

# 第一章. 搜索算法

	完备性	最优解	时间复杂度	空间复杂度	Frontier
BFS	✓	若均搜索代价	$O(b^d)$	$O(b^d)$	Queue
DFS	✗ 若无限深就找不到	✗ 多解就奇	$O(b^m)$	$O(bm)$ 空间复杂度小 占用内存少	Stack
ID3	✓	若均搜索代价	$O(b^d)$	$O(bd)$	Stack
UCS	✓	✓			Priority Queue
Greedy	✗	✗	最坏 $O(b^m)$ 最好 $O(bd)$	最坏 $O(b^m)$ 最好 $O(bd)$	按 gen 排序 → 走过的距离 Priority Queue 按 hcn 排序 → 离目标估计 Priority Queue 按 hcn + gen 排序
A*	✓	✓			

## 搜索参数

- b: 一个节点最多分支
  - m: 整个图最大深度
  - d: 解所在的深度
- 启发式属性
- Admissible:  $dcn \geq hcn$
  - Consistent:  $d(n, p) \geq h(n) - h(p)$
  - $h(n) = 0$  将同时满足以上 2 个属性

## CSP 约束满足问题

前提假设: single agent + fully observable + deterministic + discrete environment

### 局部搜索:

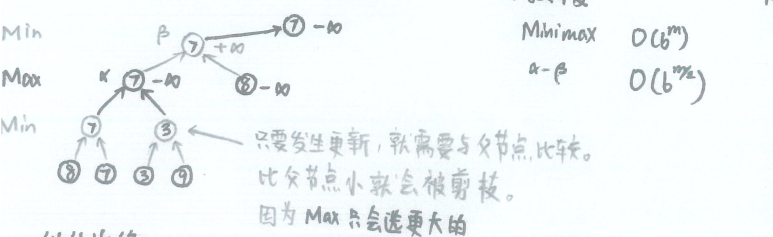
从完全放置开始, 此时不需要满足条件  
尝试重新分配变量来满足 constraint  
每次分配一个变量, 使得冲突最少

### 启发式 Backtracking

- 选择下一个 variable  
Least remaining variable (LRV)  
Most constraint variable (MCV)
- 选择下一个 value  
Least constraint assignment (LCA) 选择产生限制最少的值
- 早期失败检测  
Forward checking 检查所有变量至少还有一个可选值  
Arc consistency

# 第二章. Game

## Minimax & $\alpha$ - $\beta$ 剪枝 (DFS 算法)



## 纳什均衡

囚徒困境 (鼓励不合作)

	A 招	A 不招
B 招	-5, -5	-10, 0
B 不招	0, 10	-1, -1

猎鹿游戏 (鼓励合作)

	A 鹿	A 兔
B 鹿	2, 2	1, 0
B 兔	0, 1	1, 1

示弱游戏

	A 直	A 转
B 直	10, 10	-1, 1
B 转	1, 1	0, 0

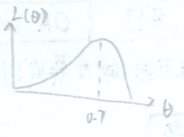
定理: 任何博弈, 只要选择有限, 则必然有一个纳什均衡  
If a player has dominant strategy. 纳什为 this player use dominant the other one use best respond  
If both player has dominant strategy. 纳什为两人都用自己的 dominant strategy

# 第三章. 贝叶斯

基本公式  $P(A|B) = \frac{P(A \cap B)}{P(B)} \Leftrightarrow P(A \cap B) = P(A|B)P(B) = P(B|A) \cdot P(A)$   
 $P(A \cap B) = P(A) \cdot P(B)$  相互独立  
 $P(B) = P(B|A_1)P(A_1) + P(B|A_2) \cdot P(A_2) \dots P(B|A_n) \cdot P(A_n)$   
 $P(A|B) = \frac{P(B|A) \cdot P(A)}{P(B|A) \cdot P(A) + P(B|\neg A) \cdot P(\neg A)}$

## 最大类似估计 (MLE)

$\theta = \arg \max P(x_0 | \theta)$   
 $L(\theta) = \theta^7 (1 - \theta)^3$   
 抛硬币, 7 次上, 3 次下  
 事件 概率



## 最大后验概率 (MAP)

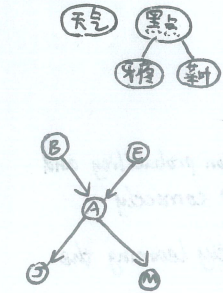
后验概率  $P(H|X) = \frac{P(X|H)P(H)}{P(X)}$   
 $\theta = \arg \max P(\theta | x_0)$   
 $\theta = \arg \max P(x_0 | \theta) \cdot P(\theta)$   
 $P(X|H) = \frac{25}{50} = \frac{1}{2}$   
 $P(H) = \frac{30}{100} = \frac{3}{10}$   
 $P(X) = \frac{35}{100} = \frac{7}{20}$   
 $P(H|X) = \frac{5}{7}$   
 选 100, 正常 70, 垃圾 30  
 垃圾中出现 10  
 垃圾中出现 25  
 垃圾属于垃圾的概率为 5/7

## 朴素贝叶斯

$P(X = (x_1, x_2) | Y = 1) = P(X = 0 | Y = 1) \cdot P(X = 2 | Y = 1) = \frac{6}{16}$   
 $P(X = (x_1, x_2) | Y = 0) = P(X = 0 | Y = 0) \cdot P(X = 2 | Y = 0) = \frac{1}{16}$   
 谁大  $K = (0, 2)$  就是哪一类 Y

# 第四章

## 贝叶斯网络



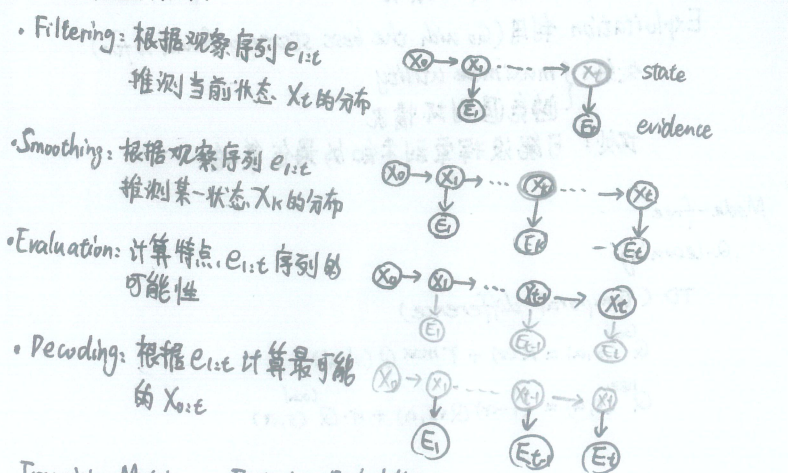
## 基本概念

$P(Z, X) = P(Z)P(X)$ 是独立	$P(Z, X) = P(Z Y)P(X Y)P(Y)$ 不是独立
$P(Z, X Y) = \frac{P(Y X, Z)P(X)P(Z)}{P(Y)}$ $\neq P(Z Y)P(X Y)$ 不是条件独立	$P(Z, X Y) = P(Z Y)P(X Y)$ 是条件独立

## 链式法则:

$P(A_1, A_2, \dots, A_n) = \prod_{i=1}^n P(A_i | \text{Parent}(A_i))$   
 $= \prod_{i=1}^n P(A_i | A_1, \dots, A_{i-1})$   
 $= P(A_1) \cdot P(A_2 | A_1) \cdot P(A_3 | A_1, A_2) \dots P(A_n | A_1, \dots, A_{n-1})$   
 $P(J, M, A, B, E) = P(B) \cdot P(E) \cdot P(A|B, E) \cdot P(J|A) \cdot P(M|A)$   
 $P(B, J) = \sum_E \sum_A \sum_M P(B, E, A, J, M)$

## 隐马尔可夫模型 HMM



## Transition Matrix

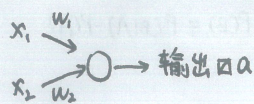
状态转移矩阵  
从一 state 到另一个 state 的概率

## Emission Probability

映射矩阵  
从一 state 到观测现象的概率



## 感知机 Perceptron

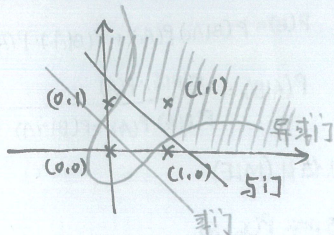


输出 =  $\begin{cases} 0, & w \cdot x + b \leq 0 \\ 1, & w \cdot x + b > 0 \end{cases}$

可为以下门电路

与门	AND Gate
与非门	NAND Gate
或门	OR Gate

但无法做为异或门 XOR



## 第六章

### 深度学习

如何训练神经网络

1. 从 training dataset 中随机选择一个 training token  $(x_i, y_i)$
2. 计算 neural net prediction  $f(x_i)$
3. 计算 loss 例如  $L = -\log f_{y_i}(x_i)$
4. Back-propagate 寻找梯度  $\frac{\partial L}{\partial W^{(l)}}$  和  $\frac{\partial L}{\partial W^{(u)}}$
5. 梯度更新  $W^{(l)} \leftarrow W^{(l)} - \eta \frac{\partial L}{\partial W^{(l)}}$
6. 重复过程直至 loss 足够小

## 第五章

### 马尔可夫决策过程 MDP

组成:

- state  $s$
- Action  $a$  每个 state  $s$  有可选的行动  $A(s)$
- Transition model  $P(s'|s, a)$
- Reward function  $R(s)$

解 solution:

- Policy  $\pi(s)$

### 贝尔曼公式

$$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s, a) U(s')$$

$\gamma \in (0, 1)$

$$U(s_0, s_1, s_2, \dots) = R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots = \sum_{t=0}^{\infty} \gamma^t R(s_t) \leq \frac{R_{\max}}{1-\gamma}$$

### 强化学习

Model-based: Learn the model of MDP (transition probability and reward) and try to solve the MDP correctly

Model-free: Learn how to act without explicitly learning the transition probability  $P(s'|s, a)$

Q-learning: learn an action-utility function  $Q(s, a)$  that tells us the value of doing action  $a$  in state  $s$

Model-based:

Exploration 探索 (采取一个后果未知的 action)

好处:  $\begin{cases} \text{获取更准确的环境模型} \\ \text{可能发现最多的 reward} \end{cases}$

坏处:  $\begin{cases} \text{探索不是 maximize utility} \\ \text{可能遇到坏情况} \end{cases}$

Exploitation 利用 (Go with the best strategy found so far)

好处:  $\begin{cases} \text{maximize utility} \\ \text{避免遇到坏情况} \end{cases}$

坏处: 可能没探索到未知的最佳策略

Model-free

Q-learning

TD (Temporal difference)

$$Q^{(local)}(s, a) = R(s) + \gamma \max_{a'} Q(s', a')$$

$$Q^{(new)}(s, a) = (1-\alpha) Q^{(local)}(s, a) + \alpha \cdot Q^{(local)}(s, a)$$