



第十五章 计算复杂度理论与机器学习中的算法

湖南大学信息科学与工程学院



15.1 图灵机的思想与模型简介

15.2 计算复杂性理论简介

15.3 常见 NP 完全问题

15.4 机器学习中的算法

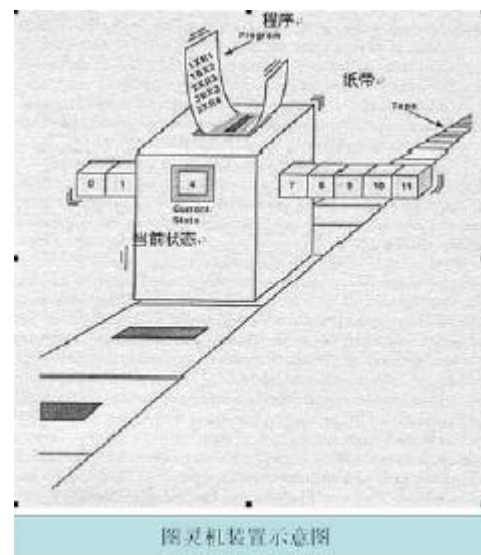
15.1 图灵机的思想与模型简介



图灵是谁？ 图灵及其贡献

- ◆ **图灵**(Alan Turing, 1912~1954), 出生于英国伦敦, 19 岁入剑桥皇家学院, 22 岁当选为皇家学会会员。
- ◆ 1937 年, 发表了论文《论可计算数及其在判定问题中的应用》, 提出了**图灵机模型**, 后来, 冯·诺依曼根据这个模型设计出历史上第一台电子计算机。
- ◆ 1950 年, 发表了划时代的文章:《机器能思考吗?》, 成为了人工智能的开山之作。
- ◆ 计算机界于1966年设立了最高荣誉奖: **ACM 图灵奖**。

你能查阅一下哪些人获得图灵奖了吗?
因为什么贡献而获奖呢?

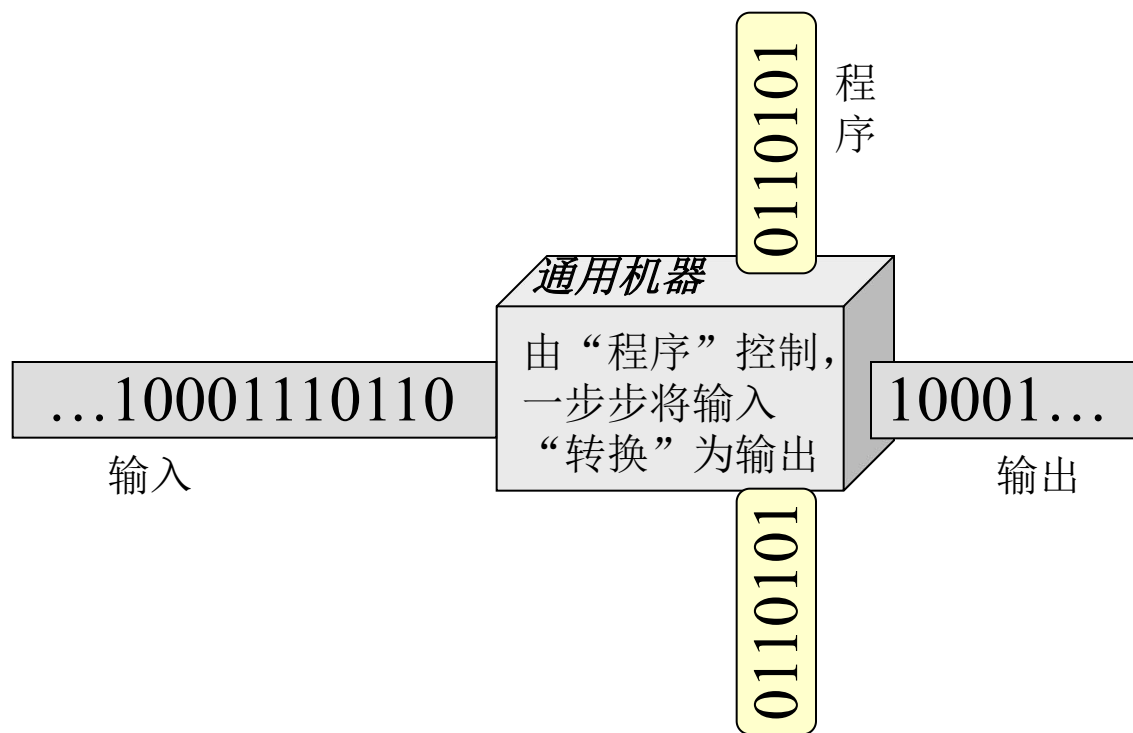


15.1 图灵机的思想与模型简介



图灵认为什么是计算？ 计算

◆ 所谓**计算**就是计算者(人或机器)对一条两端可无限延长的纸带上的一串0或1，执行指令一步一步地改变纸带上的0或1，经过有限步骤最后得到一个满足预先规定的符号串的**变换过程**。



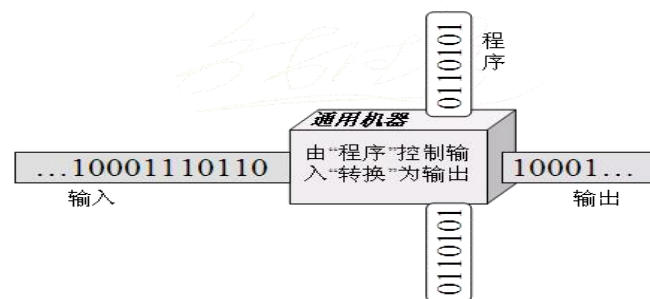
15.1 图灵机的思想与模型简介



图灵机的思想

是关于数据、指令、程序及程序/指令自动执行的基本思想。

- ◆ 输入被制成一串0和1的纸带，送入机器中----**数据**。如00010000100011...
- ◆ 机器可对输入纸带执行的**基本动作**包括：“翻转0为1”，或“翻转1为0”，“前移一位”，“停止”。
- ◆ 对基本动作的控制----**指令**，机器是按照指令的控制选择执行哪一个动作，指令也可以用0和1来表示：**01**表示“翻转0为1”（当输入为1时不变），**10**表示“翻转1为0”（当输入0时不变），**11**表示“前移一位”，**00**表示“停止”。
- ◆ 输入如何变为输出的控制可以用指令编写一个**程序**来完成，如：011110110111011100...
- ◆ 机器能够读取程序，按程序中的指令顺序读取指令，读一条指令**执行**一条指令。由此实现**自动计算**。



15.1 图灵机的思想与模型简介



图灵机是什么？

图灵机模型

- ◆ 基本的**图灵机模型**为一个七元组,如右图示意
- ◆ 几点结论:
- ◆ (1) 图灵机是一种思想模型, 它由一个控制器(有限状态转换器), 一条可无限延伸的带子和一个在带子上左右移动的读写头构成。
- ◆ (2) **程序是五元组 $\langle q, X, Y, R(\text{或}L\text{或}N), p \rangle$ 形式的指令集**。其定义了机器在一个特定状态 q 下从方格中读入一个特定字符 X 时所采取的动作作为在该方格中写入符号 Y , 然后向右移一格 R (或向左移一格 L 或不移动 N), 同时将机器状态设为 p 供下一条指令使用。

$$M = (Q, \Sigma, \Gamma, \delta, q_0, B, F)$$

其中:

Q : 状态的有穷集合

q_0 : 开始状态

F : 终止状态集合

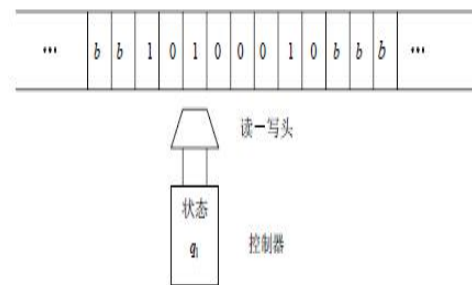
Γ : 带符号表

B : 空白符号

Σ : 输入字母表

δ : 移动函数, (1): $\delta(q, X) = (p, Y, R)$ 表示 M 在状态 q 读入符号 X , 将状态改为 p , 并在这个 X 所在的带方格中印刷符号 Y , 然后将读头向右移动一格。

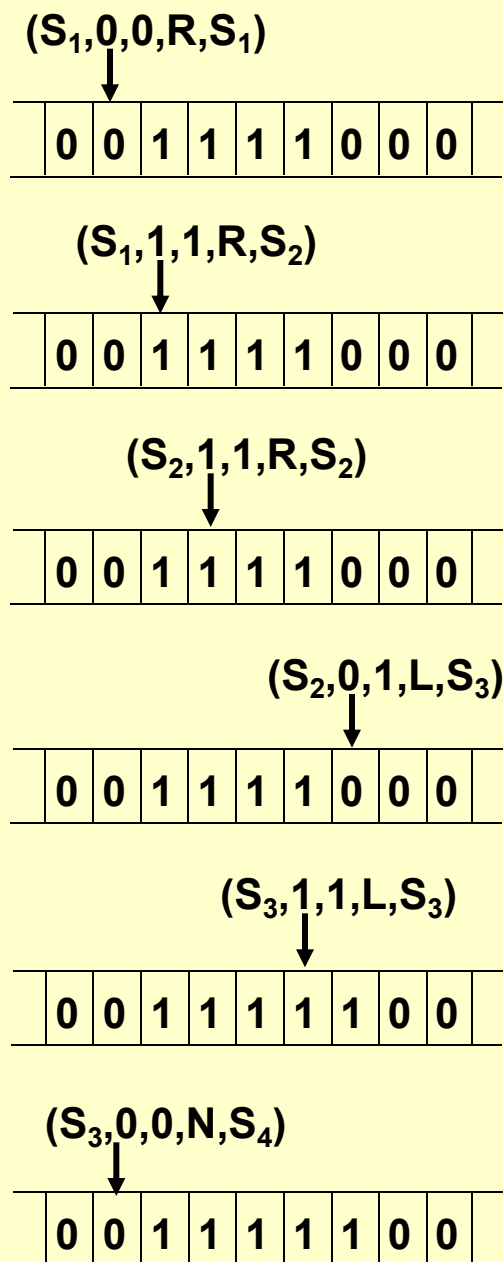
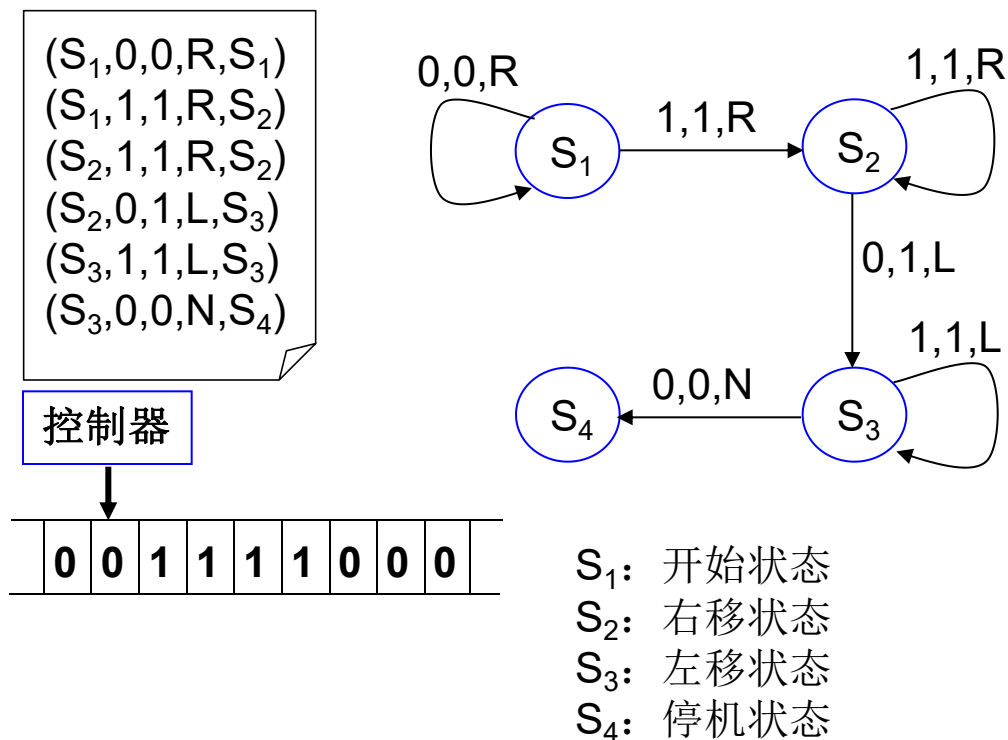
(2): $\delta(q, X) = (p, Y, L)$ 表示 M 在状态 q 读入符号 X , 将状态改为 p , 并在这个 X 所在的带方格中印刷符号 Y , 然后将读头向左移动一格。



15.1 图灵机的思想与模型简介



图灵机模型示例。(注:圆圈内的的是状态, 箭线上的是
<X,Y,R>, 其含义见前页)



功能：将一串1的后面再加一位1

15.1 图灵机的思想与模型简介



几点结论(续):

◆(3)图灵机模型被认为是计算机的基本理论模型

----计算机是使用相应的程序来完成任何设定好的任务。图灵机是一种离散的、有穷的、构造性的问题求解思路，一个问题的求解可以通过构造其图灵机(即程序)来解决。

◆(4)图灵认为：凡是能用算法方法解决的问题也一定能用图灵机解决；凡是图灵机解决不了的问题任何算法也解决不了----图灵可计算性问题。



15.1 图灵机的思想与模型简介

15.2 计算复杂性理论简介

15.3 常见 NP 完全问题

15.4 机器学习中的算法

最优化问题 vs 判定问题



最优化问题： 给定问题，找出所有满足条件的解中值最优的那个

判定问题： 给定问题，判断是否有满足条件的解

判定问题的形式化描述简单，而且很多优化问题的难度与判定问题的难度相关



- **最优化问题**：给定一个带权图 $G=(V, E, W)$ ，计算 V 中 s 到 V 中点 t 之间的最短路径
- **判定问题**：给定一个带权图 $G=(V, E, W)$ ，计算 V 中 s 到 V 中点 t 之间是否有一条路径
- **判定问题和最优化问题之间的关系**：给定一个带权图 $G=(V, E, W)$ ，计算 V 中 s 到 V 中点 t 之间是否有一条长度为 k 的路径



- **最优化形式：** 求一个整数序列中出现频率最高的数
- **判定形式：** 一个整数序列中是否存在出现频率为 k 的数。
- **最优化 \rightarrow 判定：** 枚举 k ，返回判定有解的最大的一个。

- 给定一个简单无向图，要给图的每一个顶点着色，要求相邻的顶点着不同的颜色。
- 最优化形式：最少需要多少种颜色
- 判定形式：是否存在最多只需要 k 种颜色的解
- 最优化 \rightarrow 判定：枚举 k ，返回判定有解的最大的一个。



瓶颈生成树

一个无向图 G 上的瓶颈生成树是 G 上一种特殊的生成树。一个瓶颈生成树 T 上权重最大边的权重是 G 中所有生成树中最小的。 T 上最大权重的边的权重称为 T 的值。

求解算法： 1. 求出边权值的中位数（类似于求 n th element一类问题） M ，以此将图 G 的边按权值分成两部分，一部分小于等于 M ，另一部分大于 M ；

2. 利用 b （P372）提出的方法判断图 G 瓶颈生成树的 T 值是否不超过 M ，也就是看这个 T 值位于大小哪半边；

3. 若位于小半边，则将大半边里的边删除，并回到步骤1；

4. 若位于大半边，则小半边组成的图必不连通，将其连通分量各收缩成一个点，再和大半边重新组成一个图 G_2 ，并回到步骤1。

P 类、 EXP 类和 R 类问题



P 类：在多项式时间可解的判定问题类 ($O(n^k)$)

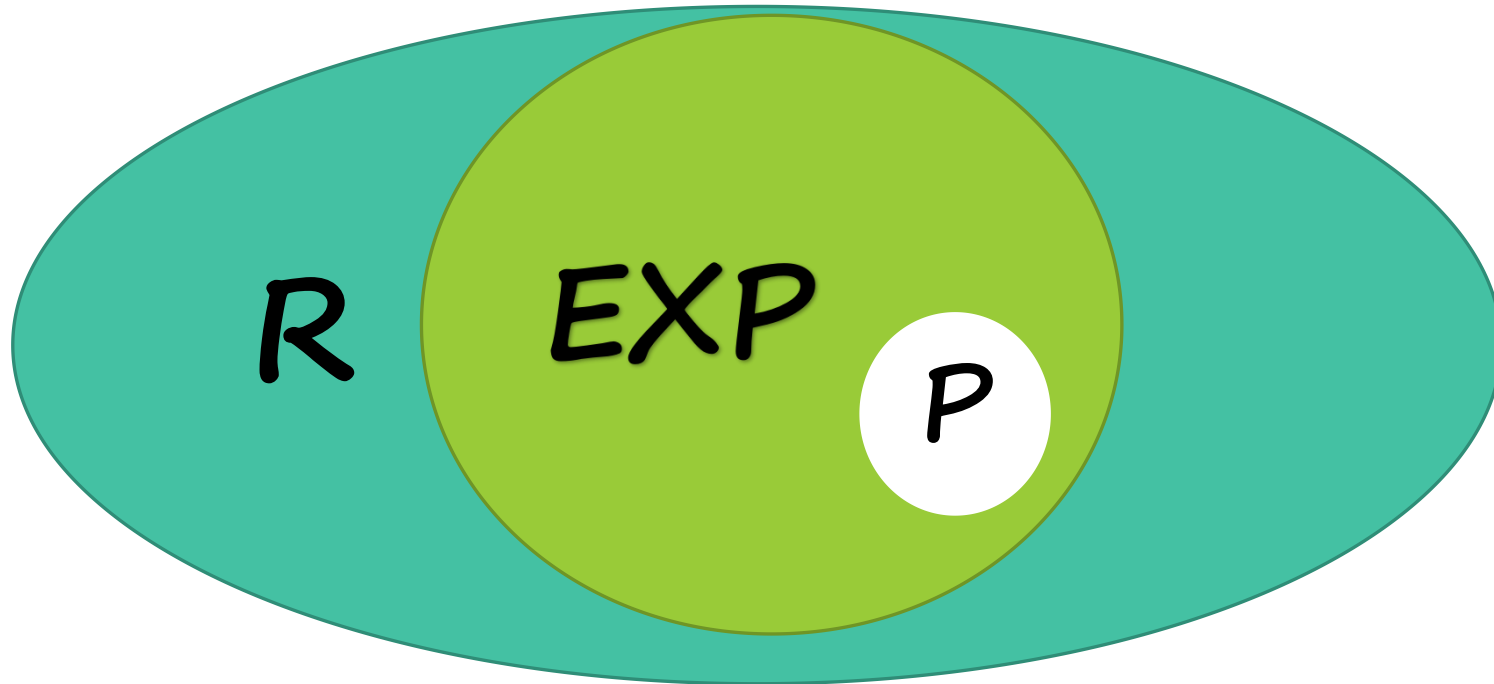
- 最短路径问题

EXP 类：在指数级时间可解的判定问题类

- 围棋问题

R 类：在有限的时间里可解的判定问题类，即计算机可解的问题

All Problems



NP类问题



定义：每个解可以在多项式时间进行检查的判定问题类

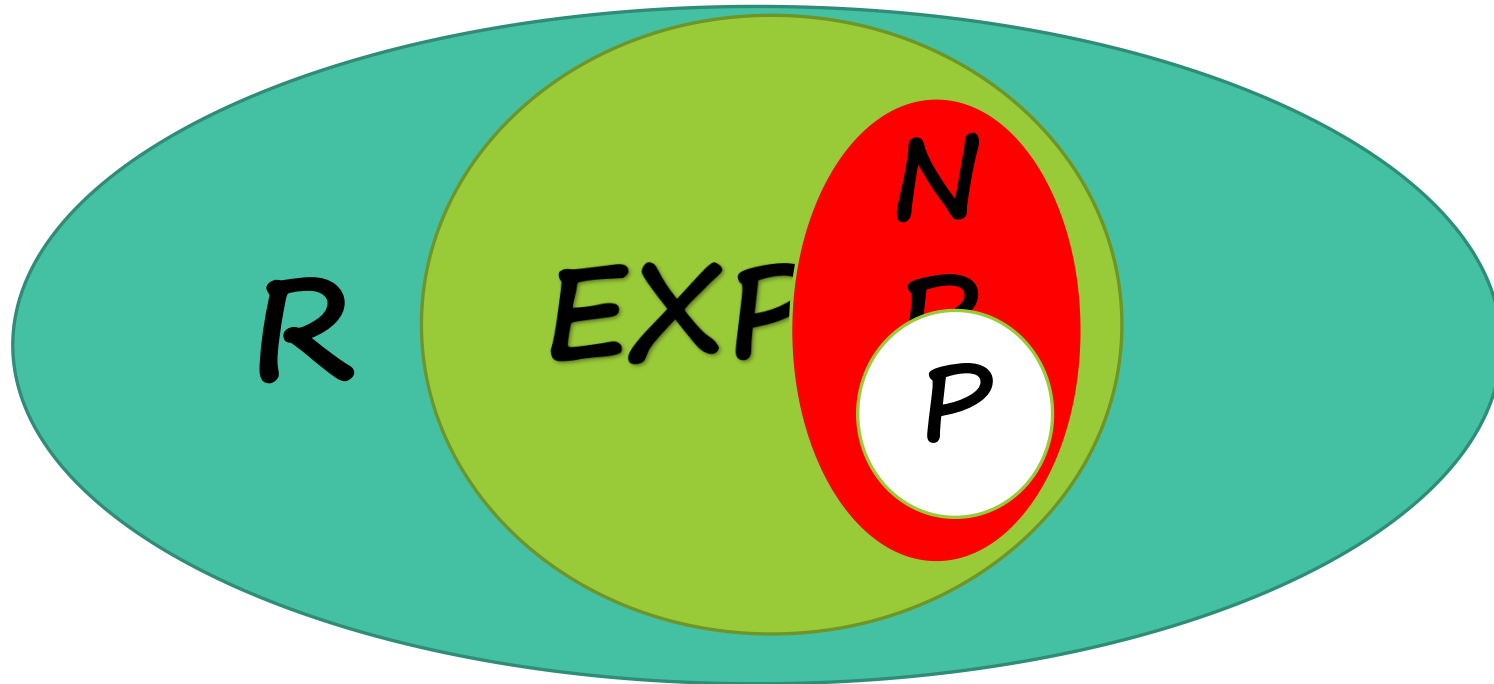
到目前还没有找到多项式时间的确定型算法

是否为难解的问题（目前还不清楚），即 P 是否等于 NP 不确定

猜测： $P \neq NP$

练习：NP问题是已经被证明地必须在指数时间内被解决（解决是求解的意思）？

All Problems



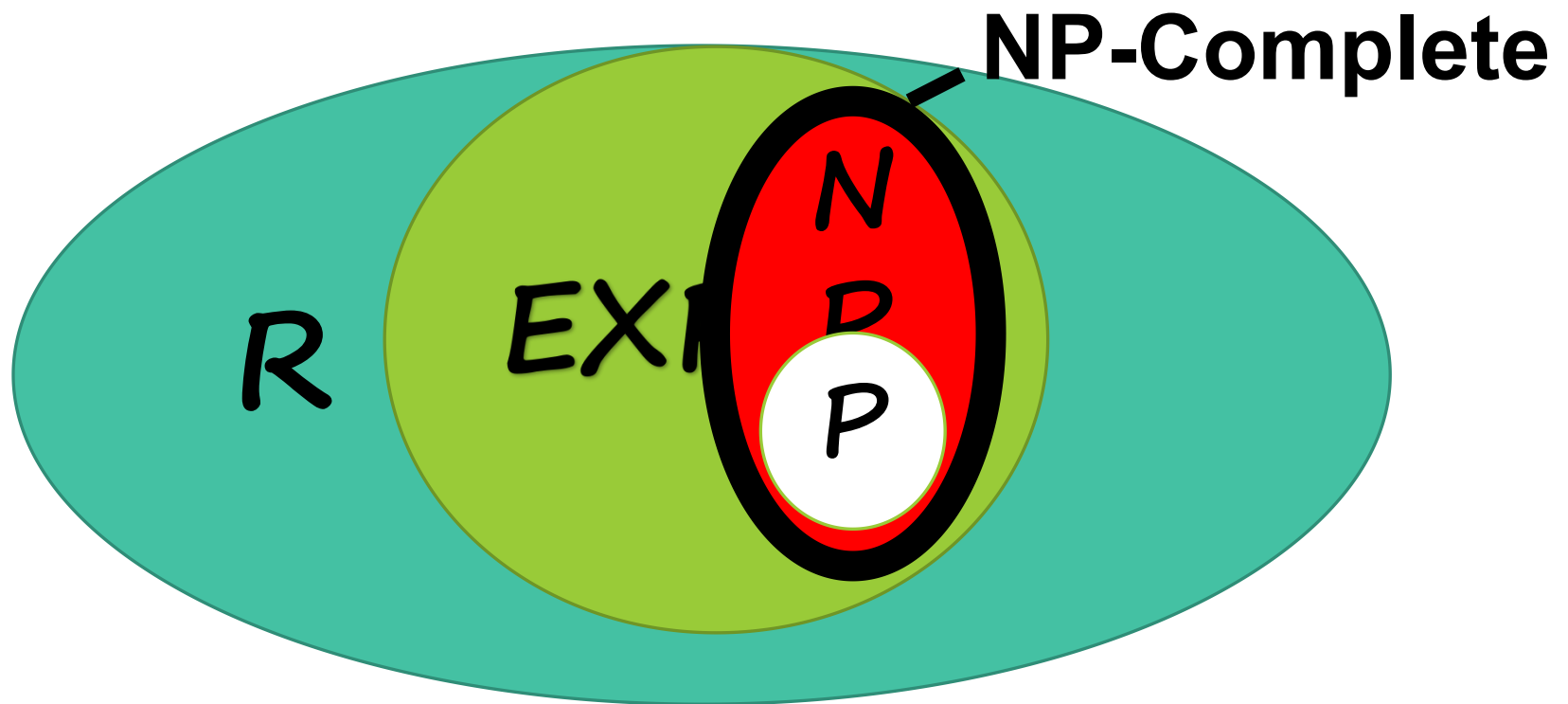
NP -hard类和 NP -complete类问题



NP -hard类: 所有不比 NP 类问题容易的问题

NP -complete类: NP 问题 \cap NP - hard

All Problems





多项式时间归约

- 多项式时间归约是比较两个问题的相对难度的重要手段。
- 对于两个问题 X 和 Y ，用 $T(X)$ 和 $T(Y)$ 表示它们的时间复杂度。如果 $T(Y)=f(T(X))$ ，其中 f 是一个多项式函数，则写作 $Y \leq_p X$ ，即 Y 可以在多项式时间内归约到 X 。通俗地讲 X 至少和 Y 一样难。
- 定理1：设 $Y \leq_p X$ 。如果 X 存在多项式时间解法，则 Y 同样存在多项式时间解法。
- 定理2：设 $Y \leq_p X$ 。如果 Y 不存在多项式时间解法，则 X 同样不存在多项式时间解法。
- 定理3：设 $Z \leq_p Y$ ， $Y \leq_p X$ ，则 $Z \leq_p X$

0-1 整数背包问题 --- NPC 问题



动态规划算法时间复杂度分析

时间代价 $O(n \times S)$, 空间代价 $O(n \times S)$

注意：这是伪多项式时间，因为 S 是作为一个整数输入

设 S 的位数是 $L = \log_2 S$, 那么相当于时间代价和空间代价是 $O(n2^L)$

15.1 图灵机的思想与模型简介

15.2 计算复杂性理论简介

15.3 常见 NP 完全问题

15.4 机器学习中的算法



- ◆ 1971年, 斯蒂芬.库克 (*Stephen Cook*) 发表Cook定理。
- ◆ 1972年, 理查德.卡普 (*Richard Karp*) 提出并证明了21个NP完全问题。

卡普的21个NPC问题



布尔可满足性问题 (*Satisfiability*)

0-1整数规划 (*0-1 integer programming*)

分团问题 (*Clique*, 参考独立集)

集合配置问题 (*Set packing*)

最小顶点覆盖问题 (*Vertex cover*)

集合覆盖问题 (*Set covering*)

反馈节点集问题 (*Feedback node set*)

反馈弧集问题 (*Feedback arc set*)

有向哈密顿回路问题 (*Directed Hamiltonian cycle*)

无向哈密顿回路问题 (*Undirected Hamiltonian cycle*)

卡普的21个NPC问题



三元布尔可满足性问题 (*3-SAT*)

图着色问题 (*Chromatic number*)

分团覆盖问题 (*Clique cover*)

精确覆盖问题 (*Exact cover*)

命中集问题 (*Hitting set*)

斯坦纳树问题 (*Steiner tree*)

三维匹配问题 (*3-dimensional matching*)

背包问题 (*Knapsack*)

作业排序 (*Job sequencing*)

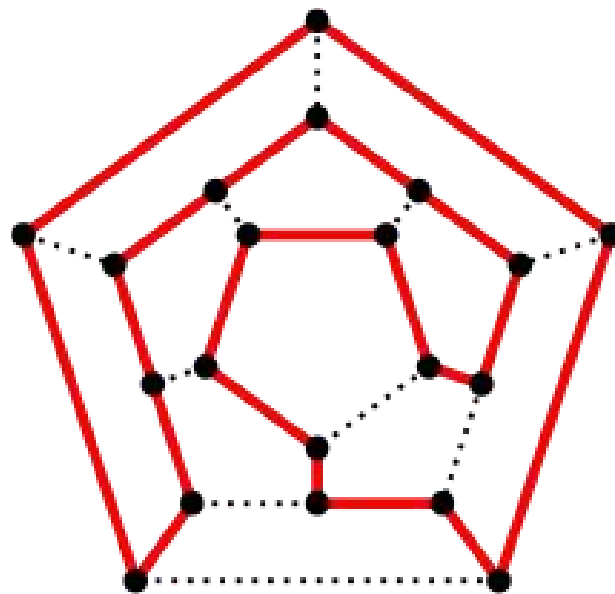
划分问题 (*Partition*)

最大割问题 (*Max cut*)

哈密顿回路问题

实例：图 $G=(V,E)$ 且 $|V|=n$

问： G 中是否包含一条哈密顿回路





15.1 图灵机的思想与模型简介

15.2 计算复杂性理论简介

15.3 常见 NP 完全问题

15.4 机器学习中的算法



机器学习算法的基本问题

输入：给定一个数据集 $D = \{x_1, x_2, \dots, x_m\}$,

**其中每个元素 $x_i = \{x_{i1}, x_{i2}, \dots, x_{id}\}$ 是一个 d 维
向量并被称为一个实例 (instance)**

输出：能对这些实例进行标记的一个模型

**按照所输出的模型，每个示例 x_i 会得到一个
值 y_i 作为标记，所有的标记组成的集合 Y 成
为“标记空间”**



机器学习算法的分类

分类：输入数据集中实例有标记

聚类：输入数据集中实例无标记



现有机器学习算法利用输入数据多轮迭代地生成模型，每轮迭代中都是贪心地选择当前最优模型来生成。

为了进一步降低计算空间进而提高效率，在贪心的基础上，很多机器学习算法在执行过程中还加上了随机的思想。它们随机地在下一步可能的最优模型中选择若干分支。



决策树是一种树形结构，其中每个内部节点表示一个属性上的测试，每个分支代表一个测试输出，每个叶节点代表一种类别

示例数据（西瓜数据集2.0）



编号	色泽	根蒂	敲声	纹理	脐部	触感	好瓜
1	青绿	蜷缩	浊响	清晰	凹陷	硬滑	是
2	乌黑	蜷缩	沉闷	清晰	凹陷	硬滑	是
3	乌黑	蜷缩	浊响	清晰	凹陷	硬滑	是
4	青绿	蜷缩	沉闷	清晰	凹陷	硬滑	是
5	浅白	蜷缩	浊响	清晰	凹陷	硬滑	是
6	青绿	稍蜷	浊响	清晰	稍凹	软粘	是
7	乌黑	稍蜷	浊响	稍糊	稍凹	软粘	是
8	乌黑	稍蜷	浊响	清晰	稍凹	硬滑	是
9	乌黑	稍蜷	沉闷	稍糊	稍凹	硬滑	否
10	青绿	硬挺	清脆	清晰	平坦	软粘	否
11	浅白	硬挺	清脆	模糊	平坦	硬滑	否
12	浅白	蜷缩	浊响	模糊	平坦	软粘	否
13	青绿	稍蜷	浊响	稍糊	凹陷	硬滑	否
14	浅白	稍蜷	沉闷	稍糊	凹陷	硬滑	否
15	乌黑	稍蜷	浊响	清晰	稍凹	软粘	否
16	浅白	蜷缩	浊响	模糊	平坦	硬滑	否
17	青绿	蜷缩	沉闷	稍糊	稍凹	硬滑	否



决策树典型算法

Hunt算法

输入: 训练集 $D = \{(x_1, y_1), (x_2, y_2), \dots, (x_m, y_m)\}$;

属性集 $A = \{a_1, a_2, \dots, a_d\}$.

过程: 函数 TreeGenerate(D, A)

1: 生成结点 node;

2: if D 中样本全属于同一类别 C then

3: 将 node 标记为 C 类叶结点; return

无需划分

4: end if

5: if $A = \emptyset$ OR D 中样本在 A 上取值相同 then

6: 将 node 标记为叶结点, 其类别标记为 D 中样本数最多的类; return

7: end if

无法划分

8: 从 A 中选择最优划分属性 a_* ;

9: for a_* 的每一个值 a_*^v do

10: 为 node 生成一个分支; 令 D_v 表示 D 中在 a_* 上取值为 a_*^v 的样本子集;

11: if D_v 为空 then

12: 将分支结点标记为叶结点, 其类别标记为 D 中样本最多的类; return

13: else

14: 以 TreeGenerate($D_v, A \setminus \{a_*\}$) 为分支结点

不能划分

15: end if

16: end for

输出: 以 node 为根结点的一棵决策树

递归返回, 情形(1).

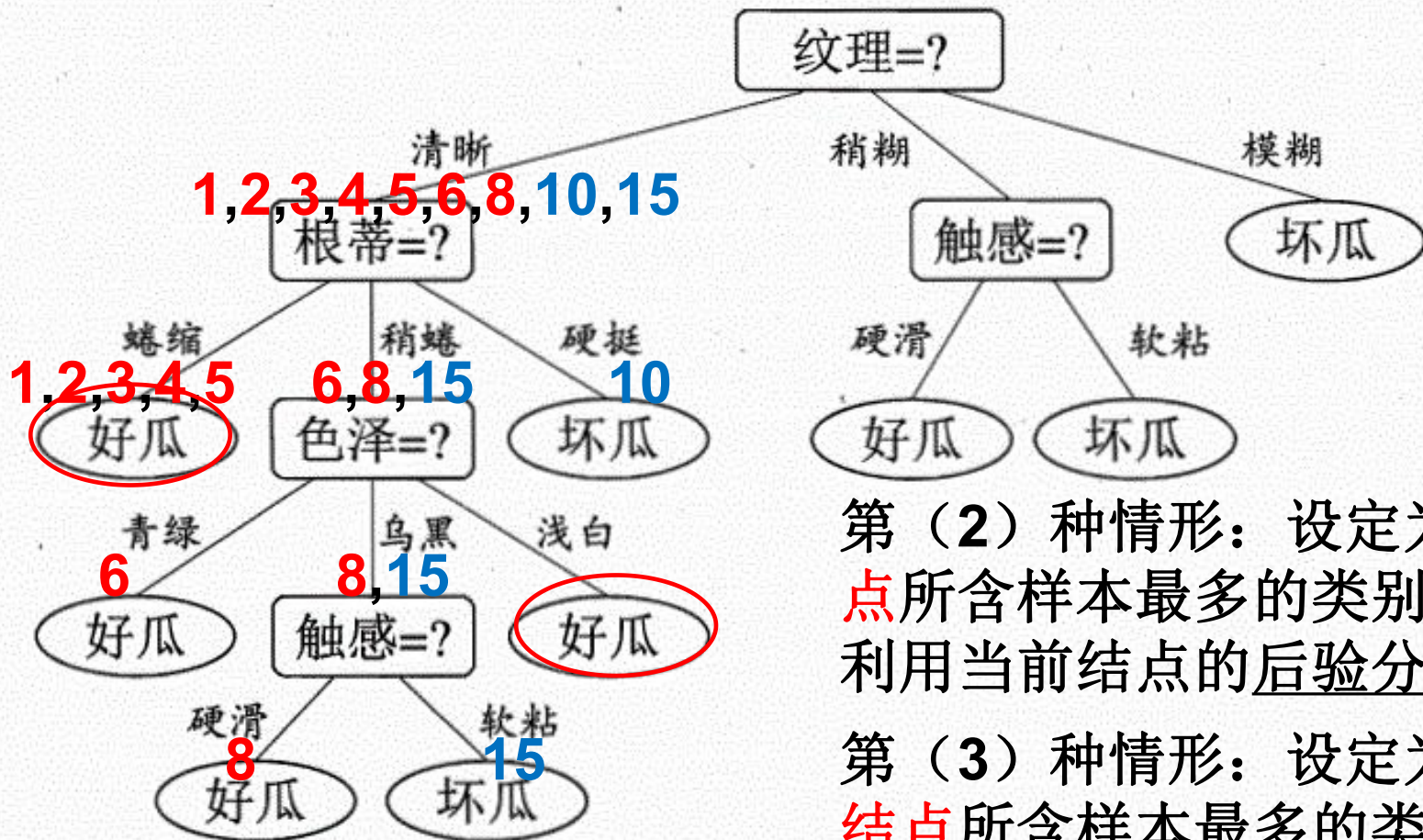
递归返回, 情形(2).

我们将在下一节讨论如何获得最优划分属性.

递归返回, 情形(3).

从 A 中去掉 a_* .

示例决策树



第（2）种情形：设定为**该结点**所含样本最多的类别
利用当前结点的后验分布

第（3）种情形：设定为**其父结点**所含样本最多的类别
把父结点的样本分布作为当前结点的先验分布

图 4.4 在西瓜数据集 2.0 上基于信息增益生成的决策树

决策树学习的关键是算法的第8行：贪心地

选择最优划分属性

什么样的划分属性是最优的？

我们希望决策树的分支结点所包含的样本尽可能属于同一类别，即结点的“纯度”越来越高，可以高效地从根结点到达叶结点，得到决策结果。



信息熵与信息增益

“信息熵” (information entropy) 是度量样本集合纯度最常用的一种指标。假定当前样本集合 D 中第 k 类样本所占的比例为 p_k ($k = 1, 2, \dots, |\mathcal{Y}|$), 则 D 的信息熵定义为

$$\text{Ent}(D) = - \sum_{k=1}^{|\mathcal{Y}|} p_k \log_2 p_k . \quad (4.1)$$

$\text{Ent}(D)$ 的值越小, 则 D 的纯度越高。对于二分类任务 $|\mathcal{Y}| = 2$

假定离散属性 a 有 V 个可能的取值 $\{a^1, a^2, \dots, a^V\}$, 若使用 a 来对样本集 D 进行划分, 则会产生 V 个分支结点, 其中第 v 个分支结点包含了 D 中所有在属性 a 上取值为 a^v 的样本, 记为 D^v . 我们可根据式(4.1) 计算出 D^v 的信息熵, 再考虑到不同的分支结点所包含的样本数不同, 给分支结点赋予权重 $|D^v|/|D|$, 即样本数越多的分支结点的影响越大, 于是可计算出用属性 a 对样本集 D 进行划分所获得的“信息增益” (information gain)

$$\text{Gain}(D, a) = \text{Ent}(D) - \sum_{v=1}^V \frac{|D^v|}{|D|} \text{Ent}(D^v) . \quad (4.2)$$



信息熵示例

举例：求解划分根结点的最优划分属性

数据集包含**17**个训练样例：

8个正例（好瓜）占 $p_1 = \frac{8}{17}$

9个反例（坏瓜）占 $p_2 = \frac{9}{17}$

对于二分类任务 $|y| = 2$

以属性“色泽”为例计算其信息增益

根结点的信息熵：

$$\text{Ent}(D) = - \sum_{k=1}^2 p_k \log_2 p_k = - \left(\frac{8}{17} \log_2 \frac{8}{17} + \frac{9}{17} \log_2 \frac{9}{17} \right) = 0.998 .$$



信息增益示例

用“色泽”将根结点划分后获得**3**个分支结点的信息熵分别为：

$$\text{Ent}(D^1) = - \left(\frac{3}{6} \log_2 \frac{3}{6} + \frac{3}{6} \log_2 \frac{3}{6} \right) = 1.000 ,$$

$$\text{Ent}(D^2) = - \left(\frac{4}{6} \log_2 \frac{4}{6} + \frac{2}{6} \log_2 \frac{2}{6} \right) = 0.918 ,$$

$$\text{Ent}(D^3) = - \left(\frac{1}{5} \log_2 \frac{1}{5} + \frac{4}{5} \log_2 \frac{4}{5} \right) = 0.722 ,$$

属性“色泽”的信息增益为：

$$\begin{aligned} \text{Gain}(D, \text{色泽}) &= \text{Ent}(D) - \sum_{v=1}^3 \frac{|D^v|}{|D|} \text{Ent}(D^v) \\ &= 0.998 - \left(\frac{6}{17} \times 1.000 + \frac{6}{17} \times 0.918 + \frac{5}{17} \times 0.722 \right) \\ &= 0.109 . \end{aligned}$$

基于信息增益的决策树示例

类似的, 我们可计算出其他属性的信息增益:

$$\text{Gain}(D, \text{根蒂}) = 0.143; \quad \text{Gain}(D, \text{敲声}) = 0.141;$$

$$\text{Gain}(D, \text{纹理}) = 0.381; \quad \text{Gain}(D, \text{脐部}) = 0.289;$$

$$\text{Gain}(D, \text{触感}) = 0.006.$$

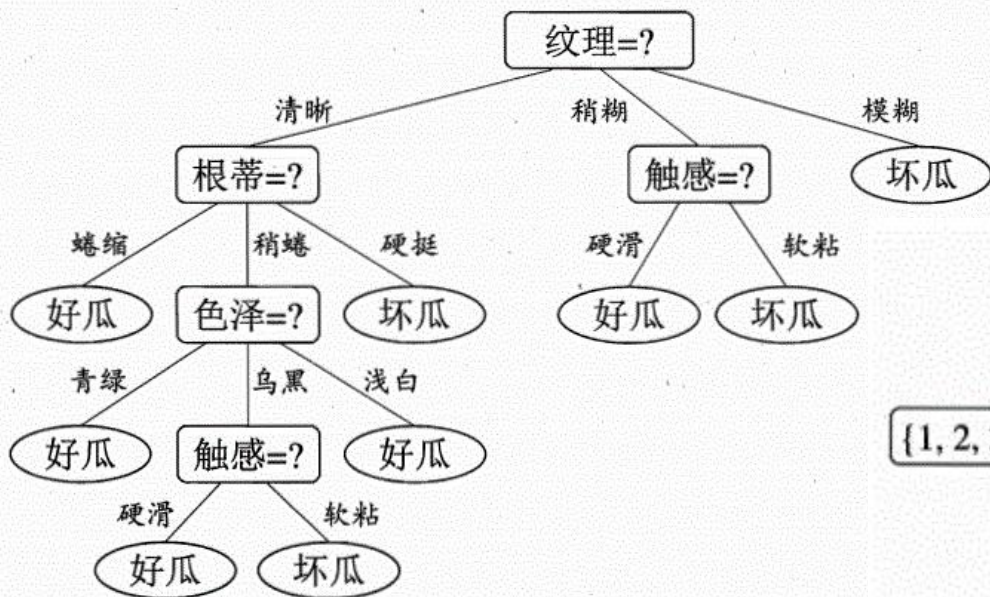


图 4.4 在西瓜数据集 2.0 上基于信息增益生成的决策树



图 4.3 基于“纹理”属性对根结点划分



信息增益的不足

若把“编号”也作为一个候选划分属性，则属性“编号”的信息增益为：

根结点的信息熵仍为： $Ent(D) = 0.998$

用“编号”将根结点划分后获得17个分支结点的信息熵均为：

$$Ent(D^1) = \dots = Ent(D^{17}) = -\left(\frac{1}{1} \log_2 \frac{1}{1} + \frac{0}{1} \log_2 \frac{0}{1}\right) = 0$$

则“编号”的信息增益为：

$$Gain(D, \text{编号}) = Ent(D) - \sum_{v=1}^{17} \frac{1}{17} Ent(D^v) = 0.998$$

远大于其他候选属性

信息增益准则对可取值数目较多的属性有所偏好

增益率

$$\text{Gain_ratio}(D, a) = \frac{\text{Gain}(D, a)}{\text{IV}(a)}, \quad (4.3)$$

其中

$$\text{IV}(a) = - \sum_{v=1}^V \frac{|D^v|}{|D|} \log_2 \frac{|D^v|}{|D|} \quad (4.4)$$

称为属性 a 的“固有价值” (intrinsic value) [Quinlan, 1993]. 属性 a 的可能取值数目越多(即 V 越大), 则 $\text{IV}(a)$ 的值通常会越大.

增益率准则对可取值数目较少的属性有所偏好

著名的**C4.5**决策树算法综合了信息增益准则和信息率准则的特点: 先从候选划分属性中找出信息增益高于平均水平的属性, 再从中选择增益率最高的。

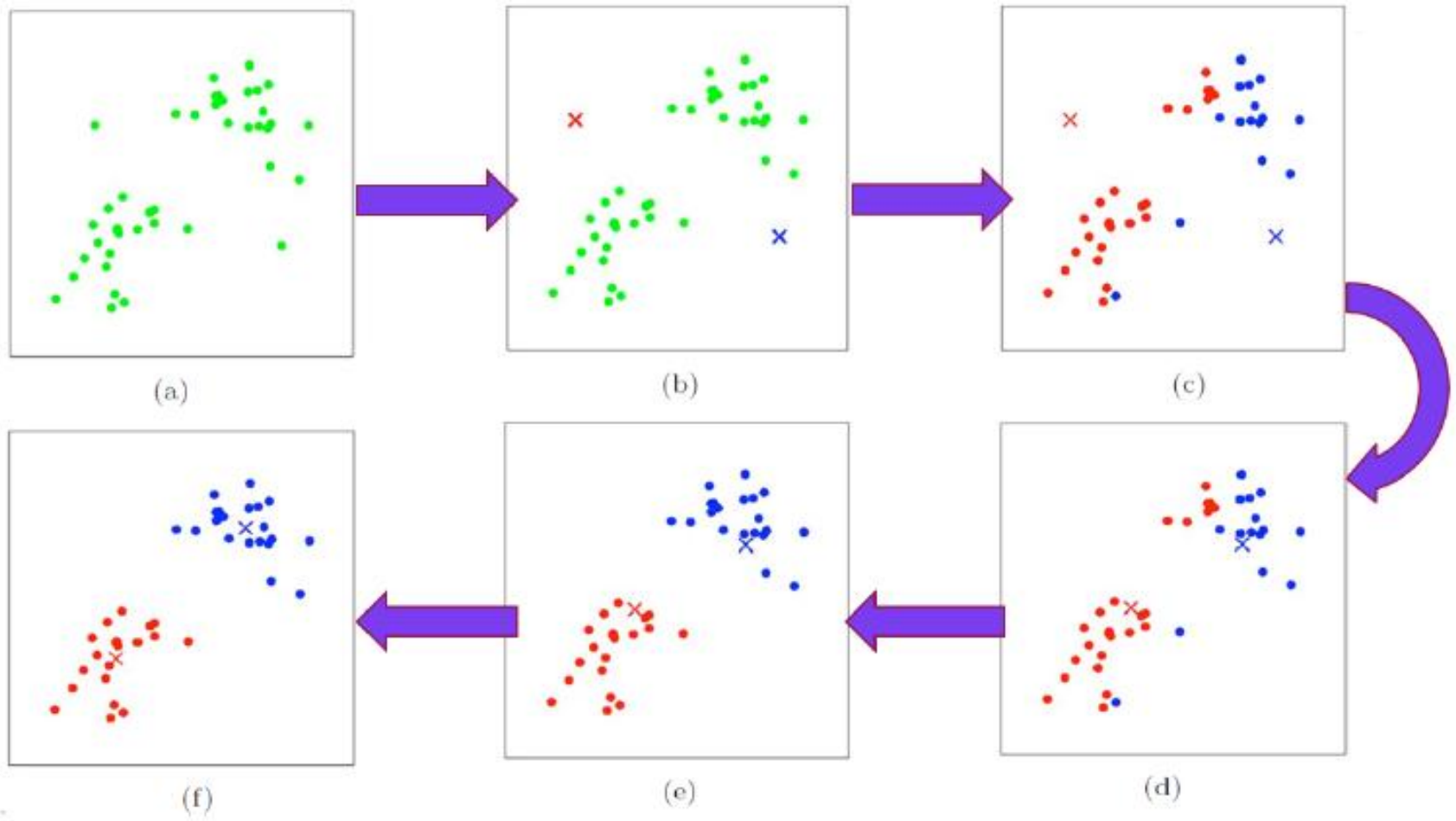


k-均值 (k-means) 算法，是一种得到最广泛使用的聚类算法

它将各个聚类子集内的所有数据样本的均值作为该聚类的代表点，其他点和其最近的代表点同一个类

- 1. 为每个聚类随机确定一个实例作为初始聚类中心，这样就有K个初始聚类中心**
- 2. 将每个实例按照最小距离原则分配到最近聚类（即贪心选择）**
- 3. 使用每个聚类中实例均值作为新聚类中心**
- 4. 重复步骤2.3直到聚类中心不再变化**
- 5. 结束，得到K个聚类**

算法运行实例



距离定义：典型定义为欧式距离：

$$d(x_i, x_j) = \sqrt{\sum_{k=1}^d (x_{ik} - x_{jk})^2}$$

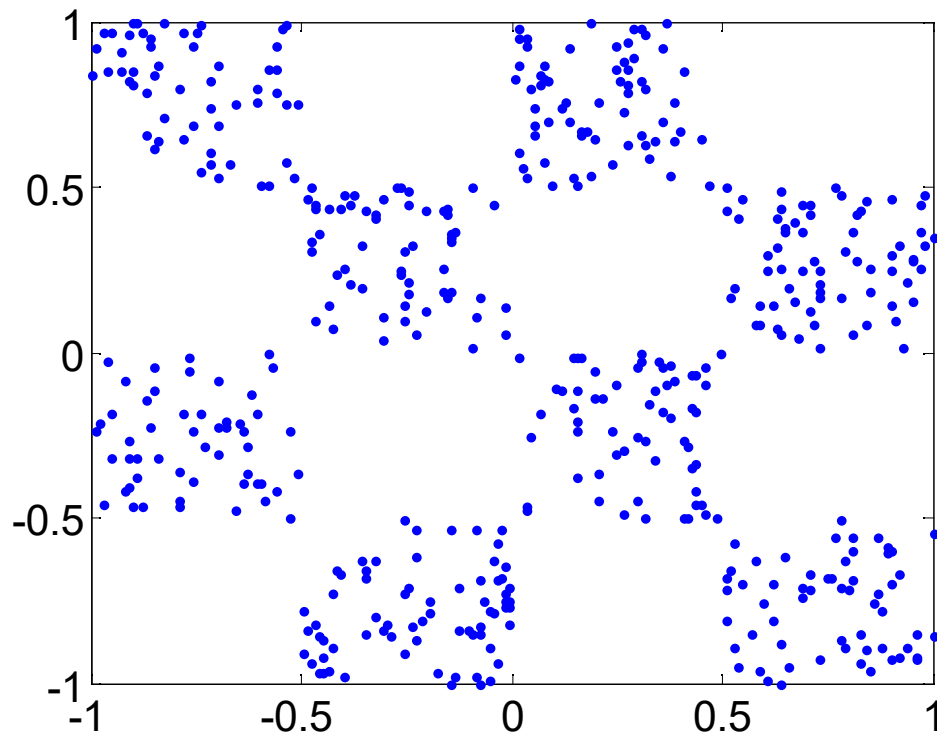


- **必须事先给出k（要生成的簇的数目），
而且对初值敏感；**
- **对于“噪声”和孤立点数据是敏感的，少
量的该类实例能够对结果产生极大的影响
；**
- **不能保证找到全局最优解，只能确保局部
最优解**



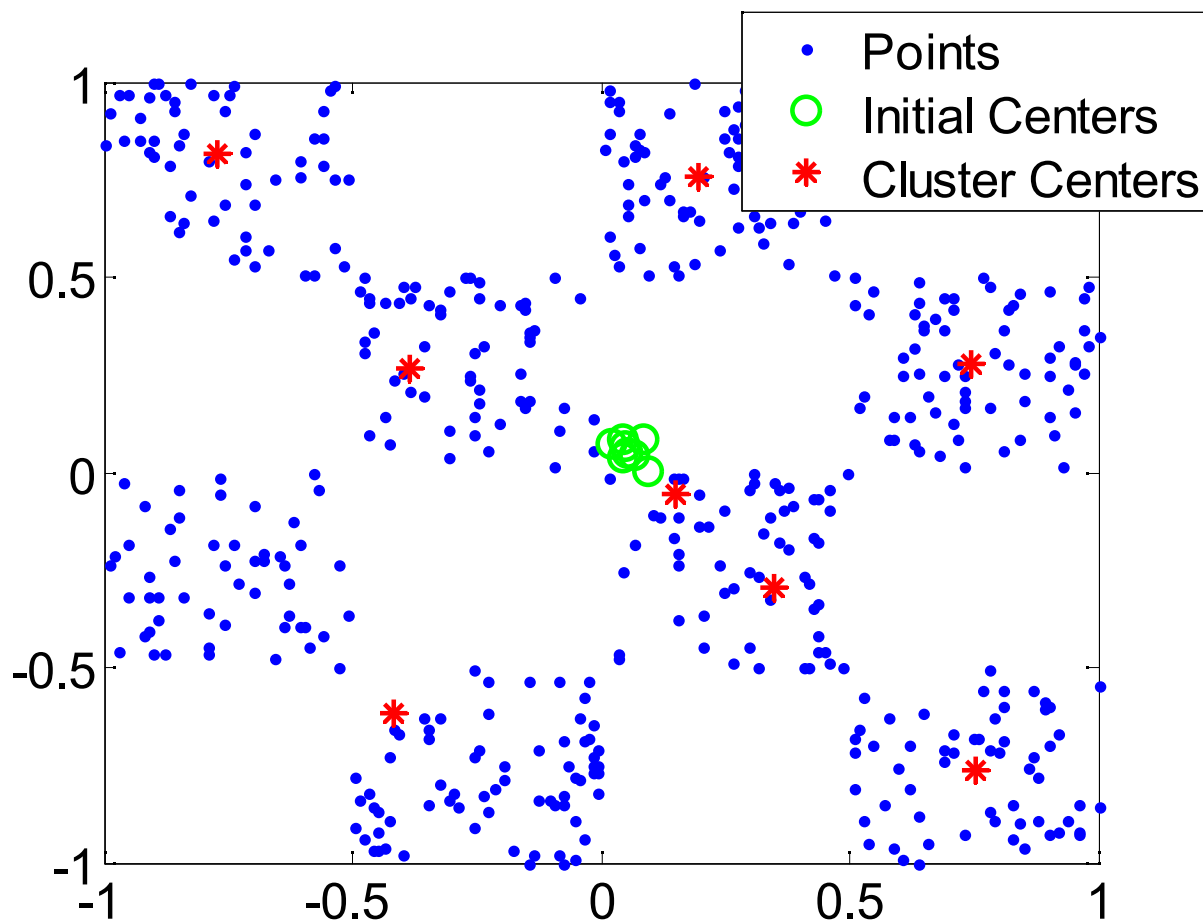
初始中心的选取对算法的影响

棋盘格数据集(Checkerboard data set)



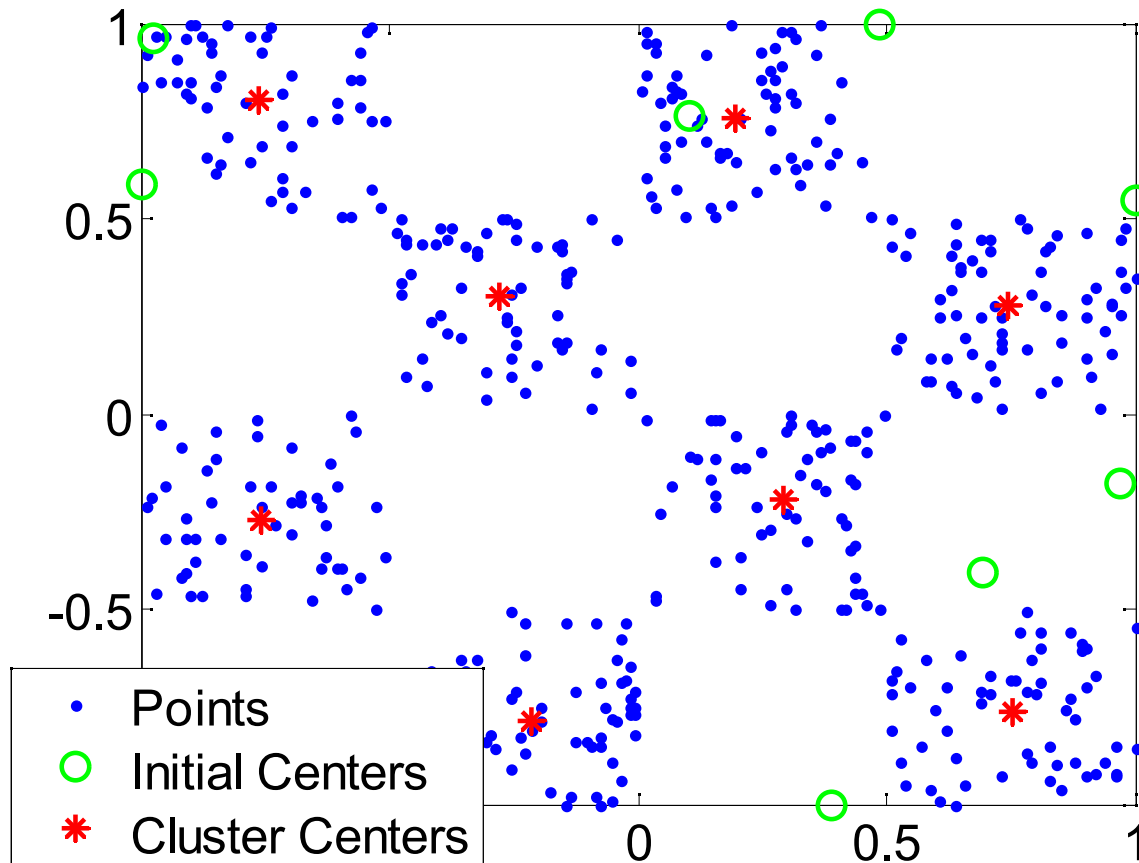
初始中心的选取对算法的影响

初始聚类中心均在中心附近



初始中心的选取对算法的影响

初始聚类中心在平面内随机选取





《机器学习》，周志华著，清华大学出版社

《统计学习方法》，李航著，清华大学出版

社