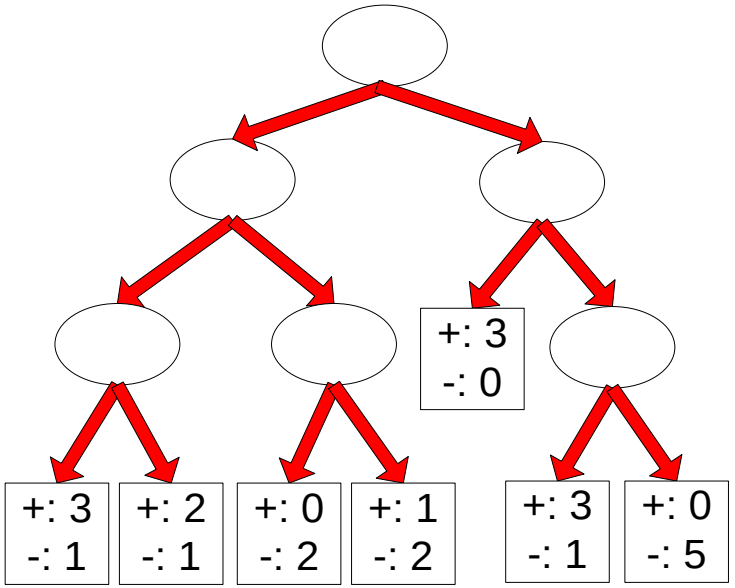
评估决策树的复杂度
Estimating the Complexity of Decision Trees

具有 *k* 个叶节点的决策树 T 的**悲观错误估计（Pessimistic Error Estimate）：**

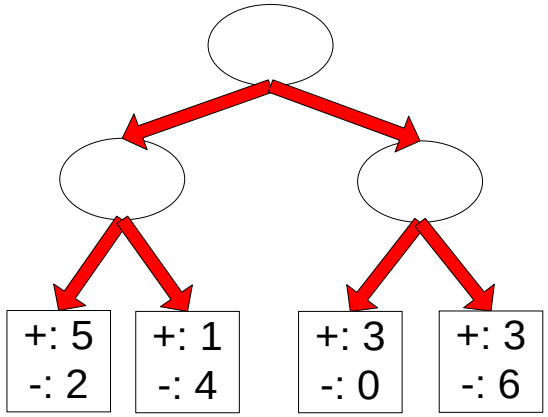
$$err_{gen}(T) = err(T) + \Omega \times \frac{k}{N_{train}}$$

- err(T): 训练集上的错误率
- Ω: 超参数(类似于 α)
  - 增加叶节点的关联成本（惩罚）
- k: 叶节点个数
- N_{train}: 训练样本总数

评估决策树的复杂度：实例
Estimating the Complexity of Decision Trees: Example



Decision Tree, T_L



Decision Tree, T_R

训练误差

$e(T_L) = 4/24$

$e(T_R) = 6/24$

$\Omega = 1$

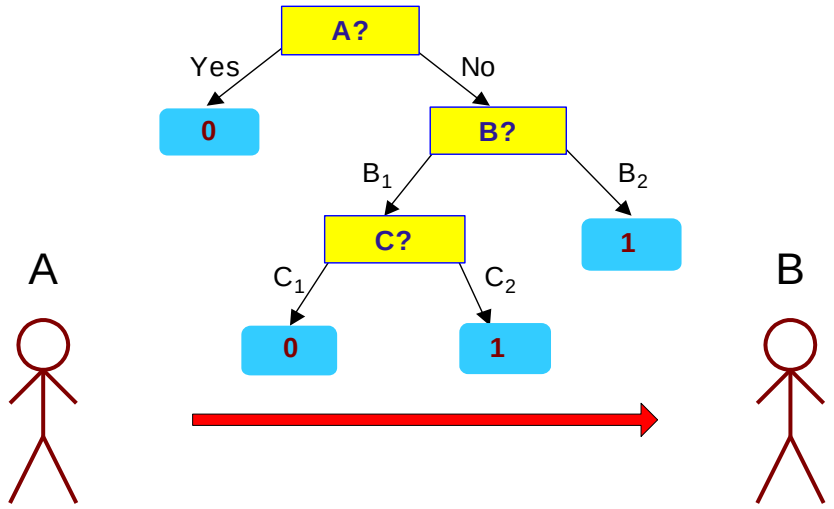
悲观错误估计--泛化误差

$e_{gen}(T_L) = 4/24 + 1*7/24 = 11/24 = 0.458$

$e_{gen}(T_R) = 6/24 + 1*4/24 = 10/24 = 0.417$

最小描述长度原则 Minimum Description Length (MDL)

X	y
X_1	1
X_2	0
X_3	0
X_4	1
…	…
X_n	1



X	y
X_1	?
X_2	?
X_3	?
X_4	?
…	…
X_n	?

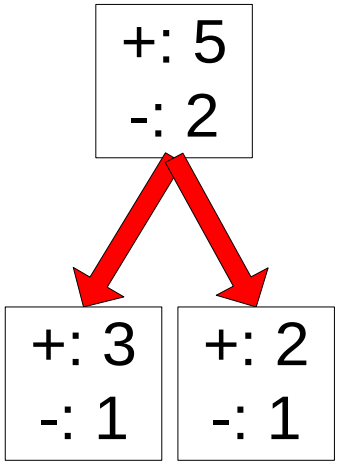
Cost(Model,Data) = Cost(Data|Model) + α x Cost(Model)

- Cost 是编码所需的比特数
- 搜索传输成本最低的模型

Cost(Data|Model)：误分类记录编码的开销。

Cost(Model)：模型编码开销，此例中包括子节点编码和分裂条件的编码。

估计统计上界 Estimating Statistical Bounds



错误率统计上界： $e'(N,e,\alpha) = \dfrac{e + \dfrac{z_{\alpha/2}^2}{2N} + z_{\alpha/2}\sqrt{\dfrac{e(1-e)}{N} + \dfrac{z_{\alpha/2}^2}{4N^2}}}{1 + \dfrac{z_{\alpha/2}^2}{N}}$

分裂前： e = 2/7， e’(7, 2/7, 0.25) = 0.503

总训练误差： e’(T) = 7 × 0.503 = 3.521

分裂后：

$e(T_L)$ = 1/4， e’(4, 1/4, 0.25) = 0.537

$e(T_R)$ = 1/3， e’(3, 1/3, 0.25) = 0.650

总训练误差： e’(T) = 4 × 0.537 + 3 × 0.650 = 4.098

因此，不分裂

决策树模型选择 Model Selection for Decision Trees

预剪枝Pre-Pruning （提前终止规则 Early Stopping Rule)

- 在成为完整树之前停止算法
- 节点的典型停止条件：
  - 如果所有实例都属于同一类，则停止
  - 如果所有属性值都相同，则停止
- 更多限制条件：
  - 如果实例数量小于用户指定的阈值，则停止
  - 如果实例的类分布独立于可用特征，则停止(e.g., 使用 χ^2 测试)
  - 如果扩展当前节点不能改进不纯度，则停止 (e.g., 基尼系数或信息增益).
  - 如果估计的泛化误差低于某个阈值，则停止

决策树模型选择 Model Selection for Decision Trees

后剪枝 Post-pruning

- 全面增长决策树
- 替换子树
  - 以自下而上的方式修剪决策树的节点
  - 如果修剪后泛化错误有所改善，请用叶节点替换子树
  - 叶节点的类标签由子树中的大多数实例类决定
- 子树提升Subtree raising
  - 用子树中最常使用的分支代替子树
  - Subtree raising selects a subtree and replaces it with the child one (i.e., a "sub-subtree" replaces its parent)

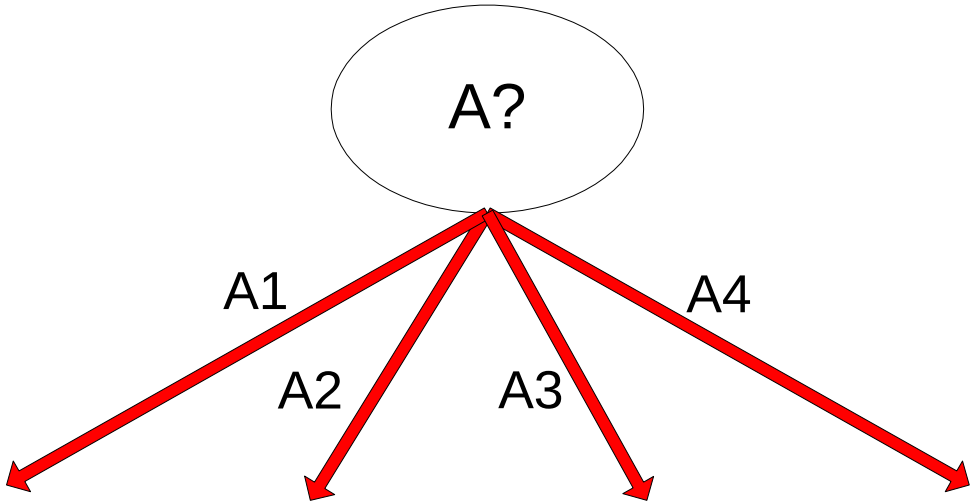
Post-Pruning 例子

Class = Yes	20
Class = No	10
Error = 10/30	

训练误差(分裂前) = 10/30
悲观错误Pessimistic error = (10 + 0.5)/30 = 10.5/30

训练误差(分裂后) = 9/30
悲观错误(分裂后) = (9 + 4 × 0.5)/30 = 11/30

剪枝? **剪枝!**



Class = Yes	8
Class = No	4

Class = Yes	3
Class = No	4

Class = Yes	4
Class = No	1

Class = Yes	5
Class = No	1

Post-Pruning 例子

Decision Tree:

```
depth = 1 :
|   breadth > 7 : class 1
|   breadth <= 7 :
|   |   breadth <= 3 :
|   |   |   ImagePages > 0.375 : class 0
|   |   |   ImagePages <= 0.375 :
|   |   |   |   totalPages <= 6 : class 1
|   |   |   |   totalPages > 6 :
|   |   |   |   |   breadth <= 1 : class 1
|   |   |   |   |   breadth > 1 : class 0
|   |   width > 3 :
|   |   |   MultiIP = 0:
|   |   |   |   ImagePages <= 0.1333 : class 1
|   |   |   |   ImagePages > 0.1333 :
|   |   |   |   |   breadth <= 6 : class 0
|   |   |   |   |   breadth > 6 : class 1
|   |   |   MultiIP = 1:
|   |   |   |   TotalTime <= 361 : class 0
|   |   |   |   TotalTime > 361 : class 1
depth > 1 :
|   MultiAgent = 0:
|   |   depth > 2 : class 0
|   |   depth <= 2 :
|   |   |   MultiIP = 1: class 0
|   |   |   MultiIP = 0:
|   |   |   |   breadth <= 6 : class 0
|   |   |   |   breadth > 6 :
|   |   |   |   |   RepeatedAccess <= 0.0322 : class 0
|   |   |   |   |   RepeatedAccess > 0.0322 : class 1
|   MultiAgent = 1:
|   |   totalPages <= 81 : class 0
|   |   totalPages > 81 : class 1
```

Subtree Raising

Subtree Replacement

Simplified Decision Tree:

```
depth = 1 :
|   ImagePages <= 0.1333 : class 1
|   ImagePages > 0.1333 :
|   |   breadth <= 6 : class 0
|   |   breadth > 6 : class 1
depth > 1 :
|   MultiAgent = 0: class 0
|   MultiAgent = 1:
|   |   totalPages <= 81 : class 0
|   |   totalPages > 81 : class 1
```

模型评估 Model Evaluation

目的:

- 评估分类器在未知数据（测试集）上的性能

保持（Holdout）

- k% 作为训练集，(100-k)% 作为测试集
  - 用于训练的被标记样本较少
  - 模型可能高度依赖于训练集和检验集的构成

- 随机二次抽样（Random subsampling）：重复的 holdout

模型评估 Model Evaluation

目的:

- 评估分类器在未知数据（测试集）上的性能

保持（Holdout）

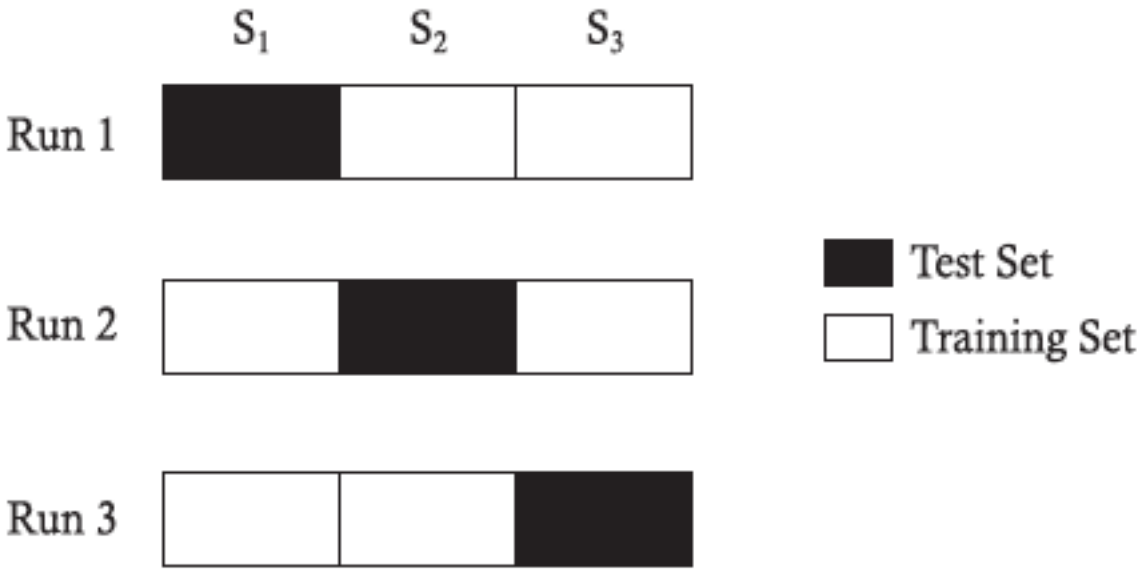
- k% 作为训练集，(100-k)% 作为测试集
- 随机二次抽样（Random subsampling）: 重复的 holdout

交叉验证 Cross validation

- 每个记录用于训练的次数相同，并且恰好检验一次
- 将数据分为k个不相交子集
- k折（k-fold）: 在 k-1 个子集上训练, 在另外一个子集上测试
- 留一（Leave-one-out）： k=n，n是数据集大小

交叉验证示例 Cross-validation Example

3-fold cross-validation



交叉验证的变体 Variations on Cross-validation

重复交叉验证Repeated cross-validation

- 多次执行交叉验证
- 给出广义误差方差（the variance of the generalization error）的估计

分层交叉验证Stratified cross-validation

- 保证训练和测试中类别标签的百分比相同
- **当类别不平衡且样本很小时，这一点很重要**

使用嵌套（nested）的交叉验证方法进行模型选择和评估

作业

杭电网络教学平台上

1. 注意截止时间
2. 需要提交（而非仅保存）

谢谢！

数据挖掘
教师：王东京
学院：计算机学院
邮箱：dongjing.wang@hdu.edu.cn