分类与预测

2021/10/11

挖掘建模

- 经过数据探索与数据预处理部分，得到了可以直接建模的数据。根据挖掘目标和数据形式可以建立**分类与预测**、**聚类分析**、**关联规则**、**时序模式**、**偏差检测**、**智能推荐**等模型，帮助企业提取数据中蕴含的商业价值，提高企业的竞争力。

目录

分类与预测

- 就餐饮企业而言，经常会碰到这样的问题：

  1）如何基于菜品历史销售情况，以及节假日、气候和竞争对手等影响因素，对菜品销量进行趋势预测？

  2）如何预测在未来一段时间哪些顾客会流失，哪些顾客最有可能会成为VIP客户？

  3）如何预测一种新产品的销售量，以及在哪种类型的客户中会较受欢迎？

- 除此之外，餐厅经理需要通过数据分析来帮助他了解具有某些特征的顾客的消费习惯；餐饮企业老板希望知道下个月的销售收入，原材料采购需要投入多少，这些都是分类与预测的例子。

分类与预测——实现过程

- **分类**：通过学习得到一个目标函数F，把数据X映射到预先定义好的类标号（离散、无序的）Y。
- **预测**：建立连续值函数模型F，预测给定自变量X的条件下因变量Y的值。

$$F(X) \rightarrow Y$$

X – 训练数据

Y – 标签

F – 模型

分类： Y为离散值

预测： Y为连续值

分类与预测——实现过程

- 分类和预测的实现过程类似，以分类模型为例，实现过程如图：

◆ 训练集（Training data）：数据库中为建立模型而被分析的数据元组形成训练集。

◆ 验证集：用于在训练过程中检验模型的状态，收敛情况。验证集通常用于调整超参数，根据几组模型验证集上的表现决定哪组超参数拥有最好的性能。

◆ 测试集：用于评估分类模型的准确率。

◆ 训练集就像是学生的课本，学生 根据课本里的内容来掌握知识，验证集就像是作业，通过作业可以知道 不同学生学习情况、进步的速度快慢，而最终的测试集就像是考试，考的题是平常都没有见过，考察学生举一反三的能力。

分类与预测——实现过程

分类算法有**两步过程**：

1. 第一步，建立一个模型，描述预定数据类集和概念集
   - 假定每个元组属于一个预定义的类，由一个类标号属性确定
   - 学习模型可以用分类规则、决策树或数学公式的形式提供
2. 使用模型，对将来的或未知的对象进行分类
   - 首先评估模型的预测准确率
     - 对每个测试样本，将已知的类标号和该样本的学习模型类预测比较
     - 模型在给定测试集上的准确率是正确被模型分类的测试样本的百分比
     - 测试集要独立于训练样本集，否则会出现“过分适应数据”的情况

预测模型的实现也有两步：

1. 第一步是通过训练集建立预测属性（数值型的）的函数模型
2. 第二步是预测，模型通过检验后再进行预测或控制。

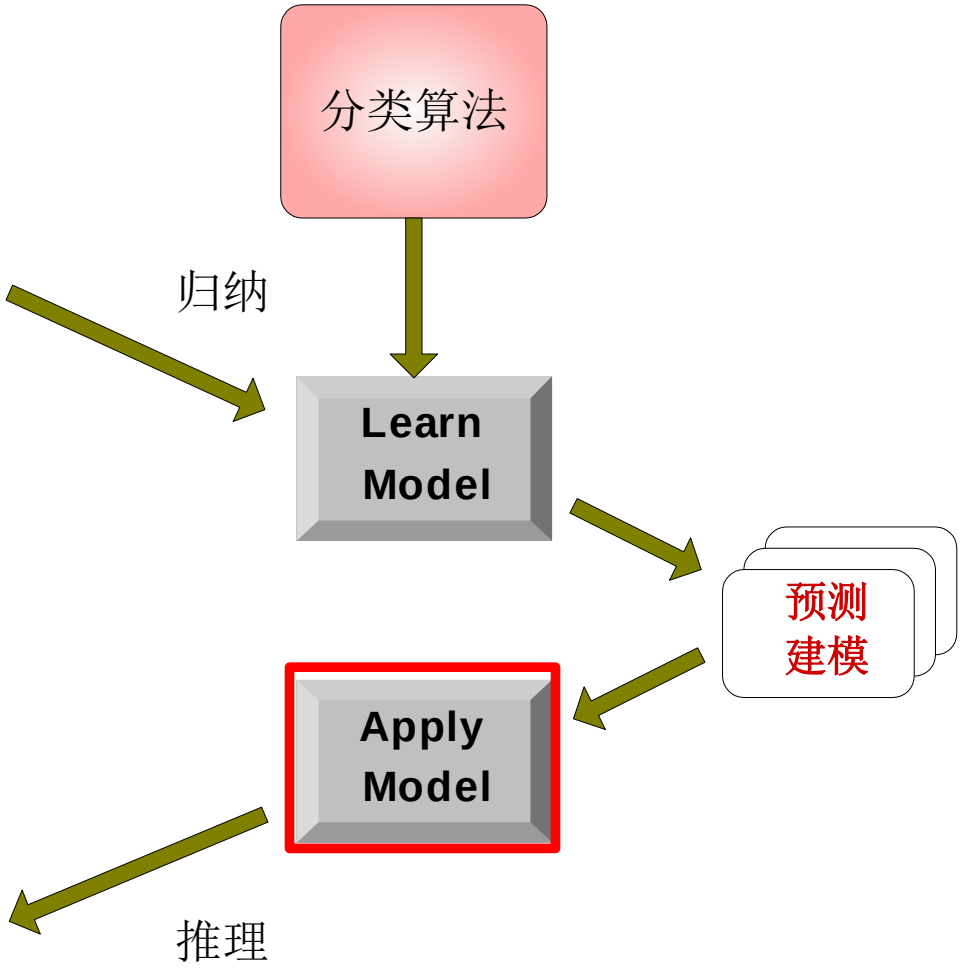
分类与预测——实现过程

Tid	偿还借款	婚姻状况	年收入	是否欺诈
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

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



有监督的学习 VS. 无监督的学习

- 有监督的学习（用于分类）
  - 模型的学习在被告知每个训练样本属于哪个类的“监督”下进行
  - 新数据使用训练数据集中得到的规则进行分类
- 无监督的学习（用于聚类）
  - 每个训练样本的类编号是未知的，要学习的类集合或数量也可能是事先未知的
  - 通过一系列的度量、观察来建立数据中的类编号或进行聚类

分类与预测算法评价

- 分类与预测模型对训练集进行预测而得出的准确率并不能很好地反映预测模型未来的性能，为了有效判断一个预测模型的性能表现，需要一组没有参与预测模型建立的数据集，并在该数据集上评价预测模型的准确率，这组独立的数据集叫测试集。

分类算法评价

- 混淆矩阵

混淆矩阵（Confusion Matrix）是模式识别领域中一种常用的表达形式。它描绘样本数据的真实属性与识别结果类型之间的关系，是评价分类器性能的一种常用方法。

混淆矩阵		真实值	
		Positive	Negative
预测值	Positive	TP	FP（Type II）
	Negative	FN（Type I）	TN

1）True positives(TP)： 被正确地划分为正例的个数，即实际为正例且被分类器划分为正例的实例数；
2）False positives(FP)：被错误地划分为正例的个数，即实际为负例但被分类器划分为正例的实例数；
3）False negatives(FN)：被错误地划分为负例的个数，即实际为正例但被分类器划分为负例的实例数；
4）True negatives(TN)：被正确地划分为负例的个数，即实际为负例且被分类器划分为负例的实例数。

分类与预测算法评价

正确率 （**Accuracy**） accuracy = （TP+TN）/(P+N)

分对的样本数除以所有的样本数，通常来说，正确率越高，分类器越好；

错误率（**Error rate**) error rate = (FP+FN)/(P+N)

分错的样本比率。 error rate =1 - accuracy

灵敏度（**Sensitive**） sensitive = TP/(TP+FN)

表示的是所有正例中被分对的比例，衡量了分类器对正例的识别能力；

特效度（**Specificity**) specificity = TN/(TN+FP)

表示的是所有负例中被分对的比例，衡量了分类器对负例的识别能力；

精度（Precision） precision=TP/（TP+FP）

精确性的度量，表示被分为正例的示例中实际为正例的比例

召回率（Recall） recall=TP/(TP+FN)=sensitive

覆盖面的度量，度量有多个正例被分为正例，与sensitive一样

负正类率（FPR） FPR=FP/(FP+TN)

分类器错认为正类的负实例占所有负实例的比例

混淆矩阵		真实值	
		Positive	Negative
预测值	Positive	TP	FP (Type II)
	Negative	FN (Type I)	TN

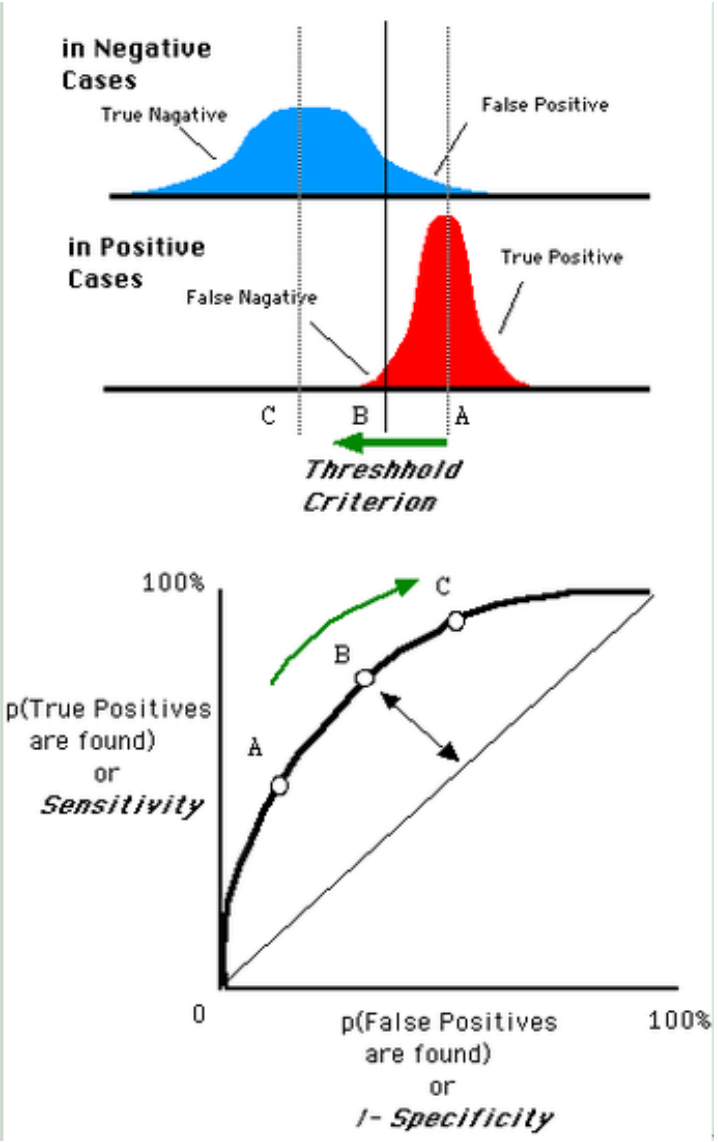
分类与预测算法评价

- ROC曲线

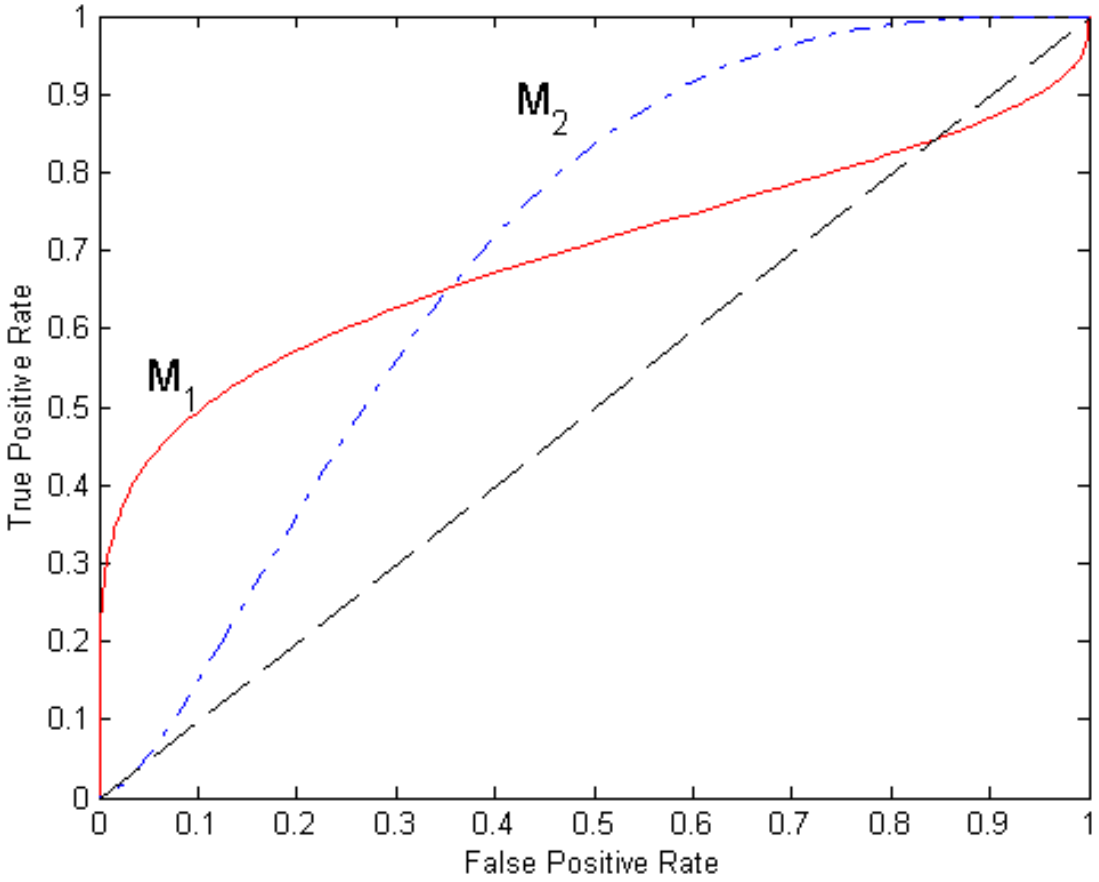
受试者工作特性（Receiver Operating Characteristic，ROC）曲线是一种非常有效的模型评价方法，可为选定临界值给出定量提示。将灵敏度（Sensitivity）设在纵轴，1-特异性（1-Specificity，也就是FPR ）设在横轴，就可得出ROC曲线图。该曲线下的积分面积（Area）大小与每种方法优劣密切相关，反映分类器正确分类的统计概率，其值越接近1说明该算法效果越好。

ROC 曲线上有几个关键点：

- （TPR=0，FPR=0）：把每个实例都预测为负类的模型
- （TPR=1，FPR=1）：把每个实例都预测为正类的模型
- （TPR=1，FPR=0）：理想模型



使用ROC曲线比较模型

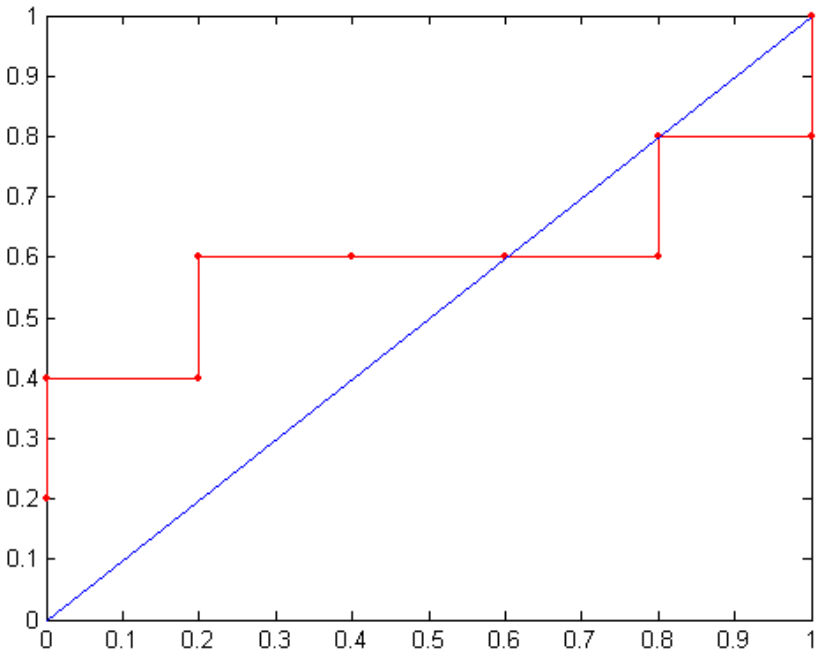


- 没有哪个模型能够压倒对方
  - FRR<0.36,M_1 较好
  - FRR>0.36, M_2较好
- ROC曲线下方的面积
  - 理想情况:
    - 面积= 1
  - 随机猜测:
    - 面积 = 0.5

怎样产生ROC曲线

Class	+	-	+	-	-	-	+	-	+	+	
Threshold >=	0.25	0.43	0.53	0.76	0.85	0.85	0.85	0.87	0.93	0.95	1.00
TP	5	4	4	3	3	3	3	2	2	1	0
FP	5	5	4	4	3	2	1	1	0	0	0
TN	0	0	1	1	2	3	4	4	5	5	5
FN	0	1	1	2	2	2	2	3	3	4	5
TPR	1	0.8	0.8	0.6	0.6	0.6	0.6	0.4	0.4	0.2	0
FPR	1	1	0.8	0.8	0.6	0.4	0.2	0.2	0	0	0

ROC 曲线:



预测算法评价指标

- 绝对误差与相对误差

设 Y表示实际值， $\hat{Y}$表示预测值，则称 E为绝对误差（AbsoluteError），计算公式如下：

$$E = Y - \hat{Y}$$

e为相对误差（RelativeError），计算公式如下：

$$e = \frac{Y - \hat{Y}}{Y}$$

有时相对误差也用百分数表示：

$$e = \frac{Y - \hat{Y}}{Y} * 100\%$$

预测算法评价

- 平均绝对误差

平均绝对误差（MeanAbsoluteError，MAE）定义如下：

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|E_i| = \frac{1}{n}\sum_{i=1}^{n}|Y_i - \hat{Y}_i|$$

式中各项的含义如下：

- MAE ：平均绝对误差
- E_i ：第 i个实际值与预测值的绝对误差
- Y_i ：第 i个实际值
- $\hat{Y}_i$ ：第 i个预测值

由于预测误差有正有负，为了避免正负相抵消，故取误差的绝对值进行综合并取其平均数，这是误差分析的综合指标法之一。

预测算法评价

- 均方误差

均方误差（MeanSquaredError，MSE）定义如下：

$$MSE = \frac{1}{n}\sum_{i=1}^{n} E_i^2 = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

式中，MSE表示均方差，其他符号同前。

本方法用于还原平方失真程度。

均方误差是预测误差平方之和的平均数，它避免了正负误差不能相加的问题。由于对误差 进行了平方，加强了数值大的误差在指标中的作用，从而提高了这个指标的灵敏性，是一大优点。均方误差是误差分析的综合指标法之一。

预测算法评价

- 均方根误差

均方根误差（RootMeanSquaredError，RMSE）定义如下：

$$RMSE = \sqrt{\frac{1}{n}\sum_{i=1}^{n} E_i^2} = \sqrt{\frac{1}{n}\sum_{i=1}^{n} (Y_i - \hat{Y}_i)^2}$$

式中，$RMSE$表示均方根误差，其他符号同前。

这是均方误差的平方根，代表了预测值的离散程度，也叫标准误差，最佳拟合情况为 。均方根误差也是误差分析的综合指标之一。

- 平均绝对百分误差

平均绝对百分误差（MeanAbsolute PercentageError，MAPE）定义如下：

$$MAPE = \frac{1}{n}\sum_{i=1}^{n} |E_i/Y_i| = \frac{1}{n}\sum_{i=1}^{n} |(Y_i - \hat{Y}_i)/Y_i|$$

式中，$MAPE$表示平均绝对百分误差。一般认为 小于10时，预测精度较高。

目录

分类与预测——常用的分类与预测算法

- 主要分类与预测算法简介：

算法名称	算法描述
决策树	它采用自顶向下的递归方式，在决策树的内部结点进行属性值的比较，并根据不同的属性值从该结点向下分支，叶结点是要学习划分的类。
回归分析	回归分析是确定预测属性（数值型）与其他变量间相互依赖的定量。关系的最常用的统计学方法。包括线性回归、非线性回归、Logistic回归、岭回归、主成分回归、偏最小二乘回归等模型。
贝叶斯网络	贝叶斯网络又称信度网络，是Bayes方法的扩展，是目前不确定知识表达和推理领域最有效的理论模型之一。
支持向量机	SVM支持向量机根据有限的样本信息在模型的复杂性和学习能力之间寻求最佳折衷，以获得最好的推广能力。
人工神经网络	一种模仿大脑神经网络结构和功能而建立的信息处理系统，表示神经网络的输入与输出变量之间关系的模型。

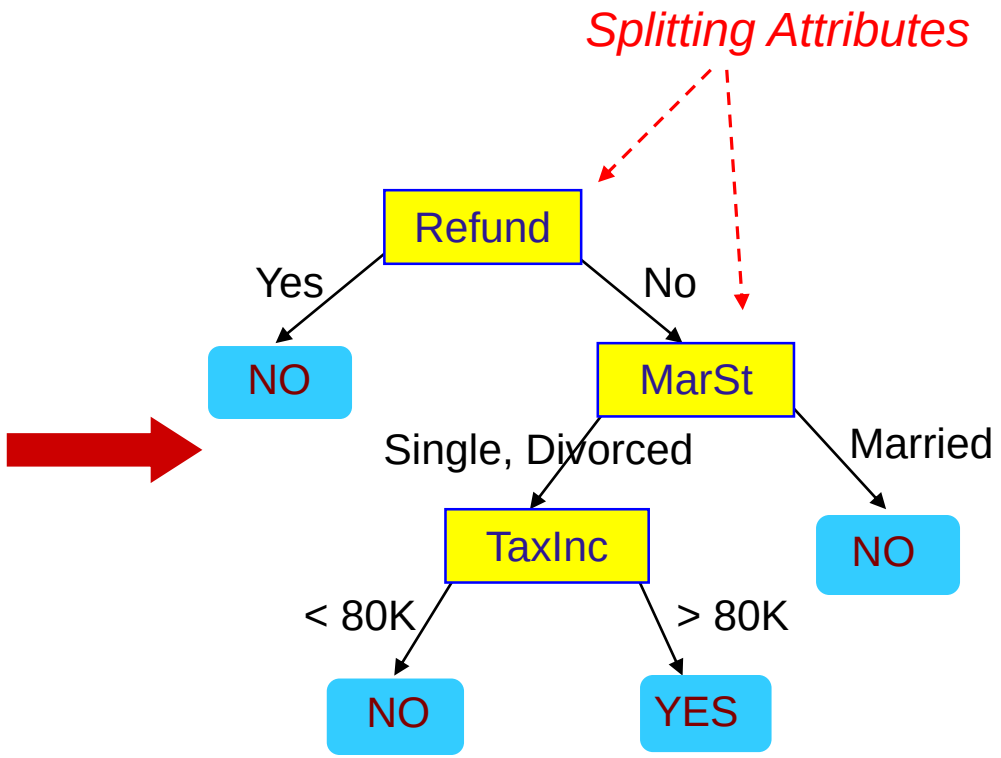
分类与预测——决策树

- 决策树方法在分类、预测、规则提取等领域有着广泛应用。在20世纪70年代后期和80年代初期，机器学习研究者J.Ross Quinilan提出了ID3算法以后，决策树在机器学习、数据挖掘邻域得到极大的发展。
- Quinilan后来又提出了C4.5，成为新的监督学习算法的性能比较基准。1984年几位统计学家提出了CART分类算法。ID3和ART算法大约同时被提出，但都是采用类似的方法从训练元组中学习决策树。
- 什么是决策树？
  - 类似于流程图的树结构
  - 每个内部节点表示在一个属性上的测试
  - 每个分枝代表一个测试输出
  - 每个树叶节点代表类或类分布

一个决策树的例子

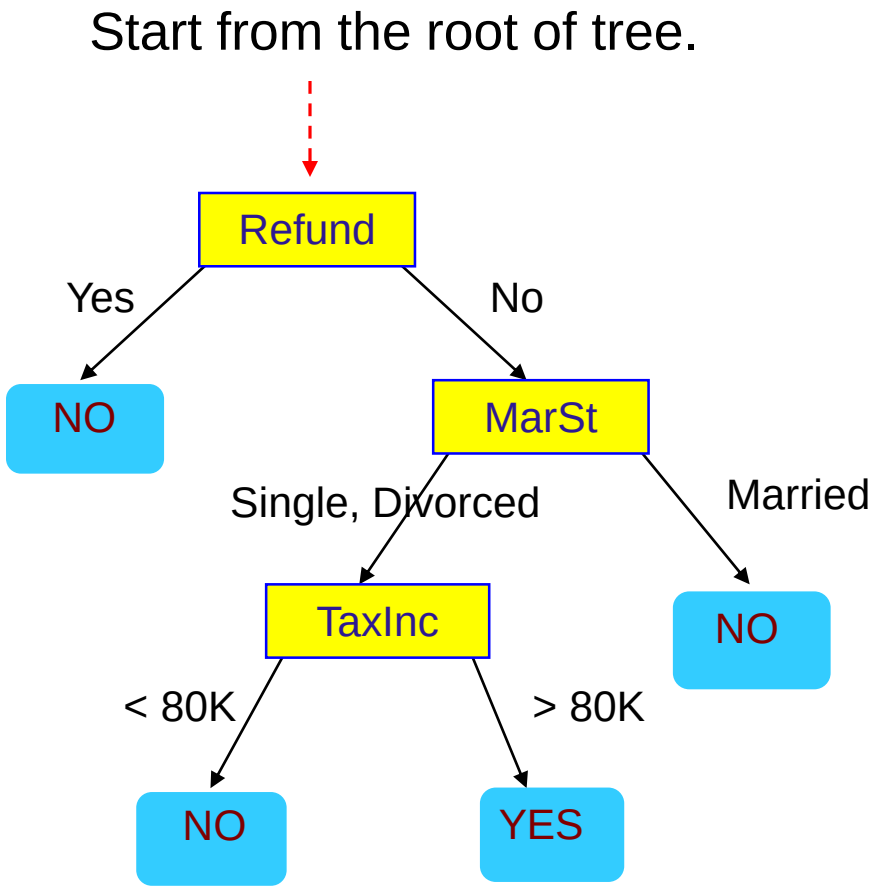
	categorical	categorical	continuous	class
Tid	**Refund**	**Marital Status**	**Taxable Income**	**Cheat**
1	Yes	Single	125K	**No**
2	No	Married	100K	**No**
3	No	Single	70K	**No**
4	Yes	Married	120K	**No**
5	No	Divorced	95K	**Yes**
6	No	Married	60K	**No**
7	Yes	Divorced	220K	**No**
8	No	Single	85K	**Yes**
9	No	Married	75K	**No**
10	No	Single	90K	**Yes**

训练数据



模型：决策树

应用决策树进行分类



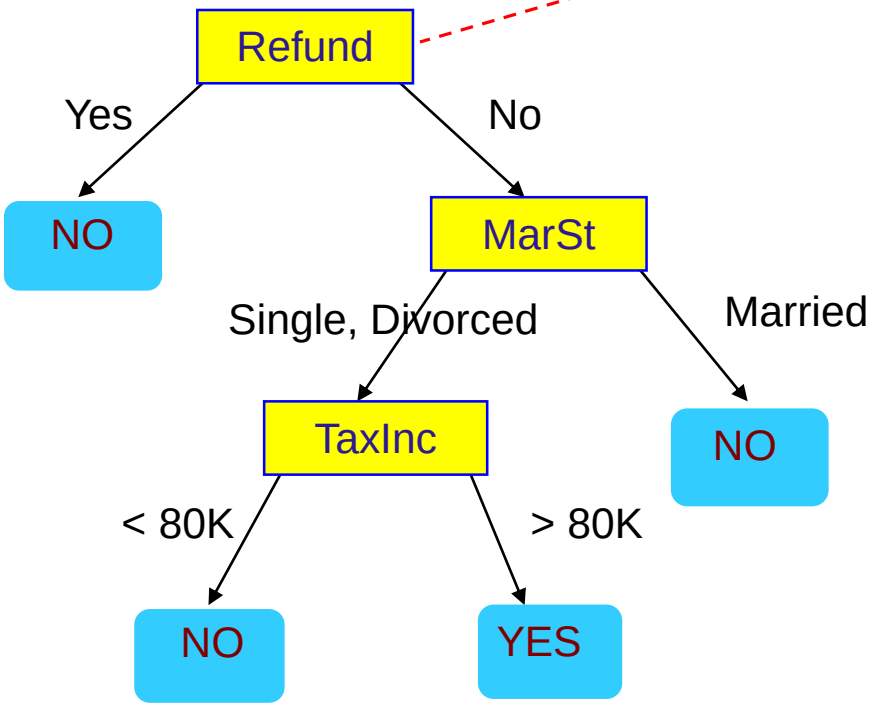
测试数据

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?

应用决策树进行分类

测试数据

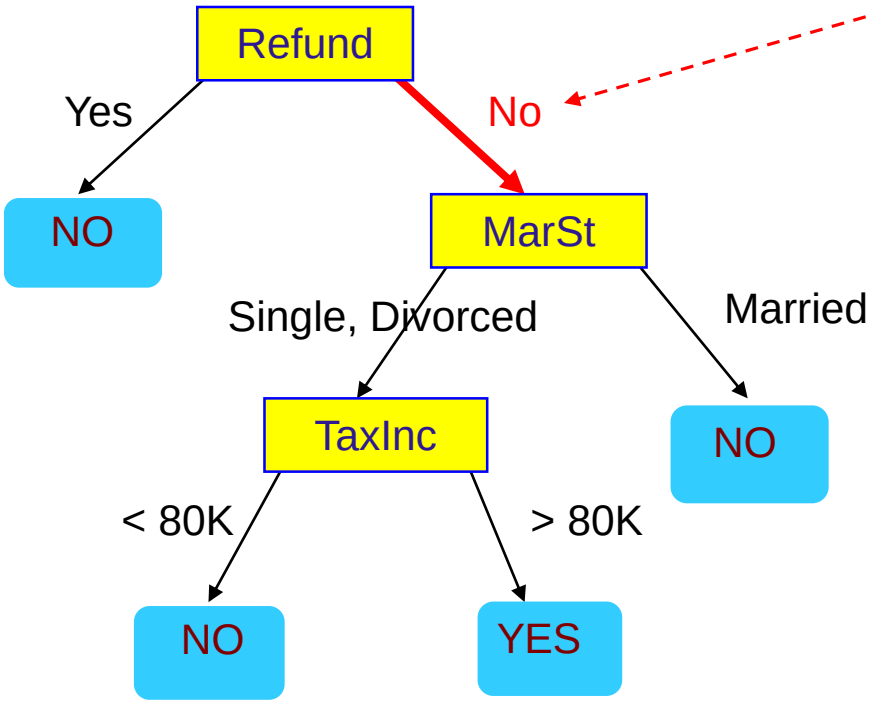
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



应用决策树进行分类

测试数据

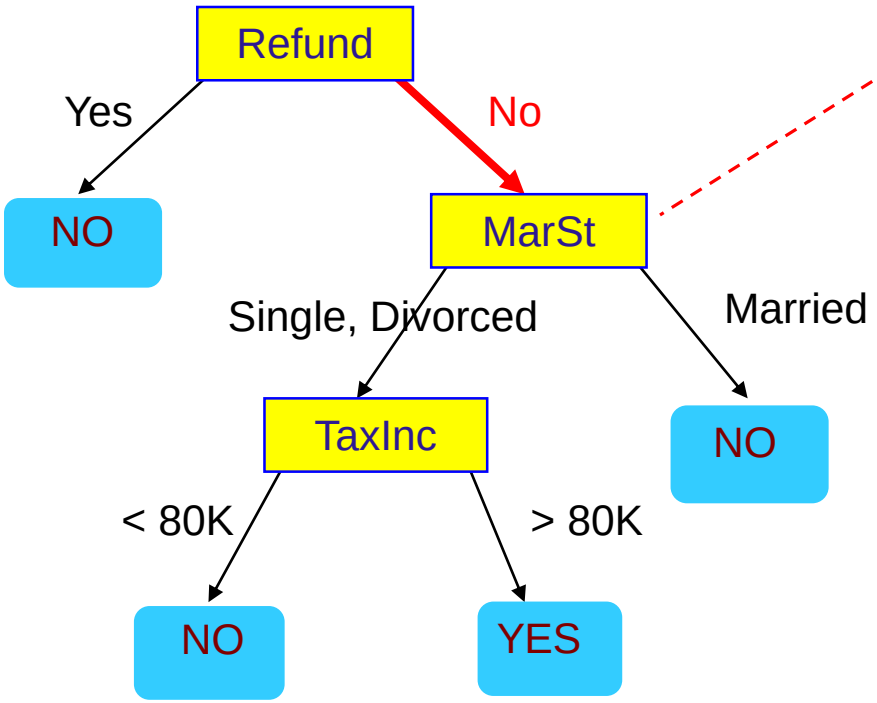
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



应用决策树进行分类

测试数据

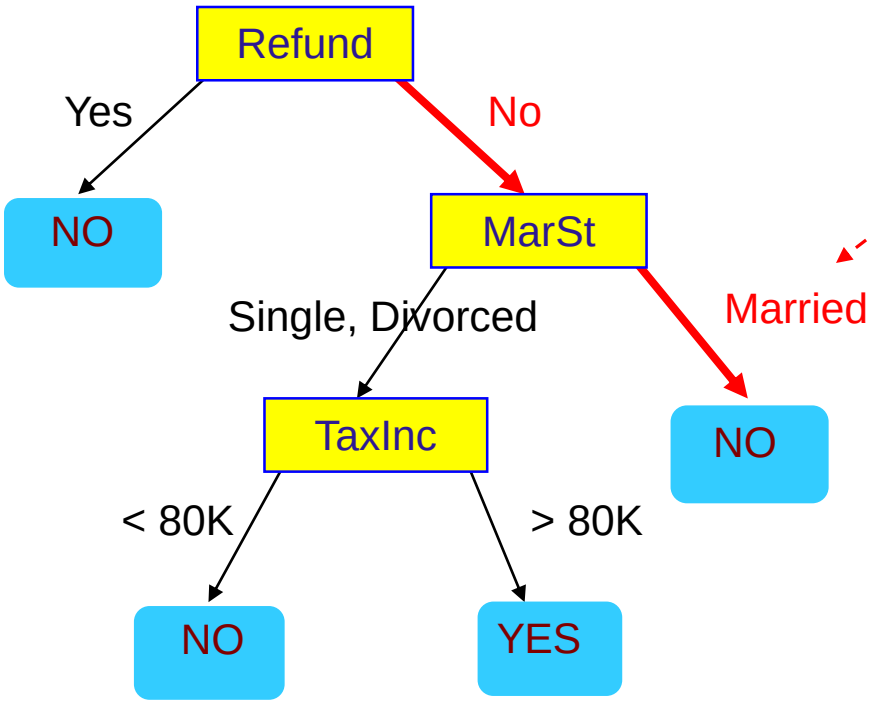
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



应用决策树进行分类

测试数据

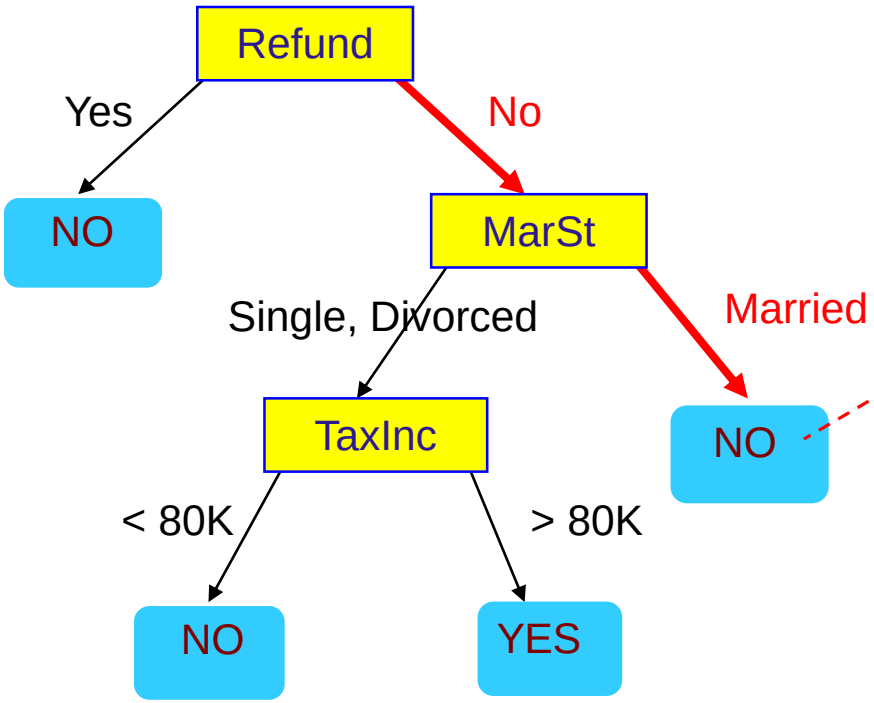
Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



应用决策树进行分类

测试数据

Refund	Marital Status	Taxable Income	Cheat
No	Married	80K	?



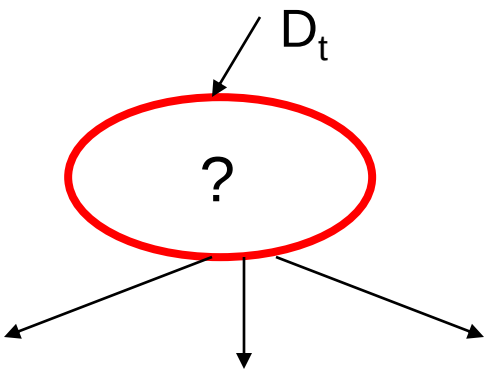
Assign Cheat to “No”

基本原理

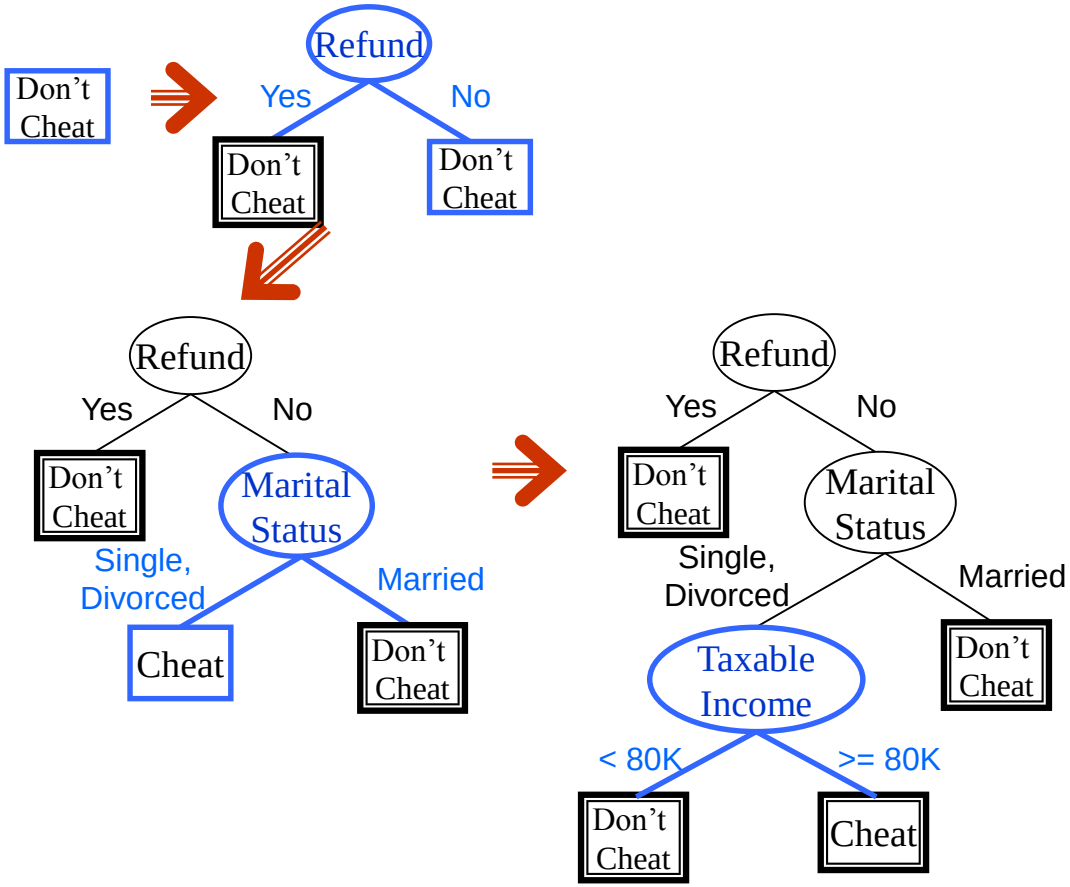
- 采用贪心策略构建决策树.

  在选择划分数据的属性时，采取一系列局部最优决策来构造决策树.
- 设 D_t 是与结点 t相关联的训练记录集
- 算法步骤:
  - 如果D_t 中所有记录都属于同一个类 y_t，则t是叶结点，用y_t标记
  - 如果 D_t 中包含属于多个类的记录，则选择一个属性测试条件，将记录划分成较小的子集。对于测试条件的每个输出，创建一个子结点，并根据测试结果将D_t中的记录分布到子结点中。然后，对于每个子结点，递归地调用该算法

Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes



基本原理

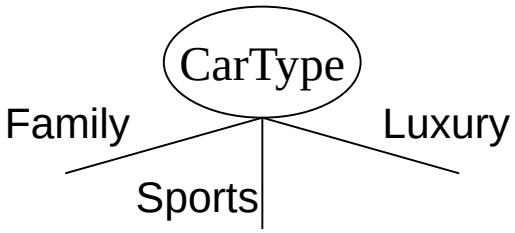


Tid	Refund	Marital Status	Taxable Income	Cheat
1	Yes	Single	125K	No
2	No	Married	100K	No
3	No	Single	70K	No
4	Yes	Married	120K	No
5	No	Divorced	95K	Yes
6	No	Married	60K	No
7	Yes	Divorced	220K	No
8	No	Single	85K	Yes
9	No	Married	75K	No
10	No	Single	90K	Yes

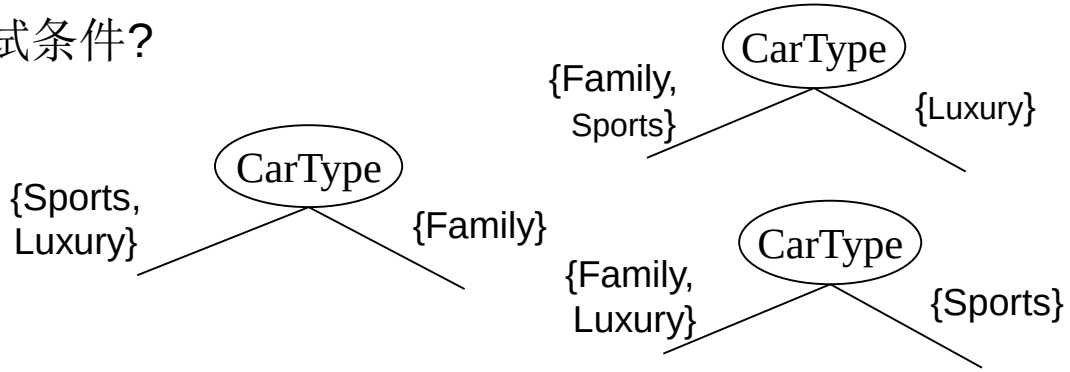
- 如何分裂训练记录
  - 怎样为不同类型的属性指定测试条件?
  - 怎样评估每种测试条件?
- 如何停止分裂过程

如何分裂训练记录？

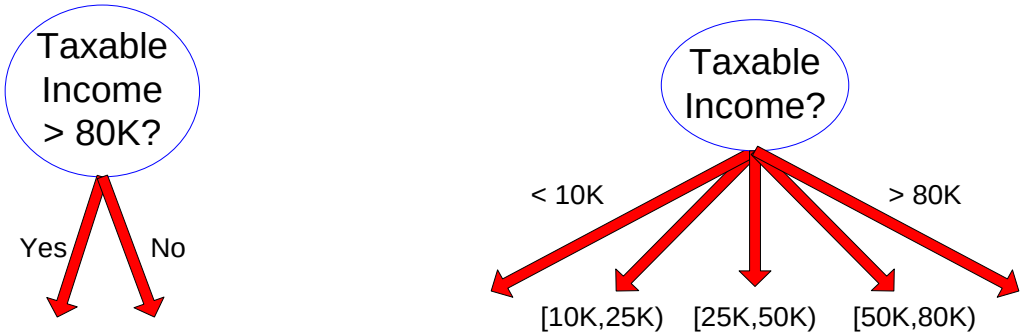
➢怎样为不同类型的属性指定测试条件?



离散值：多路划分



离散值：二路划分



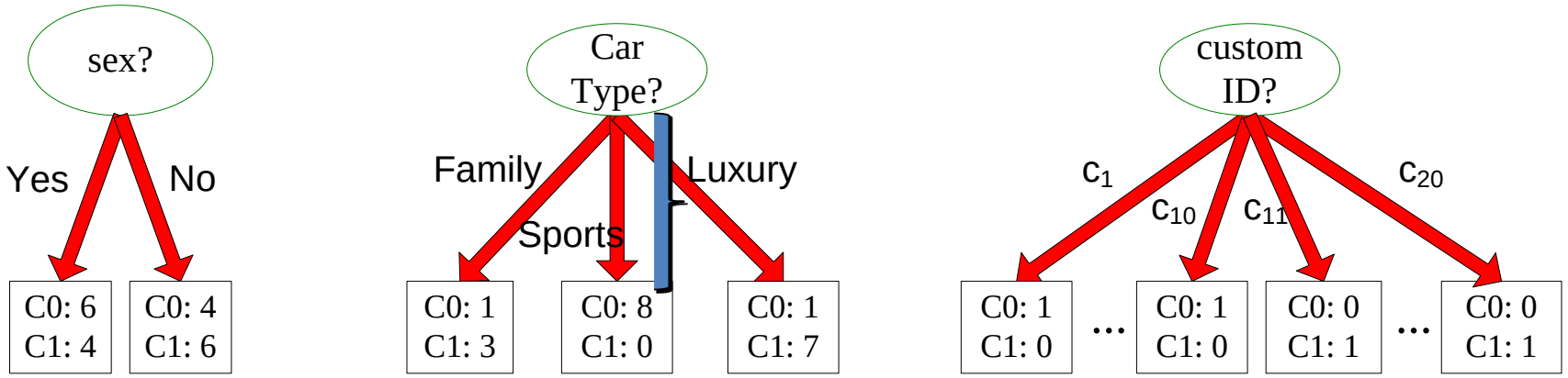
(i) Binary split

(ii) Multi-way split

连续值：二路划分/多路划分

怎样评估每种测试条件?

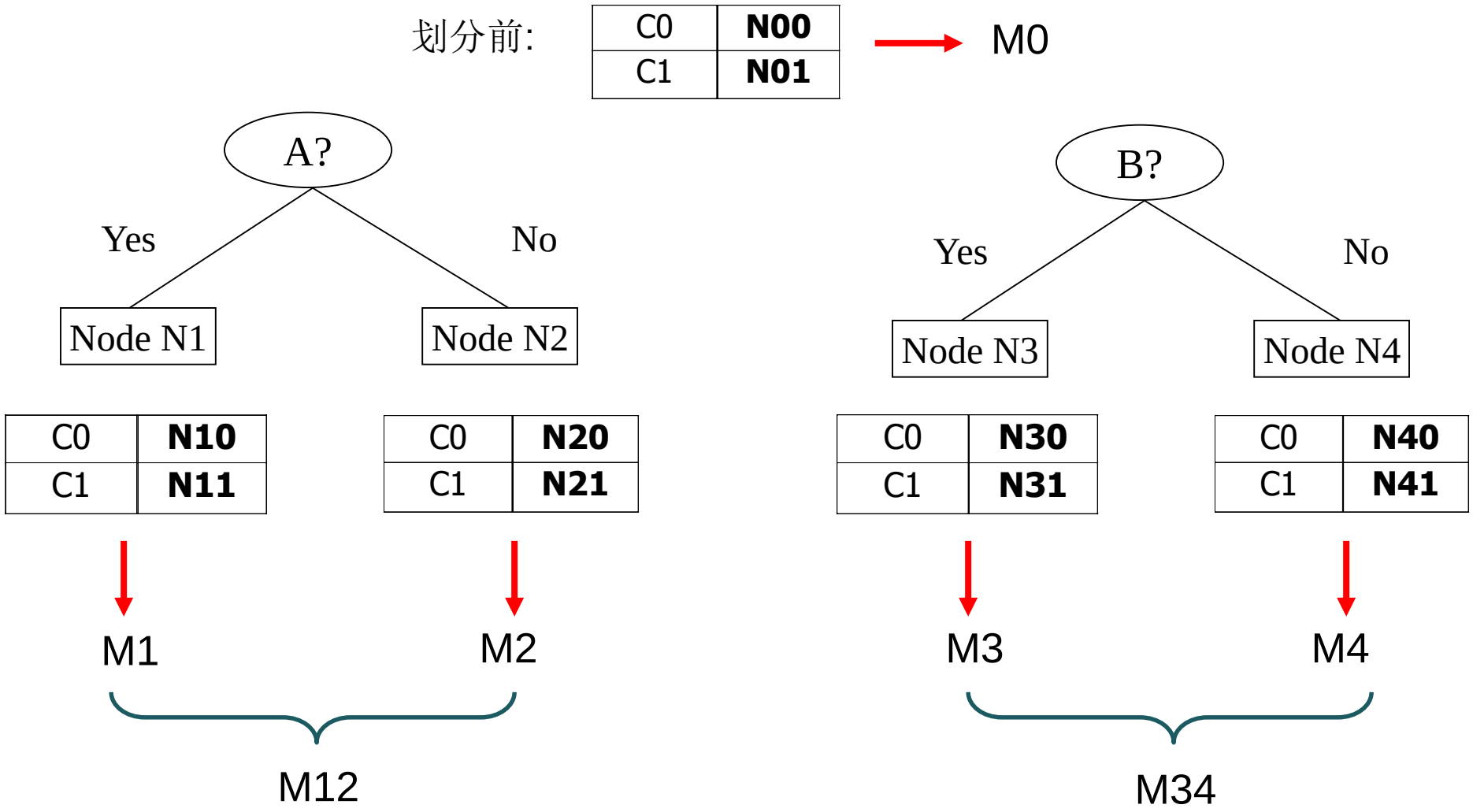
- 选择最佳划分的度量通常是根据划分后子结点不纯性的程度。
- 不纯性越小，类分布随机性越小，可预测性越大



节点不纯性的测量方法

- Gini
- Entropy
- Classification error

怎样评估每种测试条件?



Gain = M0 – M12 vs M0 – M34

Gain越大越好，也就是新划分的纯度越小越好

不纯性的测量：Gini

◆ 给定结点t的Gini值计算：

$$GINI(t) = 1 - \sum_j [p(j|t)]^2$$

($p(j \mid t)$ 是在结点t中，类j发生的概率).

– 当类分布均衡时，Gini值达到最大值 (1 - 1/k) : k为类的个数

– 相反当只有一个类时，Gini值达到最小值0

C1	0
C2	6
Gini=0.000	

C1	1
C2	5
Gini=0.278	

C1	2
C2	4
Gini=0.444	

C1	3
C2	3
Gini=0.500	

◆ 当一个结点 t 分割成 k 个部分 (孩子), 划分的质量可由下面公式计算

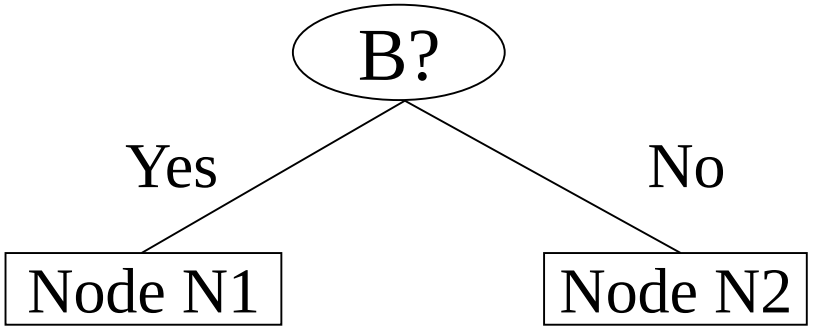
$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

n_i = 孩子结点 i的记录数,

n = 父结点 t的记录数.

计算Gini的例子

	Parent
C1	6
C2	6
Gini = 0.500	



	N1	N2
C1	5	1
C2	2	4
Gini=0.368		

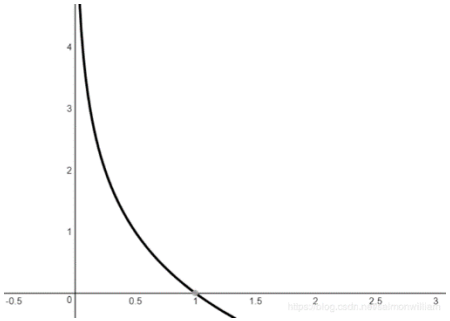
Gini(N1)
$= 1 - (5/7)^2 - (2/7)^2 = 0.403$

Gini(N2)
$= 1 - (1/5)^2 - (4/5)^2 = 0.32$

$\text{Gini}_{\text{split}}$
$= 7/12 * 0.403 + 5/12 * 0.32$
$= 0.368$

不纯性的测量：Entropy

- 定义：给定一个概率空间 $\{X, \chi, q(x)\}$事件$x_k \in \chi$ 的自信息定义为$\mathrm{I}(x_k) = -log_2^{q_k}$ 因$q_k \in [0,1]$故 $\mathrm{I}(x_k) \geq 0$



自信息反映了事件 x_k 发生所需要的信息量。

$\mathrm{I}(x_k)$值越大说明需要越多的信息才能确定事件 x_k 的发生，其随机性也越大，而当 x_k发生时所携带的信息量也越大。反过来，$\mathrm{I}(x_k)$ 值越小，需要较少信息量就能确定x_k 的发生，即事件随机性较小，当其发生时所携信息量就少。$\mathrm{I}(x_k)$是对不确定性大小的一种刻画。

- 定义：在概率空间 $\{X, \chi, q(x)\}$ 上定义的随机变量I(X)的数学期望

$$\mathrm{E}(\mathrm{I}(\mathrm{x})) = \sum_{x \in \chi} q(x) I(x) = -\sum_{x \in \chi} q(x) log_2^{q(x)}$$

称为随机变量X的平均自信息，又称X的信息熵或熵记为H(X)

- H(X)特性：
  - 非负性：H大于等于0
  - 连续性：H对任意q连续
  - 极值性：当q都等于1\K时 H达到最大值logK 证明可参考：
    https://blog.csdn.net/salmonwilliam/article/details/88971713

不纯性的测量：Entropy

- 给定结点t的Entropy值计算（这里log以2为底）：

$$Entropy(t) = -\sum_j p(j|t)\log_2^{p(j|t)}$$

($p(j \mid t)$ 是在结点t中，类j发生的概率).

  - 当类分布均衡时，Entropy值达到最大值 ($\log_2^k$)，k为类的个数
  - 相反当只有一个类时， Entropy值达到最小值0

C1	**0**
C2	**6**
Entropy=0.000	

C1	**1**
C2	**5**
Entropy=0.650	

C1	**2**
C2	**4**
Entropy=0.918	

C1	**3**
C2	**3**
Entropy=1	

- 当一个结点 t 分割成 k 个部分 (孩子), 划分的质量可由下面公式计算

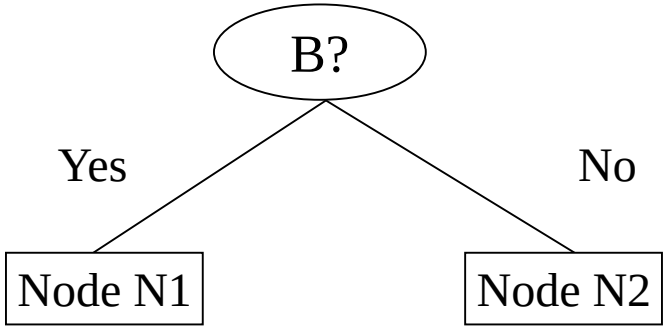
$$\text{Entropy}_{split} = \sum_{i=1}^{k} \frac{n_i}{n} \text{Entropy}\ (i)$$

n_i = 孩子结点 i的记录数,

n = 父结点 t的记录数.

计算Entropy的例子

	Parent
C1	**6**
C2	**6**
Entropy = 1	



	N1	N2
C1	**5**	**1**
C2	**2**	**4**
Entropy=0.804		

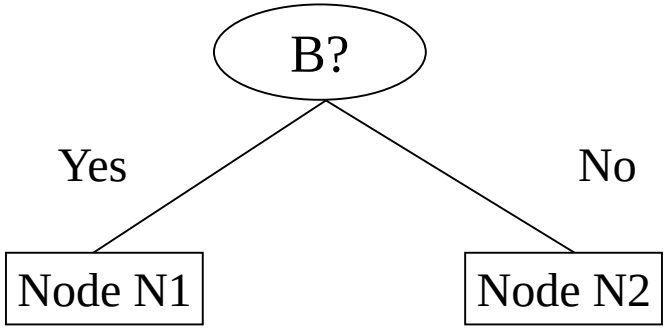
Entropy(N1)
$= -5/7\log_2(5/7) - 2/7\log_2(2/7) = 0.863$

Entropy(N2)
$= -1/5\log_2(1/5) - 4/5\log_2(4/5) = 0.722$

Entropy $_{split}$
$= 7/12 * 0.863 + 5/12 * 0.722 = 0.804$

计算Entropy的例子

	Parent
C1	**6**
C2	**6**
Entropy = 1	



	N1	N2
C1	**5**	**1**
C2	**2**	**4**
Entropy=0.804		

Entropy(N1)
$= -5/7\log_2(5/7) - 2/7\log_2(2/7) = 0.863$

Entropy(N2)
$= -1/5\log_2(1/5) - 4/5\log_2(4/5) = 0.722$

Entropy $_{split}$
$= 7/12 * 0.863 + 5/12 * 0.722 = 0.804$

基于 Information Gain的划分...

- Information Gain:

$$GAIN_{\text{split}} = \text{Entropy}(t) - \text{Entropy}_{split}$$

  - 熵和Gini指标等不纯性趋向于有利于具有大量不同值的属性!如：利用雇员id产生更纯的划分，但它却毫无用处
  - 每个划分相关联的记录数太少，将不能做出可靠的预测
  - 解决该问题的策略有两种：
    - ✓ 限制测试条件只能是二元划分
    - ✓ 使用增益率。K越大Split Info越大增益率越小

- 增益率（Gain Ratio）：

$$GainRATIO_{split} = \frac{GAIN_{Split}}{SplitINFO}$$

$$SplitINFO = -\sum_{i=1}^{k} \frac{n_i}{n} \log \frac{n_i}{n}$$

基于 Classification Error的划分

- 给定结点t的 Classification Error值计算 :

$$Error(t) = 1 - \max_i P(i|t)$$

  - 当类分布均衡时， error值达到最大值 (1 - 1/k)，这里k为类的个数
  - 相反当只有一个类时， error值达到最小值0

C1	**0**
C2	**6**

P(C1) = 0/6 = 0　　P(C2) = 6/6 = 1

Error = 1 – max (0, 1) = 1 – 1 = 0

C1	**1**
C2	**5**

P(C1) = 1/6　　P(C2) = 5/6

Error = 1 – max (1/6, 5/6) = 1 – 5/6 = 1/6

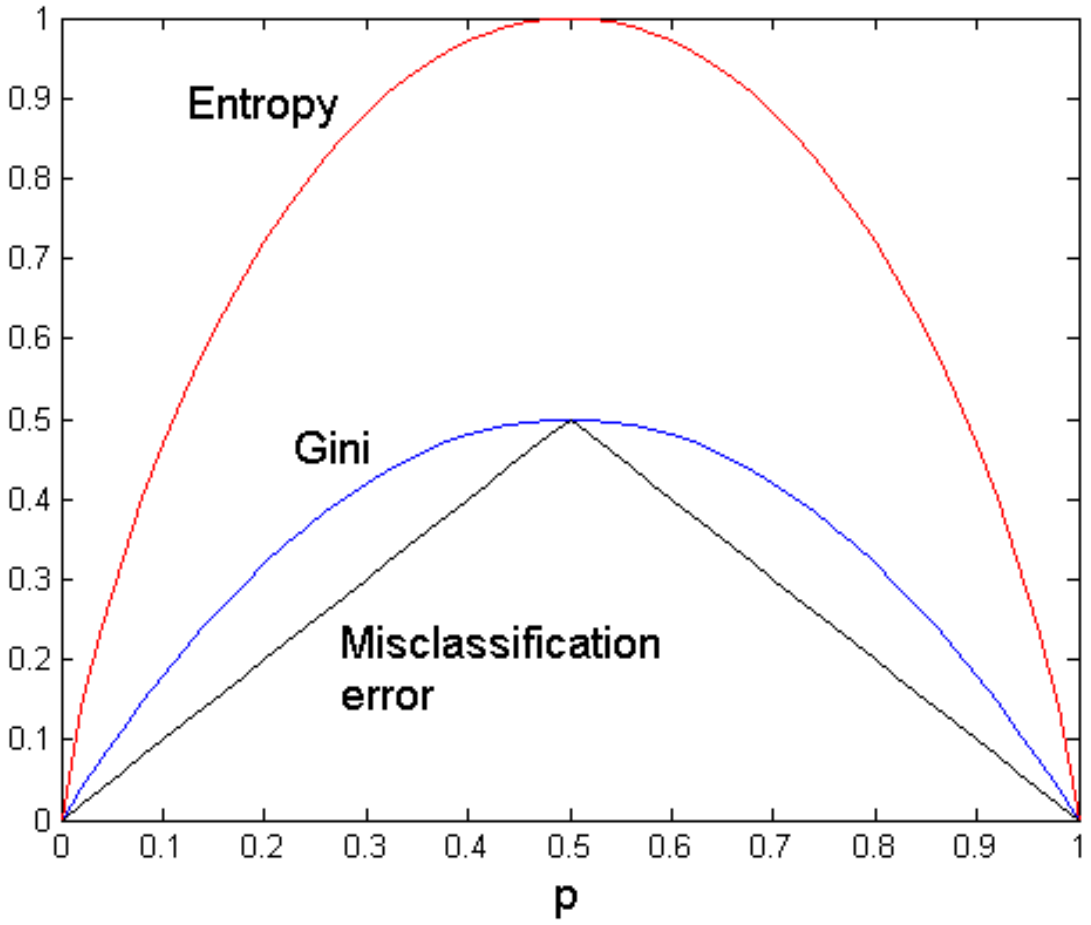
C1	**2**
C2	**4**

P(C1) = 2/6　　P(C2) = 4/6

Error = 1 – max (2/6, 4/6) = 1 – 4/6 = 1/3

不纯性度量之间的比较

◆ 针对二元分类问题（p为其中一个类的概率，1-p为另一个类的概率）：

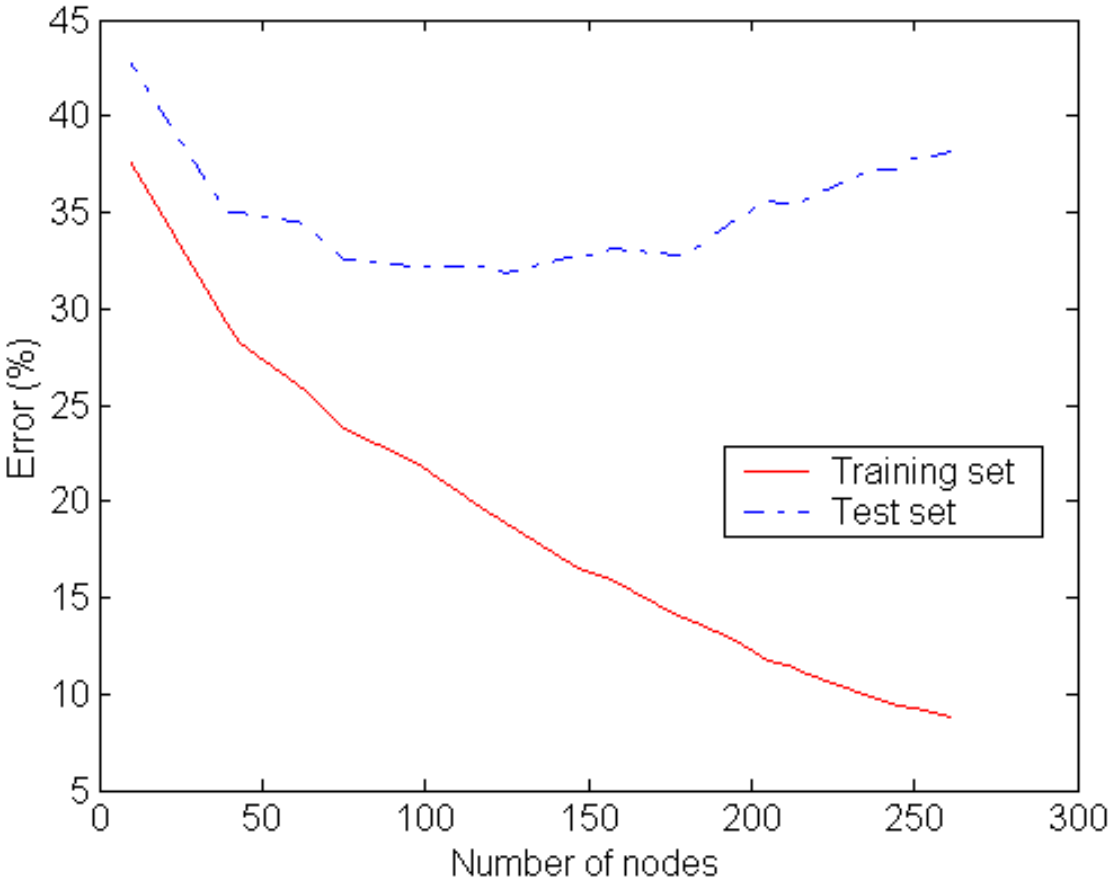


模型过分拟合和拟合不足

- 分类模型的误差大致分为两种：
  - 训练误差：是在训练记录上误分类样本比例
  - 泛化误差：是模型在未知记录上的期望误差
- 一个好的分类模型不仅要能够很好的拟合训练数据，而且对未知样本也要能准确分类。
- 换句话说，一个好的分类模型必须具有低训练误差和低泛化误差。
- 当训练数据拟合太好的模型，其泛化误差可能比具有较高训练误差的模型高，这种情况成为模型过分拟合

模型过分拟合和拟合不足

- 当决策树很小时，训练和测试误差都很大，这种情况称为模型拟合不足。出现拟合不足的原因是模型尚未学习到数据的真实结构。
- 随着决策树中结点数的增加，模型的训练误差和测试误差都会随之下降。
- 当树的规模变得太大时，即使训练误差还在继续降低，但是检验误差开始增大，导致模型过分拟合



停止分裂过程

- 当所有的记录属于同一类时，停止分裂
- 当所有的记录都有相同的属性时，停止分裂
- 提前终止树的生长

处理决策树中的过分拟合

- 先剪枝 (Early Stopping Rule)
  - 树增长算法在产生完全拟合整个训练数据集的之前就停止决策树的生长
  - 为了做到这一点，需要采用更具限制性的结束条件:
    - ✓ 当结点的记录数少于一定阈值，则停止生长
    - ✓ 当不纯性度量的增益低于某个确定的阈值时，则停止生长 (e.g., information gain).
- 缺点：很难为提前终止选取正确的阈值:
  - 阈值太高，导致拟合不足
  - 阈值太低，导致不能充分解决过分拟合的问题。

处理决策树中的过分拟合...

◆ 后剪枝

- ➢ 在该方法中，初始决策树按照最大规模生长，然后进行剪枝的步骤，按照自底向上的方式修剪完全增长的决策树。
- ➢ 修剪有两种做法:
  - ✓ 用新的叶结点替换子树，该叶结点的类标号由子树下记录中的多数类确定
  - ✓ 用子树中最常用的分支代替子树

◆ 比较

- ➢ 与先剪枝相比，后剪枝技术倾向于产生更好的结果。
- ➢ 因为不像先剪枝，后剪枝是根据完全增长的决策树作出的剪枝决策，先剪枝则可能过早终止决策树的生长。
- ➢ 然而，对于后剪枝，当子树被剪掉后，生长完全决策树的额外开销就被浪费了。

分类与预测——决策树

- 常用的决策树算法如下表所示。

决策树算法	算法描述
ID3算法	采用二叉树，在决策树的各级节点上使用信息增益方法作为属性的选择标准，来帮助确定生成每个节点时所应采用的合适属性。
C4.5算法	C4.5决策树生成算法相对于ID3算法的重要改进是使用信息增益率来选择节点属性。C4.5算法可以克服ID3算法存在的不足：ID3算法只适用于离散的描述属性，而C4.5算法既能够处理离散的描述属性，也可以处理连续的描述属性。
CART算法	CART决策树是一种十分有效的非参数分类和回归方法，通过构建树、修剪树、评估树来构建一个二叉树。当终结点是连续变量时，该树为回归树；当终结点是分类变量，该树为分类树。

sklearn中训练决策树的默认算法是CART

sklearn决策树

DecisionTreeClassifier(criterion="gini", splitter="best", max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0., max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0., min_impurity_split=None, class_weight=None, presort=False)

参数含义：

1.criterion:string, optional (default="gini") (1).criterion='gini',分裂节点时评价准则是Gini指数。 (2).criterion='entropy',分裂节点时的评价指标是信息增益。

2.max_depth:int or None, optional (default=None)。指定树的最大深度。 如果为None，表示树的深度不限。直到所有的叶子节点都是纯净的，即叶子节点 中所有的样本点都属于同一个类别。或者每个叶子节点包含的样本数小于min_samples_split。

3.splitter:string, optional (default="best")。指定分裂节点时的策略。 (1).splitter='best',表示选择最优的分裂策略。 (2).splitter='random',表示选择最好的随机切分策略。

4.min_samples_split:int, float, optional (default=2)。表示分裂一个内部节点需要的最少样本数。 (1).如果为整数，则min_samples_split就是最少样本数。 (2).如果为浮点数(0到1之间)，则每次分裂最少样本数为ceil(min_samples_split * n_samples)

5.min_samples_leaf: int, float, optional (default=1)。指定每个叶子节点需要的最少样本数。 (1).如果为整数，则min_samples_split就是最少样本数。 (2).如果为浮点数(0到1之间)，则每个叶子节点最少样本数为ceil(min_samples_leaf * n_samples)

6.min_weight_fraction_leaf:float, optional (default=0.) 叶子节点最小的样本权重和。这个值限制了叶子节点所有样本权重和的最小值，如果小于这个值，则会和兄弟节点一起被剪枝。 默认是0，就是不考虑权重问题。一般来说，如果我们有较多样本有缺失值，或者分类树样本的分布类别偏差很大，就会引入样本权重，这时我们就要注意这个值了。

7.max_features:int, float, string or None, optional (default=None). 搜寻最佳划分的时候考虑的特征数量。 (1).如果为整数，每次分裂只考虑max_features个特征。 (2).如果为浮点数(0到1之间)，每次切分只考虑int(max_features * n_features)个特征。 (3).如果为'auto'或者'sqrt',则每次切分只考虑sqrt(n_features)个特征 (4).如果为'log2',则每次切分只考虑log2(n_features)个特征。 (5).如果为None,则每次切分考虑n_features个特征。 (6).如果已经考虑了max_features个特征，但还是没有找到一个有效的切分，那么还会继续寻找 下一个特征，直到找到一个有效的切分为止。

sklearn决策树-续1

DecisionTreeClassifier(criterion="gini", splitter="best", max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0., max_features=None, random_state=None, max_leaf_nodes=None, min_impurity_decrease=0., min_impurity_split=None, class_weight=None, presort=False)

参数含义：

8.random_state:int, RandomState instance or None, optional (default=None) (1).如果为整数，则它指定了随机数生成器的种子。(2).如果为RandomState实例，则指定了随机数生成器。 (3).如果为None，则使用默认的随机数生成器。

9.max_leaf_nodes: int or None, optional (default=None)。指定了叶子节点的最大数量。 (1).如果为None,叶子节点数量不限。(2).如果为整数，则max_depth被忽略。

10.min_impurity_decrease:float, optional (default=0.) 如果节点的分裂导致不纯度的减少(分裂后样本比分裂前更加纯净)大于或等于min_impurity_decrease，则分裂该节点。 加权不纯度的减少量计算公式为： min_impurity_decrease=N_t / N * (impurity - N_t_R / N_t * right_impurity - N_t_L / N_t * left_impurity) 其中N是样本的总数，N_t是当前节点的样本数，N_t_L是分裂后左子节点的样本数， N_t_R是分裂后右子节点的样本数。impurity指当前节点的基尼指数，right_impurity指 分裂后右子节点的基尼指数。left_impurity指分裂后左子节点的基尼指数。

11.min_impurity_split:float 树生长过程中早停止的阈值。如果当前节点的不纯度高于阈值，节点将分裂，否则它是叶子节点。这个参数已经被弃用。用min_impurity_decrease代替了min_impurity_split。

12.class_weight:dict, list of dicts, "balanced" or None, default=None 类别权重的形式为{class_label: weight} (1).如果没有给出每个类别的权重，则每个类别的权重都为1。 (2).如果class_weight='balanced'，则分类的权重与样本中每个类别出现的频率成反比。 计算公式为：n_samples / (n_classes * np.bincount(y)) (3).如果sample_weight提供了样本权重(由fit方法提供)，则这些权重都会乘以sample_weight。

13.presort:bool, optional (default=False) 指定是否需要提前排序数据从而加速训练中寻找最优切分的过程。设置为True时，对于大数据集 会减慢总体的训练过程；但是对于一个小数据集或者设定了最大深度的情况下，会加速训练过程。

一个决策树的例子

表5-5　处理后的数据集

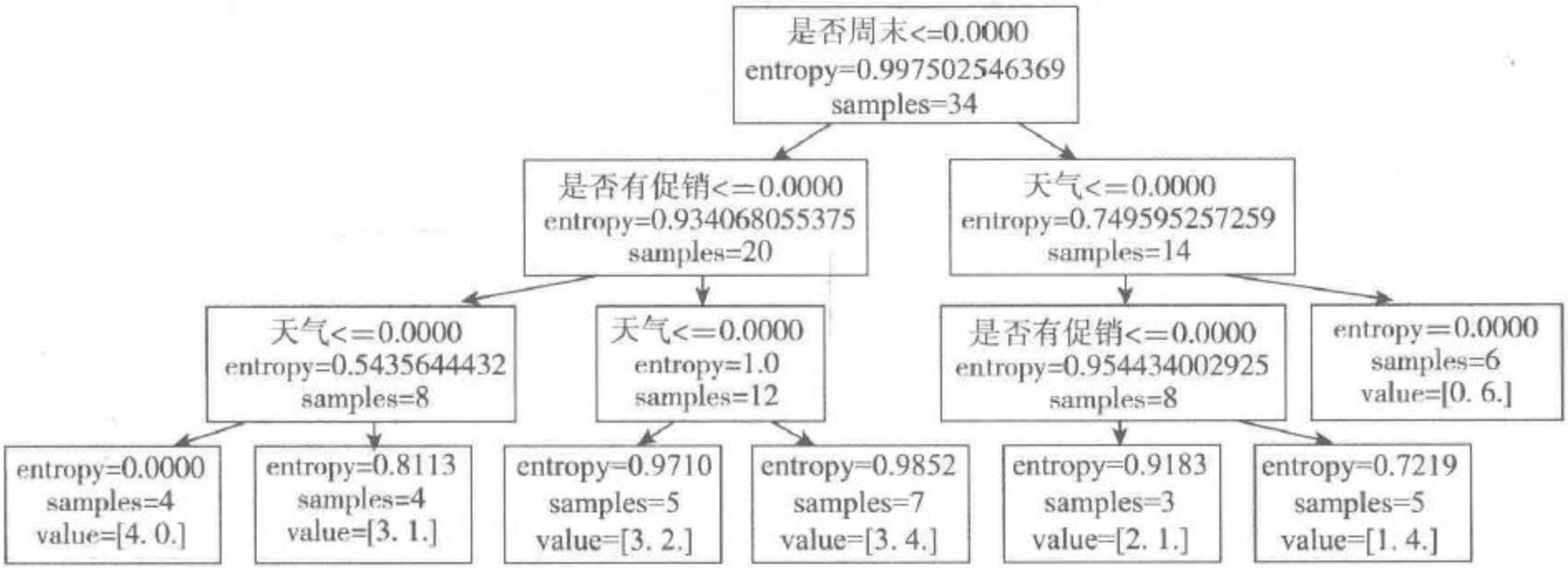
序　　号	天　　气	是否周末	是否有促销	销　　量
1	坏	是	是	高
2	坏	是	是	高
3	坏	是	是	高
4	坏	否	是	高
…	…	…	…	…
32	好	否	是	低
33	好	否	否	低
34	好	否	否	低

代码清单5-2　决策树算法预测销量高低代码

```
#-*- coding: utf-8 -*-
#使用ID3决策树算法预测销量高低
import pandas as pd

#参数初始化
filename = '../data/sales_data.xls'
data = pd.read_excel(filename, index_col = u'序号') #导入数据

#数据是类别标签，要将它转换为数据
#用1来表示“好” “是” “高”这三个属性，用-1来表示“坏” “否” “低”
data[data == u'好'] = 1
data[data == u'是'] = 1
data[data == u'高'] = 1
data[data != 1] = -1
x = data.iloc[:,:3].as_matrix().astype(int)
y = data.iloc[:,3].as_matrix().astype(int)

from sklearn.tree import DecisionTreeClassifier as DTC
dtc = DTC(criterion='entropy') #建立决策树模型，基于信息熵
dtc.fit(x, y) #训练模型

#导入相关函数，可视化决策树。
#导出的结果是一个dot文件，需要安装Graphviz才能将它转换为pdf或png等格式。
from sklearn.tree import export_graphviz
from sklearn.externals.six import StringIO
with open("tree.dot", 'w') as f:
  f = export_graphviz(dtc, feature_names = x.columns, out_file = f)
```

代码详见：示例程序 /code/decision_tree.py

生成的决策树


是否周末<=0.0000
entropy=0.997502546369
samples=34
是否有促销<=0.0000
entropy=0.934068055375
samples=20
天气<=0.0000
entropy=0.749595257259
samples=14
天气<=0.0000
entropy=0.5435644432
samples=8
天气<=0.0000
entropy=1.0
samples=12
是否有促销<=0.0000
entropy=0.954434002925
samples=8
entropy=0.0000
samples=6
value=[0. 6.]
entropy=0.0000
samples=4
value=[4. 0.]
entropy=0.8113
samples=4
value=[3. 1.]
entropy=0.9710
samples=5
value=[3. 2.]
entropy=0.9852
samples=7
value=[3. 4.]
entropy=0.9183
samples=3
value=[2. 1.]
entropy=0.7219
samples=5
value=[1. 4.]

扩展阅读

历史

奠定基础的热烈时期 1950s-1960s

停滞不前的冷静时期 1960s-1970s

重拾希望的复兴时期 1970s-1980s

现代机器学习的成型时期 1990s-2000s

大放光芒的蓬勃发展时期 2000s-now

唐纳德·赫布

阿兰·图灵

赫伯特·西蒙

阿瑟·塞缪尔

罗森·布拉特

马文·明斯基

杰弗里·辛顿

燕乐存

奠定基础的热烈时期

1950s
-1960s

1902年，James的神经元相互连接
1943年，McCulloch和Pitts发现了神经元的“兴奋”和“抑制”机制
1949年，Hebb的学习律。

基于最小二乘的Rosenblatt的感知机(1956)，其本质是多变量空间上的平均(回归)。

基函数：
$L = \beta_1 D + \beta_2 I + \beta_3 G + \beta_4 S$
设计算法，确定β，获得模型

贡献是：多变量回归的计算方法(神经网络)。

疑问是：只能解决线性问题，不能满足实际的需要。埋下被批评的口实。

奠定基础的热烈时期

1950s
-1960s

1949年，Turing的图灵测试。

2014年6月8日，一台计算机（计算机尤金·古斯特曼是一个聊天机器人，一个电脑程序）成功让人类相信它是一个13岁的男孩，成为有史以来首台通过图灵测试的计算机。这被认为是人工智能发展的一个里程碑事件。

奠定基础的热烈时期

1950s
-1960s

1952年，IBM科学家Samuel的跳棋程序。

4年后，这个程序战胜了塞缪尔本人

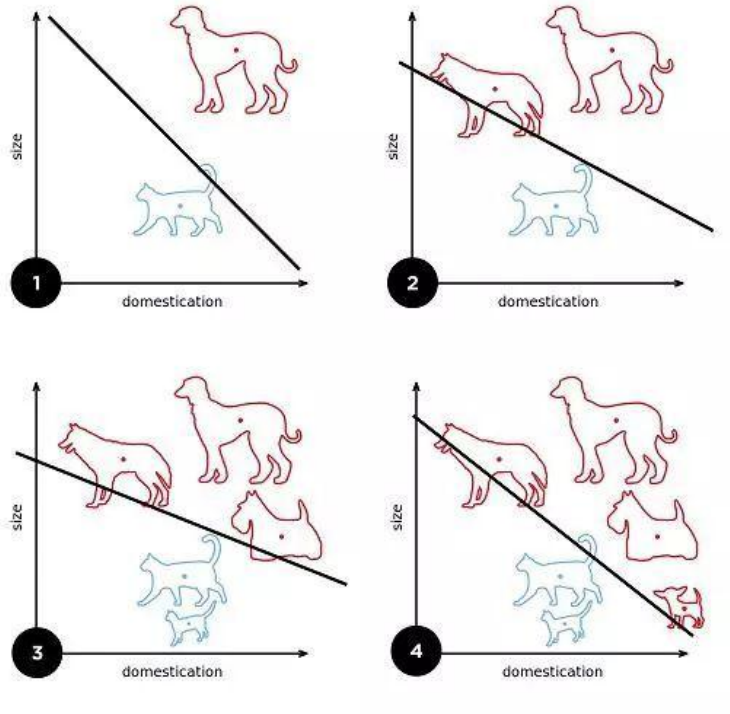
又过了3年，战胜了美国一个保持8年不败的冠军

“机器学习”：“可以提供计算机能力而无需显式编程的研究领域”。

1950s
-1960s

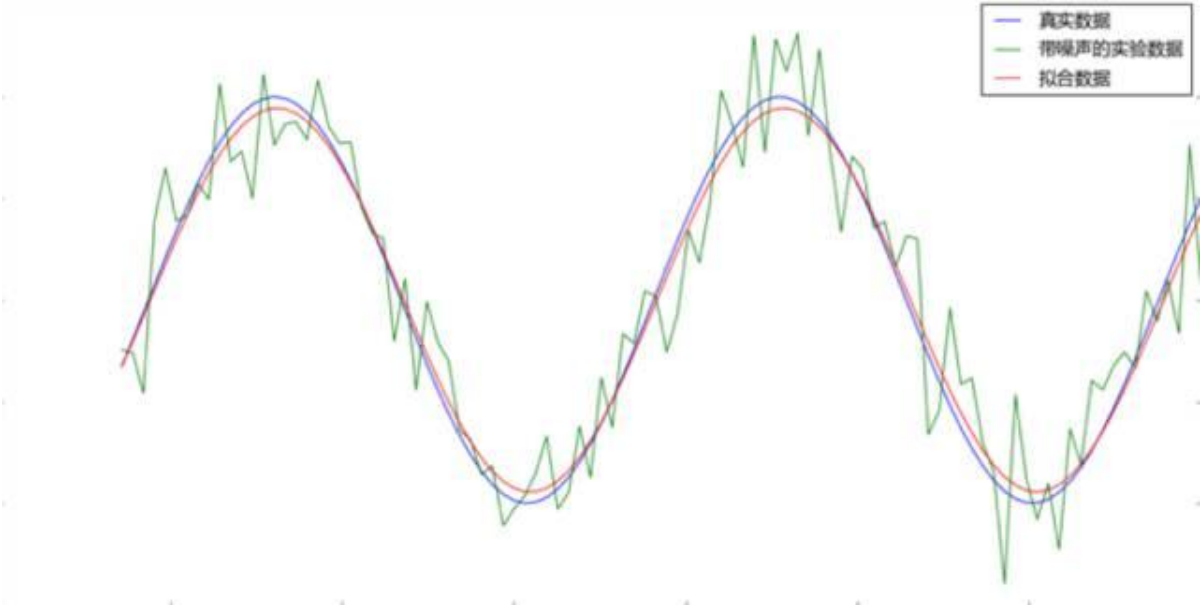
1957年，Rosenblatt设计出了第一个计算机神经网络——感知机（the perceptron），它模拟了人脑的运作方式。

$$f(x) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 & \text{otherwise} \end{cases}$$



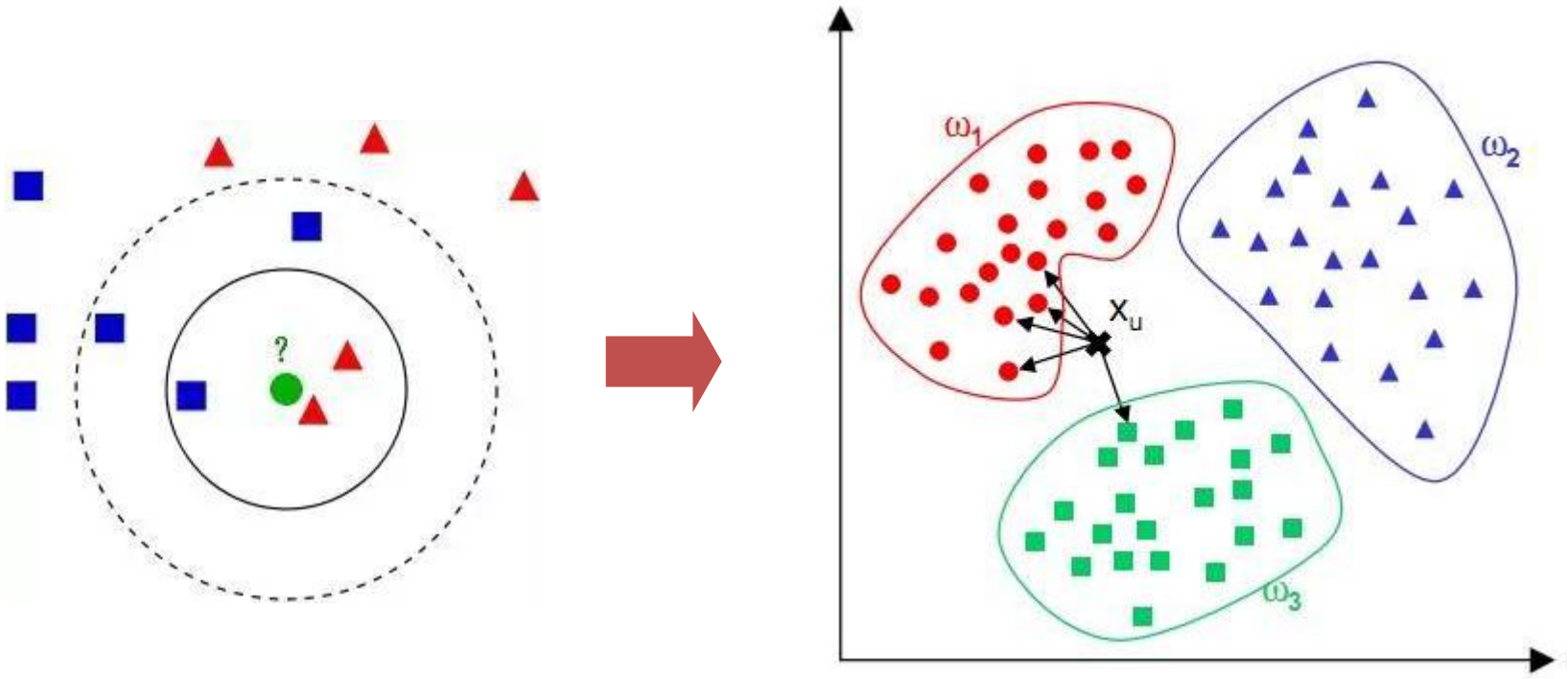
1950s
-1960s

1960年，维德罗首次使用Delta学习规则（即最小二乘法）用于感知器的训练步骤，创造了一个良好的线性分类器。



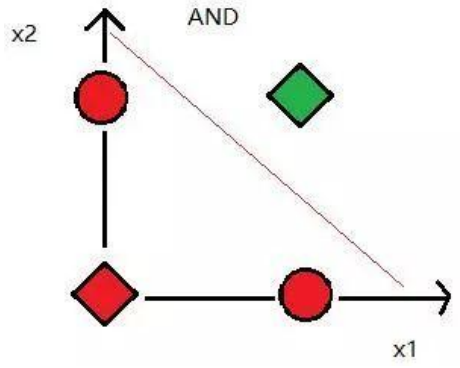
1950s
-1960s

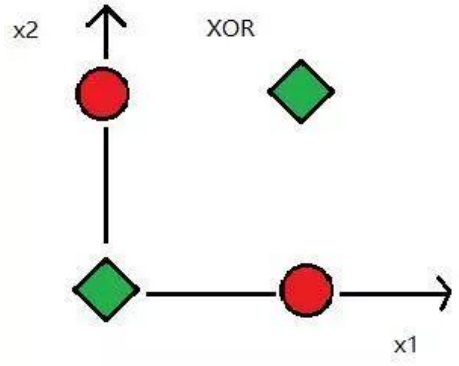
1967年，最近邻算法（The nearest neighbor algorithm）出现，由此计算机可以进行简单的模式识别。

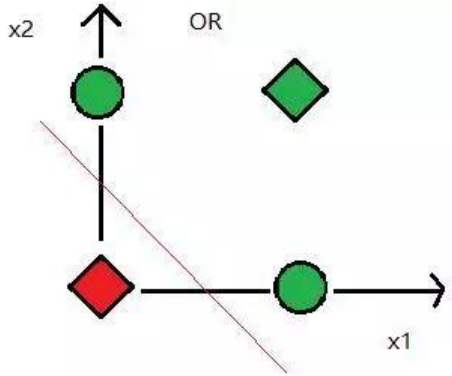


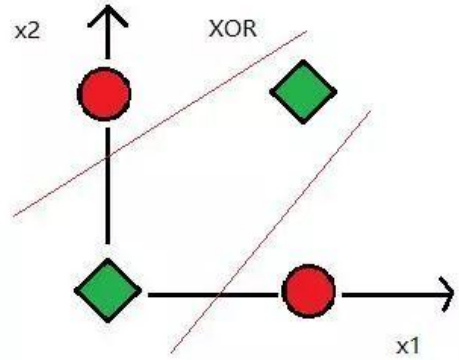
1950s
-1960s

1969年，马文·明斯基提出了著名的XOR问题,指出感知机在线性不可分的数据分布上是失效的。









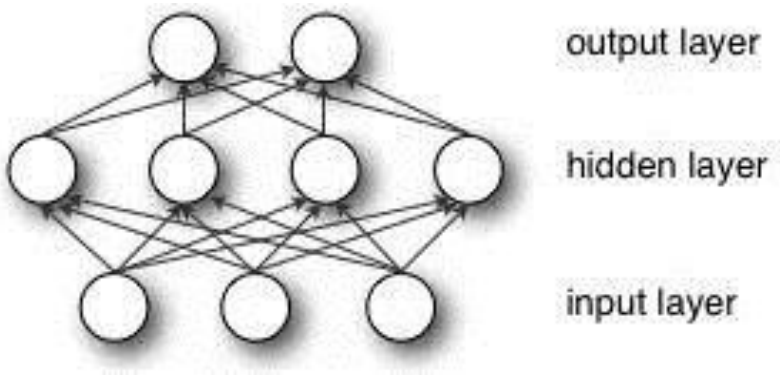
停滞不前的冷静时期

1960s
-1970s

重拾希望的复兴时期

1970s -1980s

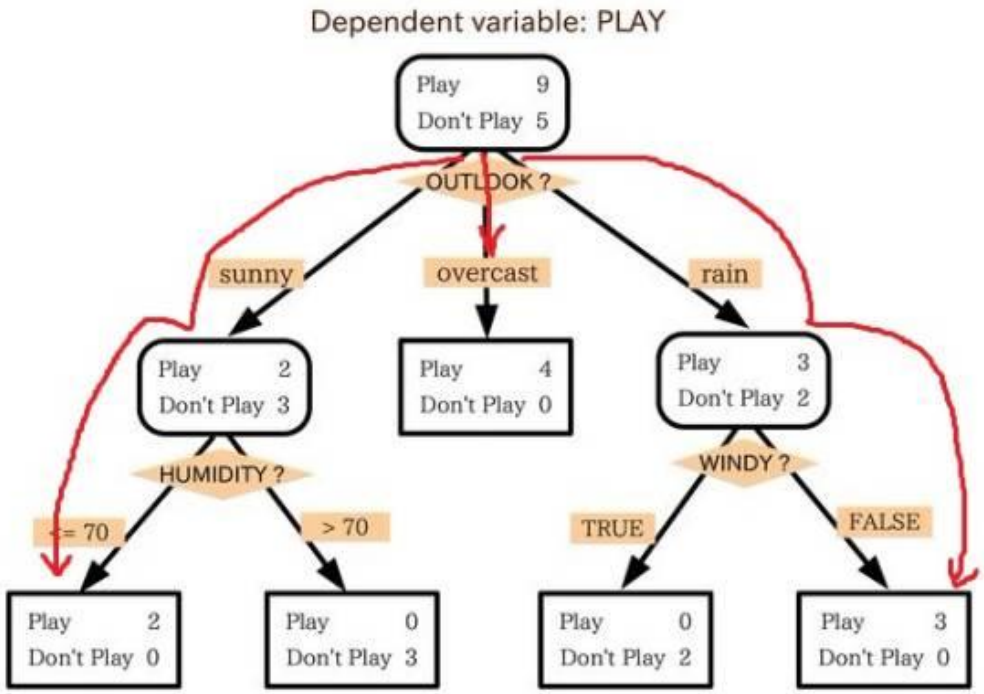
1981年，Werbos提出了多层感知机（人工神经网络），MLP。



在1985-1986年，神经网络研究人员（鲁梅尔哈特，辛顿，威廉姆斯-赫，尼尔森）相继提出了使用BP算法训练的多参数线性规划（MLP）的理念，成为后来深度学习的基石。

1970s
-1980s

1986年,昆兰提出了一种非常出名的机器学习算法，“决策树”，更具体的说是ID3算法。



打网球的天气分类决策

1990s
-2000s

1990年, Vapnik和Cortes提出支持向量机（SVM）

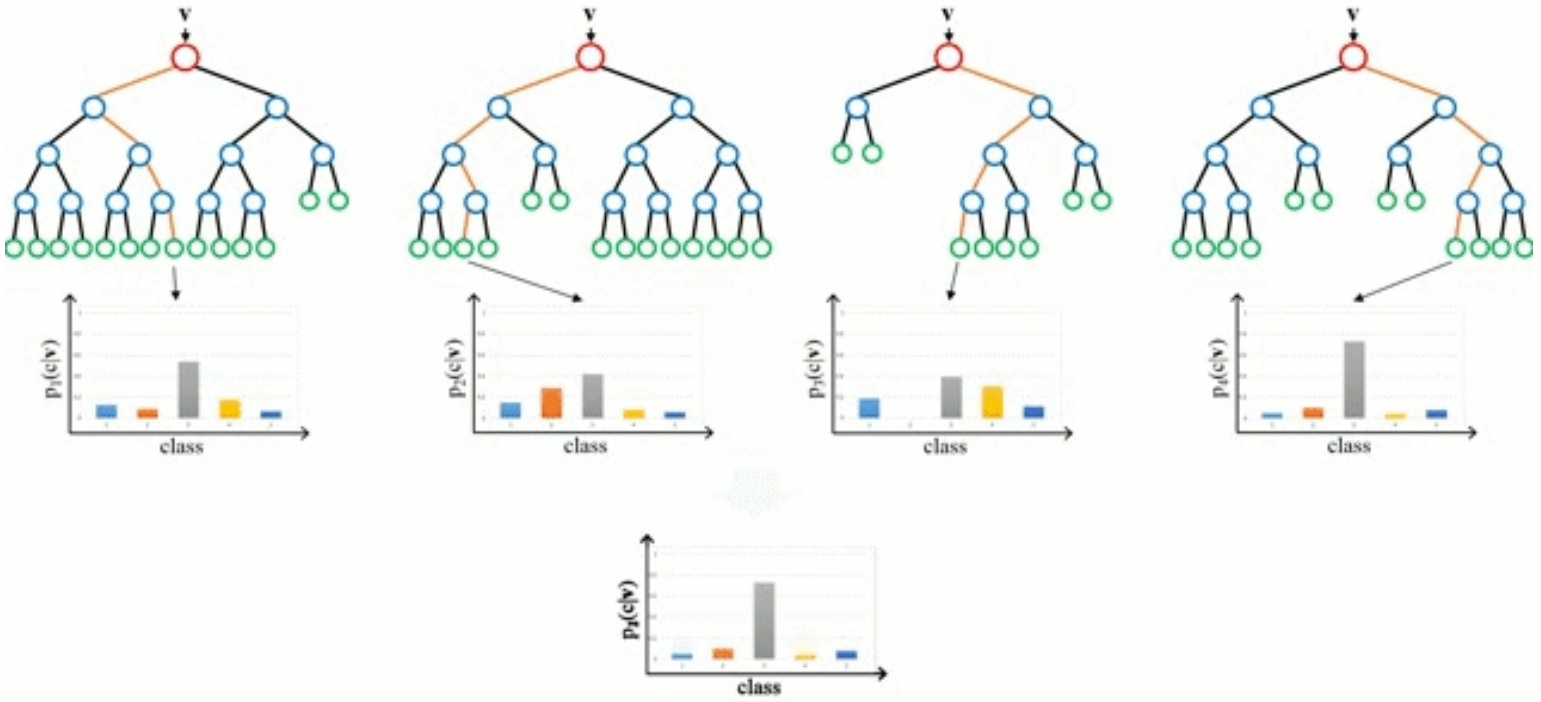
2000年提出带核函数的SVM

Hochreiter 等 人 1991 年 和 Hochreiter等人在2001年的研究表明在应用BP算法学习时，NN神经元饱和后会出现梯度损失（gradient loss）的情况。

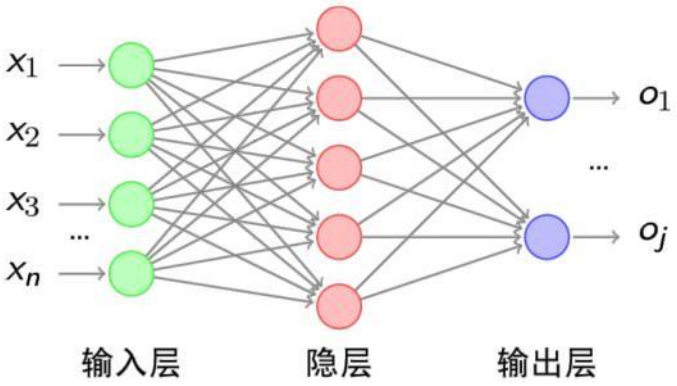
SVM在许多以前由NN占据的任务中获得了更好的效果。

1990s
-2000s

2001年, Breiman提出随机森林



2000s
-now



浅层学习

2006年，Hinton提出了神经网络Deep Learning算法，使神经网络的能力大大提高，向支持向量机发出挑战。

深层学习