提醒

- 今天提交作业2

今天的内容

- 命题逻辑 剩余部分
- 逻辑型的智能体
- 概率

简要回顾上次内容

知识(knowledge)

知识库（knowledge base）= 在形式语言中定义的一个句子的集合

声明式（declarative）法构建智能体(或其他系统):

- 告诉 智能体它需要知道的(或让它自己*学习*这些知识)
- 然后它可以询问 自己在一个环境里如何行动———回答应遵循知识库里的知识

在***知识层(knowledge level)***描述智能体

- 具体说明智能体 *知道 什么*, 它的目标是什么，和实现细节无关

一个推理算法可以回答任何可以回答的问题

- 比较而言，一个搜索算法只能回答 “如何从A 到 B” 的问题

知识库	领域特定事实 (Domain-specific facts)
推理引擎	领域独立的通用算法和代码

逻辑（Logic）

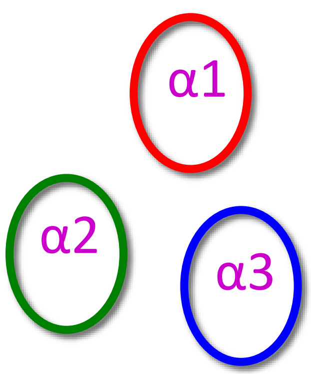
语法(Syntax): 定义句子(什么样的句子是允许的)

语义（Semantics）：

- *可能的世界 (possible worlds)**有哪些*?
- 这些句子在哪些世界里为 *真*？(句子真实性的*定义*)

语义空间

语法空间



推理的内容: 蕴涵/导出(entailment)

蕴涵: $\alpha \models \beta$ (“α 导出(entails) β” or “β 遵循于(follows from) α”)当且仅当在 α 为真的每个世界里, β 也是真

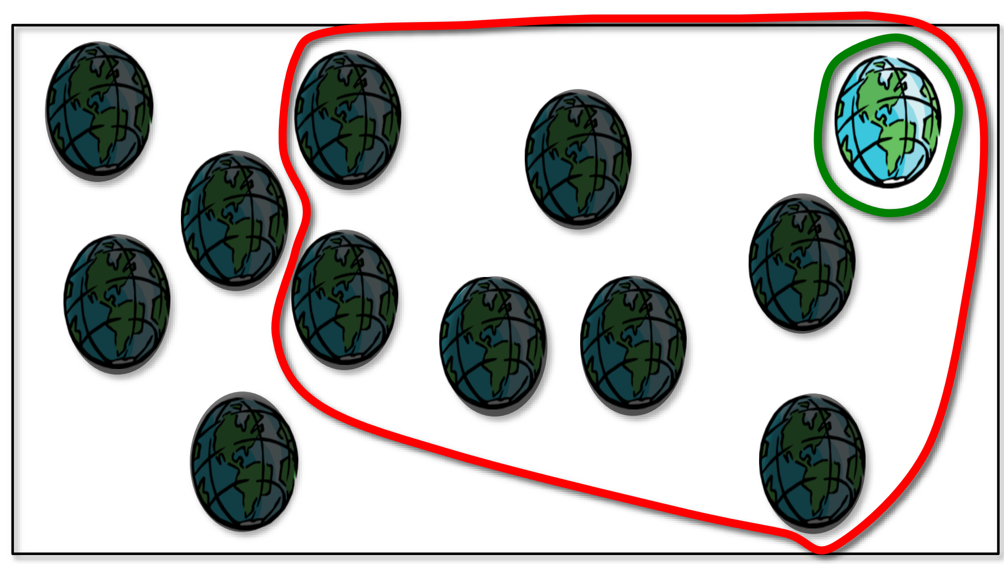
- 换句话说, α-的世界（为真的那些世界） 是 β-的世界的一个子集 [***models***(α) $\subseteq$ ***models***(β)]

例如, $\alpha_2 \models \alpha_1$

(比如 α_2 是 $\neg Q \wedge R \wedge S \wedge W$
α_1 是 $\neg Q$)

α_1

α_2



推理的过程: 证明

证明--指 α 和 β 之间的导出（蕴涵）关系的***演示证明***

方法 1: ***模型检查 (model-checking)***

- 对于每一个可能的世界里, 如果 α 为真， 那么确认 β 也真
- 有限多的世界里可行（比如命题逻辑）；但不容易对于一阶谓词逻辑

方法 2: ***定理证明 (theorem-proving)***

- 搜寻一系列的证明步骤 (应用 推理规则(***inference rules)***) 从 α 引导到 β
- 例如, 从 $P \wedge (P \Rightarrow Q)$, 推理出 Q 通过 肯定前件式推理(***Modus Ponens)***

合理性 (Sound) 算法: 所有被推理证明出来的，实际上也都是被蕴涵的

完全性 (Complete) 算法: 所有被蕴涵的（句子）， 都可以被推理证明出来

命题逻辑（Propositional logic）：语法

给定: 一组 命题字符 {X_1,X_2,…, X_n ,P, Q, R, North,…}（可为真或假）
- (True 和 False 也包含其中，真值固定)

X_i 是一个句子（原子语句）

复杂句
- 如果 α 是句子，那么 $\neg\alpha$ 是一个句子（否定）
- 如果α 和 β是句子，那么$\alpha \wedge \beta$是一个句子 （结合/“与”）
- 如果α 和 β是句子，那么$\alpha \vee \beta$是一个句子 （分离/“或”）
- 如果α 和 β是句子，那么$\alpha \Rightarrow \beta$是一个句子 （隐含,或条件的if …then）
- 如果α 和 β是句子，那么$\alpha \Leftrightarrow \beta$是一个句子 （双向条件的 if and only if ）
- 逻辑连接符，和（）的组合

文字(literal): 原子语句和否定的原子语句

没有其他样式的句子!

命题逻辑: 语义

给定一个模型（model），决定一个句子的真值

真值表

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

逻辑上的一致性

$$
\begin{array}{rcll}
(\alpha \land \beta) & \equiv & (\beta \land \alpha) & \text{commutativity of } \land \\
(\alpha \lor \beta) & \equiv & (\beta \lor \alpha) & \text{commutativity of } \lor \\
((\alpha \land \beta) \land \gamma) & \equiv & (\alpha \land (\beta \land \gamma)) & \text{associativity of } \land \\
((\alpha \lor \beta) \lor \gamma) & \equiv & (\alpha \lor (\beta \lor \gamma)) & \text{associativity of } \lor \\
\neg(\neg\alpha) & \equiv & \alpha & \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) & \equiv & (\neg\beta \Rightarrow \neg\alpha) & \text{contraposition} \\
(\alpha \Rightarrow \beta) & \equiv & (\neg\alpha \lor \beta) & \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) & \equiv & ((\alpha \Rightarrow \beta) \land (\beta \Rightarrow \alpha)) & \text{biconditional elimination} \\
\neg(\alpha \land \beta) & \equiv & (\neg\alpha \lor \neg\beta) & \text{De Morgan} \\
\neg(\alpha \lor \beta) & \equiv & (\neg\alpha \land \neg\beta) & \text{De Morgan} \\
(\alpha \land (\beta \lor \gamma)) & \equiv & ((\alpha \land \beta) \lor (\alpha \land \gamma)) & \text{distributivity of } \land \text{ over } \lor \\
(\alpha \lor (\beta \land \gamma)) & \equiv & ((\alpha \lor \beta) \land (\alpha \lor \gamma)) & \text{distributivity of } \lor \text{ over } \land
\end{array}
$$

简单的定理证明过程: 前向推理 (Forward chaining)

应用肯定前件式推理(Modus Ponens) 产生新的事实:

- 给定 $X_1 \wedge X_2 \wedge \ldots X_n \Rightarrow Y$ 和 $X_1, X_2, \ldots, X_n$
- 推理出 Y

前向推理持续应用这个规则, 不断添加新的事实, 直到没有可添加的为止

要求 KB（知识库） 只包含 ***确定子句(definite clauses)***:

- 一组分离(disjunction)的文字(literals),只有一个是正的（其他都含否定符）; 可转成以下形式
- (字符的结合(conjunction)) $\Rightarrow$ 字符; 或
- 一个单一的字符 (注意 X 相当于 $True \Rightarrow X$)

前向链接算法(Forward chaining)

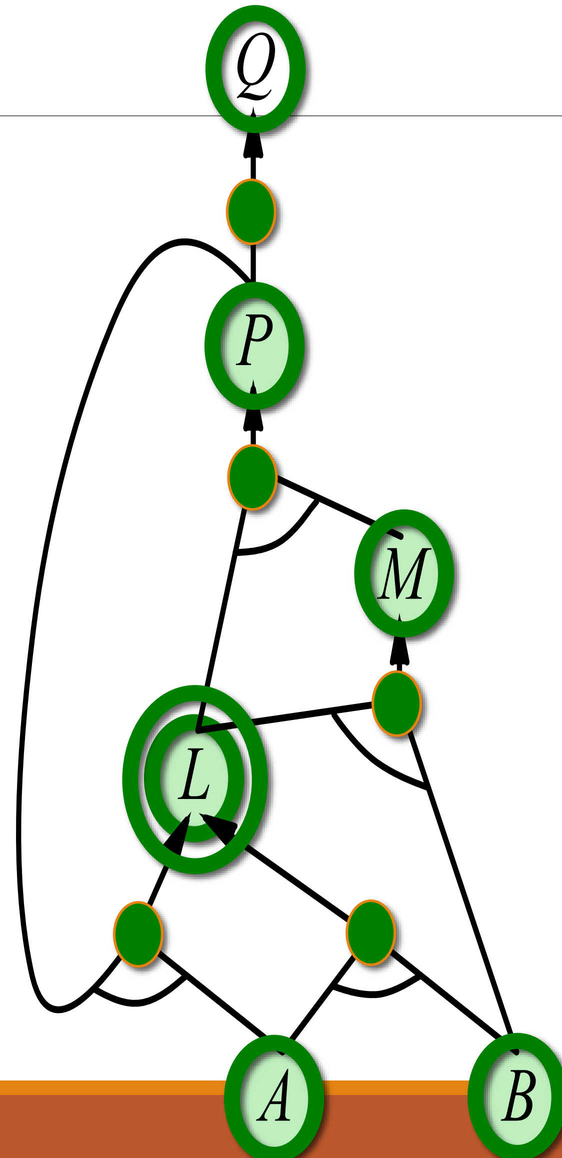
```
function PL-FC-ENTAILS?(KB, q) returns true or false
  count ← 一个表，count[c] 是子句 c 的前提中的还未知的字符数量
  inferred ← 一个表，inferred[s] 初始化为 false 对于所有字符 s
  agenda ← 一个字符队列，初始化为 KB 里 (为真的)所有字符
  while agenda is not empty do
      p ← Pop(agenda)
      if p = q then return true
      if inferred[p] = false then
          inferred[p]←true
          for each 子句 c in KB where p 在 c的前提里 do
              减一 count[c]
              if count[c] = 0 then add c的结论 to agenda
  return false
```

前向链接推理举例: 证明 Q（被蕴涵）

子句	COUNT	INFERRED
$P \Rightarrow Q$	~~1~~ 0	A ~~false~~ true
$L \wedge M \Rightarrow P$	~~2~~ ~~1~~ 0	B ~~false~~ true
$B \wedge L \Rightarrow M$	~~2~~ ~~1~~ 0	L ~~false~~ true
$A \wedge P \Rightarrow L$	~~2~~ ~~1~~ 0	M ~~false~~ true
$A \wedge B \Rightarrow L$	~~2~~ ~~1~~ 0	P ~~false~~ true
A	0	Q ~~false~~ true
B	0	

AGENDA

~~A~~ ~~B~~ ~~L~~ ~~M~~ ~~P~~ ~~L~~ ~~Q~~



前向链接(FC)推理的性质

定理: FC 是合理的(sound) 和 完全的(complete)，对于确定子句(definite-clause)组成的KBs

合理性: 遵循于肯定前件式（ Modus Ponens）的合理性

完全性证明:

1. 假设FC 达到了一个固点，即 没有新的原子语句被推导出
2. 最终的 *inferred* 表可以被考虑成一个模型 ***m***, 即字符被赋给了 true/false 值
3. 在原始的 KB 中的每一个子句在 ***m*** 里为真

   证明: 假定一个子句 $a_1 \wedge \ldots \wedge a_k \Rightarrow b$ 在 ***m*** 中为假

   那么 $a_1 \wedge \ldots \wedge a_k$ 必为真在 ***m*** 并且 b 为假 在 ***m***

   如此说明算法还未达到一个固点!（与假设矛盾）
4. 因此 ***m*** 是 KB 的一个模型 （KB 在 m 里为真）
5. 如果 KB |= q, q 则在KB中的每一个模型里为真, 包括 ***m***； 即q可以被算法推导出来

A xxxx false true

B xxxx false true

L xxxx false true

M xxxx false true

P xxxx false true

Q xxxx false true

简单的模型检查(model checking)

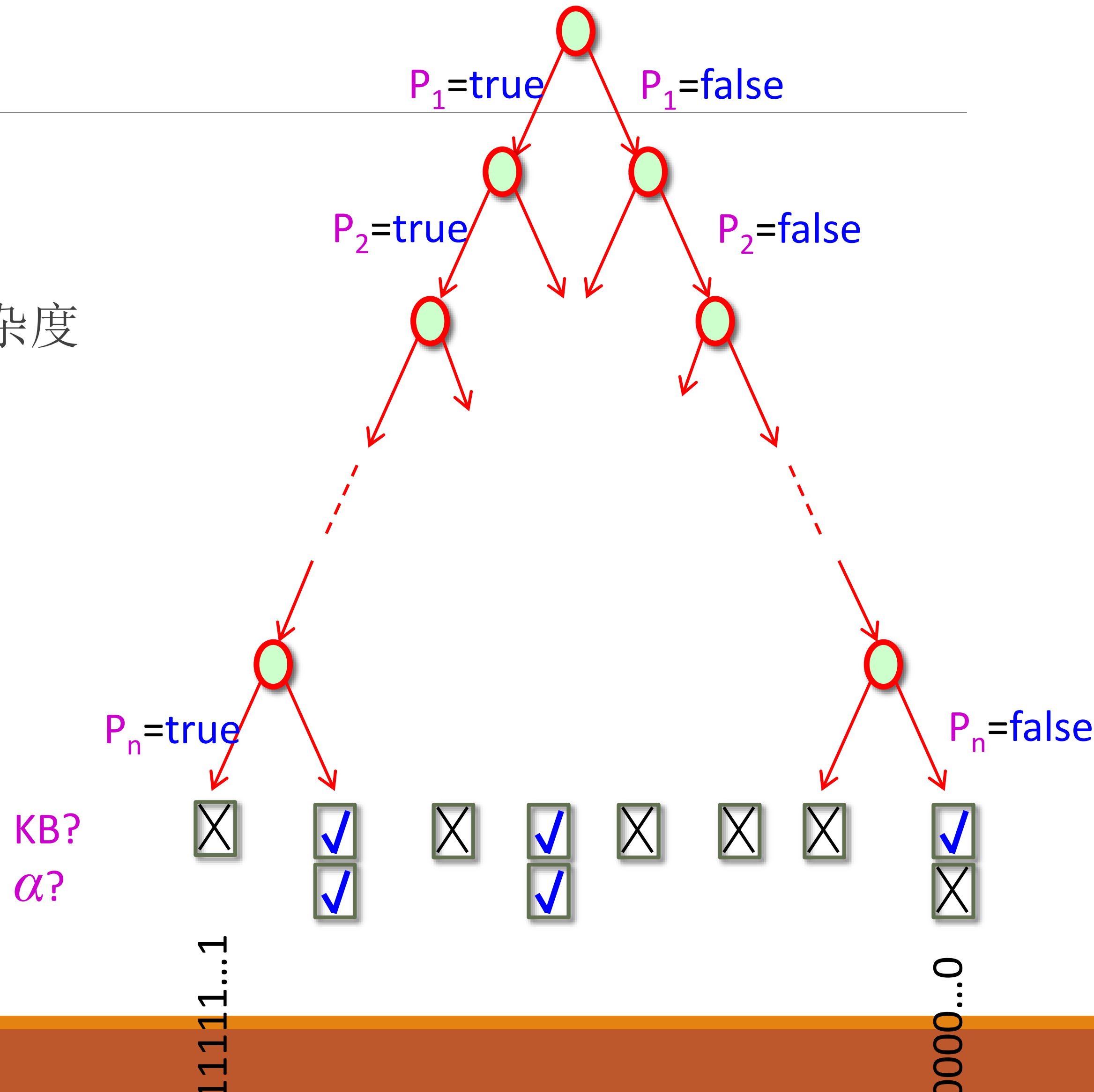
```
function TT-ENTAILS?(KB, α) returns true or false
  return TT-CHECK-ALL(KB,α,symbols(KB) ∪ symbols(α),{})
function TT-CHECK-ALL(KB,α,symbols,model) returns true or false
  if empty?(symbols) then
      if PL-TRUE?(KB,model) then return PL-TRUE?(α,model)
      else return true
  else
      P ← first(symbols)
      rest ← rest(symbols)
      return and (TT-CHECK-ALL(KB,α,rest,model ∪ {P = true})
                 TT-CHECK-ALL(KB,α,rest,model ∪ {P = false }))
```

简单的模型检查, 继续

深度优先，类似于回溯算法(backtracking)的递归

O(2^n) 时间复杂度, 线性空间复杂度

可以有更高效的算法!



可满足性 和 导出(蕴涵)

一个语句是 ***可满足的*** ，如果它至少在一个世界里为真 (参见 CSPs!)

假设我们有一个超高效的 SAT solver; 我们如何能用它来测试蕴涵关系?

- 假定 $\alpha \models \beta$
- 那么 $\alpha \Rightarrow \beta$ 为真 在所有世界（演绎公理）
- 因此 $\neg(\alpha \Rightarrow \beta)$ 为假 在所有世界
- 因此 $\alpha \wedge \neg\beta$为假 在所有世界, i.e., 不可满足的(unsatisfiable)

所以, 把否定的结论添加到 所知道的语句里, 测试其不可满足性 (un)satisfiability; 也叫 归谬法(reductio ad absurdum)

高效的 SAT solvers 需要 合取范式(***conjunctive normal form)***

合取范式(CNF)

替换双向条件，用两个暗示条件

替换 $\alpha \Rightarrow \beta$ 用 $\neg\alpha \vee \beta$

分配 v 到 ∧

每个句子可以被表达为 …

每个子句是一个 *文字* … *析取*

每个文字是一个正…字符或一个…定的…

通过一序列标准的变换 转成 CNF:

- $R \Rightarrow (D \Leftrightarrow U)$
- $R \Rightarrow ((D \Rightarrow U) \wedge (U \Rightarrow D))$
- $\neg R \vee ((\neg D \vee U) \wedge (\neg U \vee D))$
- $(\neg R \vee \neg D \vee U) \wedge$
  $(\neg R \vee \neg U \vee D)$

高效的SAT问题求解器(SAT solvers)

DPLL (Davis-Putnam-Logemann-Loveland) 是 现代SAT求解器的核心算法

本质上是一个对模型的回溯搜索，和一些额外的技术:

- ***提早终止***: 如果
  - 所有子句都被满足; e.g., $(A \vee B) \wedge (A \vee \neg C)$ 被满足，通过 {A=true}
  - 任何一个子句为假; e.g., $(A \vee B) \wedge (A \vee \neg C)$ 为假，当 {A=false,B=false}

- ***纯文字（字符）***：如果一个字符在剩下所有未满足的子句里的符号都是统一的，那么赋给这个字符那个值
  - 例如, A 是纯的，并且正号的 $(A \vee B) \wedge (A \vee \neg C) \wedge (C \vee \neg B)$ 所以赋给 true

- ***单元子句***: 如果一个子句只剩下一个单一的文字，那么给这个字符赋值使之满足该子句
  - 例如, 如果 A=false, $(A \vee B) \wedge (A \vee \neg C)$ becomes $(\text{false} \vee B) \wedge (\text{false} \vee \neg C)$, i.e. $(B) \wedge (\neg C)$
  - 满足单元子句的过程中经常会导致进一步的传递，产生新的单元子句。

DPLL 算法

```
function DPLL(子句集,字符集,模型) returns true or false
  if every 子句 in 子句集 is true in 模型 then return true
  if 某个 子句 in 子句集 is false in 模型 then return false
  P,value ←FIND-PURE-SYMBOL(字符集,子句集,模型)
  if P is non-null then return DPLL(子句集, 字符集–P, 模型∪{P=value})
  P,value ←FIND-UNIT-CLAUSE(子句集,模型)
  if P is non-null then return DPLL(子句集, 字符集–P, 模型∪{P=value})
  P ← First(字符集); rest ← Rest(字符集)
  return or(DPLL(子句集,rest,模型∪{P=true}),
            DPLL(子句集,rest,模型∪{P=false}))
```

效率

DPLL的简单实现: 求解 ~100 变量

额外技巧:

- 变量和值的选取排序 (参见 CSPs)
- 分治法（divide and conquer）
- 记录下 无法求解的情况，作为额外的子句，用来避免重蹈覆辙
- 索引和 增量计算技巧，使得DPLL 算法的每一步都是高效的 (通常 O(1))
  - 索引子句中每个变量（字符）的符号（正或是否定的）
  - 变量赋值过程中持续记录已满足的子句数量
  - 持续记录每个子句中剩余的文字符号的数量

DPLL的真正实现: 可以求解 ~10,000,000 变量

SAT 求解器在现实中的应用

电路验证: 超大规模集成电路 是否给出正确的计算?

软件验证: 程序是否计算正确的结果?

软件综合: 哪些程序计算正确结果?

协议验证: 这个安全协议能否被攻破?

协议合成: 哪些协议对于这个任务是安全的?

规划: 智能体的行为规划?

总结

- 一种可能的智能体框架: 知识 + 推理
- 逻辑 提供了一种对知识进行编码的正规方法
  - 一个逻辑的定义: 语法, 可能世界的集合, 真值条件
- 逻辑推理计算句子间的导出（蕴涵）关系

人工智能导论：逻辑型的智能体

一个基于知识的智能体

function KB-AGENT(percept) **returns** 一个行动
内部记录：KB, 知识库
t, 整数, 初始为 0

TELL(KB, MAKE-PERCEPT-SENTENCE(percept, t))

action ← ASK(KB, MAKE-ACTION-QUERY(t))

TELL(KB, MAKE-ACTION-SENTENCE(action, t))

t←t+1

return action

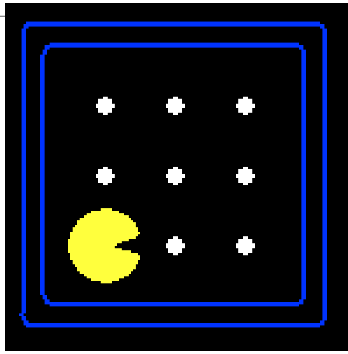
举例：部分可观察的 Pacman

Pacman 行动只能根据局部的感知信息

- 四个布尔感知变量 代表在每个方向是否有墙

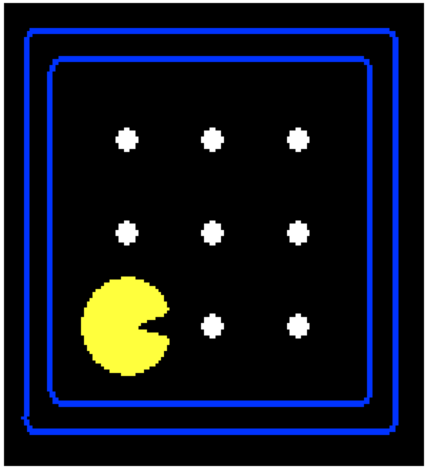
它需要什么知识能开始行动?

- *传感模型*: 句子 说明 当前感知变量是如何被当前的状态变量所决定的
- *转换模型*: 句子 说明 下一个状态变量是如何被当前状态变量和Pacman的行动所决定的
- *初始条件*: Pacman对于初始状态的知识
- *领域约束*: 普通的事实, 例如，Pacman智能在一个时间做一件事，并且在一个时间只能出现在一个地方



所涉及到的Pacman的变量

- Pacman的位置
  - At_1,1_0 (Pacman 在[1,1] 在时刻 0)   At_3,3_1 等
- 墙的位置
  - Wall_0,0   Wall_0,1 …
- 感知到的
  - Blocked_W_0 (向西被墙阻挡， 在时刻 0) 等.
- 行动
  - W_0 (Pacman 向西移动， 在时刻 0), E_0 等.



NxN 个世界， T 个时刻 => $N^2T + N^2 + 4T + 4T = O(N^2T)$ 变量数

2^{N^2T} 可能的世界!， N=10, T=100 => 10^{3010}

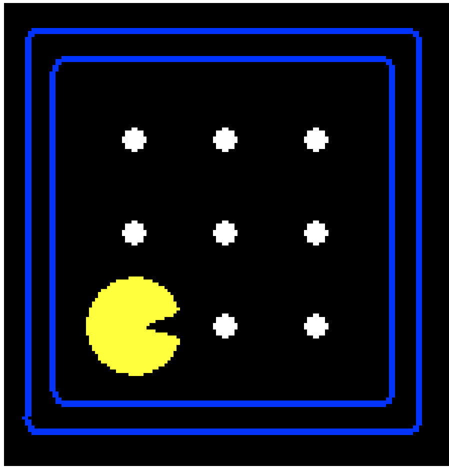
传感模型

描述如何产生 Pacman的感知

Pacman 感觉到向西有一面墙在时刻 t， *当且仅当* 他在位置 x,y 并且 有一面墙在位置 x-1,y

- Blocked_W_0 ⇔
- ((At_1,1_0 ∧ Wall_0,1) v
  (At_1,2_0 ∧ Wall_0,2) v
  (At_1,3_0 ∧ Wall_0,3) v ....)

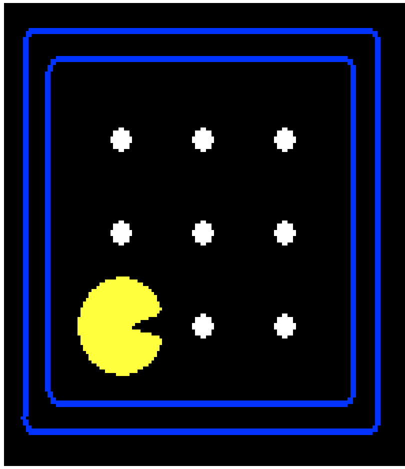
- 这样的句子有多少？



另一种传感器模型的问题

如果在时刻 t 他在位置 x,y 并且 有一堵墙在位置 x-1,y, 则 Pacman 在时刻 t 感知到一堵墙在他的西面

- At_1,1_0 ∧ Wall_0,1 ⇒ Blocked_W_0
- At_1,1_1 ∧ Wall_0,1 ⇒ Blocked_W_1
- ....
- At_3,3_9 ∧ Wall_3,4 ⇒ Blocked_N_9



- 这种表达的问题
  - 不完整
  - 只是说 在这些条件下感知变量为真
  - 但没说 感知变量何时为假

如果左边为假，右边是真还是假？

转换模型 (transition model)

- 每个 ***状态变量*** 在每个时刻如何获得它的值?
- 部分可感知的Pacman里的状态变量 是 At_x,y_t , 例如, At_3,3_17
- 一个状态变量获得它的值，根据 ***后继状态公理 (successor-state axiom)***
  - $X_t \Leftrightarrow [X_{t-1} \wedge \neg$(某个 action_{t-1} 使之为 false)] v
    [$\neg X_{t-1} \wedge$ (某个 action_{t-1} 使之为 true)]
- 对于 Pacman 的位置:
  - At_3,3_17 ⇔ [At_3,3_16 ∧ ¬((¬Wall_3,4 ∧ N_16) v (¬Wall_4,3 ∧ E_16) v …)]
    v [¬At_3,3_16 ∧ ((At_3,2_16 ∧ ¬Wall_3,3 ∧ N_16) v
    (At_2,3_16 ∧ ¬Wall_3,3 ∧ N_16) v …)]

初始状态

- 智能体可能知道它的初始位置:
  - At_1,1_0 ∧ ¬At_1,2_0 ∧ ¬At_1,3_0 ....
- 或者, 它可能不知道:
  - At_1,1_0 v At_1,2_0 v At_1,3_0 v … v At_3,3_0
- 我们也需要一个 ***值域约束*** – 一个时间只能做一件事!
  - ¬(W_0 ∧ E_0) ∧ ¬(W_0 ∧ S_0) ∧ …
  - ¬(W_1 ∧ E_1) ∧ ¬(W_1 ∧ S_1) ∧ …
  - … ∧ (W_0 v E_0 v N_0 v S_0) ∧ …

状态估计

回忆智能体在部分可观察情况下的 ***信念状态(belief state)*** 定义:

- 给定行动和当前的感知，与之相符合的世界状态的集合
- *状态估计* 是指保持(预测/更新)当前的信念状态

对于一个逻辑型的智能体, 计算在当前状态下哪些变量为真，只不过是一个逻辑推理的问题

- 例如, 询问是否 KB ∧ <actions> ∧ <percepts> |= Wall_2,2
- 简单但效率低: 每一步的推理涉及到一个智能体整个行动感知的历史

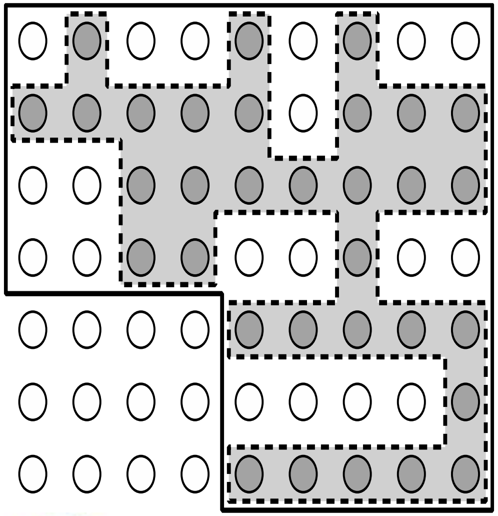
状态估计，继续

一个更“积极主动的”状态估计形式:

- 在每个行动和感知以后
  - 对每个状态变量 X_t
    - 如果 X_t 是被蕴涵的, 则加到 KB
    - 如果 $\neg X_t$ 是被蕴涵的, 则加到 KB

对于准确的状态评估是否这就足够?

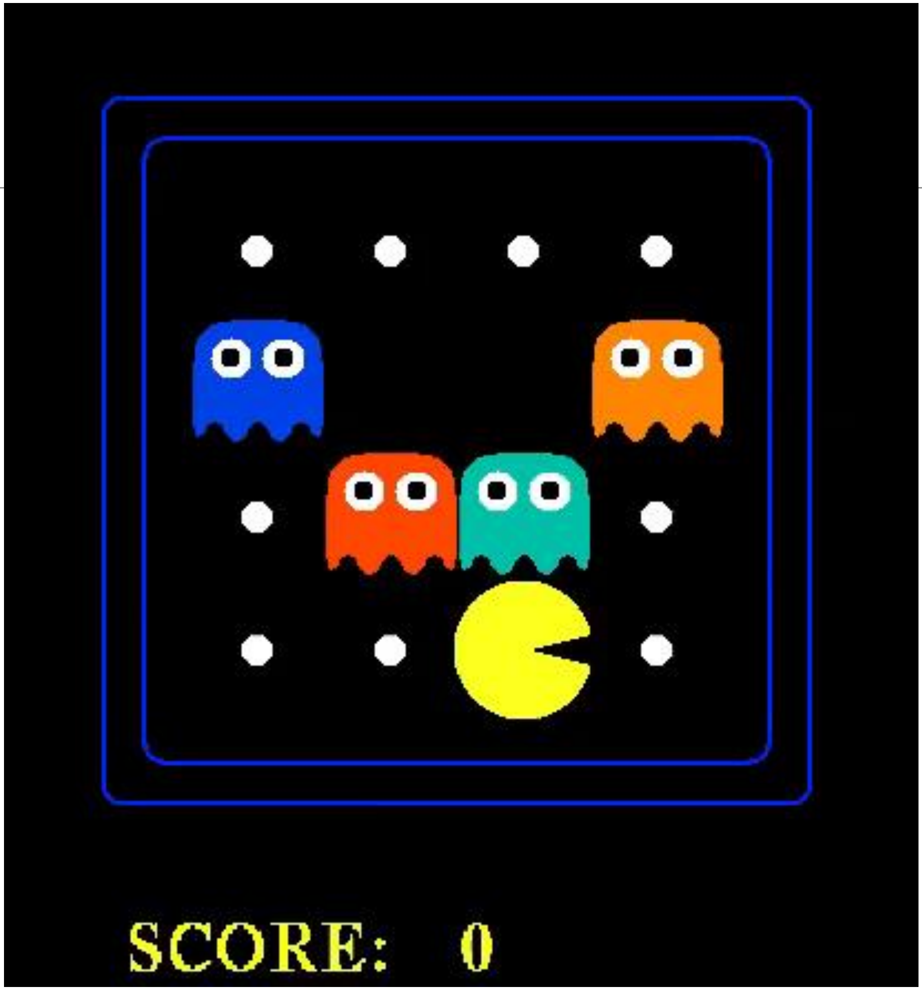
- 不是! 可能的情况是 X_t 或 $\neg X_t$ 都不被蕴涵, 并且 Y_t 或 $\neg Y_t$ 也都不被蕴涵, 但是某个约束, 例如, $X_t \vee Y_t$, *是* 被蕴涵的
  - 例如: 初始不确定性的智能体的位置

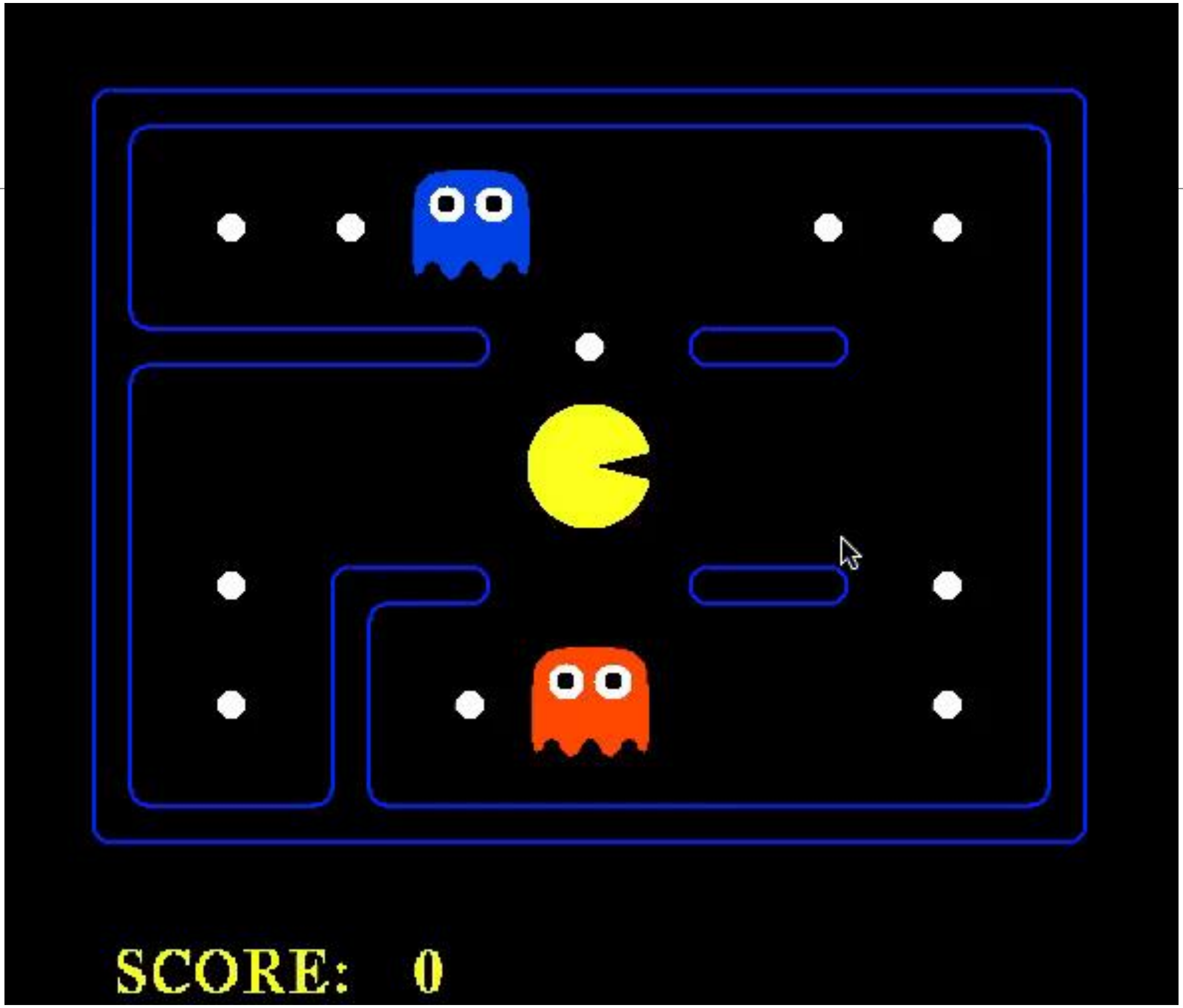


普遍来讲，完美的状态估计是很难达到的

可满足(satisfiability)来解规划(Planning)问题

- 给定一个超高效的 SAT 求解器，我们能用它来规划智能体的行动吗?
- 是的, 对于完全可观察的, 决定性的环境:
  - 规划问题是可解的 当且仅当 存在某个可满足的赋值对于所有变量
  - 相应的行动变量的真值构成了解
- 对于时间 T = 1 到无穷, 按以下内容构建知识库 (KB)， 并运行SAT solver:
  - 初始状态, 值域约束
  - 截至到时刻 T 的转换模型语句(包括后继状态转换公理，对于所有可能的行动)
  - 目标(Goal) 在时刻 T 为真


SCORE: 0


SCORE: 0


SCORE: 0

总结

- 声明法/陈述法(declarative approach) 构建智能体的框架
  - 知识库里是陈述语句（包括公理，感知到的事实）
  - 逻辑推理– 事实被蕴涵的推理证明，规划智能体行动
- 现代超高效的SAT 求解器，使得此法可在实践中可行
- 弱点：语句表达
  - 例如“对于每个时刻 t”，“对于每个方块位置[x,y]”
  - 一阶逻辑(first order logic) 提高了语句的表达性；其逻辑推理方法与命题逻辑的方法一致

人工智能导论：概率

人工智能的大体分类

第一部分：搜索和规划 (Search and planning)

第二部分：概率推理 (Probabilistic reasoning)

- 医疗诊断
- 语音识别
- 追踪物体
- 机器人制图
- 基因学
- 通讯纠错代码
- … 还有很多!

第三部分: 机器学习 (Machine learning)

不确定性(Uncertainty)

例如，搭乘的航班计划登机时间是 10:30 am

- 让行动 A_t = 距离登机时间提前 t 分钟从家里出发
- A_t 行动将能保证赶上飞机吗?

可能出现问题:

- 部分可观察性 (交通状况, 公交或出租车等待时间, 等)
- 可能有误差的传感器 (交通广播的报告, 百度地图，等)
- 对交通流的建模和预测非常复杂
- 缺乏对环境/世界动态性的知识 (车轮胎损坏? 道路临时施工堵塞?安检中可能出现的情况等)

对不确定性的应对

忽视它？ 不行；为什么？

对逻辑规则的修改

- $A_{1440} \rightarrow_{0.9999}$ *赶上飞机*
- *赶上飞机* $\rightarrow_{0.95} \neg$ *交通堵塞*
- 因此，$A_{1440} \rightarrow_{0.949} \neg$*交通堵塞*
  - 逻辑关系不准确；难以穷尽各种因素

概率(Probability)

- 根据现有的情况和所采取的行动 A_{120}, 那么赶上飞机的概率是 0.92

概率(Probability)

概率是对 复杂，不确定情况某种程度上的总结代表

- 懒惰性***(laziness)***: 太多的意外情况难以罗列, 等
- 无知性***(ignorance)***: 对某些情况缺乏了解和知识，等

主观的***(Subjective)*** or 贝叶斯(***Bayesian)*** 概率:

- 根据自己的知识状态来决定相关命题的概率
- 例如, P(*赶上飞机* | A_{120}, 晴天) = 0.92

命题有关概率随新知识观察的变化:

- 例如, P(*赶上飞机* | A_{120} , 晴天, *桥上不堵车*) = 0.96

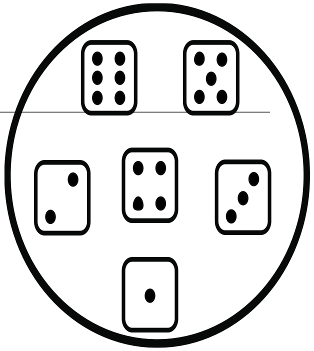
决策(Decisions)

- 假设我们有：
  - *P(赶上飞机* | A_{60} , *所有我所获得的信息…)* = 0.51
  - *P(赶上飞机* | A_{120} ,*所有我所获得的信息…)* = 0.97
  - *P(赶上飞机* | A_{1440} ,*所有我所获得的信息…)* = 0.9999
- 选择哪一个行动?
- 还取决于偏好(***preferences)*** , 例如, 不能错过飞机, 机场等待时间,机场的食品 等.
- ***（功用/利益 原理）Utility theory***， 用来对偏好进行表示和推理
- ***（决策原理）Decision theory*** = 功用原理 + 概率原理
- ***（最大化功用期值）Maximize expected utility*** :
  - $a^* = \mathrm{argmax}_a \sum_s P(s \mid a)\, U(s)$

概率的基本法则

开始于一组可能世界的集合 Ω

- 例如, 一个骰子的6个可能结果, {1, 2, 3, 4, 5, 6}

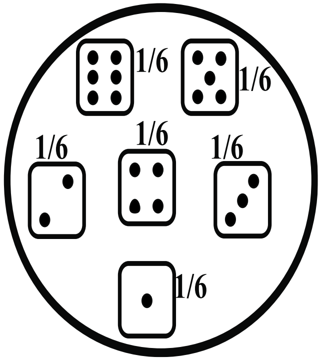


概率模型(probability model) 赋予一个数 $P(\omega)$ 给每一个世界 ω

- $P(1) = P(2) = P(3) = P(5) = P(5) = P(6) = 1/6$.

这些数必须满足

- $0 \leq P(\omega) \leq 1$
- $\sum_{\omega \in \Omega} P(\omega) = 1$



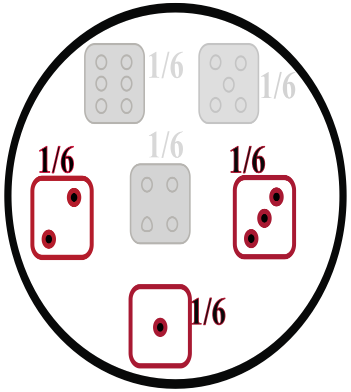
基本法则（继续）

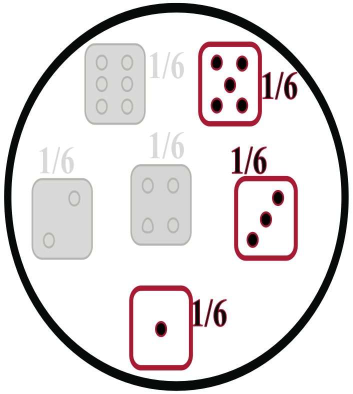
一个***事件(event)*** 是Ω的一个子集

- “投数 < 4” 是集合 {1,2,3}
- “投数是奇数”， {1,3,5}

一个事件的概率是在对应世界概率数值之和

- $P(A) = \sum_{\omega \in A} P(\omega)$
- $P(\text{投数} < 4) = P(1) + P(2) + P(3) = 1/2$

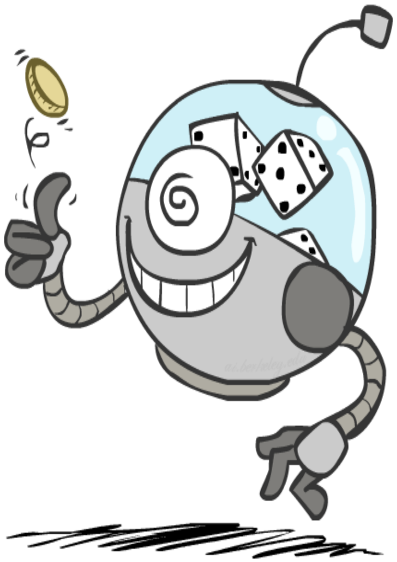




随机变量(Random Variables)

一个随机变量描述了世界中我们可能不确定的某个方面
（正式的讲，是ω上的一个决定性的函数）

- *R* = 天是否将会下雨?
- *Odd* = 骰子的投数是否将会是一个奇数?
- *T* = 天气是热还是冷?
- *D* = 花费多长时间能够到达机场?



大写字母开头

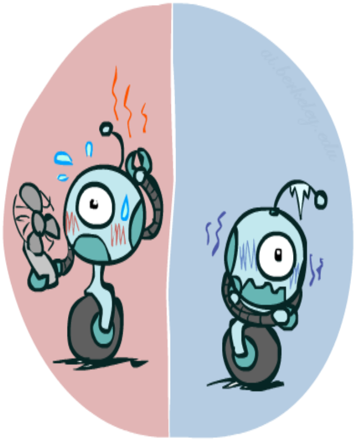
随机变量也有值域

- *Odd* in {true, false} e.g. *Odd*(1)=true, *Odd*(6) = false
  - 通常把事件 Odd=true 写成 odd, Odd=false 写成 $\neg$odd
- *T* in {hot, cold}
- *D* in [0, ∞)

概率分布(Probability Distributions)

每个概率由一个值来代表，并且加和为 1

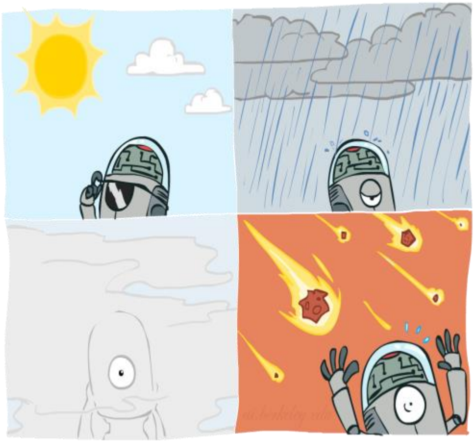
- 温度:



$P(T)$

T	P
hot	0.5
cold	0.5

- 天气:



$P(W)$

W	P
sun	0.6
rain	0.1
fog	0.3
meteor	0.0

概率分布

- 每个概率模型自动为每个随机变量固定了一个分布

$P(T)$

T	P
hot	0.5
cold	0.5

$P(W)$

W	P
sun	0.6
rain	0.1
fog	0.3
meteor	0.0

简略标识:

$$P(hot) = P(T = hot),$$

$$P(cold) = P(T = cold),$$

$$P(rain) = P(W = rain),$$

$$\ldots$$

只要值域里的每个值都唯一即可

- 一个分布是概率值的一个表
- 一个概率值 是一个单一的数

$$P(W = rain) = 0.1$$

特别的布尔记号表示:

$P(happy)$ = $P(Happy=true)$

$P(\neg happy)$ = $P(Happy=false)$

联合分布(Joint Distributions)

- 一组随机变量的*联合分布*： $X_1, X_2, \ldots X_n$

  为每一组赋值（或结果）指定了一个真实的数值:

  $$P(X_1=x_1, X_2=x_2, \ldots, X_n=x_n)$$

  $$P(x_1, x_2, \ldots, x_n)$$

- 必须遵守:

  $$P(x_1, x_2, \ldots, x_n) \geq 0$$

  $$\sum_{x_1, x_2, \ldots, x_n} P(x_1, x_2, \ldots, x_n) = 1$$

$P(T, W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

联合分布中可能的世界

- 通常情况下
  - 从随机变量和它们的值域开始
  - 构建可能的世界，即对变量的所有的赋值组合
- 例如, 两个骰子 $Roll_1$ and $Roll_2$
  - 可能的世界有多少? 6x6 = 36
  - 它们的概率是多少? 1/36 each (为什么??)
- n 个变量，每个变量的值域大小是 d， 分布的大小是多少? d^n
- 除了最小的分布以外，通常情况下的分布很难全部手写罗列出来!

事件的概率

回忆：一个事件的概率是该事件所有世界的概率值之和

所以，给定一个所有变量的联合分布，就可以计算任何事件的概率!

- 概率 hot AND sunny?
- 概率hot?
- 概率hot OR sunny?

通常我们关心的都是***部分赋值***（***partial assignments***）的事件, 比如 P(*T*=hot)

$$P(T, W)$$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

练习: 事件概率

P(X=true, Y=true) ?

$P(X, Y)$

X	Y	P
true	true	0.2
true	false	0.3
false	true	0.4
false	false	0.1

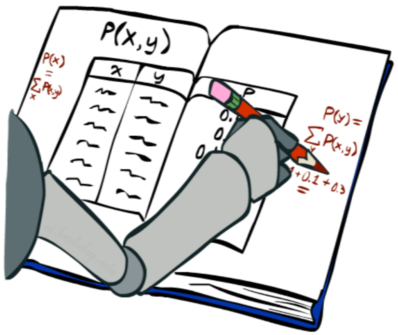
P(X=true) ?

P(X $\Rightarrow$ Y) ?

边缘分布(Marginal Distributions)

边缘分布式消除掉某些变量后的子表

边缘化 (加和): 通过求和来合并行



$P(T,W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

$P(T)$

T	P
hot	0.5
cold	0.5

$P(W)$

W	P
sun	0.6
rain	0.4

$$P(X=x_1) = \sum_{y'} P(X=x_1, Y=y')$$

练习: 边缘分布

$P(X,Y)$

X	Y	P
true	true	0.2
true	false	0.3
false	true	0.4
false	false	0.1

$P(x) = \sum_{y'} P(x, y')$

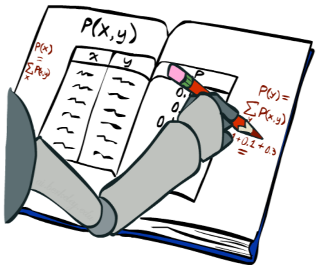
$P(y) = \sum_{x'} P(x', y)$

$P(X)$

X	P
true	
false	

$P(Y)$

Y	P
true	
false	

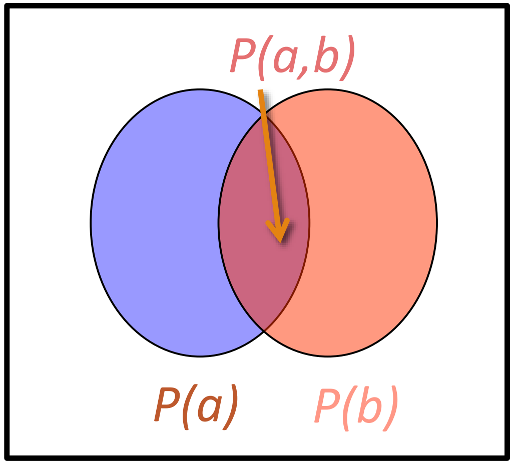


条件概率(Conditional Probabilities)

联合概率和条件概率间的简单关系

- 实际上，这也是条件概率的定义：

$$P(a \mid b) = \frac{P(a, b)}{P(b)}$$



$P(T,W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

$$P(W=s \mid T=c) = \frac{P(W=s,T=c)}{P(T=c)} = 0.2/0.5 = 0.4$$

$= P(W=s,T=c) + P(W=r,T=c)$
$= 0.2 + 0.3 = 0.5$

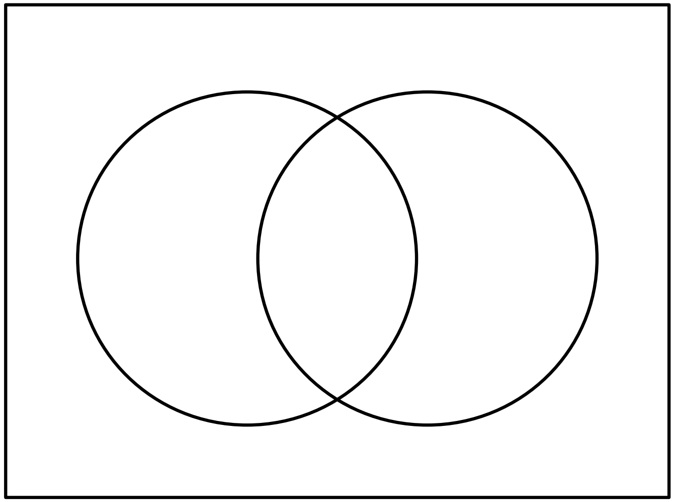
条件概率

从单一事件过渡到离散的分布：

$P(a \mid b)$

$P(A \mid b)$

$P(A \mid B)$



$P(A \mid b)$

A	B	P(A |B)
a	b	0.3
¬a	b	0.7

$P(A \mid B)$

A	B	P(A |B)
a	b	0.3
¬a	b	0.7
a	¬b	0.4
¬a	¬b	0.6

练习: 条件概率

$P(X,Y)$

X	Y	P
true	true	0.2
true	false	0.3
false	true	0.4
false	false	0.1

P(X=true | Y=true) ?

P(X=false | Y=true) ?

P(Y=false | X=true) ?

条件分布（ Conditional Distributions ）

某些变量的概率分布，当其他变量的值固定的时候，

条件分布

$P(W|T)$

$P(W|T = hot)$

W	P
sun	0.8
rain	0.2

$P(W|T = cold)$

W	P
sun	0.4
rain	0.6

联合分布

$P(T, W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

正规化/标准化(Normalization) 技巧

$P(T,W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

$$P(W=s \mid T=c) = \frac{P(W=s,T=c)}{P(T=c)} = \frac{P(W=s,T=c)}{P(W=s,T=c)+P(W=r,T=c)} = \frac{0.2}{0.2+0.3} = 0.4$$

$$P(W=r \mid T=c) = \frac{P(W=r,T=c)}{P(T=c)} = \frac{P(W=r,T=c)}{P(W=s,T=c)+P(W=r,T=c)} = \frac{0.3}{0.2+0.3} = 0.6$$

$P(W \mid T=c)$

W	P
sun	0.4
rain	0.6

正规化技巧

$$P(W=s|T=c)=\frac{P(W=s,T=c)}{P(T=c)}$$
$$=\frac{P(W=s,T=c)}{P(W=s,T=c)+P(W=r,T=c)}$$
$$=\frac{0.2}{0.2+0.3}=0.4$$

$P(T,W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

选择 那些符合证据(evidence)的联合概率

$P(c,W)$

T	W	P
cold	sun	0.2
cold	rain	0.3

正规化 这些选项(使它们的和为1)

$P(W|T=c)$

W	P
sun	0.4
rain	0.6

$$P(W=r|T=c)=\frac{P(W=r,T=c)}{P(T=c)}$$
$$=\frac{P(W=r,T=c)}{P(W=s,T=c)+P(W=r,T=c)}$$
$$=\frac{0.3}{0.2+0.3}=0.6$$

正规化技巧

$P(T,W)$

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

选择 那些符合证据(evidence)的联合概率

$P(c,W)$

T	W	P
cold	sun	0.2
cold	rain	0.3

正规化 这些选项(使它们的和为1)

$P(W|T=c)$

W	P
sun	0.4
rain	0.6

为什么是这样? 选项之和是 P(evidence)! (这里是， P(T=c))

$$P(x_1|x_2) = \frac{P(x_1,x_2)}{P(x_2)} = \frac{P(x_1,x_2)}{\sum_{x_1} P(x_1,x_2)}$$

与公式相符合

正规化的定义

- (字典解释) 使之回归到一个常态条件下

所有项之和为1

- 步骤:
  - 第一步: 计算 Z = 所有项之和
  - 第二步: 把每一项除以 Z

- 例 1

W	P
sun	0.2
rain	0.3

正规化 →

Z = 0.5

W	P
sun	0.4
rain	0.6

- 例 2

T	W	P
hot	sun	20
hot	rain	5
cold	sun	10
cold	rain	15

正规化 →

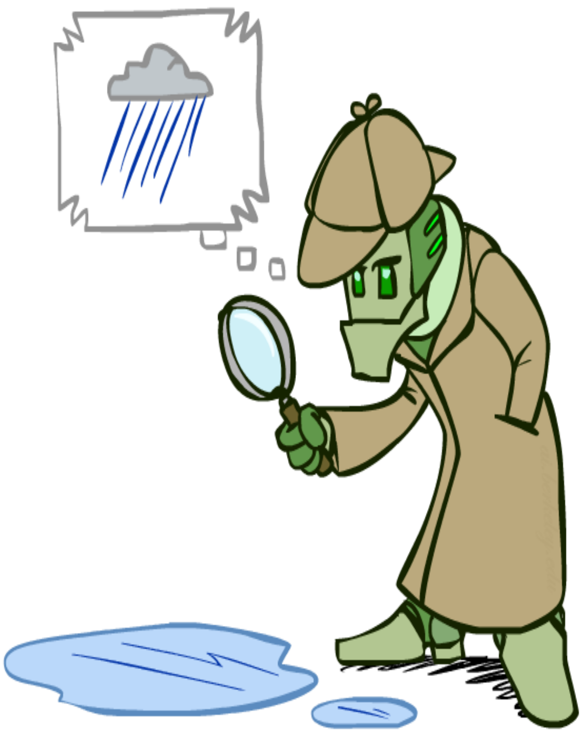
Z = 50

T	W	P
hot	sun	0.4
hot	rain	0.1
cold	sun	0.2
cold	rain	0.3

概率推理(Probabilistic Inference)

- 概率推理: 从其他已知概率里计算一个想知道的概率 (例如，从联合概率中计算条件概率)

- 通常我们计算的都是条件概率
  - P(准时到机场 | 没有交通事故发生) = 0.90
  - 这些代表了智能体在给定证据(evidence)下的*信念* *(beliefs)*



- 概率会随新的证据而变化:
  - P(准时到达 | 无交通事故, 早上5 点出发) = 0.95
  - 准时到达 | 无交通事故, 早上5 点出发, 下雨) = 0.80
  - 观察到新的证据时，会引发*信念(beliefs)* 的更新

通过列举(Enumeration)来推理

** 多个查询变量也可以*

通常情况:
- 证据变量: $E_1 \dots E_k = e_1 \dots e_k$
- 查询* 变量: Q
- 隐藏变量: $H_1 \dots H_r$

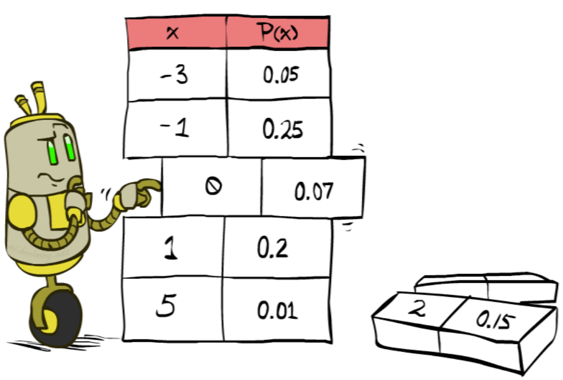
$$\left.\begin{matrix} E_1 \dots E_k = e_1 \dots e_k \\ Q \\ H_1 \dots H_r \end{matrix}\right\} X_1, X_2, \dots X_n$$

所有变量

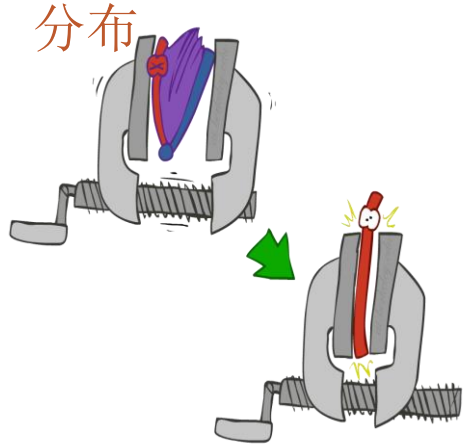
- 我们想要的:

$$P(Q|e_1 \dots e_k)$$

- 第一步: 选择和证据相一致的项



- 第二步: 求和消掉隐藏变量H，以得到查询和证据变量的联合分布



$$P(Q, e_1 \dots e_k) = \sum_{h_1 \dots h_r} \underbrace{P(Q, h_1 \dots h_r, e_1 \dots e_k)}_{X_1, X_2, \dots X_n}$$

- 第三步: 正规化

$$\times \frac{1}{Z}$$

$$Z = \sum_q P(Q, e_1 \cdots e_k)$$

$$P(Q|e_1 \cdots e_k) = \frac{1}{Z} P(Q, e_1 \cdots e_k)$$

通过列举来推理

P(W)?

P(W | winter)?

P(W | winter, hot)?

S	T	W	P
summer	hot	sun	0.30
summer	hot	rain	0.05
summer	cold	sun	0.10
summer	cold	rain	0.05
winter	hot	sun	0.10
winter	hot	rain	0.05
winter	cold	sun	0.15
winter	cold	rain	0.20

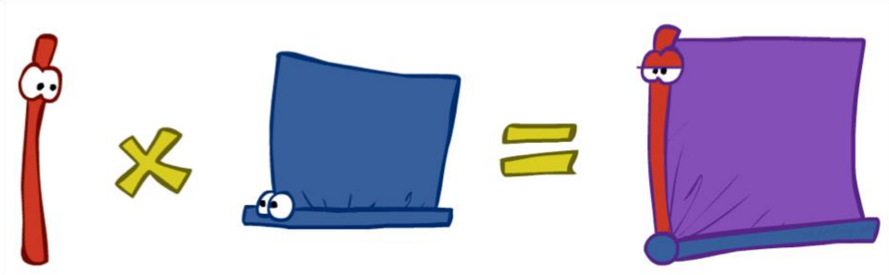
列举推理

- 明显的问题:
  - 最差情况下时间复杂度 $O(d^n)$
  - 空间复杂度 $O(d^n)$，需要存储联合分布

乘法规则(The Product Rule)

- 当已有条件分布，想要计算联合分布时

$$P(a \mid b)\, P(b) = P(a, b) \quad \Longleftrightarrow \quad P(a \mid b) = \frac{P(a, b)}{P(b)}$$



乘法规则

$$P(a \mid b)\, P(b) = P(a, b)$$

举例： $P(D \mid W)\, P(W) = P(D, W)$

$P(D \mid W)$

D	W	P
wet	sun	0.1
dry	sun	0.9
wet	rain	0.7
dry	rain	0.3

$P(W)$

W	P
sun	0.8
rain	0.2

$P(D, W)$

D	W	P
wet	sun	
dry	sun	
wet	rain	
dry	rain	

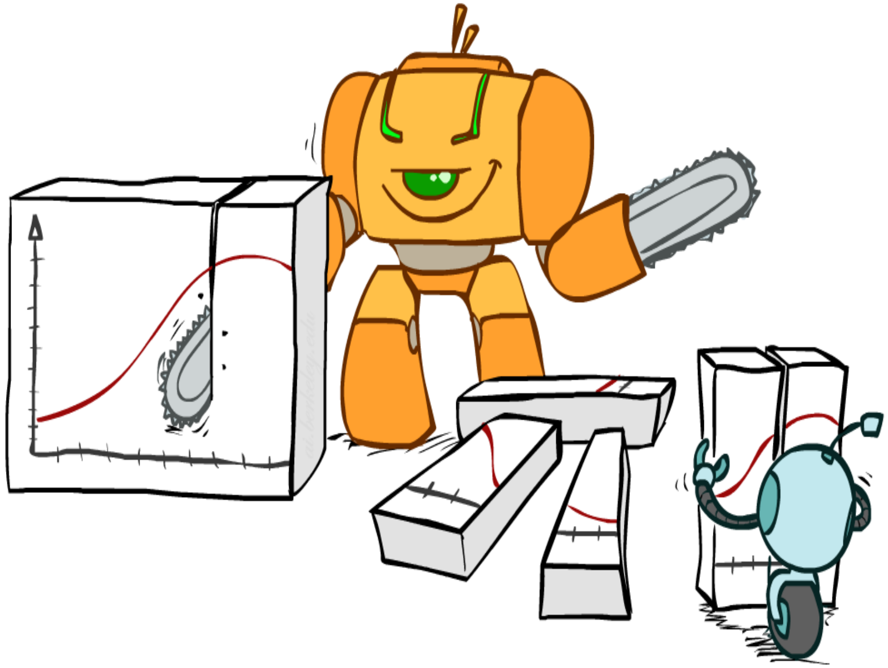
链式法则(The Chain Rule)

- 更普遍化的, 任何联合分布可以表达成条件分布的增量乘积的形式：

$$P(x_1, x_2, x_3) = P(x_3 \mid x_1, x_2)\, P(x_1, x_2) = P(x_3 \mid x_1, x_2)\, P(x_2 \mid x_1)\, P(x_1)$$

$$P(x_1, x_2, \ldots, x_n) = \prod_i P(x_i \mid x_1, \ldots, x_{i-1})$$

贝叶斯法则(Bayes Rule)



贝叶斯法则(Bayes’ Rule)

- 两种~~方法因式分解一由两个变量组成的联合分布:~~

$$P(x,y) = P(x|y)P(y) = P(y|x)P(x)$$

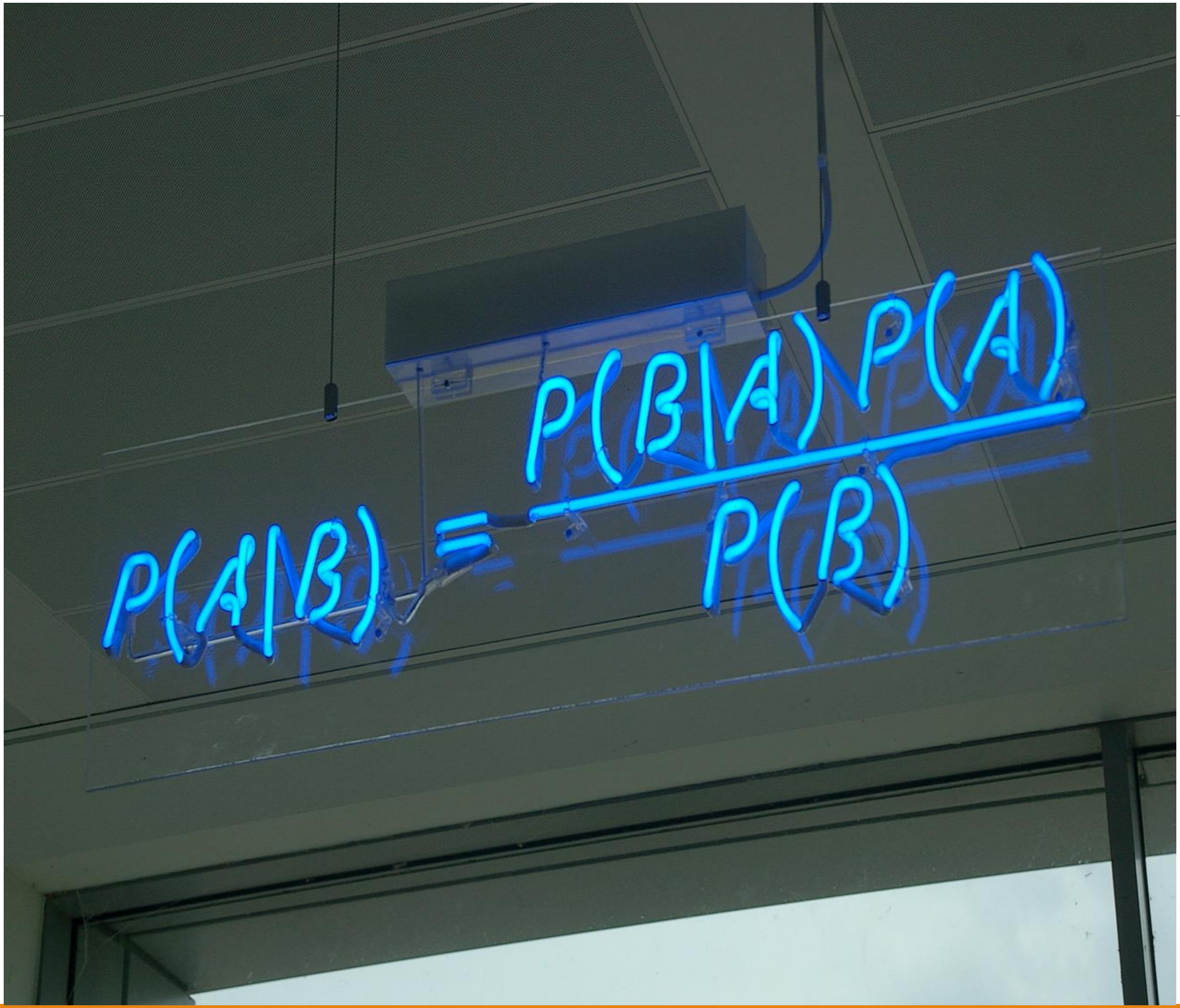
- 相除后, 我们得到:

$$P(x|y) = \frac{P(y|x)}{P(y)}P(x)$$

- 为什么这个有用?
  - 让我们计算一个条件概率，从它的相反的形式
  - 通常一个条件概率很难计算，但是相对应的另一个却很简单
  - 描述了一个“更新”步骤，从先验概率 $P(a)$ 到后验概率

    $P(a \mid b)$
  - 许多人工智能系统的基础
- 最重要的人工智能公式之一!

贝叶斯法则



用贝叶斯法则进行推断

举例: 从因果关系概率推断医疗诊断概率:

$$P(\textit{原因} \mid \textit{结果}) = \frac{P(\textit{结果} \mid \textit{原因})\, P(\textit{原因})}{P(\textit{结果})}$$

例如:

- M: meningitis, S: stiff neck

$$\left.\begin{aligned} P(m) &= 0.0001 \\ P(s \mid m) &= 0.8 \\ P(s) &= 0.01 \end{aligned}\right\} \text{已给定的}$$

$$P(m \mid s) = \frac{P(s \mid m)\, P(m)}{P(s)} = \frac{0.8 \times 0.0001}{0.01}$$

- 注意: meningitis 的后验概率还是非常小: 0.008 (但比先验概率大80倍 – 为什么?)
- 注意: 如果有了症状还是应该去检查! 为什么?

下一次的内容

- 独立性(Independence)
- 条件无关性(Conditional independence)
- 贝叶斯网络(Bayes nets)