人工智能复习笔记

制作人：匡乾 Sun Yat-sen University School of Data and Computer Science

Search
- formalization（形式化）
- Property of Search 搜索的属性
- Uninformed Search 无信息搜索
  - Breadth first 宽度优先
  - Depth first 深度优先
  - Uniform cost 一致代价搜索
  - Depth-limited search 深度受限搜索
  - Iterative deepening search 迭代加深搜索
  - Bidirectional search 双向搜索
    - 无信息搜索总结
- path checking / cycle checking 路径检测/环检测
  - 路径检测
  - 环检测
  - 总结
- Heuristic search 启发式搜索
  - Greedy best-first search (Greedy BFS) 最佳优先搜索
  - A* Search A*搜索
    - Admissible 可接纳性
    - Consistency (Monotonicity) 一致性、单调性
  - IDA* 迭代加深A*算法
- Game tree search 博弈树搜索
  - basic definition
  - MiniMax Strategy
  - Alpha-beta pruning
    - Alpha cut
    - Beta cut
  - 总结
- CSP (Constraint satisfaction problem)约束满足问题
  - Formalization 形式化
  - backtracking 回溯算法
  - Forward checking 向前检测
    - MRV (Minimum Remaining Values Heuristics) 最小剩余启发式
  - GAC (Generalized Arc Consistency) 整体边一致

KRR(Knowledge representation and reasoning) 知识表示与推理
- FOL(First-order logic) 一阶逻辑
- Clausal form
- Refutation
- Converting first-order formulas into clausal form
- Unification
- Resolution
- Answer extraction

Reasoning under Uncertainty 不确定推理
- Probability in General
- Bayesian Networks

graph + tables
Construct a Bayes Net
Inference
Variable Elimination
VE Algorithm:
D-Separation
Machine Learning 机器学习
Decision tree 决策树
Entropy 熵
Information gain 信息增益
Overfit 过度拟合
Bayes Learning 贝叶斯学习
Maximum a posteriori （极大后验MAP）
Maximum Likelihood （极大似然ML）
Articial Neural Networks 神经网络
Linear regression 线性回归
Logistic regression 逻辑回归
Forward and backward phases

Search

- Problem solving by search: formalization
- Uninformed search: Breadth-First, Uniform-Cost, Depth-First, Depth-Limited, and Iterative- Deepening
- Heuristic search: Greedy best-first, A*
- Properties of search: completeness, optimality, time and space complexity
- Path/cycle checking
- Game tree search: MiniMax, alpha-beta pruning
- CSP: Formalization, backtracking, forward checking, and GAC algorithms

formalization（形式化）

1. Formulate a **state space** (形式化状态空间)

   抽象真实问题

2. Formulate **actions** （形式化动作）

allow one to move between different states

3. Identify the **initial state** （确定初始状态）
4. Identify the **goal** or **desired condition** （确定目标）
5. Formulate heuristic （形式化启发式）

Example：

- States: the various cities you could be located in.
- Actions: drive between neighboring cities.
- Initial state: in Arad
- Goal: in Bucharest
- Solution: the route, the sequence of cities to travel through to get to Bucharest.

Property of Search 搜索的属性

- **Completeness 完备性**: will the search always find a solution if a solution exists?
- **Optimality 最优性** : will the search always find the least cost solution? (when actions have costs)
- **Time complexity 时间复杂度**: what is the maximum number of nodes than can be expanded or generated?
- **Space complexity 空间复杂度**: what is the maximum number of nodes that have to be stored in memory?

Uninformed Search 无信息搜索

Breadth first 宽度优先

将继承者放置到边界末端

example:

{0<>}

{1,2}

{2,2,3}

{2,3,3,4}

{3,3,4,3,4}

{3,4,3,4,4,5}

完备性、最优性：Yes

从小到大寻求方案，直到找到答案为止

最大继承数：b

最小解决方案步数：d

时间复杂度：$1+b+b^2+\cdots+b^d+b(b^d-1)=O(b^{d+1})$

空间复杂度：$b(b^d-1)=O(b^{d+1})$

Depth first 深度优先

将继承者放置到边界前端

example:

{0}

{1,2}

{2,3,2}

{3,4,3,2}

{4,5,4,3,2}

{5,6 4,5,4,3,2}

完备性：

- Infinite state space: No
- Finite state space with infinite paths: No
- Finite state space and prune paths with duplicate states ? Yes

最优性：No

最大继承数：b

最小解决方案步数：d

时间复杂度：$O(b^m)$ m是状态空间中最长的路径；若m远远大于d则非常糟糕，但若有多个解往往会比较快

空间复杂度：$O(bm)$ 线性，每次仅探索一条路径

Uniform cost 一致代价搜索

边界顺序由代价(cost)决定，永远扩展代价最小的路径

完备性、最优性：Yes

C*：最优结果的代价 ϵ：每一步的代价

时间、空间复杂度：$O(b^{C^*/\epsilon+1})$

Depth-limited search 深度受限搜索

设置的深度：L

- Completeness: No
- Optimality: No
- Time complexity: $O(b^L)$
- Space complexity: $O(bL)$

Iterative deepening search 迭代加深搜索

初始令L=0，并逐渐增大L

- Completeness: Yes
- Optimality: Yes if costs are uniform

时间复杂度：$O(b^d)$

空间复杂度：$O(bd)$

Bidirectional search 双向搜索

Completeness: Yes

Optimality: if edges have uniform costs

Time and space complexity: $O(b^{d/2})$

无信息搜索总结

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening	Bidirectional (if applicable)
Complete?	Yes[a]	Yes[a,b]	No	No	Yes[a]	Yes[a,d]
Time	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(b^m)$	$O(b^\ell)$	$O(b^d)$	$O(b^{d/2})$
Space	$O(b^d)$	$O(b^{1+\lfloor C^*/\epsilon \rfloor})$	$O(bm)$	$O(b\ell)$	$O(bd)$	$O(b^{d/2})$
Optimal?	Yes[c]	Yes	No	No	Yes[c]	Yes[c,d]

(BFS中的空间和时间改为$O(b^{d+1})$)

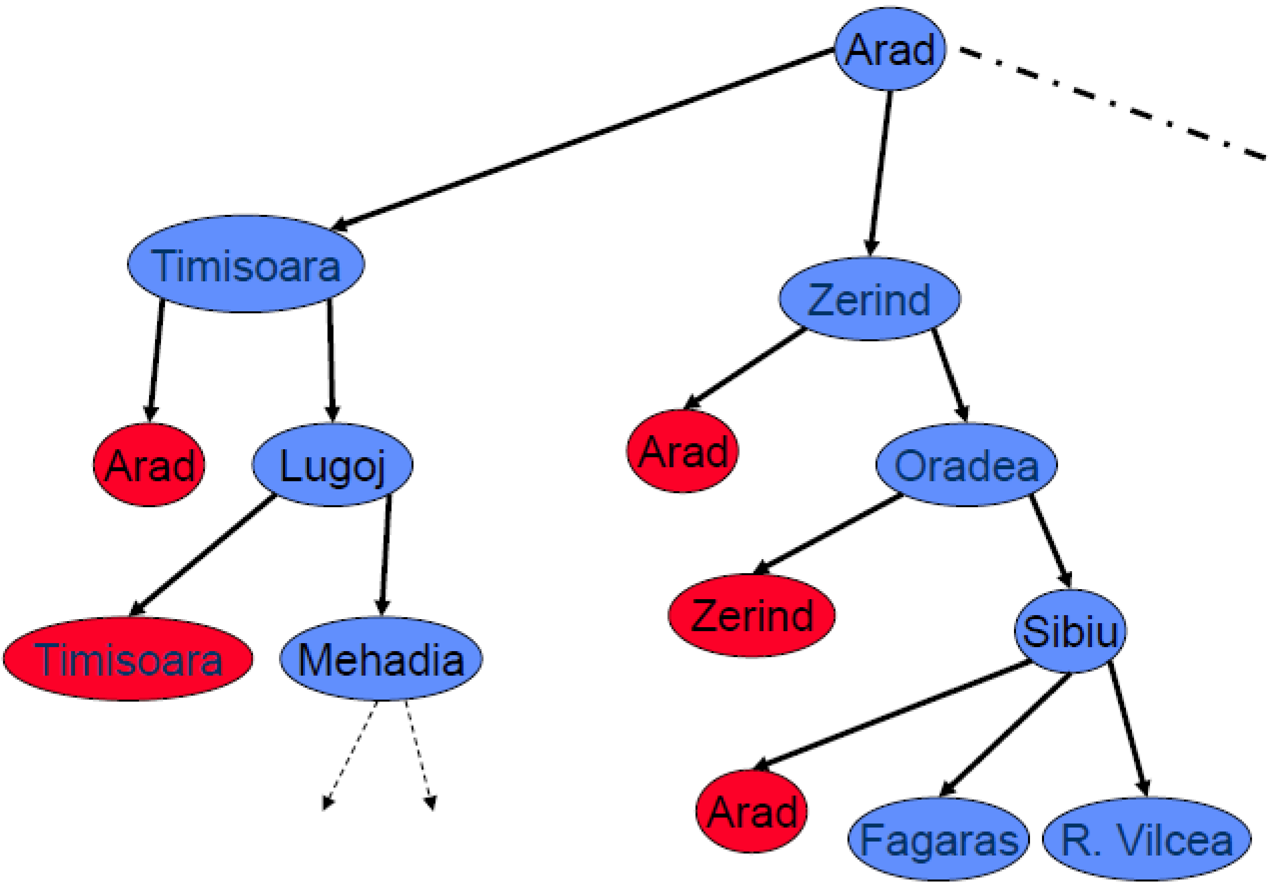
path checking / cycle checking 路径检测/环检测

路径检测

通向c的路径：$< n_1, \cdots, n_k, c >$

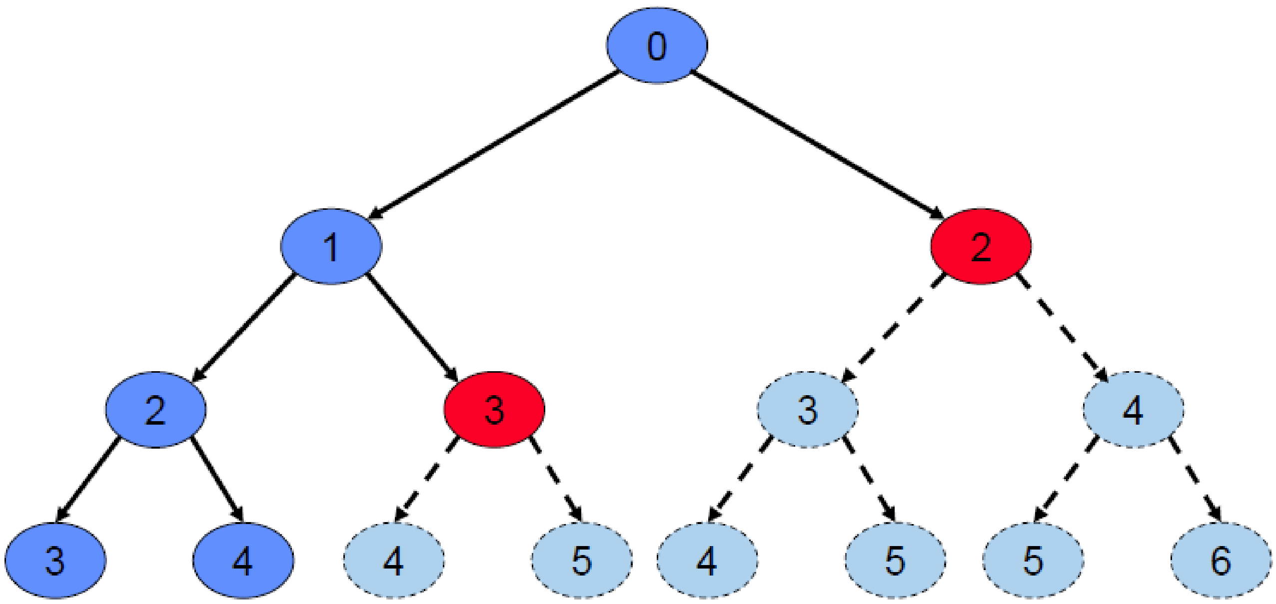
则c不能与n_i相同

总结



环检测

在整个探索过程中记录结点，确保扩展的结点c不与之前任何状态中的结点相同



总结

- Path checking: when we expand n to obtain child c, ensures that the state c is not equal to the state reached by any ancestor of c along this path
- Cycle checking: keep track of all states previously expanded during the search; when we expand n to obtain child c, ensure that c is not equal to any previously expanded state
- For uniform-cost search, cycle checking preserves optimality

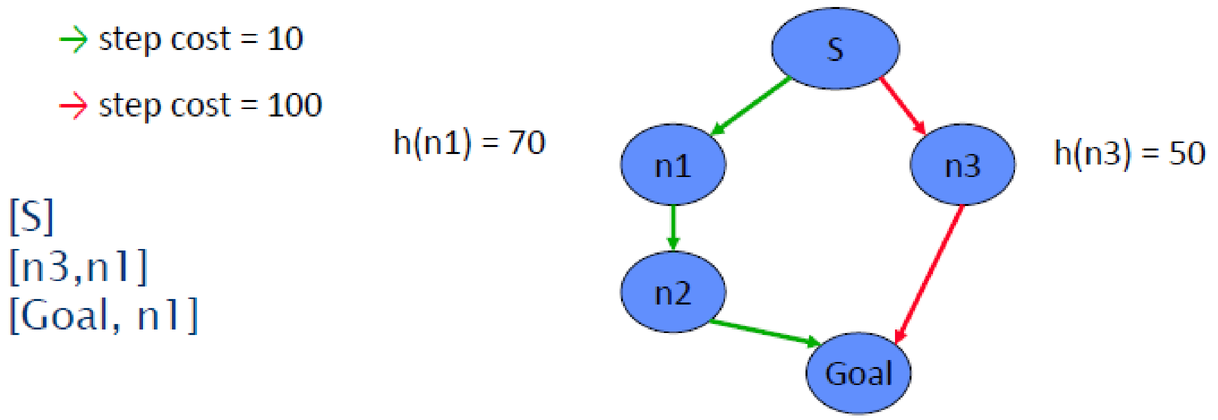
Heuristic search 启发式搜索

idea: 得到启发式函数$h(n)$，预测从当前节点n到目标节点的花费

Greedy best-first search (Greedy BFS) 最佳优先搜索

用h(n)对边界中的结点进行排序，优先获取low cost的解

该方法忽略了到达n的cost



Thus Greedy BFS is incomplete, not optimal

A* Search A*搜索

evaluation function 评估函数：$f(n) = g(n) + h(n)$

$g(n)$是到达结点n的路径花费

$h(n)$是启发式估计从结点n到达终点的花费

$f(n)$是对经过结点n到达终点的估计

Admissible 可接纳性

$h^*(n)$是从n到达终点的最佳路径的花费

$h(n)$是可容许的如果对于所有节点n都有$h(n) \leq h^*(n)$

Admissible 可容纳的启发式低估真正的花费

$h(g)=0$，如果n不能到达终点则$h(n)=\infty$

可接纳性 $\rightarrow$最佳性 Admissibility implies optimality

Consistency (Monotonicity) 一致性、单调性

$h(n)$**一致的/单调的**，如果对于任意结点$n1, n2$都有$h(n1) \leq c(n1 \rightarrow n2) + h(n2)$

一致性 $\rightarrow$ 可接纳性 Consistency implies admissibility

Note that consistency implies admissibility (proof)

- Case 1: no path from n to the goal
- Case 2: Let $n = n_1 \rightarrow n_2 \rightarrow \ldots \rightarrow n_k$ be an optimal path from n to a goal node. We prove by induction on i that for all i, $h(n_i) \leq h^*(n_i)$.

单调性保证能在第一次到达某结点就是最佳路径

若没有单调性，则需要记住之前路径的花费

性质：

1. **Proposition 1**. The f-values of nodes along a path must be non-decreasing

   f(n)单调递增

2. **Proposition 2**. If n_2 is expanded after n_1, then $f(n_1) \leq f(n_2)$

   若n2在n1后出现，则$f(n_1) \leq f(n_2)$

3. **Proposition 3**. When n is expanded every path with lower f-value has already been expanded.

   任何f花费小于f(n)的结点必然已经扩展过

4. **Proposition 4**. The first time A* expands a state, it has found the minimum cost path to that state.

   第一次扩展到的结点就是最短路径

5. 在单调性的前提下，换检测保证了最佳性

IDA* 迭代加深A*算法

迭代cutoff value为f-value，而不是原来的L（深度）

边界中以f(n)的大小来排序

Theorem. The optimal cost to nodes in the relaxed problem is an admissible heuristic for the original problem!

放松问题中的最优花费是对于原问题可接受的启发式

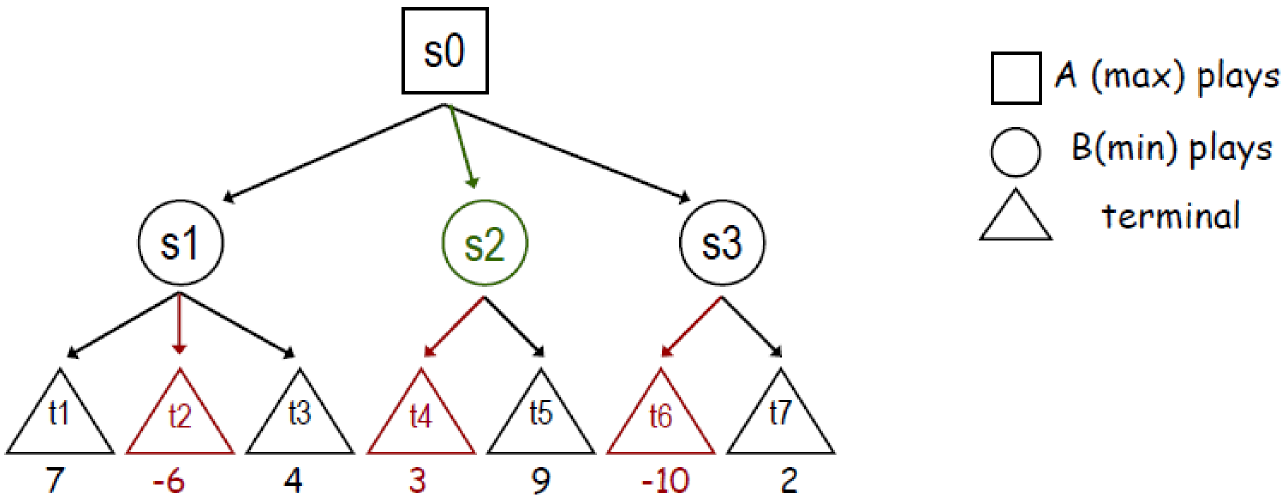
Game tree search 博弈树搜索

basic definition

- Player: A(Max), B(Min)
- State: S
- Initial state: I
- Terminal state: T
- Successors
- Utility(效益), Payoff function: V

MiniMax Strategy

- $U(n)$ = min {$U(c)$: c is a child of n} if n is a Min node
- $U(n)$ = max {$U(c)$: c is a child of n} if n is a Max node



```
DFMiniMax(n, Player) //return Utility of state n given that
                     //Player is MIN or MAX

If n is TERMINAL
Return V(n) //Return terminal states utility
            //(V is specified as part of game)

//Apply Player's moves to get successor states.
ChildList = n.Successors(Player)
If Player == MIN
    return minimum of DFMiniMax(c, MAX) over c  ∈ ChildList
Else //Player is MAX
    return maximum of DFMiniMax(c, MIN) over c  ∈ ChildList
```

Alpha-beta pruning

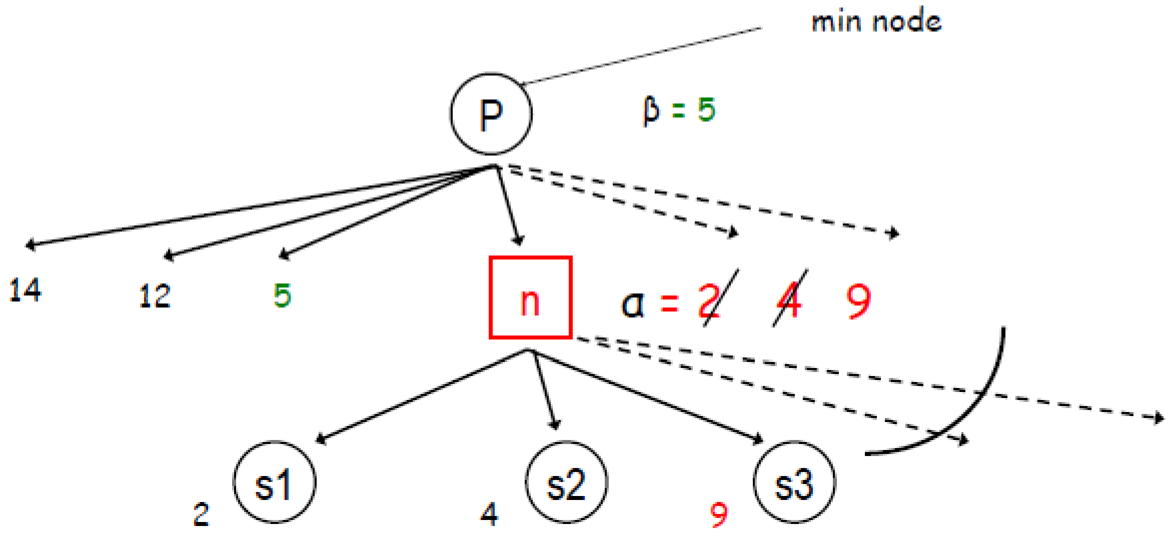
Two types of pruning:

- pruning of max nodes (α-cuts)
- pruning of min nodes (β-cuts)

Alpha cut

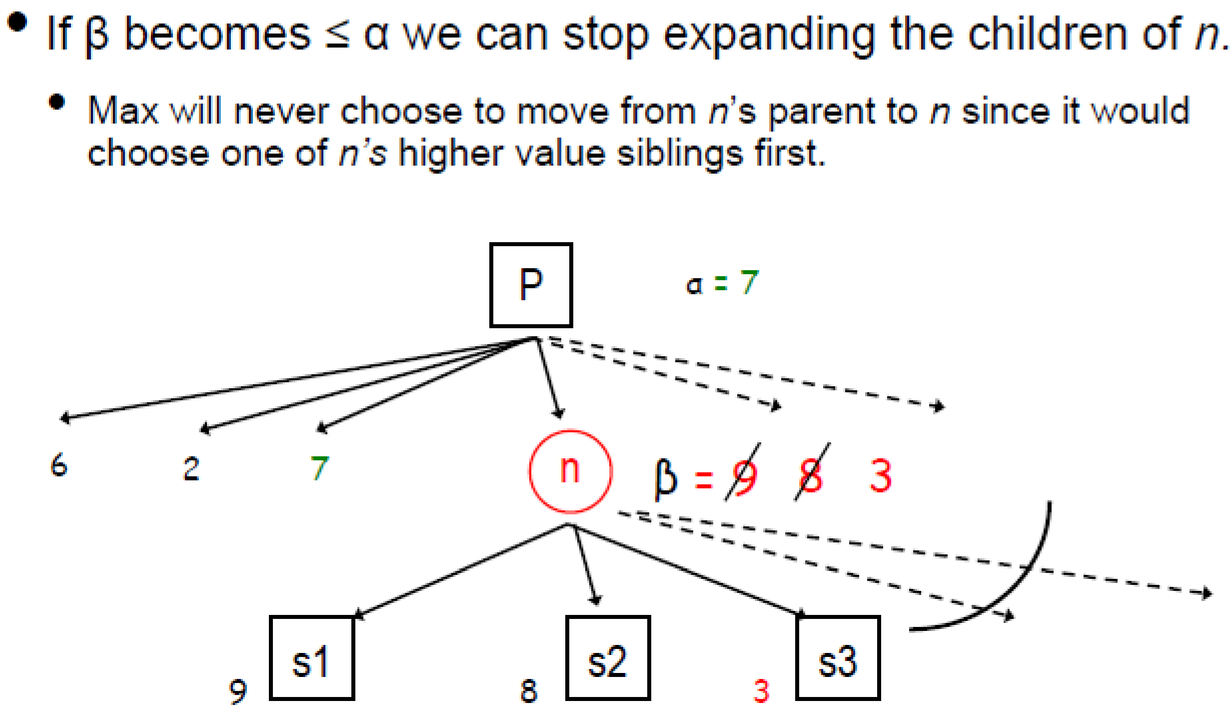
- At a Max node *n*:
  - Let β be the lowest value of *n*'s siblings examined so far (siblings to the left of *n* that have already been searched)
  - Let α be the highest value of *n*'s children examined so far (changes as children examined)

- While at a Max node *n*, if α becomes ≥ β we can stop expanding the children of *n*
  - Min will never choose to move from *n*'s parent to *n* since it would choose one of *n's* lower valued siblings first.



Beta cut

- At a Min node *n*:
  - Let α be the highest value of *n*'s sibling's examined so far (fixed when evaluating *n*)
  - Let β be the lowest value of *n*'s children examined so far (changes as children examined)



总结

当 $\beta \leq \alpha$时，进行剪枝

Minimax 需要探索 $O(b^D)$个结点，而alpha-beta剪枝需要探索$O(b^{D/2})$个结点

CSP (Constraint satisfaction problem)约束满足问题

Formalization 形式化

A CSP consists of:

- A set of variables: V1, ... ,Vn
- Each variable has a domain: Dom[Vi] ($V_i = d \iff d \in Dom[V_i]$)
- A set of constraints: C1, ... ,Cm e.g. C(V1,V2,V4)

goal: 寻找满足条件的解，使得各个变量都有取值

backtracking 回溯算法

- We pick a variable*,
- pick a value for it*,
- test the constraints that we can,
- if a constraint is unsatisfied we backtrack,
- otherwise we set another variable.
- When all the variables are set, we're done.

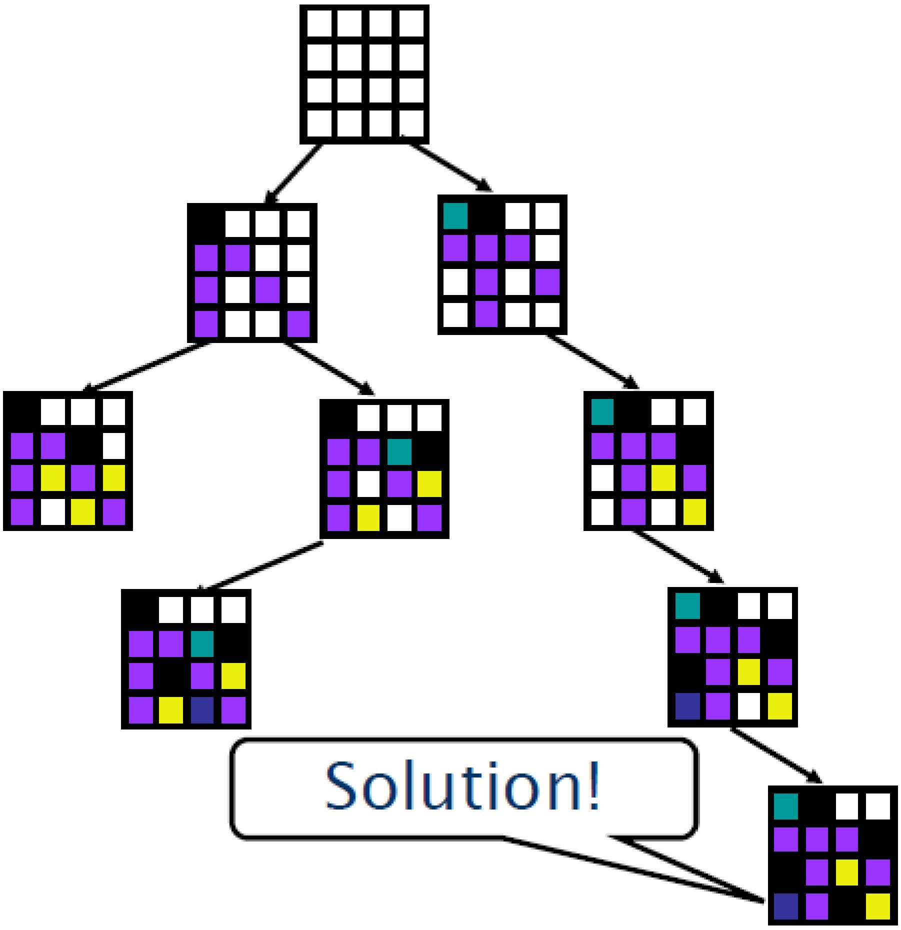
启发式应用于挑选变量和挑选值：

the order in which variables are assigned:
PickUnassignedVariable()
the order of values tried for each variable.

Forward checking 向前检测

检查那些只含有一个未实例化变量的约束，去除那个变量所有违反约束取值

同时要记住，每一个值是在哪一步被去除的



MRV (Minimum Remaining Values Heuristics) 最小剩余启发式

先执行值域较小的变量，当一个变量只有一个取值时，立即执行

- What variables would you try first?

8	1	5	6					4
6				7	5		8	
				9				
9				4	1	7		
	4						2	
		6	2	3				8
				5				
	5		9	1				6
1					7	8	9	5

Domain of each variable: {1, ..., 9}

(1, 5) **impossible values:**
Row: {1, 4, 5, 6, 8}
Column: {1, 3, 4, 5, 7, 9}
Subsquare: {5, 6, 7, 9}
→ Domain = {2}

(9, 5) **impossible values:**
Row: {1, 5, 7, 8, 9}
Column: {1, 3, 4, 5, 7, 9}
Subsquare: {1, 5, 7, 9}
→Domain = {2, 6}

After assigning value 2 to cell (1,5): Domain = {6}

Most restricted variables! = MRV

GAC (Generalized Arc Consistency) 整体边一致

Some definition:

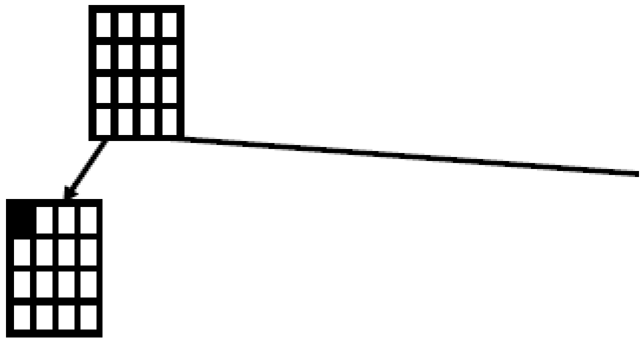
- C(X,Y) is consistent $\iff$ $\forall x, \exists y$ 满足 C
- C(V1,V2, ... ,Vn) 关于Vi is GAC $\iff$ $\forall Vi, \exists V1, ... Vi-1, Vi+1, ... Vn$ 满足 C
- A constraint(C) is GAC $\iff$ 关于它的任何变量都是GAC的
- A CSP is GAC $\iff$ 所有限制(C)都是GAC的

如果对于变量V，取值d不能得到一个解，这说d是arc inconsistent（边不一致的）

C(X,Y): $X > Y$, Dom(X)={1,5,11}, Dom(Y)={3,8,15}

- For X=1 there is no value of Y s.t. $1 > Y$, so remove 1 from domain X
- For Y=15 there is no value of X s.t. $X > 15$, so remove 15 from domain Y
- We obtain Dom(X)={5,11} and Dom(Y)={3,8}.

GAC检查的过程需要不断的循环，因为一个定义域改变可能引起其它定义域变化



V1 = 1

arc consistency stages:

1. V2 = {3,4}, V3 = {2,4}, V4 = {2, 3}
   V2=1,2 & V3 = 1,3 & V3 = 1,4 are inconsistent with V1=1.
2. V2 = {4} (**V2=3 is inconsistent** with both values in CurDom[V3]
3. V3 = {2} (**V3 = 2 is inconsistent with values in CurDom[V2]**
4. V4 = {} (both values for V4 inconsistent with values in CurDom[V3]

DWO

GAC必须在每个节点都检查所有限制(C)

Example:http://www.cs.toronto.edu/~fbacchus/csc384/Lectures/Tutorial3_CSP.pdf

KRR(Knowledge representation and reasoning) 知识表示与推理

- First-order logic: syntax and semantics
- Soundness and completeness of proof procedures
- Converting first-order formulas into clausal form
- Unification and MGU
- Resolution proof: forward chaining and refutation
- Answer extraction

知识表示假设：所有AI system都是基于知识的(knowledge-based)

FOL(First-order logic) 一阶逻辑

$a \rightarrow b \iff \neg a \vee b$

$a \leftrightarrow b \iff (a \rightarrow b) \wedge (b \rightarrow a)$

Clausal form

e.g., $p \vee \neg r \vee s$, written $(p, \neg r, s)$

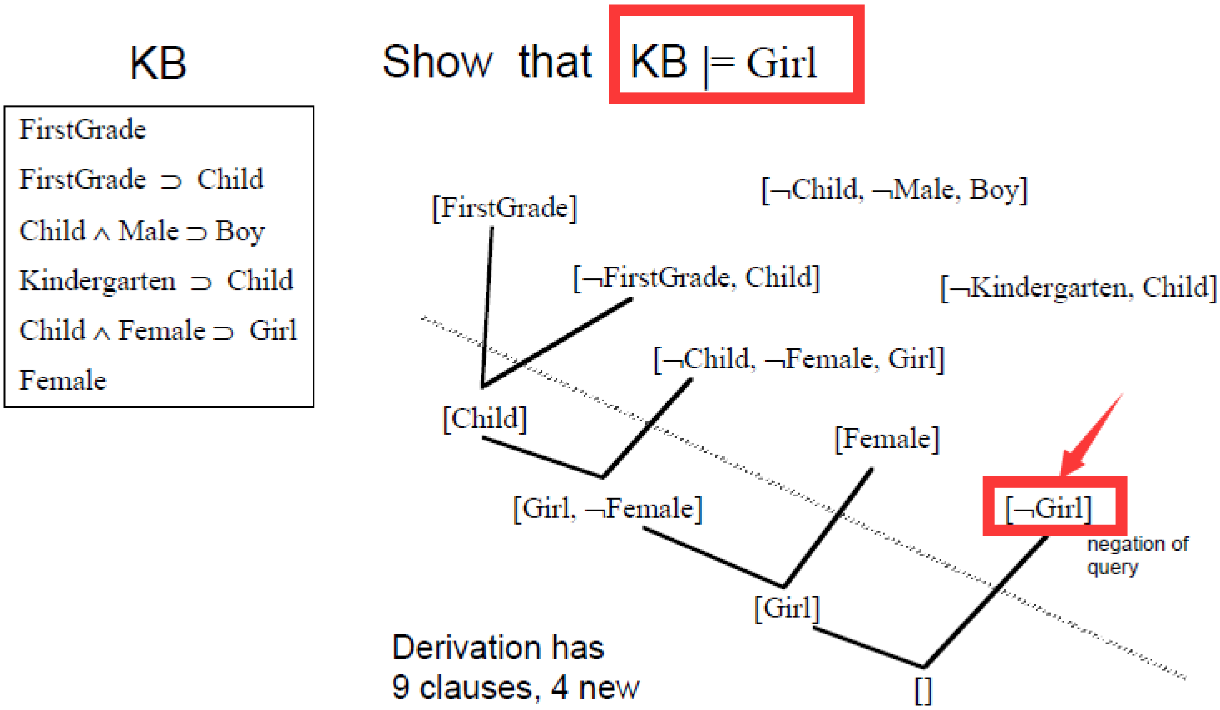
Proposition. $\{p\} \cup c_1, \{\neg p\} \cup c_2 \models c_1 \cup c_2$

Refutation

$KB \models \alpha$ iff $KB \wedge \neg\alpha$ is unsatisfiable

Thus to check if $KB \models \alpha$,

- put KB and $\neg\alpha$ into clausal form to get S,
- check if $S \vdash ()$



Converting first-order formulas into clausal form

Step:

1. Eliminate Implications （消去蕴含）

   $A \rightarrow B \iff \neg A \vee B$

2. Move negations inwards using （将括号外，量词外的非挪到里面）

   - $\neg(A \vee B) \Leftrightarrow \neg A \wedge \neg B,\ \neg(A \wedge B) \Leftrightarrow \neg A \vee \neg B$
   - $\neg\exists x.A \Leftrightarrow \forall x.\neg A,\ \neg\forall x.A \Leftrightarrow \exists x.\neg A,\ \neg\neg A \Leftrightarrow A$

3. Standardize Variables （规范变量名称，使每个量化变量都unnique）

   $$\forall x\{\neg P(x) \vee [\forall y(\neg P(y) \vee P(f(x,y))) \wedge \exists y(Q(x,y) \vee \neg P(y))]\}$$

   3. Standardize Variables (Rename variables so that each quantified variable is unique)

   $$\forall x\{\neg P(x) \vee [\forall y(\neg P(y) \vee P(f(x,y))) \wedge \exists z(Q(x,z) \vee \neg P(z))]\}$$

4. Skolemize （将所有带有存在量词的变量，转换为关于全称量词变量的函数）

$$\forall x\{\neg P(x) \vee [\forall y(\neg P(y) \vee P(f(x,y))) \wedge \exists z(Q(x,z) \vee \neg P(z))]\}$$

4. Skolemize (Remove existential quantifiers by introducing new function symbols)

$$\forall x\{\neg P(x) \vee [\forall y(\neg P(y) \vee P(f(x,y))) \wedge (Q(x,g(x)) \vee \neg P(g(x)))]\}$$

5. Convert to prenex form （转换为前束范式，即将所有量词提到最前面）

6. Disjunctions over conjunctions （把交提出来）

$A \vee (B \wedge C) \iff (A \vee B) \wedge (A \vee C)$

7. Flatten nested conjunctions and disjunctions （不知道干嘛的）

8. Convert to Clauses （去除量词，把交分开）

$$\begin{aligned}\forall x \forall y\{(\neg P(x) \vee \neg P(y) \vee P(f(x,y))) \wedge \\ (\neg P(x) \vee Q(x,g(x)) \vee \neg P(g(x)))\}\end{aligned}$$

8. Convert to Clauses (remove quantifiers and break apart conjunctions).

a) $\neg P(x) \vee \neg P(y) \vee P(f(x,y))$
b) $\neg P(x) \vee Q(x,g(x)) \vee \neg P(g(x))$

Unification

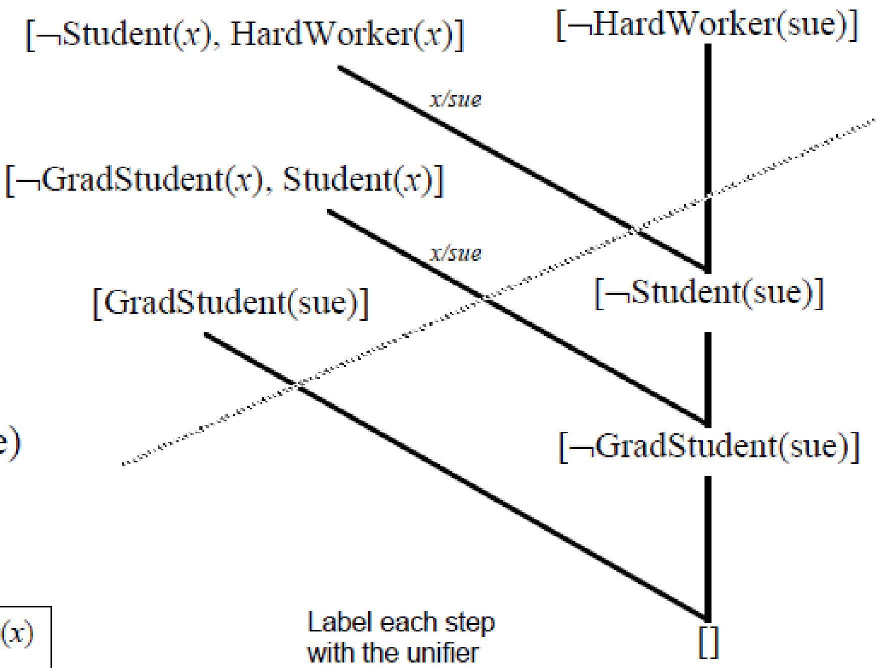
- Let $\theta = \{x = f(y), y = z\}$, $\sigma = \{x = a, y = b, z = y\}$
- Step 1. Get $S = \{x = f(b), y = y, x = a, y = b, z = y\}$
- Step 2. Delete $y = y$.
- Step 3. Delete $x = a$.
- The result is $S = \{x = f(b), y = b, z = y\}$

Resolution

example:

1. $(P(x), Q(g(x)))$
2. $(R(a), Q(z), \neg P(a))$
3. R[1a,2c]{X=a} $(Q(g(a)), R(a), Q(z))$

- "R" means resolution step.



$$\text{KB} \overset{?}{\models} \text{HardWorker(sue)}$$

KB

$\forall x$ GradStudent(x) $\supset$ Student(x)
$\forall x$ Student(x) $\supset$ HardWorker(x)
GradStudent(sue)

Label each step with the unifier

Point to relevant literals in clauses

Prove that $\exists y \forall x P(x, y) \models \forall x \exists y P(x, y)$

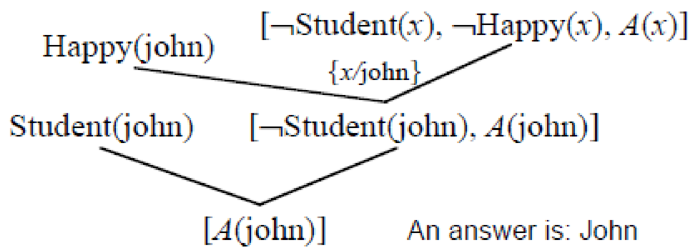
- $\exists y \forall x P(x, y) \Rightarrow 1.P(x, a)$
- $\neg \forall x \exists y P(x, y) \Leftrightarrow \exists x \forall y \neg P(x, y) \Rightarrow 2.\neg P(b, y)$
- R[1,2]$\{x = b, y = a\}()$

Answer extraction

- We can also answer wh- questions
- Replace query $\exists x P(x)$ by $\exists x[P(x) \wedge \neg answer(x)]$
- Instead of deriving (), derive any clause containing just the answer predicate

KB: Student(john)
Student(jane)
Happy(john)

Q: $\exists x[\text{Student}(x) \wedge \text{Happy}(x)]$



直接在Clausal form下的query插入answer(x)即可

Reasoning under Uncertainty 不确定推理

- Bayesian networks: graphs + tables, inference
- Variable elimination algorithm
- Use D-separation to determine independence

Probability in General

- $Pr(U) = 1$
- $Pr(A) \in [0, 1]$
- $Pr(A \cup B) = Pr(A) + Pr(B) - Pr(A \cap B)$

$$Pr(\{V_1 = a\}) = \sum_{x_2 \in D[V_2]} \cdots \sum_{x_n \in D[V_n]} Pr(V_1 = a, V_2 = x_2, \ldots, V_n = x_n)$$

Conditional probabilities（条件概率）：

$$Pr(B|A) = Pr(B \cap A)/Pr(A)$$

全集分割:

$$B_1, B_2, \ldots, B_k$$

（不交，不漏）

- $B_i \cap B_j = \emptyset$, $i \neq j$ (mutually exclusive)
- $B_1 \cup B_2 \cup \ldots \cup B_k = U$ (exhaustive)

In probabilities:

- $Pr(B_i \cap B_j) = 0$, $i \neq j$
- $Pr(B_1 \cup B_2 \cup \ldots \cup B_k) = 1$

Sumout rule:

$$Pr(A) = Pr(A \cap B_1) + \ldots + Pr(A \cap B_k)$$

In conditional probabilities:

$$Pr(A) = Pr(A|B_1)Pr(B_1) + \ldots + Pr(A|B_k)Pr(B_k)$$

Independent:

$Pr(B|A) = Pr(B)$ (B is independent of A)

- If A and B are independent, then
  $Pr(A \cap B) = Pr(A) \cdot Pr(B)$
- If given A, B and C are conditionally independent, then
  $Pr(B \cap C|A) = Pr(B|A) \cdot Pr(C|A)$

Bayes rule:

$Pr(Y|X) = Pr(X|Y)Pr(Y)/Pr(X)$

Chain rule:

$$Pr(A_1 \cap A_2 \cap \ldots \cap A_n) = Pr(A_1|A_2 \cap \ldots \cap A_n) \cdot$$
$$Pr(A_2|A_3 \cap \ldots \cap A_n) \cdot \ldots \cdot Pr(A_{n-1}|A_n) \cdot Pr(A_n)$$

Notation / Terminology:

Pr(X) == Pr(X=d) for all d in Dom[X]

$$\sum_{d \in Dom[X]} Pr(X = d) = 1$$

Inference:

- Computing Pr(a) in more concrete terms:
  - Pr(c) = Pr(c|e)Pr(e) + Pr(c|~e)Pr(~e)

    = 0.9 * 0.7 + 0.5 * 0.3 = 0.78
  - Pr(~c) = Pr(~c|e)Pr(e) + Pr(~c|~e)Pr(~e) = 0.22
    - Pr(~c) = 1 – Pr(c), as well
  - Pr(a) = Pr(a|c)Pr(c) + Pr(a|~c)Pr(~c)

    = 0.3 * 0.78 + 1.0 * 0.22 = 0.454
  - Pr(~a) = 1 – Pr(a) = 0.546

Bayesian Networks

graph + tables

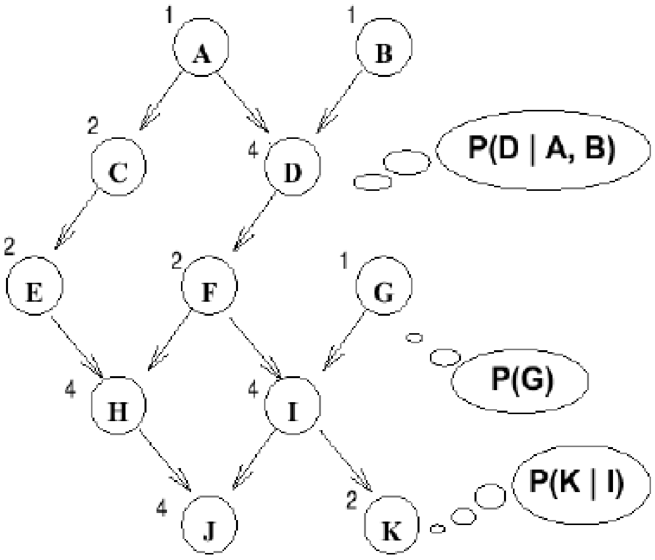
A BN over variables $\{X_1, X_2, \ldots, X_n\}$ consists of:

- a DAG (directed acyclic graph) whose nodes are the variables
- a set of CPTs (conditional probability tables) $Pr(X_i|Par(X_i))$ for each X_i

example:

Pr(A,B,C,D,E,F,G,H,I,J,K) =

Pr(A)
x Pr(B)
x Pr(C|A)
x Pr(D|A,B)
x Pr(E|C)
x Pr(F|D)
x Pr(G)
x Pr(H|E,F)
x Pr(I|F,G)
x Pr(J|H,I)
x Pr(K|I)



Construct a Bayes Net

- Step 1 Apply the Chain Rule

$$Pr(X_1, \dots, X_n) = Pr(X_n|X_1, \dots, X_{n-1}) Pr(X_{n-1}|X_1, \dots, X_{n-2}) \dots Pr(X_1)$$

- Step 2 移除所有无关变量

$$Pr(X_n|Par(X_n)) Pr(X_{n-1}|Par(X_{n-1})) \dots Pr(X_1)$$

- Step 3 建立一个图(DAG)
- Step 4 确定CPT(conditional probability table)条件概率表格

Inference

Given

1) a **Bayes net**

$Pr(X_1, X_2, \ldots, X_n)$

$= Pr(X_n \mid Par(X_n)) * Pr(X_{n-1} \mid Par(X_{n-1}) * \cdots * Pr(X_1 \mid Par(X_1))$

2) some **Evidence**, E

E = {a set of values for some of the variables}

We want to

- **compute the new probability distribution**

$Pr(X_k \mid E)$

That is, we want to figure out

$Pr(X_k = d \mid E)$ for all $d \in Dom[X_k]$

Variable Elimination

Variable elimination uses

- the product decomposition, and
- the summing out rule

- In general, at each stage VE will compute a table of numbers: one for each different instantiation of the variables in the sum.

- Let f(**X**,**Y**) & g(**Y**,**Z**) be two factors with variables **Y** in common
- The ***product*** of f and g, denoted $h = f \times g$ (or sometimes just h = fg), is defined:

$$h(\mathbf{X},\mathbf{Y},\mathbf{Z}) = f(\mathbf{X},\mathbf{Y}) \times g(\mathbf{Y},\mathbf{Z})$$

f(A,B)		g(B,C)		h(A,B,C)			
ab	0.9	bc	0.7	abc	0.63	ab~c	0.27
a~b	0.1	b~c	0.3	a~bc	0.08	a~b~c	0.02
~ab	0.4	~bc	0.8	~abc	0.28	~ab~c	0.12
~a~b	0.6	~b~c	0.2	~a~bc	0.48	~a~b~c	0.12

restrict a Factor:

- Let f(X,**Y**) be a factor with variable X (**Y** is a set)
- We ***restrict*** factor f *to* X=*a* by setting X to the value *a* and "deleting" incompatible elements of f's domain . Define $h = f_{X=a}$ as: h(**Y**) = f(a,**Y**)

f(A,B)		$h(B) = f_{A=a}$	
ab	0.9	b	0.9
a~b	0.1	~b	0.1
~ab	0.4		
~a~b	0.6		

VE Algorithm:

Given:

- Bayes Net with CPTs factors F,
- query variable Q,
- evidence variables E (observed to have values e),
- remaining variables Z.

Now Compute Pr(QjE)

1. Replace each factor $f \in F$ that mentions a variable(s) in **E** with its restriction $f_{\mathbf{E}=e}$ (this might yield a "constant" factor)
2. For each Z_j – in the order given –eliminate $Z_j \in \mathbf{Z}$ as follows:
   1. Let $f_1, f_2, \ldots, f_k$ be the factors in F that include Z_j
   2. Compute new factor $g_j = \sum_{Z_j} f_1 \times f_2 \times \ldots \times f_k$
   3. Remove the factors f_i from F and add new factor g_j to F
3. The remaining factors refer only to the query variable Q. Take their product and normalize to produce Pr(Q|**E**).

1. 用已知事实替换变量
2. 将变量Zj用关于其它变量的函数表示，从而消去Zj

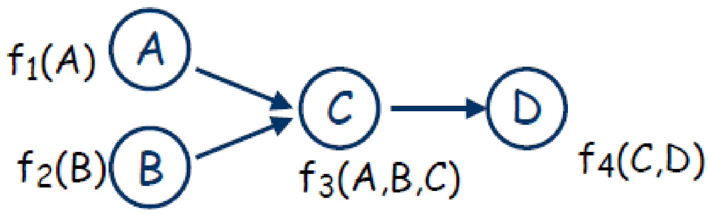
   将包含Zj的用fi表示，并将它们全部消去最后加入新产生的gi
3. 最后只剩下查询变量

Factors: $f_1(A)$ $f_2(B)$ $f_3(A,B,C)$ $f_4(C,D)$

Query: P(A)?

Evidence: D = d

Elim. Order: C, B



Restriction: replace $f_4(C,D)$ with $f_5(C) = f_4(C,d)$

Step 1: Eliminating C: Compute & Add $f_6(A,B) = \Sigma_C f_5(C) f_3(A,B,C)$

Remove: $f_3(A,B,C)$, $f_5(C)$

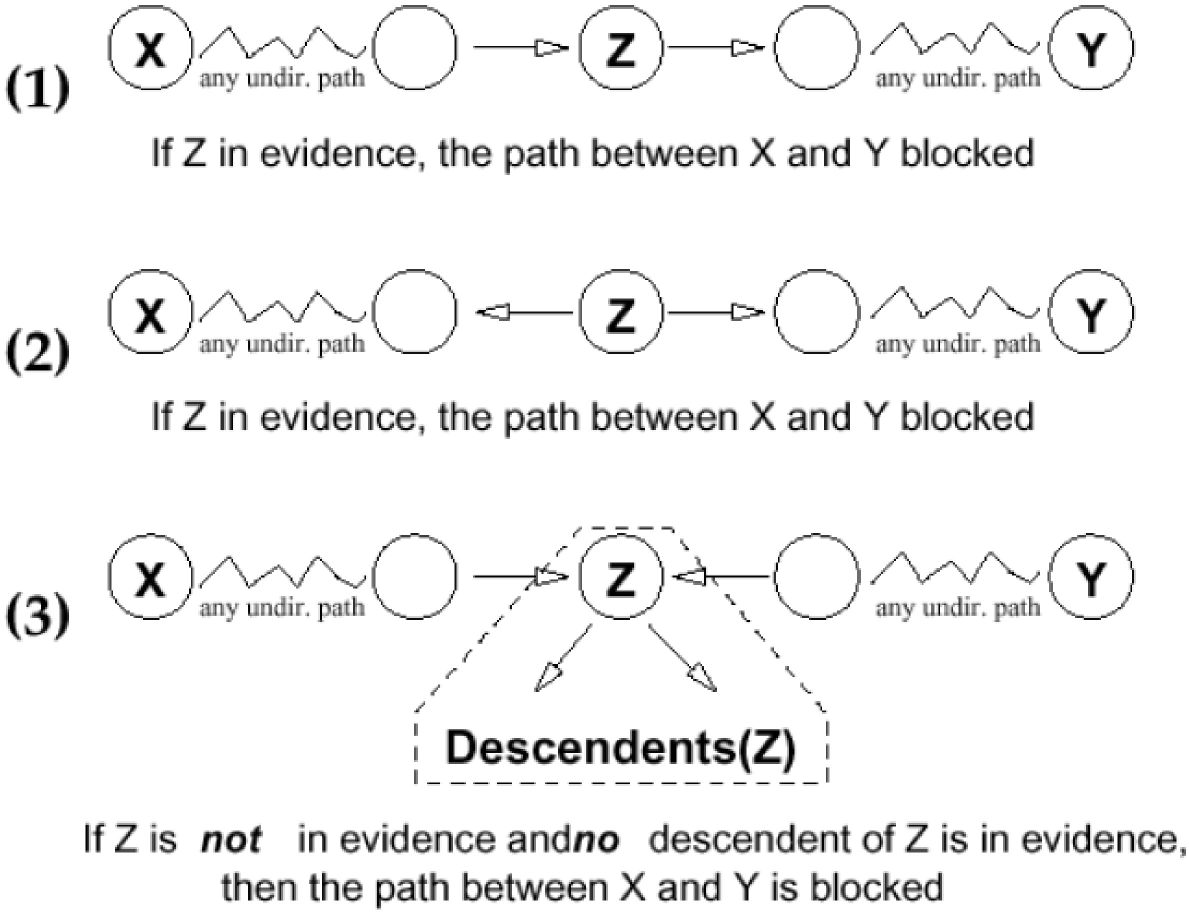
Step 2: Eliminating B: Compute & Add $f_7(A) = \Sigma_B f_6(A,B) f_2(B)$

Remove: $f_6(A,B)$, $f_2(B)$

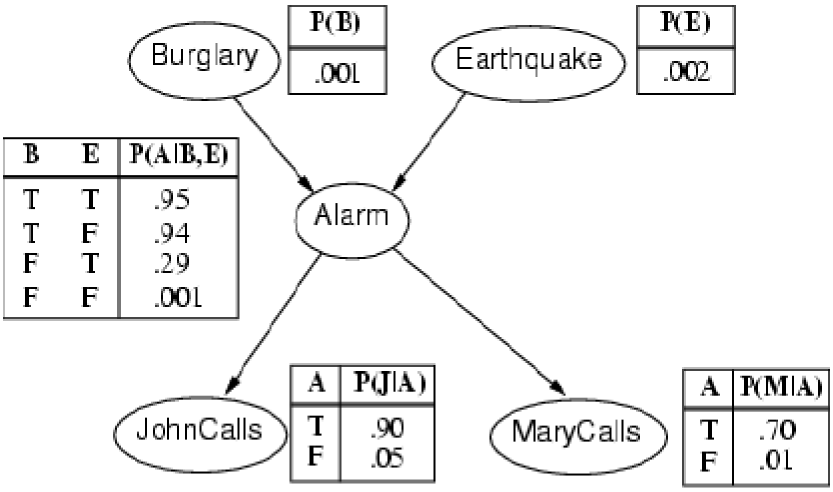
Last factors: $f_7(A)$, $f_1(A)$. The product $f_1(A) \times f_7(A)$ is (unnormalized) posterior. So… $P(A|d) = \alpha f_1(A) \times f_7(A)$

where $\alpha = 1/\sum_A f_1(A) f_7(A)$ ← ****Note the Normalization Constant!****

D-Separation



example:



- A and M are dependent given J
- B and M are independent, given A
- J and M are dependent, but independent given A
- B and E are independent
- B and E are dependent, given A, J, or M

Machine Learning 机器学习

What is ML?

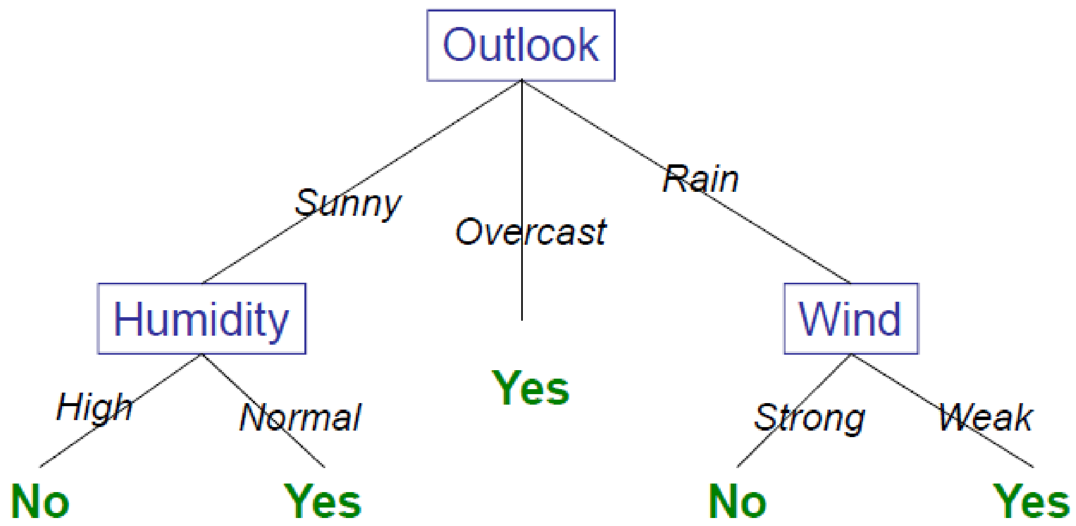
performance in T measured by P improves with E
(experience E, task T, performance measure P)

- Decision-tree learning
- Naive Bayes learning
- K-means and EM
- Chain rule for computing partial derivatives
- Linear and logistic regression
- Backpropagation
- Q-learning

Decision tree 决策树

结点：标识属性
边：标识属性值
叶子：标识输出值

example:



An instance
<Outlook=Sunny, Temp=Hot, Humidity=High, Wind=Strong>

Classification: No

为了构造一个尽量小的树，我们应该优先选择具有代表性的属性

function DECISION-TREE-LEARNING(*examples*, *attributes*, *parent_examples*) **returns** a tree

if *examples* is empty **then return** PLURALITY-VALUE(*parent_examples*)
else if all *examples* have the same classification **then return** the classification
else if *attributes* is empty **then return** PLURALITY-VALUE(*examples*)
else
  $A \leftarrow \operatorname{argmax}_{a \in attributes}$ IMPORTANCE(a, *examples*)
  tree ← a new decision tree with root test A
  for each value v_k of A **do**
    $exs \leftarrow \{e : e \in examples$ **and** $e.A = v_k\}$
    subtree ← DECISION-TREE-LEARNING(*exs*, *attributes* − A, *examples*)
    add a branch to *tree* with label $(A = v_k)$ and subtree *subtree*
  return *tree*

Entropy 熵

$$H(V) = -\sum_k P(v_k) \log_2 P(V_k)$$

The entropy of a Boolean random variable that is true with probability q :

$$B(q) = -(q \log_2 q + (1-q) \log_2(1-q))$$

如果训练集有p个正确n个不正确的例子，则熵为：

$$H(Goal) = B(\frac{p}{p+n})$$

Information gain 信息增益

So the expected entropy remaining after testing attribute A is

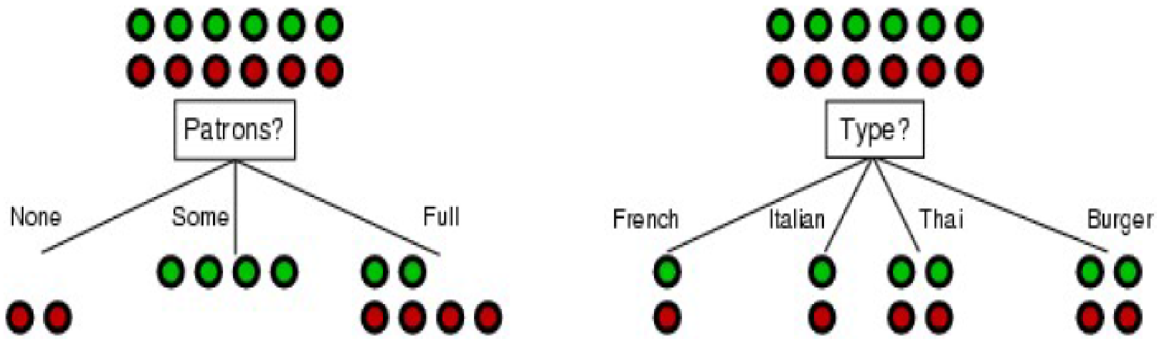
$$Remainder(A) = \sum_{k=1}^{d} \frac{p_k + n_k}{p+n} B(\frac{p_k}{p_k + n_k}).$$

pk / nk: positive/negative examples of the subset

The information gain (IG) from the attribute test on A is the expected reduction in entropy:

$$Gain(A) = B(\frac{p}{p+n}) - Remainder(A)$$

example:



- For the training set, $p = n = 6$, $B(6/12) = 1$
- $Gain(Pat) = 1 - [\frac{2}{12}B(\frac{0}{2}) + \frac{4}{12}B(\frac{4}{4}) + \frac{6}{12}B(\frac{2}{6})] \approx 0.541$
- $Gain(Type) = 1 - [\frac{2}{12}B(\frac{1}{2}) + \frac{2}{12}B(\frac{1}{2}) + \frac{4}{12}B(\frac{2}{4}) + \frac{4}{12}B(\frac{2}{4})] = 0$

Overfit 过度拟合

可以通过剪枝来去除关联度小的结点从而避免过度拟合

Bayes Learning 贝叶斯学习

- Prior: $Pr(H)$
- Likelihood: $Pr(d|H)$
- Evidence: $d = \langle d_1, d_2, \ldots, d_n \rangle$
- Computing the posterior using Bayes'Theorem:

$$Pr(H|d) = \alpha Pr(d|H)Pr(H)$$

$$P(X|d) = \sum_i P(X|d, h_i)P(h_i|d) = \sum_i P(X|h_i)P(h_i|d)$$

Maximum a posteriori （极大后验MAP）

- Idea: make prediction based on most probable hypothesis
  - $h_{\text{MAP}} = \text{argmax}_{h_i} P(h_i|d)$
  - $P(X|d) \approx P(X|h_{\text{MAP}})$

$$h_{\text{MAP}} = \text{argmax}_h P(h)P(d|h)$$

需要考虑各个糖果方案出现的可能性即可得到h_{MAP}

Maximum Likelihood （极大似然ML）

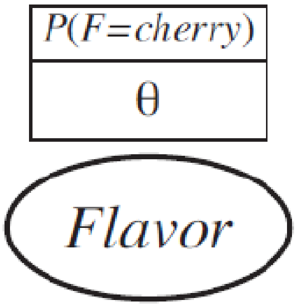
$$h_{\text{ML}} = \text{argmax}_h P(d|h)$$

$$P(X|d) \approx P(X|h_{\text{ML}})$$

无需考虑各个糖果方案出现的可能性即可得到h_{ML}

example:

- Hypothesis h_θ
  - $P(cherry) = \theta$ and $P(lime) = 1 - \theta$
- Data d:
  - c cherries and l limes



- $c/\theta - l/(1-\theta) = 0 \Rightarrow \theta = c/(c+l)$

Articial Neural Networks 神经网络

Loss function:

- Three commonly used loss functions:
  - Absolute value loss: $L_1(y, y') = |y - y'|$
  - Squared error loss: $L_2(y, y') = (y - y')^2$
  - 0/1 loss: $L_{0/1}(y, y') = 0$ if $y = y'$, else 1

Linear regression 线性回归

梯度下降：

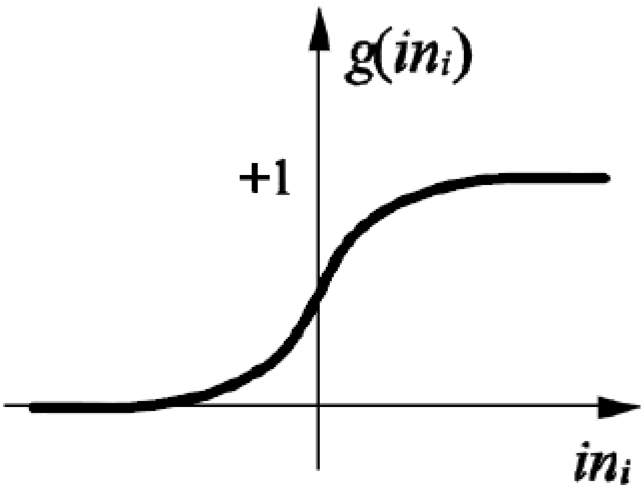
$$w_i \leftarrow w_i - \alpha \partial Loss(w)/\partial w_i$$

α：学习率

- $h_w(x) = w \cdot x = \sum_i w_i x_i$
- Squared error loss: $Loss(w) = (y - h_w(x))^2$
- Chain rule: $\partial g(f(x))/\partial x = g'(f(x)) \partial f(x)/\partial x$
- $\partial Loss(w)/\partial w_i = -2(y - h_w(x)) x_i$
- $w_i \leftarrow w_i + \alpha (y - h_w(x)) x_i$

Logistic regression 逻辑回归

Sigmoid



(b)

$$g(x) = 1/(1+e^{-x})$$

logistic function:

$$g(x) = 1/(1 + e^{-x})$$

- $g(x) = 1/(1 + e^{-x})$
- $h_w(x) = g(w \cdot x)$
- $g' = g(1 - g)$
- $Loss(w) = (y - h_w(x))^2$
- $\partial Loss(w)/\partial w_i = -2(y - h_w(x))g'(w \cdot x)x_i$
  $= -2(y - h_w(x))h_w(x)(1 - h_w(x))x_i$
- $w_i \leftarrow w_i + \alpha(y - h_w(x))h_w(x)(1 - h_w(x))x_i$

initialize w arbitrarily
repeat
for each e in examples do
$p \leftarrow g(w \cdot x(e))$
$\delta \leftarrow y(e) - p$
for each i do
$w_i \leftarrow w_i + \alpha\delta p(1-p)x_i$
until some stopping criterion is satisfied
return w

Forward and backward phases

Forward phase:

Output a_j at unit j: $a_j = g(in_j)$ where $in_j = \sum_i w_{ij}a_i$

$in = \Sigma w_i a_i$ $out = g(in) = 1/1 + e^{-in}$

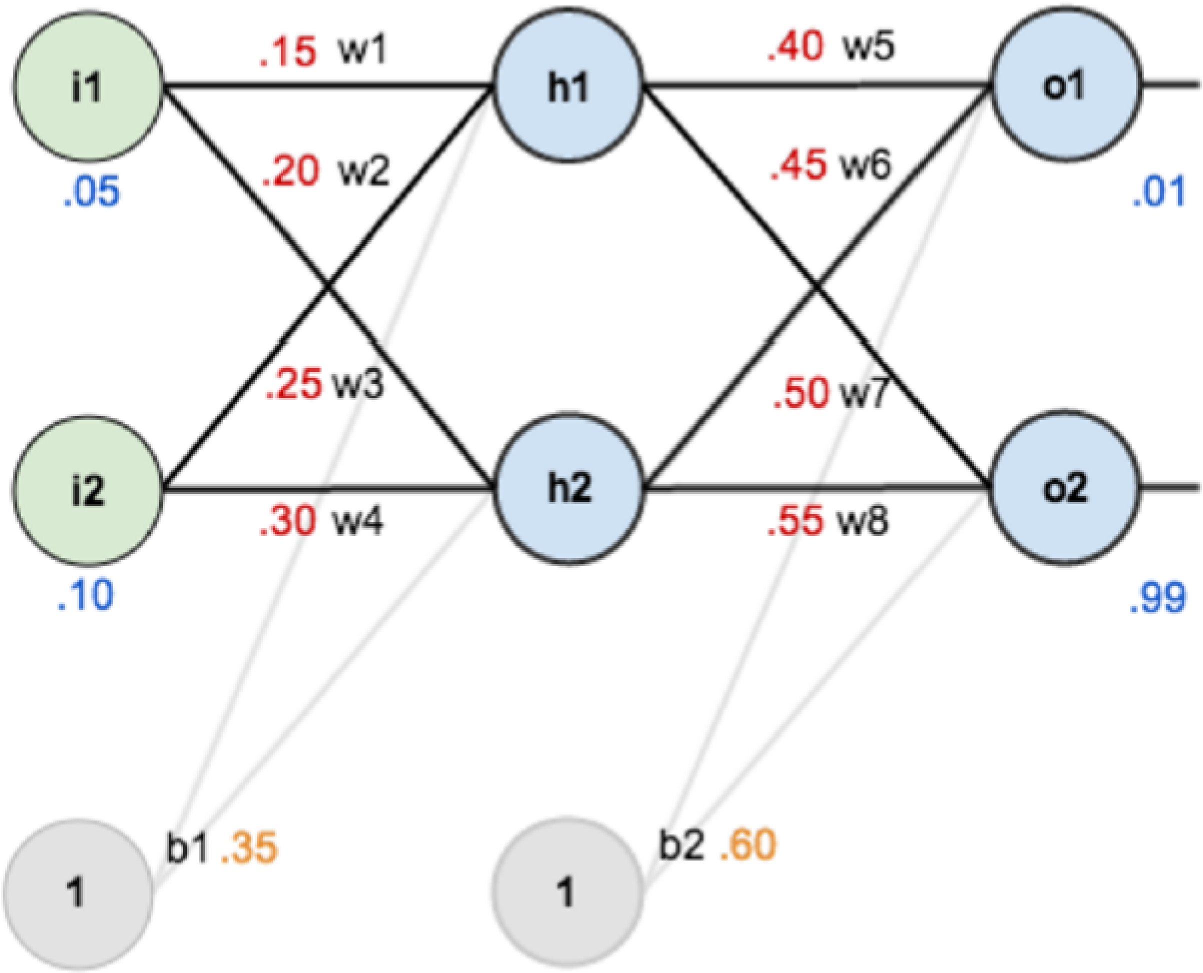
Backward phase:

- For an output unit j:
  $\Delta_j = g'(in_j)(y_j - a_j) = a_j(1-a_j)(y_j - a_j)$
- For an hidden unit i:
  $\Delta_i = g'(in_i)\sum_j w_{ij}\Delta_j = a_i(1-a_i)\sum_j w_{ij}\Delta_j$

Weight updating: $w_{ij} \leftarrow w_{ij} + \alpha a_i \delta_j$

$\Delta o = o(1-o)(y-o)$
$w^+ = w - \alpha * out * \Delta o$
$\Delta h = out(1-out)\Sigma w_i \Delta_o$

example:



- $in_{h_1} = w_1 i_1 + w_2 i_2 + b_1 = 0.05 * 0.15 + 0.10 * 0.20 + 0.35 = 0.3775$
- $out_{h_1} = g(in_{h_1}) = \frac{1}{1+e^{-0.3775}} = 0.593269992$
- $out_{h_2} = 0.596884378$
- $in_{o_1} = w_5 out_{h_1} + w_6 out_{h_2} + b_2 = 0.40 * 0.593269992 + 0.45 * 0.596884378 + 0.60 = 1.105905967$
- $out_{o_1} = g(in_{o_1}) = \frac{1}{1+e^{-1.105905967}} = 0.75136507$
- $out_{o_2} = 0.772928465$

Let $\alpha = 0.5$

- $\Delta_{o_1} = 0.75136507(1 - 0.75136507)(0.01 - 0.75136507) = -0.138498562$

- $w_5^+ = w_5 + \alpha \cdot out_{h_1} \cdot \Delta_{o_1} =$
  $0.40 - 0.5 * 0.593269992 * 0.138498562 = 0.35891648$

- $w_6^+ = w_6 + \alpha \cdot out_{h_2} \cdot \Delta_{o_1} =$
  $0.45 - 0.5 * 0.596884378 * 0.138498562 = 0.408666186$

- $\Delta_{o_2} = 0.772928465(1 - 0.772928465)(0.99 - 0.772928465) =$
  0.0380982366
- $w_7^+ = w_7 + \alpha \cdot out_{h_1} \cdot \Delta_{o_2} =$
  $0.50 + 0.5 * 0.593269992 * 0.0380982366 = 0.511301270$
- $w_8^+ = w_8 + \alpha \cdot out_{h_2} \cdot \Delta_{o_2} =$
  $0.55 + 0.5 * 0.596884378 * 0.0380982366 = 0.561370121$
- $\Delta_{h_1} = g'(in_{h_1})(w_5\Delta_{o_1} + w_7\Delta_{o_2}) =$
  $0.593269992(1 - 0.593269992)(0.40 * (-0.138498562) +$
  $0.50 * 0.0380982366) = -0.241300709 * 0.036350306$
- $w_1^+ = w_1 + \alpha \cdot i_1 \cdot \Delta_{h_1} =$
  $0.15 - 0.5 * 0.05 * 0.241300709 * 0.036350306 = 0.149780716$
- $w_2^+ = 0.19956143$
- $w_3^+ = 0.24975114$
- $w_4^+ = 0.29950229$

[回到顶部]