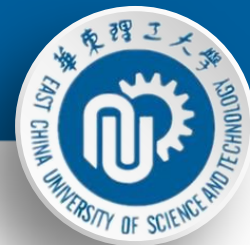


课时内容

第5章 智能搜索技术





◆ 基本粒子群优化算法

◆ 粒子速度和位置的更新

假设在 D 维搜索空间中，有 m 个粒子；

其中第 i 个粒子的位置为矢量 $\vec{x}_i = (x_{i1}, x_{i2}, \dots, x_{iD})$

其飞翔速度也是一个矢量，记为 $\vec{v}_i = (v_{i1}, v_{i2}, \dots, v_{iD})$

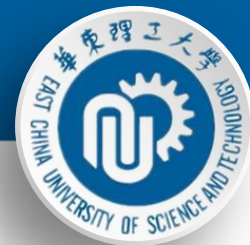
第 i 个粒子搜索到的最优位置为 $\vec{p}_i = (p_{i1}, p_{i2}, \dots, p_{iD})$

整个粒子群搜索到的最优位置为 $\vec{p}_{gbest} = (p_{gbest1}, p_{gbest2}, \dots, p_{gbestD})$

第 i 个粒子的位置和速度更新为：

$$v_{id}^{k+1} = wv_{id}^k + c_1 rand()(p_{id} - x_{id}^k) + c_2 rand()(p_{gbest} - x_{id}^k)$$

$$x_{id}^{k+1} = x_{id}^k + v_{id}^{k+1} \quad i = 1, 2, \dots, m; \quad d = 1, 2, \dots, D$$



◆ 基本粒子群优化算法

◆ 粒子速度和位置的更新

$$v_{id}^{k+1} = wv_{id}^k + c_1 rand()(p_{id} - x_{id}^k) + c_2 rand()(p_{gbest} - x_{id}^k)$$

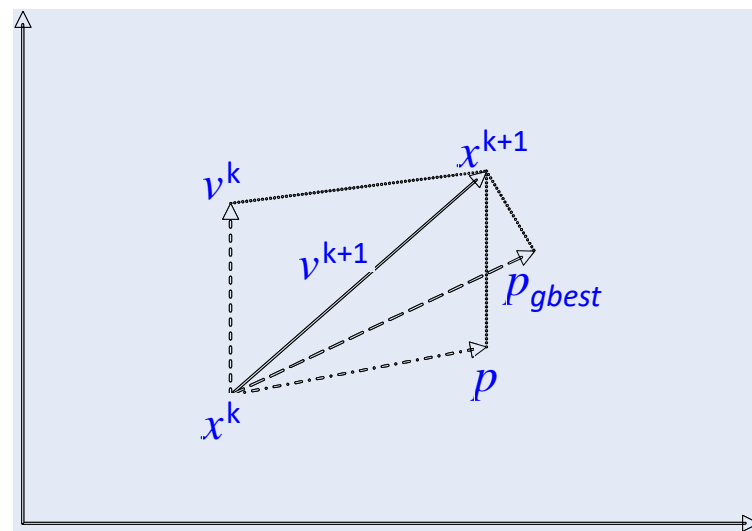
$$x_{id}^{k+1} = x_{id}^k + v_{id}^{k+1}$$

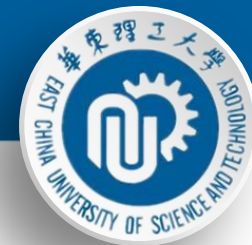
$$i = 1, 2, \dots, m; \quad d = 1, 2, \dots, D$$

其中， w 称为**惯性权重**，

c_1 和 c_2 为两个正常数，称为**加速因子**。

将 v_{id}^k 限制在一个最大速度 v_{max} 内。





◆ 基本粒子群优化算法

◆ 粒子速度和位置的更新

$$v_{id}^{k+1} = wv_{id}^k + c_1 \text{rand}() (p_{id} - x_{id}^k) + c_2 \text{rand}() (p_{gbest} - x_{id}^k)$$

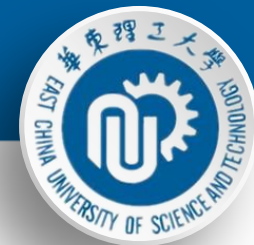
$$x_{id}^{k+1} = x_{id}^k + v_{id}^{k+1}$$

$$i = 1, 2, \dots, m; d = 1, 2, \dots, D$$

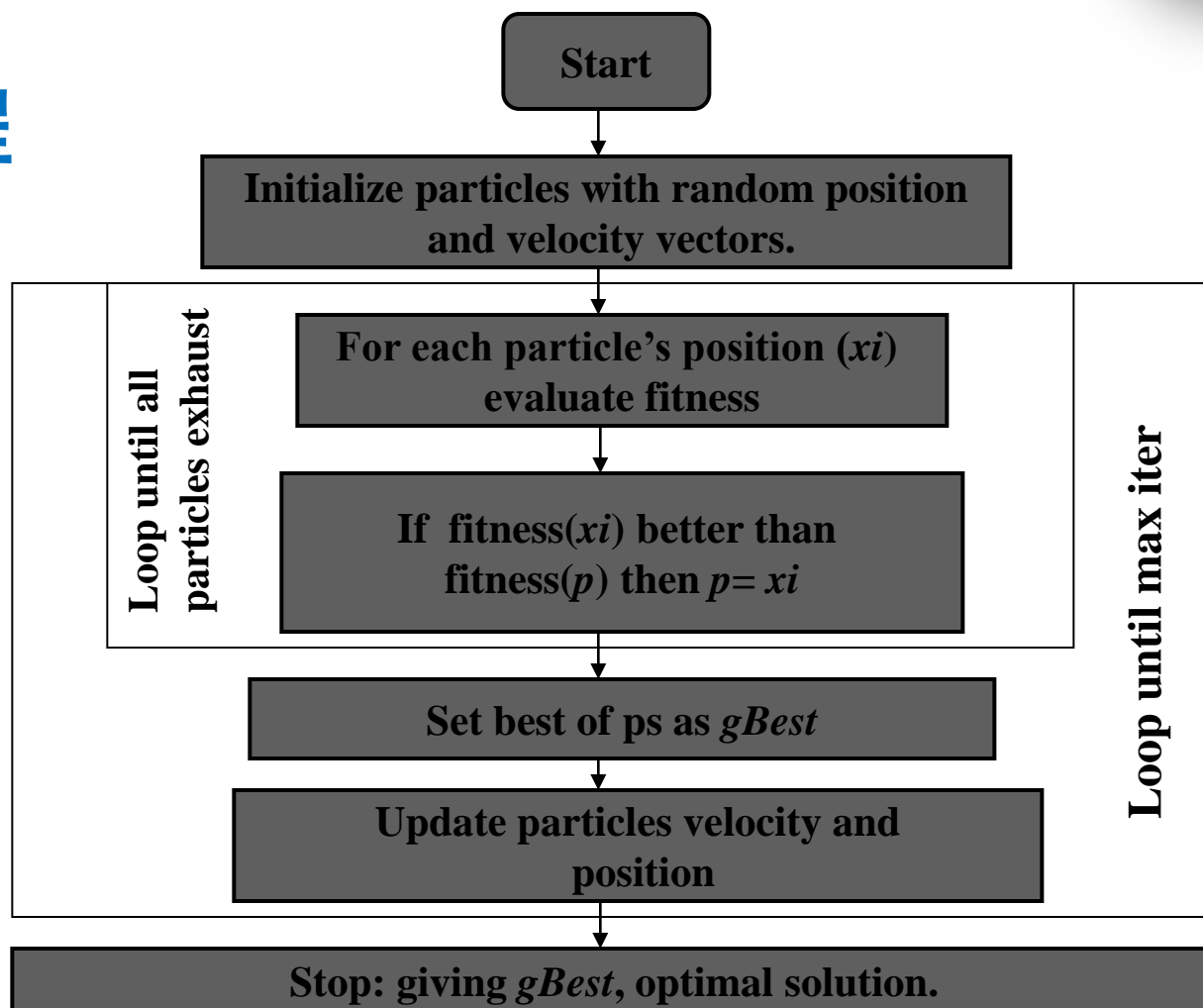
“惯性部分”，
对自身运动状态
的信任

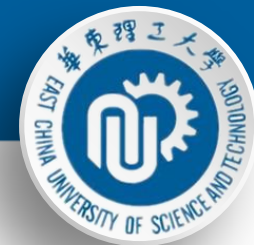
“认知部分”，对微粒本身
的思考，即来源于自己经
验的部分

“社会部分”，微粒间的信息
共享，来源于群体中的其它优
秀微粒的经验



◆ 算法流程





◆ 粒子群优化算法应用

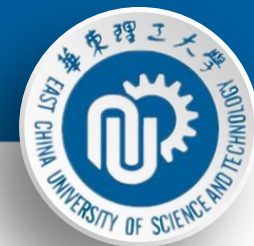
◆ 求解TSP问题

◆ 交换子和交换序

设 n 个节点的**TSP**问题的解序列为 $s=(a_i), i=1\dots n$ 。

定义**交换子**： $SO(i_1, i_2)$ 为交换解 S 中的点 a_{i_1} 和 a_{i_2} ，则 $S'=S+SO(i_1, i_2)$ 为解 S 经算子 $SO(i_1, i_2)$ 操作后的新解。

一个或多个交换子的有序队列就是**交换序**，记作 SS ， $SS=(SO_1, SO_2, \dots, SO_N)$ 。其中， SO_1, \dots, SO_N 等是交换子，之间的顺序是有意义的，意味着所有的交换子依次作用于某解上。



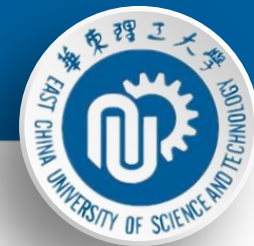
◆ 求解TSP问题

◆ 符号的定义

若干个交换序可以合并成一个新的交换序，定义 \oplus 为两个交换序的合并算子。

- ◆ 设两个交换序 SS_1 和 SS_2 按先后顺序作用于解 S 上，得到新解 S' 。假设另外有一个交换序 SS' 作用于同一解 S 上，能够得到相同的解 S' ，可定义

$$SS' = SS_1 \oplus SS_2$$



◆ 求解TSP问题

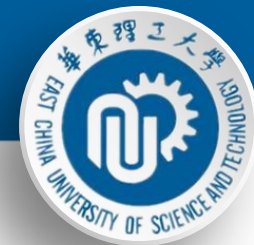
◆ 符号的定义

SS' 和 $SS_1 \oplus SS_2$ 属于同一**等价集**，在交换序等价集中，拥有最少交换子的交换序称为该等价集的**基本交换序**。

◆ 解决TSP问题的速度算式定义


$$\begin{cases} v'_{id} = v_{id} \oplus \alpha(p_{id} - x_{id}) \oplus \beta(p_{gd} - x_{id}) \\ x'_{id} = x_{id} + v'_{id} \end{cases}$$

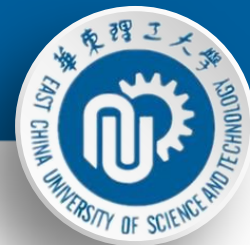
α 、 β 为 $[0,1]$ 上的随机数。 V_{id} 表示交换序， x_{id} 表示路径序列（解）。



◆ 求解TSP问题

◆ 算法流程

1. 初始化粒子群，给每个粒子一个初始解(x_{id})和随机的交换序(v_{id});
2. 如果满足结束条件，转步骤5;
3. 根据粒子当前位置 x_{id} 计算下一新解 x_{id}' ; 
4. 如果整个群体找到一个更好的解，更新 P_{gd} ，转步骤2;
5. 显示结果。



◆ 求解TSP问题

◆ 算法流程

3. 根据粒子当前位置 x_{id} 计算下一新解 x_{id}' :

1) 计算 $A=p_{id}-x_{id}$, A 是一个基本交换序, 表示 A 作用于 x_{id} 得到 p_{id} ;

2) 计算 $B=p_{gd}-x_{id}$, B 也是一个基本交换序;

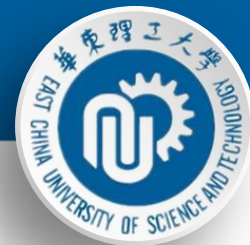
3) 更新速度 $v_{id}' = v_{id} \oplus \alpha(p_{id}-x_{id}) \oplus \beta(p_{gd}-x_{id})$

并将其转换为一个基本交换序;

4) 更新解 $x_{id}' = x_{id} + v_{id}'$;

5) 如果找到一个更好得解, 更新 p_{id} 。

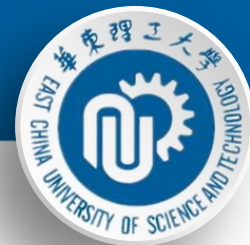




◆与遗传算法比较

◆ 共性

- 1 都属于仿生算法;
- 2 都属于全局优化方法;
- 3 都属于随机搜索算法;
- 4 都隐含并行性;
- 5 根据个体的适配信息进行搜索, 因此不受函数约束条件的限制, 如连续性、可导性等;
- 6 对高维复杂问题, 往往会遇到早熟收敛和收敛性能差的缺点, 都无法保证收敛到最优点。



◆与遗传算法比较

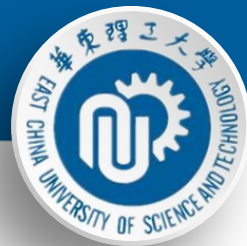
◆ 差异

1. PSO有记忆，所有粒子都保存较优解的知识，而GA，以前的知识随着种群的变化被改变；
2. PSO中的粒子是一种单向共享信息机制。而GA中的染色体之间相互共享信息，使得整个种群都向最优区域移动；
3. GA需要编码和遗传操作，而PSO没有交叉和变异操作，粒子只是通过内部速度进行更新，因此原理更简单、参数更少、实现更容易。

课时内容

第6章 机器学习





◆ 学习系统的基本模型

环境

学习系统所感知到的外界信息集合，也是学习系统的外界来源。信息的水平（一般化程度）和质量（正确性）对学习系统影响较大。

学习环节

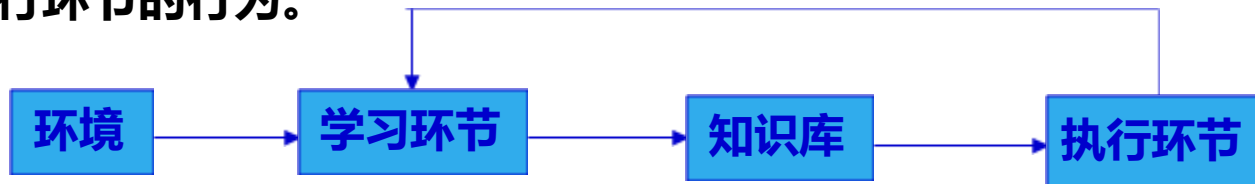
对环境提供的信息进行整理、分析归纳或类比，形成知识，并将其放入知识库。

知识库

存储经过加工后的信息（即知识）。其表示形式是否合适非常重要。

执行环节

根据知识库去执行一系列任务，并将执行结果或执行过程中获得的信息反馈给学习环节。学习环节再利用反馈信息对知识进行评价，进一步改善执行环节的行为。





◆ 示例学习的模型

示例空间

是我们向系统提供的示教例子的集合。 **研究问题：**例子质量，搜索方法。

归纳过程

是从搜索到的示例中抽象出一般性的知识的归纳过程。 **归纳方法：**常量转换为变量，去掉条件，增加选择，曲线拟合等。

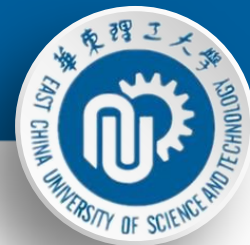
规则空间

是事务所具有的各种规律的集合。 **研究问题：**对空间的要求，搜索方法

验证过程

是要从示例空间中选择新的示例，对刚刚归纳出的规则做进一步的验证和修改。





◆ 把常量化为变量

把示例中的常量换成相应的变量即可得到一个一般性的规则。下面以扑克牌中**同花**的概念为例，进行讨论。

假设例子空间中有关于扑克牌中“同花”概念的示例：

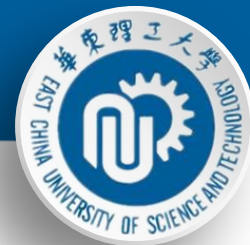
示例1：花色(c_1 , 梅花) \wedge 花色(c_2 , 梅花) \wedge 花色(c_3 , 梅花) \wedge 花色(c_4 , 梅花) \wedge 花色(c_5 , 梅花) \rightarrow 同花(c_1, c_2, c_3, c_4, c_5)

示例2：花色(c_1 , 红桃) \wedge 花色(c_2 , 红桃) \wedge 花色(c_3 , 红桃) \wedge 花色(c_4 , 红桃) \wedge 花色(c_5 , 红桃) \rightarrow 同花(c_1, c_2, c_3, c_4, c_5)

其中，示例1表示5张梅花牌是同花，示例2表示5张红桃牌是同花。

对这两个示例，**采用把常量化为变量的归纳方法**，只要把“梅花”和“红桃”用变量 x 代换，就可得到如下一般性的规则：

规则1：花色(c_1, x) \wedge 花色(c_2, x) \wedge 花色(c_3, x) \wedge 花色(c_4, x) \wedge 花色(c_5, x) \rightarrow 同花(c_1, c_2, c_3, c_4, c_5)



◆ 去掉条件

该方法是要把示例中的某些无关的子条件舍去，得到一个一般性的结论例如，有如下示例：

示例3：花色(c_1 , 红桃) \wedge 点数(c_1 , 2)

\wedge 花色(c_2 , 红桃) \wedge 点数(c_2 , 3)

\wedge 花色(c_3 , 红桃) \wedge 点数(c_3 , 4)

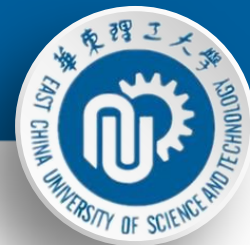
\wedge 花色(c_4 , 红桃) \wedge 点数(c_4 , 5)

\wedge 花色(c_5 , 红桃) \wedge 点数(c_5 , 6)

\rightarrow 同花(c_1, c_2, c_3, c_4, c_5)

为了学习同花的概念，除了需要把常量变为变量外，还需要把与花色无关的“点数”子条件舍去。这样也可得到上述规则1：

规则1：花色(c_1, x) \wedge 花色(c_2, x) \wedge 花色(c_3, x) \wedge 花色(c_4, x) \wedge 花色(c_5, x) \rightarrow 同花(c_1, c_2, c_3, c_4, c_5)



◆ 增加选择

该方法是要在析取条件中增加一个新的析取项。它包括前件析取法和内部析取法。

前件析取法：是通过对示例的前件的析取来形成知识的。例如：

示例4：点数(c_1 , J) \rightarrow 脸(c_1) 示例5：点数(c_1 , Q) \rightarrow 脸(c_1)

示例6：点数(c_1 , K) \rightarrow 脸(c_1)

将各示例的前件进行析取，就可得到所要求的规则：

规则2：点数(c_1 , J) \vee 点数(c_1 , Q) \vee 点数(c_1 , K) \rightarrow 脸(c_1)

内部析取法：是在示例的表示中使用集合与集合的成员关系来形成知识的。

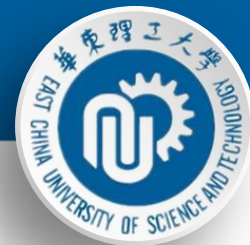
例如，有如下关于“脸牌”的示例：

示例7：点数 $c_1 \in \{J\} \rightarrow$ 脸(c_1) 示例8：点数 $c_1 \in \{Q\} \rightarrow$ 脸(c_1)

示例9：点数 $c_1 \in \{K\} \rightarrow$ 脸(c_1)

用内部析取法，可得到如下规则：

规则3：点数(c_1) $\in \{J, Q, K\} \rightarrow$ 脸(c_1)



◆ 曲线拟合

对数值问题的归纳可采用曲线拟合法。假设示例空间中的每个示例(x, y, z)都是输入x, y与输出z之间关系的三元组。例如, 有下3个示例:

示例10: (0, 2, 7)

示例11: (6, -1, 10)

示例12: (-1, -5, -16)

用最小二乘法进行曲线拟合, 可得x, y, z之间关系的规则如下:

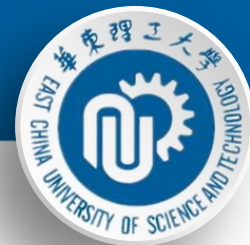
规则4: $z=2x+3y+1$

说明: 在上述前三种方法中, 方法(1)是把常量转换为变量; 方法(2)是去掉合取项 (约束条件); 方法(3)是增加析取项。它们都是要扩大条件的适用范围。从归纳速度上看, 方法(1)的归纳速度快, 但容易出错; 方法(2)归纳速度慢, 但不容易出错。因此, 在使用方法(1)时应特别小心。例如:

对示例4、示例5及示例6, 若使用方法(1), 则会归纳出如下的错误规则:

规则5: (错误) 点数(c1, x)→脸(c1)

它说明, 归纳过程是很容易出错的。



◆ 信息熵和信息增益

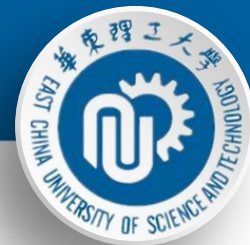
信息熵

信息熵是对信息源整体**不确定性的度量**。假设 S 为样本集， S 中所有样本的类别有 k 种，如 y_1, y_2, \dots, y_k ，各种类别样本在 S 上的概率分布分别为 $P(y_1), P(y_2), \dots, P(y_k)$ ，则 S 的信息熵可定义为：

$$\begin{aligned} E(s) &= -P(y_1)\log P(y_1) - P(y_2)\log P(y_2) - \dots - P(y_r)\log P(y_r) \\ &= -\sum_{j=1}^k P(y_j)\log P(y_j) \end{aligned}$$

其中，概率 $P(y_j)$ ($j=1, 2, \dots, k$)，实际上为 y_j 的样本在 S 中所占的比例；对数可以是以各种数为底的对数，在ID3算法中，我们取以2为底的对数。

$E(S)$ 的值越小， S 的不确定性越小，即其确定性越高。



◆ 信息熵和信息增益

信息增益 (information gain)

对两个信息量之间的差的度量。其讨论涉及到样本集S中样本的结构。对S中的每一个样本，除其类别外，还有其条件属性，或简称为属性。若假设S中的样本有m个属性，其属性集为 $x = \{x_1, x_2 \dots x_m\}$ 且每个属性均有r种不同的取值，则我们可以根据属性的不同取值将样本集S划分成r个不同的子集 $S_1, S_2, \dots S_r$ 。

此时，可得到由属性 x_i 的不同取值对样本集合S进行划分后的**加权信息熵**

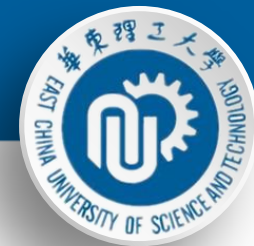
$$E(S, x_i) = \sum_{t=1}^r \frac{|S_t|}{|S|} \times E(S_t)$$

其中，t为条件属性 x_j 的属性值： S_t 为 $x_i = t$ 时的样本子集； $E(S_t)$ 为样本子集的信息熵； $|S|$ 和 $|S_t|$ 分别为样本集S和样本子集 S_t 的大小，即样本个数。

有了信息熵和加权信息熵，就可以计算信息增益。所谓信息增益就是指 $E(S)$ 和 $E(S, x_i)$ 之间的差，即

$$G(S, x_i) = E(S) - E(S, x_i) = E(S) - \sum_{t=1}^r \frac{|S_t|}{|S|} \times E(S_t)$$

可见，**信息增益所描述的是信息的确定性，其值越大，信息的确定性越高**



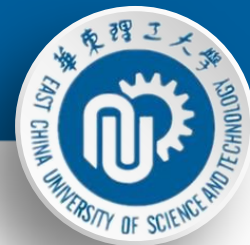
- 机器学习概述
- 记忆学习
- 示例学习
- 决策树学习
 - a. 决策树的概念
 - b. ID3算法
- 统计学习
- 集成学习
- 粗糙集知识发现
- 线性回归



◆ ID3算法的描述

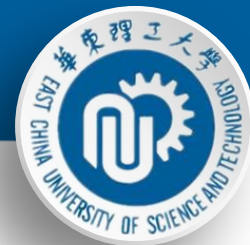
ID3算法的学习过程是一个**以整个样本集为根节点**，以**信息增益最大为原则**，选择条件属性进行扩展，逐步构造出决策树的过程。若假设 $S=\{s_1, s_2, \dots, s_n\}$ 为整个样本集， $X=\{x_1, x_2, \dots, x_m\}$ 为全体属性集， $Y=\{y_1, y_2, \dots, y_k\}$ 为样本类别。则ID3**算法描述如下**：

1. 初始化**样本集** $S=\{s_1, s_2, \dots, s_n\}$ 和**属性集** $X=\{x_1, x_2, \dots, x_m\}$ ，生成仅含根节点（ S, X ）的初始决策树。
2. 如果节点样本集中的所有样本全都属于同一类别，则将**该节点标记为叶节点**，并**标出该叶节点的类别**。算法结束。 否则执行下一步。
3. 如果**属性集为空**；或者**样本集中的所有样本在属性集上都取相同值**，即**所有样本都具有相同的属性值**，则同样将该节点标记为叶节点算法结束。 否则执行下一步。



◆ ID3算法的描述

4. 计算**每个属性的信息增益**，并选出信息增益最大的属性对当前决策树进行扩展
5. 对选定属性的每一个属性值，重复执行如下操作，至所有属性值全部处理完为止：
 - ① 为**每一个属性值生成一个分支**；并将样本集中与该分支有关的所有样本放到一起，形成该新生分支节点的样本子集；
 - ② 若样本子集为空，则**将此新生分支节点标记为叶节点**；
 - ③ 否则，若**样本子集中的所有样本均属于同一类别**，则将该节点标记为叶节点。
6. 从属性集中删除所选定的属性，得到新的属性集
7. 转到第3步。



◆ ID3算法简例(1/9)

例6.1 用ID3算法完成下述学生选课的例子，假设将决策 y 分为以下3类：

y_1 : 必修AI

y_2 : 选修AI

y_3 : 不修AI

对于电信和机电两类专业

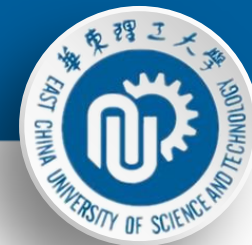
做出这些决策的依据有以下3个属性：

x_1 : **学历层次** $x_1=1$ 研究生, $x_1=2$ 本科

x_2 : **专业类别** $x_2=1$ 电信类, $x_2=2$ 机电类

x_3 : **学习基础** $x_3=1$ 修过AI, $x_3=2$ 未修AI

表1给出了一个关于选课决策的训练例子集 S 。

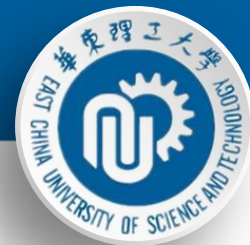


◆ ID3算法简例(2/9)

序号	属性值			决策方案 y_j
	x_1 (层次)	x_2 (专业)	x_3 (学否)	
1	1	1	1	y_3
2	1	1	2	y_1
3	1	2	1	y_3
4	1	2	2	y_2
5	2	1	1	y_3
6	2	1	2	y_2
7	2	2	1	y_3
8	2	2	2	y_3

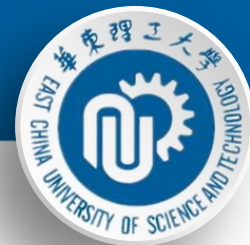
表1 学生选课决策的训练例子集

该训练例子集 S 的大小为 8。ID3 算法就是依据这些训练例子，以 (S, X) 为根节点，按照**信息熵下降最大的原则**来构造决策树的。



◆ ID3算法简例(3/9)

解：按照ID3算法，先**初始化**样本集 $S=\{1,2,3,4,5,6,7,8\}$ 和属性集 $X=\{x_1, x_2, x_3\}$ 生成仅含根节点 (S, X) 的初始决策树。其中， S 中的数字为样本集中相应样本的编号。然后通过算法第2、3步，执行其第4步，计算根节点 (S, X) 关于每一个属性的信息增益，并选择具有**最大信息增益的属性**对根节点进行扩展。



◆ ID3算法简例(4/9)

可求得各属性的信息增益为：

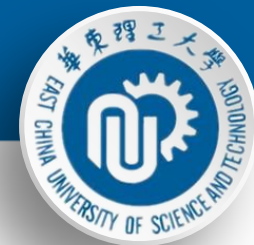
$$\begin{aligned} G((S,X), x_1) &= E(S,X) - E((S,X), x_1) \\ &= 1.2988 - 1.1557 = 0.1431 \end{aligned}$$

$$\begin{aligned} G((S,X), x_2) &= E(S,X) - E((S,X), x_2) \\ &= 1.2988 - 1.1557 = 0.1431 \end{aligned}$$

$$\begin{aligned} G((S,X), x_3) &= E(S,X) - E((S,X), x_3) \\ &= 1.2988 - 0.75 = 0.5488 \end{aligned}$$

显然， x_3 的**信息增益最大**，因此应先选择 x_3 对**根节点进行扩展**。

接着执行 5，对 x_3 的所有属性值分别生成根节点 (S,X) 的不同分支节点。先取 $x_3 = 1$ ，生成根节点 (S,X) 的左分支节点。由于 $t = 1$ ，设所得节点的样本子集为 S_1' ，则有 $S_1' = \{1,3,5,7\}$ 。又由于该样本子集 S_1' 中的所有样本均属于同一类别，故将该节点标记为叶节点，并标出其类别 y_3 。

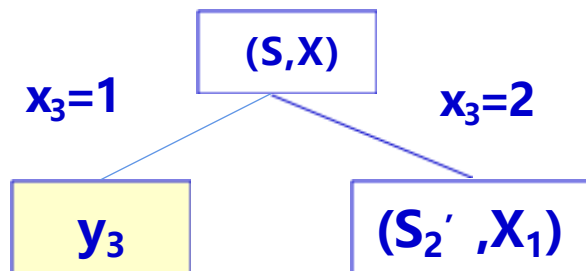


◆ ID3算法简例(5/9)

再取 $x_3=2$, 生成根节点 (S,X) 的右分支节点。由于 $t=2$, 设所得节点的样本子集为 S_2' , 则有 $S_2'=\{2,4,6,8\}$ 。

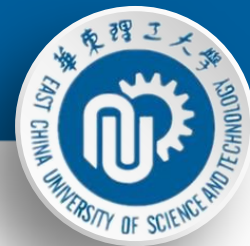
显然该样本子集非空, 且其中的样本并非同一类别, 故算法第5步全部完成。

执行第6步, 从属性集 $X=\{x_1,x_2,x_3\}$ 中删除本轮扩展所选定的属性 x_3 , 得到新的属性集 $X_1=\{x_1, x_2\}$ 。至此, 根节点 (S,X) 的扩展过程完成, 所得到的当前部分决策树如图所示。



扩展根节点后的部分决策树

然后返回算法第3步, 进入下一轮扩展过程。



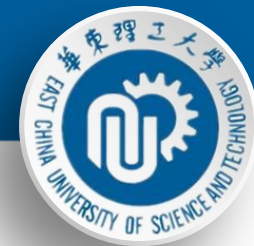
◆ ID3算法简例(6/9)

显然, 第3步的条件不满足, 接着执行算法第4步

计算节点(S_2' , X_1)下各属性的信息增益, 并选择具有最大信息增益的属性对决策树进行扩展。

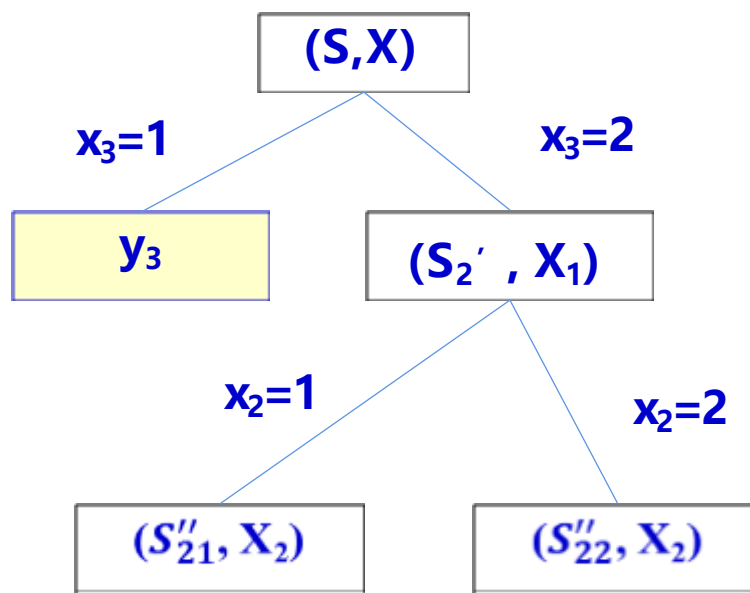
可得: x_2 的信息增益大于 x_1 的信息增益, 因此应**先扩展属性 x_2** 。

接着执行算法第5步, 对属性 x_2 的所有取值分别生成节点(S_2' , X_1)的不同分支节点。当 $x_2 = 1$ 时, 生成其左子节点; 当 $x_2 = 2$ 时, 生成其右子节点。

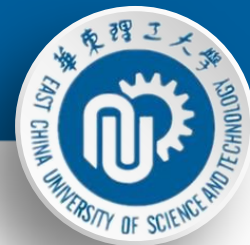


◆ ID3算法简例(7/9)

接着执行算法第6步，从当前属性集 $X_1 = \{x_1, x_2\}$ 中删除本轮扩展所选定的属性 x_2 ，得到新的属性集 $X_2 = \{x_1\}$ 。当前的部分决策树如图所示。



扩展根节点 (S_2', X_1) 后的部分决策树



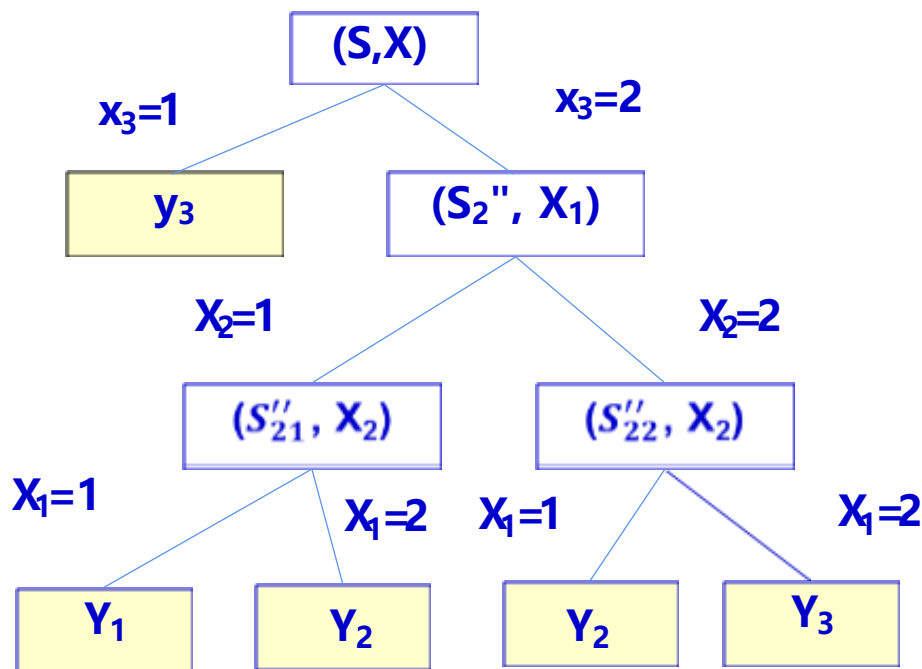
◆ ID3算法简例(8/9)

接着返回算法第3步，进入下一轮扩展过程。由于第3步中的条件都不满足，故执行第4步。由于此时属性集 X 中只有 x_1 ，无须再进行属性选择，直接执行算法第5步，对属性 x_1 的所有取值，依次完成对各非叶节点的扩展，并将所有新生分支节点标记为叶节点。

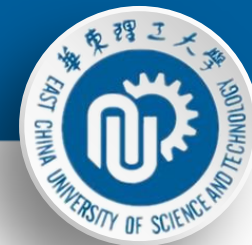
执行算法第6步，此时从 $X_2 = \{ x_1 \}$ 中删除属性 x_1 ，当前属性集为空；

返回算法第3步，此时因属性集为空，算法结束。

右图为最终所得完整决策树。

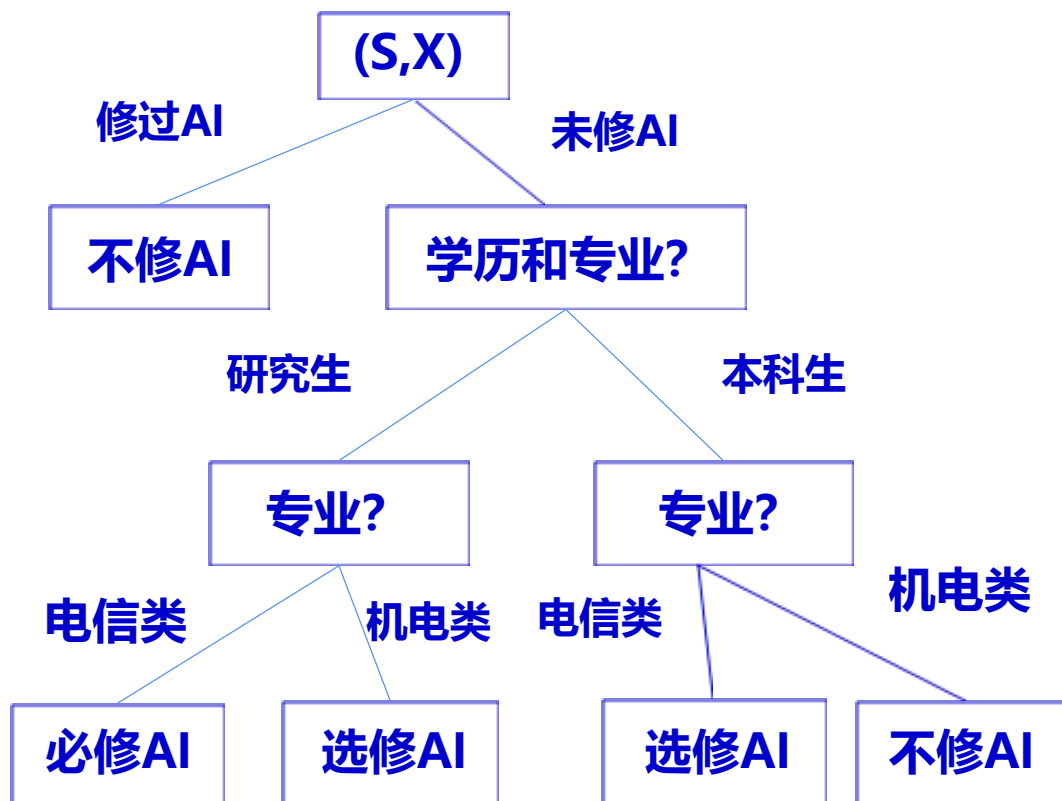


最终得到的完整决策树

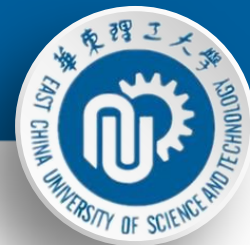


◆ ID3算法简例(9/9)

上述该决策树的含义如图所示。其中，从根节点到每个叶节点的路径都代表了一条知识。



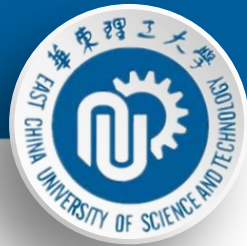
最终所得完整决策树的含义



◆ 集成学习的基本概念

集成学习是指为解决同一问题，先训练出一系列**个体学习器**（或称弱学习器），然后再根据某种规则把这些个体学习器的学习结果**整合**到一起，得到比单个个体学习器更好的学习效果。集成学习的基本结构如下图所示。

集成学习包括两大基本问题，一个是个体学习器的构造，另一个是个体学习器的合成。



◆ 集成学习的两种方式

➤ 同质集成

要求构造个体学习器时使用相同类型的学习方法，构造出来的多个个体学习器为同质学习器。

所谓同质，是指同一类型，例如要使用决策树都为决策树。

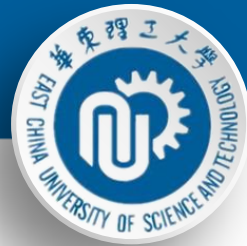
这种采用相同学习方法构造个体学习器的集成学习称为**狭义集成学习**，其个体学习器称为基学习器，所用的学习算法称为基学习算法。

➤ 异质集成

不要求构造个体学习器时使用同一类型的学习方法，而是可以异质。

所谓异质，是指不同类型，例如可以同时使用决策树和神经网络去构造个体学习器。

这种集成学习又称为**广义集成学习**，构造个体学习器所用学习算法不再称基学习算法，构造出来的个体学习器也不再称基学习器，而直接称其为个体学习器。



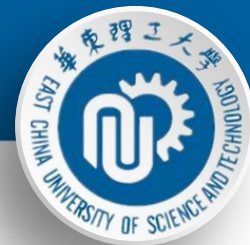
◆ 集成学习的基本类型

根据个体学习器生成方式的不同，以及个体学习器之间依赖关系的不同，集成学习可分为**Boosting方法**和**Bagging方法**两大基本类。

Boosting方法的基本思想：

- a) 从初始训练集开始，先为每个训练样本平均分配初始权重，并训练出弱学习器1；
- b) 然后通过提高错误率高的训练样本的权重，降低错误率低的训练样本的权重，得到训练样本的新的权重分布，并在在该权重分布上训练出弱学习器2；
- c) 依此逐轮迭代，直至达到最大迭代轮数，最后再将训练出来的这些弱学习器合成到一起，形成最终的强学习器。

其典型代表是AdaBoost算法和提升树(boosting tree)算法。



◆ 集成学习的基本类型

Bagging方法则不同，其基本思想为：

在给定初始训练集和弱学习算法的前提下，每轮迭代都使用可重采样的随机抽样方法从初始训练集产生出本轮的训练子集。

并利用选定的弱学习算法训练出本轮迭代的弱学习器，依此逐轮迭代，直至达到最大迭代轮数。

最后再按照某种合成方式将这这些训练出来的弱学习器合成到一起，形成最终的强学习器。

其典型代表包括bagging算法和随机森林（Random Forest）算法等。