Intelligent Agents

Deterministic	**Stochastic**	当前状态与当前行动完全决定下一个状态
Episodic	sequential	一个动作先做后做不影响后续动作
Static	dynamic	环境只能由自己改变

搜索算法

tree search 树搜索展开所有子节点

graph search 图搜索添加了一个explored集，不会展开已经展开过的节点

- 不一定比tree search快
- 最大搜索数为节点数n-1

一、Uninformed 盲目搜索

名称	描述	注意
Breadth-First Search 广度优先	依次展开每一层节点，总是找到最浅的解	新结点出现同时检查是否为解，找到解就终止； 该解不再加入前线frontier
Uniform-Cost Search 一致代价	总展开最低的path cost，即从起点到该结点总花费g(n)，总是找到最优解	新结点若为解，还不能返回，而要遍历全部可能路径比较哪一条费用最小； 解会加入frontier，因为直到被展开的一刻才能确认没有其余路径可以到达解
Depth-First Search 深度优先	展开当前结点最深处；只用于树搜索	不是解而且没有后继的结点会被删除，节约存储； 解不加入前线
Depth-limit Search 深度受限	确定一个深度l，展开当前结点最深直到l	
Iterative Deepening Search 迭代加深搜索	从$l=0$开始，增加深度直到有解	
Bidirectional	从起点Start与目标Goal同时搜	目标已知且只有一个，所有步骤

Search 双向搜索 名称	索，在中间相遇；更快，因为 描述 $b^{\frac{d}{2}}+b^{\frac{d}{2}}<b^d$	action皆可逆 注意

二、Informed Search 有信息搜索

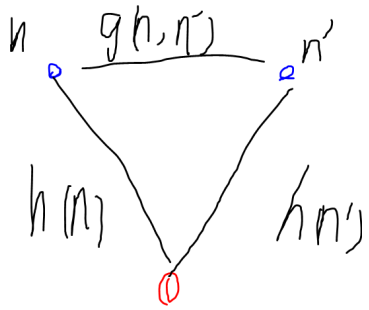
h(n)：结点n到目标结点花费的估计值，目标结点h(n)=0

g(n)：起点到结点n的实际代价

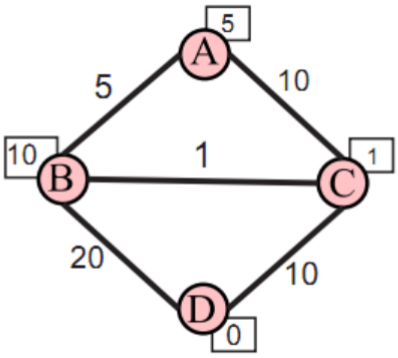
名称	描述	注意
Greedy best-first Search	每一步都试图扩展离目标最近的结点，$f(n)=h(n)$	目标结点加入前线
A* Search tree	$f(n)=g(n)+h(n)$	最优条件是admissible 目标会加入前线，在展开时返回
A* Search graph		最优条件是consistent; 到同一个目的地但更高的代价的路径不会加入前线; 目标会加入前线，在展开时返回

保证最优的条件

1. admissible：代价估计值$h(n)$一定要比实际值小，e.g.
   - 城市间直线距离＜城市间铁路距离
2. consistent：$h(n)\leq g(n,n')+h(n')$，即两个预估花费和两结点间实际花费可以构成三角形



三、例题



admissible 但不consistent，因为$h(B)=10\not\leq 1+1=g(B,C)+h(C)$

A* graph search

	1	2	3	4
node	A(0+5)	C(11)	B(15)	D(20)
frontier	C(10+1), B(5+10)	B(15), D(20+0)	D(20) $f(B \to D) = 25$因此不添加到前线	
explored	A	AC	ACB	ACB

没有找到最优解因为不consistent

A* tree search

	1	2	3	4	5	6
node	A(5)	C(11)	B(15)	C(7)	A(15)	D(16)
frontier	C(11), B(15)	B(15), D(20), B(21), A(25)	C(7), A(15), D(20), B(21), A(25), D(25)	A(15), D(20), B(21), A(25), D(25), A(21), B(17), D(16)	D(20), B(21), A(25), D(25), A(21), B(17), D(16), B(25), C(21)	D(20), B(21), A(25), D(25), A(21), B(17), B(25), C(21)

找到最优解了因为是admissible

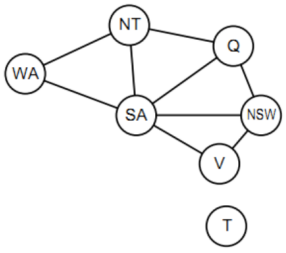
Constraint Satisfication 约束匹配

承上启下，既包含搜索算法，也包含推理算法

Constraint Satisfication Problem （CSP）：

$$(\{\text{变量集}Variables\}, \{\text{每个变量候选值域}Domain - \{x_1 values\}, \{x_2 values\}\}, \{\text{约束}C\})$$

约束图 Constraint graph：直观显示变量间约束关系



一、深度优先backtracking算法

1. 选一个未赋值变量

- minimum-remaining-values：优先选择domain size大的结点。

- degree heuristic：再优先选择关联最多（degree）的结点，尽可能排除候选项

2. 为其赋值

- least-constraining-value：优先给其他变量留尽可能多的取值，fail-last，因为只要解出一组匹配即可

> *当一个变量取值导致别的变量可选值（domain size）为0时，需要backtracking，尝试其他取值*

3. 引入推理——提早排除不可能选项

二、推理 Inference

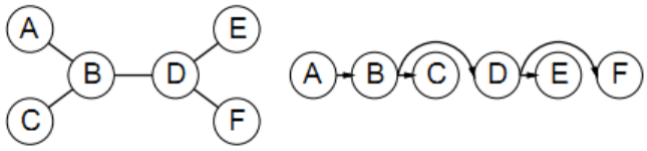
- Forward checking：每一步赋值后，只消去邻居不可能的取值
- Arc consistency：消去所有变量不可能的取值（每一步赋值后，或者赋值前先预处理数据）

> *arc consistent：两个变量任意取值，总能在对方值域中找到满足约束的对应值, e.g.*
> $Y = X^2, X = \{0, 1, 2, 3\}, Y = \{0, 1, 4, 9\}$

三、树状结构CSP

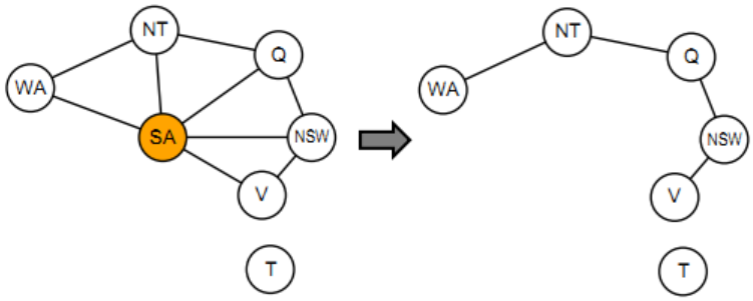
当约束图呈树状结构（没有闭环）时，可以算的比较快

构造树状结构：选取一个结点作为根，每个结点只能往前链接，永不回头



此时，若每一个节点与其后继arc-consistent，那么整个约束图自动也为arc-consistent

如果原graph有闭环，就选取$S \subset X$, 使得X去除S后为树状结构。为树状结构赋值时先去除S已经取走的值。



Propositional Logic 命题逻辑

一、定义

- Model： 使命题为真的实例
- m satisfies α : 命题α 在实例m中为真
- $M(\alpha)$ ： 全体使命题α为真的实例集合
- entail $\alpha \models \beta$ ： $M(\alpha) \subseteq M(\beta)$， 由$\alpha$可以推导出$\beta$
- $\alpha \equiv \beta$ ： $\alpha \models \beta$ 并且 $\beta \models \alpha$

$valid$	α在所有实例中均为真	$\alpha \equiv True, M(\alpha) =$ 全集
$satisfiable$	α仅在一些实例中为真	
$unsatisfiable$	$valid$反面，在所有实例中均为假	$\alpha \equiv False, M(\alpha) =$ 空集

$\neg$	非
$\wedge$	与
$\vee$	或
$\Longrightarrow$	蕴含
$\Longleftrightarrow$	当且仅当

二、推导

1. 形式变化：

蕴含化或	$\alpha \Longrightarrow \beta$	$\equiv$	$\neg\alpha \vee \beta$
德摩根法则	$\neg(\alpha \wedge \beta)$	$\equiv$	$(\neg\alpha \vee \neg\beta)$
与或分配律	$\alpha \vee (\beta \wedge \gamma)$	$\equiv$	$(\alpha \vee \beta) \wedge (\alpha \vee \gamma)$

2. 分解消元**(Resolution inference)**：

两个全或的表达式中矛盾项可以删去

$$\frac{\alpha \vee \beta,\ \neg\alpha \vee \gamma}{\beta \vee \gamma}$$

> *横线表示可以由上方两个式子推导下方的式子*

3. **Conjuctive Normal Form (CNF)**

由于Resolution inference只能用于全或表达式，在消元前要把表达式变为“外面全与，里面全或”。$e.g.\ (A \vee B) \wedge (C \vee D) \wedge E$

4. 反证法

当需要证明$KB \models \alpha$时，反证$KB \wedge \neg\alpha$为unsatisfiable，避开直接证明

如果$KB \wedge \neg\alpha$可以推导出一个矛盾（空集），则证明成功

三、例题

证明超人不存在

1. 将自然语言写作逻辑表达式

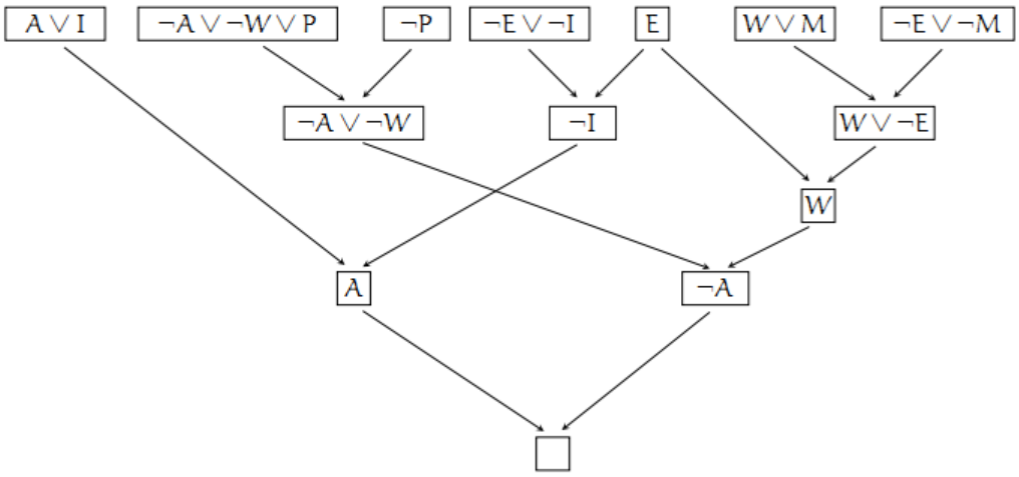
$$superman \begin{cases} Able\&Willing \rightarrow Prevent\ evil \\ unAble \rightarrow Impotent \\ unWilling \rightarrow Malevolent \\ not\ Prevent\ evil \\ Exists \rightarrow neither\ Impotent\ nor\ Malevolent \end{cases} \begin{cases} (A \wedge W) \Longrightarrow P \\ \neg A \Longrightarrow I \\ \neg W \Longrightarrow M \\ \neg P \\ E \Longrightarrow (\neg I \wedge \neg M) \end{cases}$$

2. 将每个逻辑表达式化作CNF形式

$$\begin{vmatrix} (A \wedge W) \Longrightarrow P \\ \equiv \neg(A \wedge W) \vee P \\ \equiv \neg A \vee \neg W \vee P \end{vmatrix} \begin{vmatrix} \neg A \Longrightarrow I \\ \equiv \neg\neg A \vee I \\ \equiv A \vee I \end{vmatrix} \begin{vmatrix} \neg W \Longrightarrow M \\ \equiv W \vee M \end{vmatrix} \begin{vmatrix} E \Longrightarrow (\neg I \wedge \neg M) \\ \equiv \neg E \vee (\neg I \wedge \neg M) \\ \equiv (\neg E \vee \neg I) \wedge (\neg E \vee \neg M) \end{vmatrix} \neg P$$

在知识库基础上与上待证式否命题，$\because \alpha = \neg E,\ \therefore \neg\alpha = E$

3. 消元



First-Order logic 一阶逻辑

表达力比命题逻辑强，但仍比自然语言弱；命题逻辑只有事实，而一阶逻辑还包含量词(∀, ∃)与关联

一、量词规则

1. 通常，∀和 $\Longrightarrow$ 连用

$$\forall x, At(x, TUM) \quad \Longrightarrow \quad Smart(x)$$

> *否则上句变为所有人都在TUM，并且所有人都聪明*

而∃ 和∧连用

$$\exists x, At(x, TUM) \quad \wedge Smart(x)$$

> *否则当有人不在TUM时，上句也为真*

2. 同一个量词后的变量可以互换位置或者组合，即$\forall x \forall y \equiv \forall y \forall x \equiv \forall (x, y)$,

但不同量词不能互换位置$\forall x \exists y \neq \exists y \forall x$

3. 德摩根法则：

$$\forall x, P \equiv \neg \exists x, \neg P$$
$$\exists x, P \equiv \neg \forall x, \neg P$$

二、逻辑推导

（一）、消除量词

1. 消除"任意"量词：

用有限个命题逻辑实例替换一个通用语句

$$\frac{\forall \text{变量} v，\ \text{命题} \alpha}{\text{替代}(\text{变量} v / \text{实例} g), \text{命题} \alpha}$$

比如：

$$\forall \mathbf{x}, \mathbf{King(x)} \wedge \mathbf{Greedy(x)} \implies \mathbf{Evil(x)}$$
$\theta = x/John$（用$John$实例化x）
$$King(John) \wedge Greedy(John) \implies Evil(John)$$
以此类推 $\cdots$

2. 消除“存在”量词：

用一个**Skolem Constant**常量替换，此过程称为**Skolemazation**，这样替换出来的语句将失去不确定性，特指某一个具体的对象

$$\frac{\exists v,\ \alpha}{\text{替代}(v/\text{随便一个符号}K), \alpha}$$

比如：

$$\exists \mathbf{x}, \mathbf{Crown(x)} \wedge \mathbf{OnHead(x, John)}$$
$$Crown(C_1) \wedge OnHead(C_1, John)$$

如果$\exists$ 符号外还有$\forall$符号，替换的语句会出错

比如：

$$\forall x\ [\exists y\ Animal(y) \wedge \neg Loves(x, y)]$$
$$\forall x\ [Animal(A) \wedge \neg Loves(x, A)] \qquad \text{用}A\text{替换}\exists y$$

> *由于外面还有$\forall x$， 语句变成了所有人都不爱某一只特定的动物A，意思错误*

正确的做法是将$\exists y$ 替换为关于x的函数，称为Skolem Function

$$\forall x\ [Animal(F(x)) \wedge \neg Loves(x, F(x))]$$

这样写出来的句子在消除量词$\exists$同时，仍能保持不确定性

（二）、推导方式

1. 前向链式推导

$$Horn\text{语句}：\frac{a_1, \cdots, a_n | a_1 \wedge \cdots \wedge a_n \implies \beta}{\beta}$$

> *与命题逻辑不同的是，一阶逻辑中允许变量存在，且变量的量词默认为$\forall$，因此不必消除"任意"量词*

一种简单的推导方式：从已知命题$a_1 \cdots a_n$ ，推导出新的命题加入知识库，直到推导出待证命题

2. 后向链式推导

从目标出发，不断寻找使目标成立的式子作为新目标，直到没有新目标则证明成功

$$\begin{aligned} goals： &\quad \{\text{目标命题集}\} \\ q' \leftarrow &\quad Subst(\{\text{所有替换集}\}, \text{这一步待证明子式}) \\ \theta' \leftarrow &\quad \text{下一步将替换的变量} \\ new \leftarrow &\quad \{\text{新待证命题集}\} \\ Used\ Clause \leftarrow &\quad \text{已知命题} \end{aligned}$$

3. 分解消元推导

与命题逻辑相同，把$\neg\alpha$加入知识库，使用反证法，得到矛盾即为证明成功

- 综合两个语句时如果遇到相同变量名，为避免混淆应该替换为别的字母

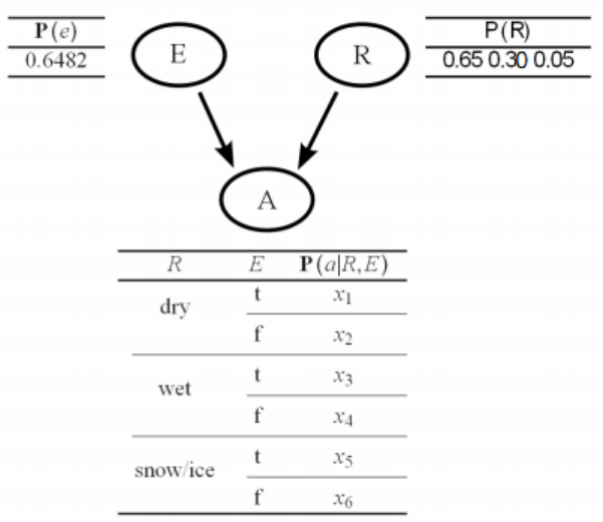
$$(\exists x\ P(x)) \vee (\exists x\ Q(x))$$
$$\rightarrow (\exists x\ P(x)) \vee (\exists y\ Q(y))$$

- 一个实例和一个通项消元时需要先把通项实例化

$$[Animal(F(x)) \vee Loves(G(x), x)] + [\neg Loves(u, v) \vee \neg Kills(u, v)]$$
替换$\theta = \{u/G(x), v/x\}$
$$Animal(F(x)) \vee \neg Kills(G(x), x)$$

贝叶斯网络

一种紧凑表示变量之间关系与概率的图



R	E	P(a|R,E)
dry	t	x_1
	f	x_2
wet	t	x_3
	f	x_4
snow/ice	t	x_5
	f	x_6

一、复习

> *例题：看到一辆蓝色的车，观察的可靠性为75%，已知有$\frac{1}{10}$的车确实为蓝色（其余为绿色），问看到蓝色车而且车的确为蓝色的概率?*

设置变量名“b”为车的确为蓝色；“ob”为观察到车为蓝色，由题设可得：
$p(b) = 0.1, p(ob|b) = 0.75, p(ob|\neg b) = 0.25$

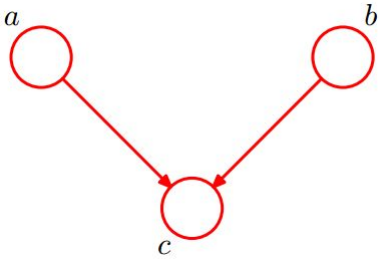
求$p(b|ob) = \frac{p(ob|b)\cdot p(b)}{p(ob)} = \alpha \cdot p(ob|b) \cdot p(b)$

1. 直接写出分母：$\frac{p(ob|b)\cdot p(b)}{p(ob)} = \frac{0.75\times 0.1}{0.75\times 0.1+0.25\times 0.9} = 0.25$
2. 使用α：$p(b|ob) = \alpha \cdot p(ob|b) \cdot p(b) = \alpha \times 0.75 \times 0.1$
   $p(\neg b|ob) = \alpha \cdot p(ob|\neg b) \cdot p(\neg b) = \alpha \times 0.25 \times 0.9$
   $\because p(b|ob) + p(\neg b|ob) = 1 \rightarrow \therefore \alpha(0.75 \times 0.1 + 0.25 \times 0.9) = 1 \rightarrow \alpha = \frac{10}{3}$
   $p(b|ob) = \alpha \cdot p(ob|b) \cdot p(b) = \alpha \times 0.75 \times 0.1 = \frac{10}{3} \times 0.75 \times 0.1 = 0.25$

二、判断独立&条件独立

D-Separation 方法——用于贝叶斯图，快速判断两个结点是否条件独立

1. 汇连collide

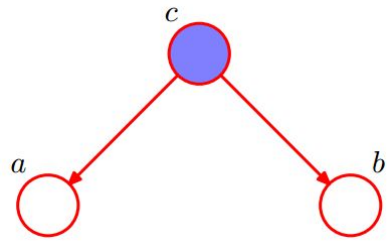


$$p(a,b,c) = p(c|a,b) \cdot p(a) \cdot p(b)$$
$$\therefore p(a,b) = \sum_c p(a,b,c) = \sum_c p(c|a,b) \cdot p(a) \cdot p(b) = p(a) \cdot p(b)$$

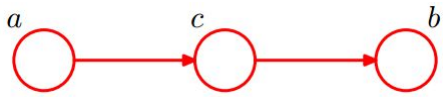
当c是collide，且不在条件集E中（未知）时，路径被阻断，a与b条件独立

C的子节点也不能在E中

2. 分连与顺连



$$p(a,b,c) = p(c) \cdot p(a|c) \cdot p(b|c)$$
$$\therefore p(a,b|c) = p(a|c) \cdot p(b|c)$$



$$p(a,b,c) = p(a) \cdot p(c|a) \cdot p(b|c)$$
$$p(a,b|c) = \frac{p(a,b,c)}{p(c)} = \frac{p(a) \cdot p(c|a) \cdot p(b|c)}{p(c)} = p(a|c) \cdot p(b|c)$$

当c不是collide，且已知的情况下，路径被阻断，a与b条件独立

Markov Blanket：一个结点，给定其Parent, Children, &children`s Parents 后，与网络中其余变量均条件独立

三、推理——enumeration

$$P(H, CD, DF, A, GF) = P(GF|DF, A)P(A|DF)P(DF|CD, H)P(CD)P(H)$$

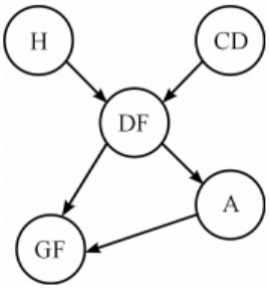
$P(h)$
0.5

$P(cd)$
0.6

DF	$\mathbf{P}(a\|DF)$
t	0.7
f	0.25

CD	H	$\mathbf{P}(df\|CD,H)$
t	t	0.15
t	f	0.01
f	t	0.99
f	f	0.1

DF	A	$\mathbf{P}(gf\|DF,A)$
t	t	0.99
t	f	0.4
f	t	0.5
f	f	0.05



$$p(CD|\neg a,gf)=\alpha\cdot\mathbf{p}(CD)\cdot\sum_{DF}\underline{\left\{\mathbf{p}(gf|DF,\neg a)\mathbf{p}(\neg a|DF)\sum_{H}[\mathbf{p}(DF|CD,H)\mathbf{p}(H)]\right\}}$$

要计算此向量需要分别求cd与$\neg cd$两个概率

$$p(cd|\neg a,gf)=\alpha\cdot p(cd)\cdot\left\{\begin{array}{c}p(gf|df,\neg a)p(\neg a|df)\cdot\left[\begin{array}{c}p(df|cd,h)p(h)\\+\\p(df|cd,\neg h)p(\neg h)\end{array}\right]\\+\\p(gh|\neg df,\neg a)p(\neg a|\neg df)\cdot\left[\begin{array}{c}p(\neg df|cd,h)p(h)\\+\\p(\neg df|cd,\neg h)p(\neg h)\end{array}\right]\end{array}\right\}$$

四、推理——variable elimination

引入新变量，矩阵运算，更快，变量间为逐元素相乘

$$p(CD|\neg a,gf)=\alpha\cdot\underbrace{p(CD)}_{f_1(CD)}\cdot\sum_{DF}\left\{\underbrace{p(gf|DF,\neg a)p(\neg a|DF)}_{f_2(DF)\quad f_3(DF)}\sum_{H}\underbrace{[p(DF|CD,H)p(H)]}_{f_4(DF,CD,H)\quad f_5(H)}\right\}$$

$$f_4\times f_5=\left\{\left[\begin{array}{cc}p(df|cd,h)&p(df|\neg cd,h)\\p(\neg df|cd,h)&p(\neg df|\neg cd,h)\end{array}\right],\left[\begin{array}{cc}p(df|cd,\neg h)&p(df|\neg cd,\neg h)\\p(\neg df|cd,\neg h)&p(\neg df|\neg cd,\neg h)\end{array}\right]\right\}\otimes\left[\begin{array}{cc}p(h)&p(\neg h)\end{array}\right]$$

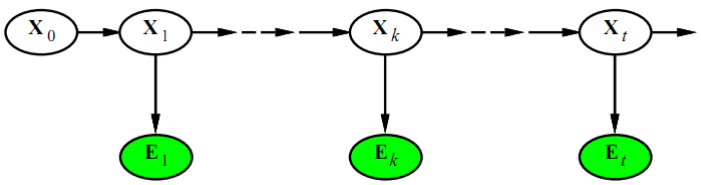
$$f_6\triangleq\sum_H f_4f_5=\left[\begin{array}{cc}p(df|cd)&p(df|\neg cd)\\p(\neg df|cd)&p(\neg df|\neg cd)\end{array}\right]$$

$$f_2f_3f_6=\left[\begin{array}{c}p(gf|df,\neg a)\\p(gf|\neg df,\neg a)\end{array}\right]\otimes\left[\begin{array}{c}p(\neg a|df)\\p(\neg a|\neg df)\end{array}\right]\otimes\left[\begin{array}{c|c}p(df|cd)&p(df|\neg cd)\\p(\neg df|cd)&p(\neg df|\neg cd)\end{array}\right]$$

$$f_7\triangleq\sum_{DF}f_2f_3f_6=\left[\begin{array}{cc}p(cd)&p(\neg cd)\end{array}\right]$$

目标$p(CD|\neg a,gf)=\alpha\cdot p(cd)\otimes f_7=\alpha\cdot\left[\begin{array}{cc}p(cd)&p(\neg cd)\end{array}\right]\otimes\left[\begin{array}{cc}p(cd)&p(\neg cd)\end{array}\right]$

隐马科夫模型HMM



不能直接观察到状态**X**（state），只能推理得出；但能观察到状态的输出**E**vidence（或**O**bservation）。又称传感器建模（传感器可能不精准，观察到事件E_1，隐藏状态可能是X_1-X_n中任何一个）

马科夫假设：

1. 下一个状态只取决于前一个状态$X_{t+1}\leftarrow X_t$

2. 可观察现象只取决于当前状态$O_t \leftarrow X_t$

一、四种问题

1. Filtering

只能用过去的数据求当下这一刻的状态，即求$P(X_{t+1}|e_{1:t+1})$。

迭代公式：

$$P(X_{t+1}|e_{1:t+1}) = \alpha \cdot \underbrace{\mathbf{P}(e_{t+1}|X_{t+1})}_{\text{传感器建模}} \cdot \sum_{x_t} \underbrace{\mathbf{P}(X_{t+1}|x_t)}_{\text{状态转移}} P(x_t|e_{1:t})$$

（矩阵形式）$f_{1:1+t} = \alpha \cdot O_{t+1} \cdot T \cdot f_{1:t}$

$$f_{1:1} = \alpha \cdot O_1 \cdot T \cdot f_0$$

2. Prediction 最简单

从当前状态不断状态转移求未来状态矩阵，$P(X_{t+k}|e_{1:t}), k \in [0, \infty) = T^k P(X_t|e_{1:t})$

3. Smoothing

用未来和过去的数据一起求某一刻状态，即求$P(X_k|e_{1:t}), k \in [1, t]$

$$\begin{aligned} P(X_k|e_{1:t}) &= P(X_k|e_{1:k}, e_{k+1:t}) \\ &= \alpha' P(X_k, e_{1:k}, e_{k+1:t}) \\ &= \alpha' \mathbf{P}(e_{k+1:t}|X_k, e_{1:k}) \cdot \mathbf{P}(X_k|e_{1:k}) \cdot P(e_{1:k}) \\ &= \alpha \cdot \underbrace{\mathbf{P}(e_{k+1:t}|X_k)}_{backward} \cdot \underbrace{\mathbf{P}(X_k|e_{1:k})}_{forward} \\ &\triangleq \alpha \cdot f_{1:k} \odot b_{k+1:t} \end{aligned}$$

forward部分与Filtering相同，只需要计算backward

$$\begin{aligned} \mathbf{P}(e_{k+1:t}|X_k) &= \sum_{x_{k+1}} \mathbf{P}(e_{k+1:t}, x_{k+1}|X_k) \\ &= \sum_{x_{k+1}} P(e_{k+1:t}|x_{k+1}, \cancel{X_k}) \cdot \mathbf{P}(x_{k+1}|X_k) \\ &= \sum_{x_{k+1}} P(e_{k+1}, e_{k+2:t}|x_{k+1}) \mathbf{P}(x_{k+1}|X_k) \\ &= \sum_{x_{k+1}} \underbrace{P(e_{k+1}|x_{k+1})}_{\text{发射矩阵}} P(e_{k+2:t}|x_{k+1}) \underbrace{\mathbf{P}(x_{k+1}|X_k)}_{\text{状态转移矩阵}} \\ \therefore b_{k+1:t} &\triangleq T^T \cdot O_{k+1} \cdot b_{k+2:t} \end{aligned}$$

4. Viterbi 算法——寻找最可能的隐藏状态序列

应用：语音识别说的话，拼音识别要打的字

> *与Smoothing的区别：smoothing只关心当下这一个状态最可能的值，而Viterbi关心最可能的一串状态序列。smoothing中，可以由t时刻所有状态转移到t+1时刻某个状态；而Viterbi假设在t时刻由某个特定状态转移到t+1时刻。因此当且仅当在第一个状态时，smoothing=Viterbi。*

$$\max_{x_1 \cdots x_t} \mathbf{P}(x_1, \cdots, x_t, X_{t+1}|e_{1:t+1}) \triangleq \overrightarrow{\mu}_{t+1}(X_{t+1})$$

$$= \alpha \underbrace{\mathbf{P}(e_{t+1}|X_{t+1})}_{\text{发射矩阵}} \max_{x_t} \left[\underbrace{\mathbf{P}(X_{t+1}|x_t)}_{\text{转移矩阵}} \max_{x_1 \cdots x_{t-1}} \underbrace{P(x_1, \cdots, x_{t-1}, x_t|e_{1:t})}_{max\text{←迭代，但只取所有路径中最可能的}} \right]$$

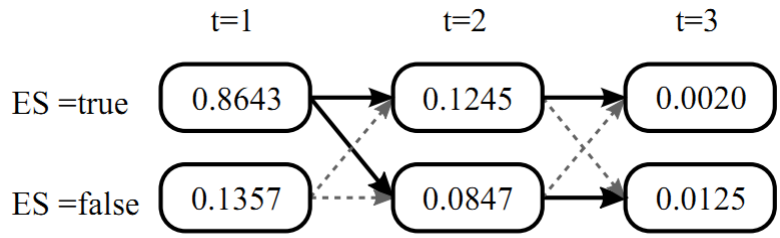
例题：

$$\mathbf{T}=\begin{pmatrix}0.8 & 0.3\\ 0.2 & 0.7\end{pmatrix},\mathbf{f_0}=\begin{pmatrix}0.7\\ 0.3\end{pmatrix},\mathbf{O_1}=\begin{pmatrix}0.72 & 0\\ 0 & 0.21\end{pmatrix},\mathbf{O_2}=\begin{pmatrix}0.18 & 0\\ 0 & 0.49\end{pmatrix},\mathbf{O_3}=\begin{pmatrix}0.02 & 0\\ 0 & 0.21\end{pmatrix}$$

$$\mu_1=f_{1:1}=\alpha O_1 T f_0=\alpha\begin{pmatrix}0.72 & 0\\ 0 & 0.21\end{pmatrix}\begin{pmatrix}0.8 & 0.3\\ 0.2 & 0.7\end{pmatrix}\begin{pmatrix}0.7\\ 0.3\end{pmatrix}=\begin{pmatrix}0.8643\\ 0.1357\end{pmatrix}$$

$$\begin{aligned}\mu_2=O_2\max\left[T\odot\mathbf{1}_N\cdot\mu_1^T\right]&=\begin{pmatrix}0.18 & 0\\ 0 & 0.49\end{pmatrix}\max\left[\begin{pmatrix}0.8 & 0.3\\ 0.2 & 0.7\end{pmatrix}\odot\left(\begin{array}{c|c}0.8643 & 0.1357\\ 0.8643 & 0.1357\end{array}\right)\right]\\&=\begin{pmatrix}0.18 & 0\\ 0 & 0.49\end{pmatrix}\cdot\max_{axis=0}\left(\begin{array}{c|c}0.6914 & 0.0407\\ 0.1729 & 0.0950\end{array}\right)\\&=\begin{pmatrix}0.1245\\ 0.0847\end{pmatrix}\end{aligned}$$

$$\begin{aligned}\mu_3=O_3\max\left[T\odot\mathbf{1}_N\cdot\mu_2^T\right]&=\begin{pmatrix}0.02 & 0\\ 0 & 0.21\end{pmatrix}\max\left[\begin{pmatrix}0.8 & 0.3\\ 0.2 & 0.7\end{pmatrix}\odot\left(\begin{array}{c|c}0.1245 & 0.0847\\ 0.1245 & 0.0847\end{array}\right)\right]\\&=\begin{pmatrix}0.02 & 0\\ 0 & 0.21\end{pmatrix}\cdot\max_{axis=0}\left(\begin{array}{c|c}0.0996 & 0.0254\\ 0.0249 & 0.0593\end{array}\right)\\&=\begin{pmatrix}0.0020\\ 0.0125\end{pmatrix}\end{aligned}$$



因此最可能的序列是T-F-F，在t=2时，虽然ES=T概率更高，但综合3天考虑后，最可能的序列不包括它。