

CS 486



Personal Notes

Marcus Chan

Taught by Pascal Poupart & Sriram Ganapathi
Subramanian

JW CS '25



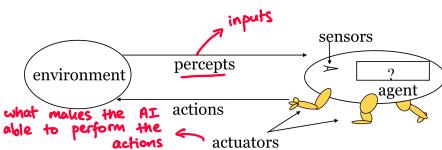
Chapter 1: Introduction

REINFORCEMENT LEARNING PROBLEM



Our goal is for the AI to learn to choose actions that maximize rewards.

AGENTS & ENVIRONMENTS



The "agent function" maps percepts to actions; ie $f: P \rightarrow A$.

The "agent program" runs on the physical architecture to produce f .

RATIONAL AGENTS

A "rational agent" chooses whichever action that maximizes the expected value of its performance measure given the percept sequence to date.

Note that rationality is not omniscience, but rather learning & autonomy.

PEAS

"PEAS" helps us specify the task environment:

- ① Performance measure;
eg safety, destination, etc
- ② Environment;
eg streets, traffic, etc
- ③ Actuators;
eg steering, brakes, etc.
- ④ Sensors;
eg GPS, engine sensors, etc.

PROPERTIES OF TASK ENVIRONMENTS

Task environments can be:

- ① fully vs partially observable:
 - fully observable: agent knows state of the world from the stimuli
 - partially observable: agent does not directly observe the world's state

② deterministic vs stochastic;

- deterministic: next state is observable at any time
- stochastic: next state is unpredictable

③ episodic vs sequential;

- episodic: agent's current action will not affect a future action
- sequential: agent's current action will affect a future action

④ static vs dynamic;

- static: model is trained once
- dynamic: model is trained continuously

⑤ discrete vs continuous;

⑥ single agent vs multiagent.

*the former option is "easier" than the latter.

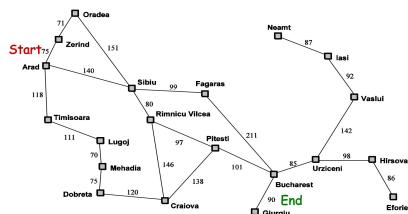
Chapter 2: Uninformed Search Techniques

SIMPLE PROBLEM SOLVING AGENT

```
function SIMPLE-PROBLEM-SOLVING-AGENT(percept) returns an action
  static: seq, an action sequence, initially empty
    state, some description of the current world state
    goal, a goal, initially null
    problem, a problem formulation
  state ← UPDATE-STATE(state, percept)
  if seq is empty then do
    goal ← FORMULATE-GOAL(state)
    problem ← FORMULATE-PROBLEM(state, goal)
    seq ← SEARCH(problem)
  action ← FIRST(seq)
  seq ← REST(seq)
  return action
```

- 💡 This can only tackle problems that are
 - ① fully observable;
 - ② deterministic;
 - ③ sequential;
 - ④ static;
 - ⑤ discrete; &
 - ⑥ single agent.

EXAMPLE: TRAVELLING IN ROMANIA



- initial state: In Arad
- actions: drive between cities
- goal test: In Bucharest
- path cost: distance between cities

EXAMPLE: 8-TILE PUZZLE



Start State



Goal State

- states: locations of 8 tiles & blank
- initial state: any state
- actions: up, down, left, right
- goal test: does state match desired configuration
- path cost: # of steps

SEARCHING

- 💡 we can visualize a state space search in terms of trees or graphs:
 - ① nodes correspond to states; &
 - ② edges correspond to taking actions.
- 💡 These "search trees" are formed using "search nodes", which have
 - ① the state associated with it;
 - ② parent node & operator applied to the parent to reach the "current" node;
 - ③ cost of the path so far; &
 - ④ depth of the node.

EXPANDING NODES

- 💡 "Expanding a node" refers to applying all legal operators to the state contained in the node & generating nodes for all corresponding successor states.



GENERIC SEARCH ALGORITHM

- 💡 Algorithm:
 - ① Initialize search with initial problem state.
 - ② Then repeat:
 - if no candidate nodes can be expanded, return failure
 - otherwise, choose a leaf node for expansion according to our search strategy.
 - if the node contains a goal state, return the solution.
 - otherwise, expand the node by applying the legal operators to the state associated within the node, & add the resulting nodes to the tree.

EVALUATING SEARCH ALGORITHMS

We can use the following properties when evaluating search algorithms:

- ① "completeness" — is the algorithm guaranteed to find a solution (if it exists)?
 - ② "optimality" — does the algorithm find the optimal solution (ie lowest path cost)?
 - ③ time & space complexity.
- We consider the following variables:
- ① "branching factor" (b) — the # of children each node has
 - ② depth of shallowest goal node (d); &
 - ③ max length of any path in the state space (m).

BREADTH-FIRST SEARCH

Refer to CS341 notes for details;
we expand all nodes on a given level before any node on the next level is expanded.

Evaluating the algorithm:

- ① Completeness: yes if $b \geq 0$
- ② Optimality: yes if all costs are same
- ③ Time: $1+b+\dots+b^d \in O(b^d)$
- ④ Space: $O(b^d)$.

* all uninformed search methods have exponential time complexity.

UNIFORM COST SEARCH

Idea: we expand the node with the lowest path cost.

We can implement this using a priority queue.

Let C^* = cost of optimal solution & $\epsilon = \min \text{action cost}$. Then

- ① Completeness: yes if $\epsilon > 0$
- ② Optimality: yes
- ③ Time: $O(b^{C^*/\epsilon})$
- ④ Space: $O(b^{C^*/\epsilon})$

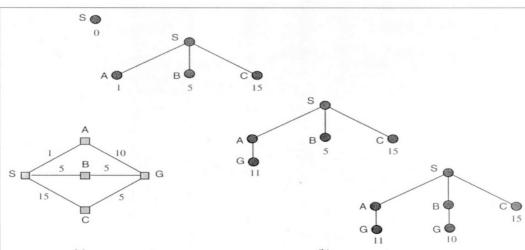


Figure 3.13 A route-finding problem. (a) The state space, showing the cost for each operator. (b) Progression of the search. Each node is labelled with $g(n)$. At the next step, the goal node with $g = 10$ will be selected.

DEPTH-FIRST SEARCH

Refer to CS341 notes for details;
we expand the deepest node in the current fringe of the search tree first.

Evaluation:

- ① Complete: no
- may get stuck going down a long path
- ② Optimal: no
- might return a solution which is deeper (ie more costly) than another solution
- ③ Time: $O(b^m)$
- note we might have $m > d$
- ④ Space: $O(bm)$

DEPTH-LIMITED SEARCH

Idea: Treat all nodes at depth l as if they have no successors.
- try to choose l based on the problem

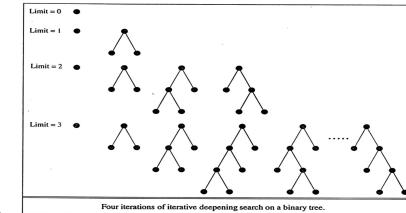
This avoids the problem of unbounded trees.

Evaluation:

- ① Time: $O(b^l)$
- ② Space: $O(b^l)$
- ③ Complete: no
- ④ Optimal: no

ITERATIVE-DEEPENING

Idea: repeatedly perform depth-limited search, but increase the limit each time.



Evaluation:

- ① Complete: yes
- ② Optimal: yes
- ③ Time: $O(b^d)$
- ④ Space: $O(bd)$

Time:

$$\begin{aligned}
 \text{(limit=1)} & \quad 1 \\
 \text{(limit=2)} & \quad 1 + b \\
 \text{(limit=3)} & \quad 1 + b + b^2 \\
 & \vdots \\
 \text{(+) (limit=d)} & \quad 1 + b + b^2 + \dots + b^d \\
 & \quad d + (d-1)b + (d-2)b^2 + \dots \\
 & \quad + b^d \in O(b^d)
 \end{aligned}$$

Chapter 3: Informed Search Techniques

MOTIVATION

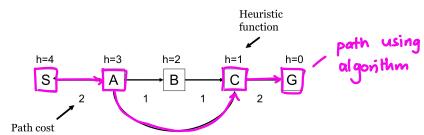
- Q₁: In search problems, we often have additional knowledge about the problem.
eg with the "travelling around Romania", we know dist. bw cities
↳ so we can find the overhead in going the wrong direction
- Q₂: Our knowledge is often about the "merit" of nodes.
- Q₃: Notions of merit:
 - ① how expensive it is to get from a state to a goal;
 - ② how easy it is to get from a state to a goal.

HEURISTIC FUNCTIONS • $h(n)$

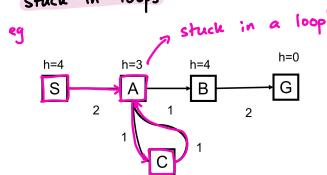
- Q₁: We need to develop domain specific "heuristic functions" $h(n)$, which guess the cost of reaching the goal from node n .
- Q₂: In general, if $h(n_1) < h(n_2)$, we guess reaching the goal is cheaper from n_1 than from n_2 . We also need
 - ① $h(n) = 0$ if n is a goal node
 - ② $h(n) > 0$ if n is not a goal node.

GREEDY BEST-FIRST SEARCH

- Q₁: Idea: Use $h(n)$ to rank the nodes in the fringe & expand the node with the lowest h -value.
(ie "greedily" trying to find the least-cost solution).
- Q₂: Example:



- Q₃: Note greedy best-first is not optimal.
eg in the above example,
 - path found has cost=6.
 - but cheaper path is $S \rightarrow A \rightarrow B \rightarrow C \rightarrow G$ with cost=6.
- Q₄: It is also not complete, as it can be stuck in loops.



- eg - but if we check for repeated states then we are okay
- Q₅: This algorithm uses exponential space & worst-case time.

ITERATIVE DEEPENING A*

(IDA*)

- Idea: Like iterative deepening search, but change f-cost rather than depth in each iteration.
- This reduces the space complexity.

SIMPLIFIED MEMORY-BOUNDED A*

(SMA*)

- Idea: Proceeds like A* but when it runs out of memory it drops the worst leaf node (ie one with highest f-value).
- If all leaf nodes have the same f-value, then drop the oldest & expand the newest.
- This is
 - ① optimal;
 - ② complete if depth of shallowest goal node < memory size.

OBTAINING HEURISTICS

- One approach to get heuristics is to think of an easier problem & let $h(n)$ be the cost of reaching the goal in the easier problem.

We can also

- ① precompute solution costs of subproblems & store them in a pattern database; or
- ② learn from experience with the problem class.

EXAMPLE: 8-PUZZLE GAME

We can relax the game in 3 ways:

- ① We can move tile from position A \rightarrow B if A is next to B (ignore whether position is blank)
- ② We can move tile from position A \rightarrow B if B is blank (ignore adjacency)
- ③ We can move tile from position A \rightarrow B regardless.

③ leads to the "misplaced tile heuristic". (h_3)

- to solve this problem we need to move each tile into its final position.
- # of moves = # of misplaced tiles
- admissible

① leads to the "manhattan distance heuristic". (h_1)

- to solve this we need to slide each tile into its final position

- admissible

Note h_1 "dominates" h_3 ; ie $h_3(n) \leq h_1(n) \forall n$.

Chapter 4: Constraint Satisfaction

INTRODUCTION

Q: These are useful for problems where the state structure is important.

Q: In many problems, the same state can be reached independent of the order in which the moves are chosen.

Q: So, we can try to solve problems efficiently by being smart about the action order.

4-QUEENS CONSTRAINT PROPAGATION

Q: Idea: Remove conflicting squares from consideration when we put a queen down.



CONSTRAINT SATISFACTION PROBLEM (CSP)

Q: A "constraint satisfaction problem" is defined by some $\{V, D, C\}$, where

① $V = \{V_1, \dots, V_n\}$ is a set of variables;

② $D = \{D_1, \dots, D_n\}$ is a set of domains, where D_i is the set of possible values for each V_i ;

&

③ $C = \{C_1, \dots, C_m\}$ is the set of constraints.

STATE

Q: A "state" is an assignment of values to some or all of the variables;

ie $V_i = x_i, V_j = x_j, \dots$.

CONSISTENT [ASSIGNMENT]

Q: We say an assignment is "consistent" if it does not violate any constraints.

SOLUTION

Q: A "solution" is a complete, consistent assignment.

EXAMPLE: 8 QUEENS AS A CSP

Q: 8 queens as a CSP:

- variables: $V_{ij}, i, j = 1, \dots, 8$

- domain of each var: $\{0, 1\}$

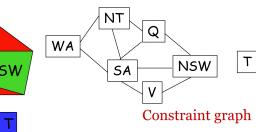
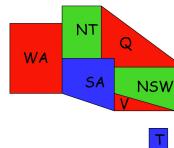
- constraints: $V_{ij} = 1 \Rightarrow V_{ik} = 0 \quad \forall k \neq j$

$V_{ij} = 1 \Rightarrow V_{kj} = 0 \quad \forall k \neq i$

similar constraint for diagonals

$$\sum_{i,j} V_{ij} = 8$$

EXAMPLE: MAP COLORING



Constraint graph

MAP COLORING AS A CSP

- variables: WA, NT, ..., T (the regions)

- each var has the same domain: {red, green, blue}

- no 2 adjacent variables have the same value

(ie $WA \neq NT$, $WA \neq SA$, etc)

PROPERTIES OF CSPS

Q: Types of variables:

① Discrete & finite;

eg 8-queens, map coloring

* we focus on this in this course.

② Discrete with infinite domains; &

eg job scheduling

③ Continuous domains.

Q: Types of constraints:

① "Unary constraint": relates a single variable to a value

- eg Queensland = blue

② "Binary constraint": relates two variables

③ "Higher order constraints": relates ≥ 3 variables.

CSPs & SEARCH

Q: We can formulate CSPs as a search problem:

① we have N variables V_1, \dots, V_n ;

② a valid assignment is $\{V_i = x_i, \dots, V_n = x_n\}$, $0 \leq i \leq n$ where values satisfy the variable constraints.

③ states: valid assignments

④ initial state: empty assignment

⑤ successor: $\{V_i = x_i, \dots, V_n = x_n\} \rightarrow \{V_i = x_i, \dots, V_n = x_n, V_{n+1} = x_{n+1}\}$

⑥ goal test: complete assignment

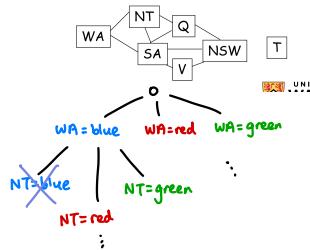
BACKTRACKING SEARCH

```

function BACKTRACKING-SEARCH(csp) returns a solution, or failure
    return RECURSIVE-BACKTRACKING({}, csp)
function RECURSIVE-BACKTRACKING(assignment, csp) returns a solution, or failure
    if assignment is complete then return assignment
    var  $\leftarrow$  SELECT-UNASSIGNED-VARIABLE(Variables[csp], assignment, csp)
    for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
        if value is consistent with assignment according to Constraints[csp] then
            add { var = value } to assignment
            result  $\leftarrow$  RECURSIVE-BACKTRACKING(assignment, csp)
            if result  $\neq$  failure then return result
            remove { var = value } from assignment
    return failure
  
```

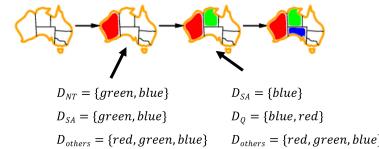
- this is DFS that choose values for one variable at a time
- we "backtrack" when a variable has no legal values to assign

EXAMPLE: MAP COLORING



MOST CONSTRAINED VARIABLE HEURISTIC

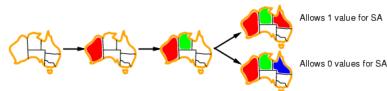
Q: Idea: Choose the variable which has the fewest "legal" moves.



Q: In a tie, choose the variable with the most constraints on the remaining variables.
(ie "most constraining variable").

LEAST CONSTRAINING VALUE HEURISTIC

Q: Idea: Given a variable, choose the "least constraining value", ie the one that rules out the fewest values in the remaining variables.

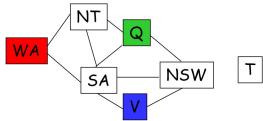


FORWARD CHECKING

Q₁: Idea: We keep track of remaining legal values for unassigned variables, & terminate search when any variable has no legal values.

Q₂: This helps us detect failure early.

EXAMPLE: MAP COLORING



WA	NT	Q	NSW	V	SA	T
RGB						
R	GB	RGB	RGB	RGB	GB	RGB
R	B	G	RB	RGB	B	RGB
R	B	G	RB	B	X	RGB

$$\begin{array}{l} \text{WA} = R \\ \text{Q} = G \\ \text{V} = B \end{array}$$

this is the empty set;
⇒ the current assignment does not lead to a solution.

Chapter 5: Uncertainty

Q₁: Refer to STAT 231/330 for more details.

Q₂: We use \sim to denote the complement of an event (ie $\sim A$).

BAYES RULE

Q₁: For 2 events A, B, note

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

Proof: $P(A)P(B|A) = P(A \wedge B) = P(B)P(A|B)$.

$$\therefore P(B|A) = \frac{P(B)P(A|B)}{P(A)}.$$

Q₂: In particular, it allows us to compute a belief about hypothesis B given evidence A.

Q₃: More general forms:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B|A)P(A) + P(B|\sim A)P(\sim A)}$$

$$P(A|B \wedge X) = \frac{P(B|A \wedge X)P(A|X)}{P(B|X)}$$

$$P(A=v_i|B) = \frac{P(B|A=v_i)P(A=v_i)}{\sum_{k=1}^n P(B|A=v_k)P(A=v_k)}$$

PROBABILISTIC INFERENCE

Q₁: Idea: Given a prior distribution $P(X)$ over variables X of interest & given new evidence $E=e$ for some variable E , revise our degrees of belief; ie the "posterior" $P(X|E=e)$.

ISSUES

Q₁: Specifying the full joint distribution for X_1, \dots, X_n requires an exponential number of possible "worlds".

Q₂: So, inference is also slow since we need to sum over these exponential number of worlds:

$$P(X_1, \dots, X_{i-1}, X_{i+1}, \dots, X_n | X_i) = \frac{P(X_1, \dots, X_n)}{\sum_{x_1} \sum_{x_{i-1}} \sum_{x_{i+1}} \dots \sum_{x_n} P(X_1, \dots, X_n)}$$

CONDITIONAL INDEPENDENCE

Q₁: Two variables X, Y are "conditionally independent" given Z if

$$P(X=x | Z=z) = P(X=x | Y=y, Z=z)$$

$$\Leftrightarrow P(X=x, Y=y | Z=z) = P(X=x | Z=z)P(Y=y | Z=z)$$

$$\Leftrightarrow \forall x \in \text{dom}(X), y \in \text{dom}(Y), z \in \text{dom}(Z)$$

Q₂: If we know the value of Z , nothing we learn about Y will influence our beliefs about X .

VALUE OF INDEPENDENCE

Q₁: If X_1, \dots, X_n are mutually independent, then we can specify the full joint distribution using only n parameters (ie linear) instead of $2^n - 1$ (ie exponential).

Q₂: Although most domains do not exhibit complete mutual independence, they do instead exhibit a fair amount of conditional independence.

NOTATION: $P(X)$

Q₁: We define " $P(X)$ " as the marginal distribution over X .

- $P(X=x)$ is a number, $P(X)$ is a distribution.

NOTATION: $P(X|Y)$

Q₁: We define " $P(X|Y)$ " as the family of conditional distributions over X ; one for each $y \in \text{dom}(Y)$.

EXPLOITING CONDITIONAL INDEPENDENCE: CHAIN RULE

Consider a story:

- If Pascal woke up too early E , Pascal probably needs coffee C ; if Pascal needs coffee, he's likely grumpy G . If he is grumpy then it's possible that the lecture won't go smoothly L . If the lecture does not go smoothly then the students will likely be sad S .



E - Pascal woke up too early G - Pascal is grumpy S - Students are sad
 C - Pascal needs coffee L - The lecture did not go smoothly

S is independent of E, C, G given L
 L is independent of E, C given G & so on.

$$\Rightarrow P(S|L, G, C, E) = P(S|L)$$

$$P(L|G, C, E) = P(L|G)$$

$$P(G|C, E) = P(G|C)$$

Then

$$\begin{aligned} P(S, L, G, C, E) &= P(S|L, G, C, E) P(L|G, C, E) P(G|C, E) \cdot \\ &\quad P(C|E) P(E) \\ &= \underline{P(S|L) P(L|G) P(G|C) P(C|E) P(E)}. \end{aligned}$$

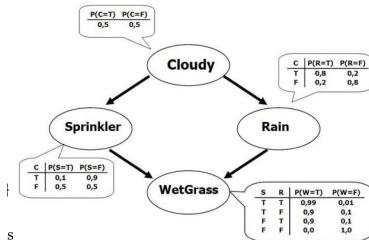
Q: In this, we can specify the full joint distribution by specifying the five local conditional distributions.

Chapter 6:

Bayesian Networks

BAYESIAN / BELIEF / PROBABILISTIC NETWORKS (BN)

B1 "Bayesian networks" are graphical representations of the direct dependencies over a set of variables, alongside a set of conditional probability tables (CPT) quantifying the strength of the influences.



B2 In particular, it has

- ① A DAG with nodes = variables X_i , &
- ② A set of CPTs $P(X_i | \text{Parents}(X_i))$ for each X_i .

B3 Key notions:

- ① parents/children of a node;
- ② ancestors/descendants of a node; &
- ③ family: set of nodes consisting of X_i & its parents.

SEMANTICS OF A BAYES NET

B1 Idea: Every X_i is conditionally independent of all its non-descendants given its parents; ie

$$P(X_i | S \cup \text{Par}(X_i)) = P(X_i | \text{Par}(X_i))$$

$\forall S \subseteq \text{NonDescendants}(X_i)$

B2 Also, the joint distribution is recoverable using the parameters (CPTs) in the BN:

$$\begin{aligned} P(x_1, \dots, x_n) &= P(x_n | x_{n-1}, \dots, x_1) P(x_{n-1} | x_{n-2}, \dots, x_1) \dots P(x_1) \\ &= P(x_n | \text{Par}(x_n)) P(x_{n-1} | \text{Par}(x_{n-1})) \dots P(x_1). \end{aligned}$$

CONSTRUCTING A BN

B1 Idea:

- ① Take any ordering of the variables, and then for X_n to X_1 :
 - let $\text{Par}(X_n)$ be any subset $S \subseteq \{X_1, \dots, X_{n-1}\}$ such that X_n is independent of $\{X_1, \dots, X_{n-1}\} - S$ given S .
 - Continue this for X_{n-1}, \dots, X_1 .

② In the end, we get a DAG, which is also a BN by construction.

B2 Note the order in which we consider the variables changes the resultant BN!

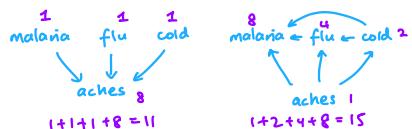
eg
order: mal, cold, flu, aches
malaria → flu → cold
aches ← aches ← aches

order: aches, cold, flu, malaria
malaria ← flu ← cold
aches ← aches ← aches

COMPACTNESS

B1 In a BN, if each rv is directly influenced by at most k others, then each CPT will have at most 2^k entries.

B2 So, the entire network of n variables is specified by $n \cdot 2^k$ parameters.



d-SEPARATION

- Q: First, we say a set of variables E "d-separates" X & Y if it "blocks" every undirected path in the BN between X & Y.

TESTING INDEPENDENCE

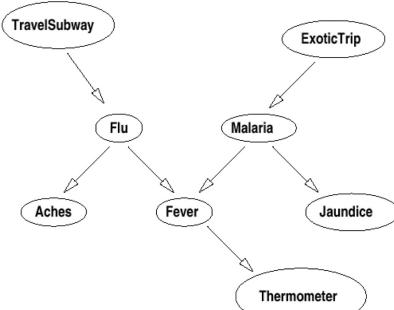
- Q: Then, X & Y are conditionally independent given evidence E if E d-separates X & Y.

BLOCKING IN d-SEPARATION

- Q: Let P be an undirected path from X → Y. Then the evidence set E "blocks" path P if
- ① one arc on P goes into Z & one goes out of Z, & Z ∉ E;
 - ② both arcs on P leave Z & Z ∉ E; or
 - ③ both arcs on P enter Z & neither Z nor any of its descendants are in E.
- $X \rightsquigarrow Z \rightsquigarrow Y$
- $X \leftarrowtail Z \rightarrowtail Y$
- $X \rightsquigarrow Z \rightsquigarrow Y$



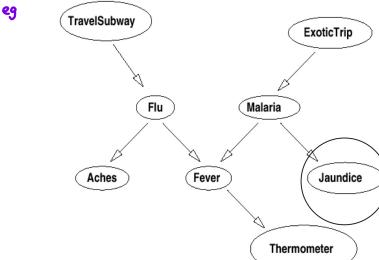
EXAMPLE



- ① subway & thermometer
 - dependent
 - but independent given the flu
 - ↳ since flu blocks the only path (rule 1)
- ② aches & fever
 - dependent
 - but independent given the flu
 - ↳ since flu blocks the only path (rule 2)
- ③ flu & malaria
 - independent
 - dependent given fever or thermometer
 - ↳ rule 3
- ④ subway & exotic trip
 - independent
 - dependent given thermometer
 - ↳ rule 3

SIMPLE FORWARD INFERENCE (CHAIN)

- Q: Idea: To compute the marginal distribution, we can use simple forward "propagation" of probabilities.



$$\begin{aligned}
 P(J) &= \sum_{M, ET} P(J, M, ET) \quad (\text{marginalization}) \\
 &= \sum_{M, ET} P(J|M, ET) P(M|ET) P(ET) \\
 &\quad (\text{chain rule}) \\
 &= \sum_{M, ET} P(J|M) P(M|ET) P(ET) \\
 &\quad (\text{conditional independence}) \\
 &= \sum_M P(J|M) \sum_{ET} P(M|ET) P(ET)
 \end{aligned}$$

all these terms can now be found in the CPTs.

- Q: We can do something similar if we have "upstream" evidence.

$$\begin{aligned}
 \text{eg } P(J|ET) &= \sum_m P(J, m|ET) \\
 &= \sum_m P(J|m, ET) P(m|ET) \\
 &= \sum_m P(J|m) P(m|ET)
 \end{aligned}$$

terms found in CPTs

SIMPLE BACKWARD INFERENCE

- Q: Idea: For "downstream" evidence, we must reason backwards, which we can use Bayes' rule:

$$\begin{aligned}
 \text{eg (using same BN as above)} \quad P(LET|J) &= \alpha P(J|ET) P(ET), \quad \alpha = \frac{1}{P(J)} \\
 &= \alpha \sum_m P(J, m|ET) P(ET) \\
 &= \alpha \sum_m P(J|m, ET) P(m|ET) P(ET) \\
 &= \alpha \sum_m P(J|m) P(m|ET) P(ET).
 \end{aligned}$$

We can then calculate $P(J) = \sum_{ET} P(J|ET) P(ET)$.

VARIABLE ELIMINATION

The "variable elimination" algorithm is a general inference tool for BNs.

FACTORS

A "factor" is a function $f(X_1, \dots, X_k)$.

We can represent factors as a table of numbers, one for each instantiation of the variables X_1, \dots, X_k .

We denote $f(X, Y)$ to be a factor over the variables $X \cup Y$, where X & Y are sets of variables.

Note each CPT in a Bayes net is a factor of its family.

eg $P(C|A, B) \rightarrow$ factor of A, B, C .

PRODUCT OF FACTORS: fg

Let $f(X, Y), g(Y, Z)$ be factors with variables Y in common.

Then the "product" of f & g , $h=fg$, is defined to be

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

eg

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

SUM VARIABLE OUT OF A FACTOR: $\sum_x f$

Let $f(X, Y)$ be a factor, where X is a variable & Y is a variable set.

Then, we can "sum out" variable X from f to produce a new factor $h = \sum_x f$, where

$$h(Y) = \sum_{x \in \text{Dom}(X)} f(x, Y).$$

eg

	$f(A, B)$	$h(B)$
ab 0.9	b 1.3	
a~b 0.1	~b 0.7	
~ab 0.4		
~a~b 0.6		

RESTRICTING FACTORS: $f|_{X=x}$

Let $f(X, Y)$ be a factor with variable X & variable set Y .

Then, we "restrict" factor f to $X=x$, ie $h=f|_{X=x}$, by doing

$$h(Y) = f(x, Y).$$

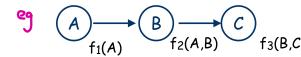
eg

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

NO EVIDENCE CASE

Idea: Computing prior probability of the query variable X can be seen as applying these operations on factors.

EXAMPLE 1

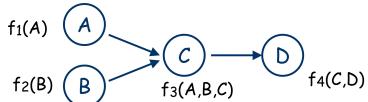


$$\begin{aligned} P(C) &= \sum_{A,B} P(C|B) P(B|A) P(A) \\ &= \sum_B P(C|B) \sum_A P(B|A) P(A) \\ &= \sum_B f_3(B, C) \sum_A f_2(A, B) f_1(A) \\ &\quad \text{---} \\ &= \sum_B f_3(B, C) f_4(B) \\ &\quad \text{---} \\ &= f_5(C). \end{aligned}$$

Numerical example:

	$f_1(A)$	$f_2(A, B)$	$f_3(B, C)$	$f_4(B)$	$f_5(C)$
a 0.9	ab 0.9	bc 0.7	b 0.85	c 0.625	
a~b 0.1	a~b 0.1	b~c 0.3	~b 0.15	~c 0.375	
~ab 0.4		~bc 0.2			
~a~b 0.6		~b~c 0.8			

EXAMPLE 2



eg $P(D) = \sum_{A,B,C} P(D|C) P(C|B,A) P(B) P(A)$

 $= \sum_C P(D|C) \sum_B P(B) \sum_A P(C|B,A) P(A)$
 $= \sum_C f_4(C,D) \sum_B f_2(B) \sum_A f_3(A,B,C) f_1(A)$
 $= \sum_C f_4(C,D) \sum_B f_2(B) f_5(B,C)$
 $= \sum_C f_4(C,D) f_6(C)$
 $= f_7(D)$

*define f_5, f_6, f_7
according to the
brackets

ALGORITHM (NO EVIDENCE)

Input: query var Q , remaining vars Z , & $F = \text{set of factors corresponding to CPTs}$ for $\{Q\} \cup Z$.

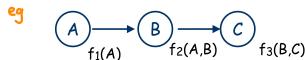
- Choose an elimination ordering Z_1, \dots, Z_n of variables in Z .
- For each Z_j -- in the order given -- eliminate $Z_j \in Z$ as follows:
 - Compute new factor $g_j = \sum_{Z_j} f_1 \times f_2 \times \dots \times f_k$, where the f_i are the factors in F that include Z_j
 - Remove the factors f_i (that mention Z_j) from F and add new factor g_j to F
- The remaining factors refer only to the query variable Q . Take their product and normalize to produce $P(Q)$

eg query: $P(D)$
elim. order: A, B, C

Steps:

- add $f_5(B,C) = \sum_A f_3(A,B,C) f_1(A)$;
remove $f_1(A), f_3(A,B,C)$
- add $f_6(C) = \sum_B f_2(B) f_5(B,C)$
- we don't need to sumout f_3 as we removed it in step ①
remove $f_2(B), f_5(B,C)$
- add $f_7(D) = \sum_C f_4(C,D) f_6(C)$
remove $f_4(C,D), f_6(C)$
- The remaining factor $f_7(D)$ is our (possibly unnormalized) probability $P(D)$

EVIDENCE CASE



eg $P(A|C=c) = \alpha P(A) P(C=c|B) P(B|A)$ (Bayes' thm)

 $= \alpha P(A) \sum_B P(C=c|B) P(B|A)$
 $= \alpha f_1(A) \sum_B f_3(B,c) f_2(A,B)$
 $= \alpha f_1(A) \sum_B f_4(B) f_2(A,B)$
 $= \alpha f_1(A) f_5(A)$
 $= f_6(A)$

ALGORITHM (WITH EVIDENCE)

Input: Given query var Q , evidence vars E (observed to be e), remaining vars Z & set of factors involving CPTs for $\{Q\} \cup Z$, F :

- Replace each factor $f \in F$ that mentions a variable(s) in E with its restriction $f|_{E=e}$ (somewhat abusing notation)
- Choose an elimination ordering Z_1, \dots, Z_n of variables in Z .
- For each Z_j -- in the order given -- eliminate $Z_j \in Z$ as follows:
 - Compute new factor $g_j = \sum_{Z_j} f_1 \times f_2 \times \dots \times f_k$, where the f_i are the factors in F that include Z_j
 - Remove the factors f_i (that mention Z_j) from F and add new factor g_j to F
- The remaining factors refer only to the query variable Q . Take their product and normalize to produce $P(Q)$

eg same example
Query: $P(A|D=d)$

- replace $f_4(C,D)$ with $f_5(C) = f_4(C,d)$
- proceed similar to before

ANALYSIS

\bullet_1 After eliminating z_j , the factors remaining in set F refer only to x_{j+1}, \dots, z_n & Q .

\bullet_2 Also, no factor mentions any evidence variable E after the initial restriction.

\bullet_3 Note

- ① The number of iterations is linear in the # of variables;
- ② The complexity is exponential in the # of variables.

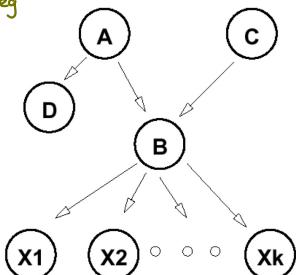
POLYTREES

\bullet_1 Polytrees are basically "trees" (ie no undirected cycles) that can have multiple start nodes.

\bullet_2 Idea: In these, the inference is linear wrt the size of the network.

\bullet_3 To do this, we eliminate only "singly-connected nodes".

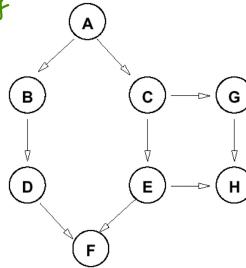
eg



- eliminate D, A, C, X_1, \dots, X_k
- if we eliminate B before these, we get factors that include all of A, C, X_1, \dots, X_k !

LEAST NEIGHBORS HEURISTIC

eg



- A, F, H, G, B, C, E is good

↳ we eliminate the nodes with 2 neighbors first
↳ leaving the nodes with 3 neighbors at the end.

- if we started with B , the ordering would be bad since the size of the factors is larger.

\bullet_1 Idea: When choosing an ordering, prioritize nodes with the least number of neighbors.

RELEVANCE

\bullet_1 Motivation: Certain variables have no impact on the query.

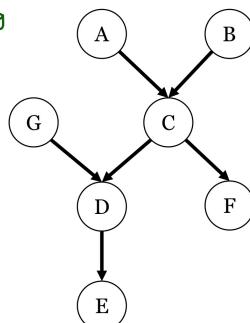
eg $A \rightarrow B \rightarrow C$

- To calculate $P(A)$, we only need to look at A 's CPT!
- However, if we do var elimination, we get trivial factors (ie whose value is just 1)

\bullet_2 Thus, when considering variables, we can restrict our attention to only the "relevant" ones:

- ie given query Q & evidence E :
- ① Q is relevant;
 - ② If Z is relevant, $\text{Parents}(Z)$ are relevant; &
 - ③ If $E' \in E$ is a descendant of a relevant node, then E' is relevant.

eg



① $Q = P(F)$

relevant: F, C, B, A

② $Q = P(C|F, E)$

relevant: F, C, B, A, E, D, Q

③ $Q = P(C|F, E, C)$

relevant: whole graph, but really none except C, F since C cuts off all influence of others.

Chapter 7: Causal Inference

CAUSALITY

"Causality" is the study of how things influence each other & how causes lead to effects.

CAUSAL DEPENDENCE

We say " X causes Y " if changes to X induce changes in Y .

Note joint distributions captures correlations between X & Y , not causations.

- $P(Y|X) \neq X \text{ causes } Y$

CAUSAL BAYESIAN NETWORK

A "causal Bayesian network" is one where all edges indicate direct causal effects.



CAUSAL INFERENCE

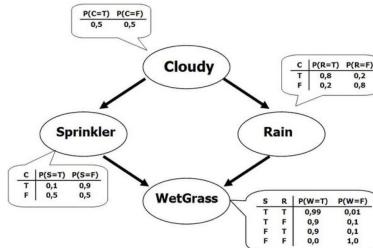
Causal networks can solve "intervention queries"; ie what the effect of an action is.
- but non-causal networks cannot.

OBSERVATION VS INTERVENTION

Observational queries are in the form "What is the likelihood of Y given X ?" ie $P(Y|X=x)$.

Interventional queries are in the form "How does doing X affect Y ?"
ie $P(Y|\text{do}(X=x))$.
- the "do" keyword specifies the query is an intervention.

EXAMPLE: CAUSAL GRAPH



observational: $P(WG|S=\text{true})$

- factors: $P(C), P(R|C), P(S|C), P(WG|S, R)$

- evidence: $S = \text{true}$

- eliminate: C, R

interventional: $P(WG|\text{do}(S=\text{true}))$

- we can remove the CPT from S since we "explicitly set" $S = \text{true}$.

- factors: $P(C), P(R|C), P(WG|S, R)$

- evidence: $S = \text{true}$

- eliminate: C, R

INFERENCE WITH THE DO OPERATOR

To do inference for $P(X|\text{do}(Y=y), Z=z)$:

① Remove edges pointing to Y &

$P(Y|\text{Parents}(Y))$

② Perform variable elimination as usual
(evidence is $Y=y, Z=z$).

COUNTER-FACTUAL ANALYSIS

- "Counter-factual analysis" explores outcomes that did not occur, but could have occurred under different conditions.
 - basically a "what-if?" analysis
 This can help test causal relationships.
 eg "would the patient have died if he was not treated"

STRUCTURAL CAUSAL MODEL / SCM

- Idea: We want to separate causal relations from "noise".

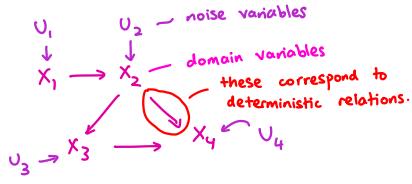
A "structural causal model" consists of

- ① X : endogenous/domain variables
- ② U : exogenous variables/noise
- ③ Only deterministic relations given by equations in the form

$$X_i = f(\text{parents}(X_i), U_i)$$

where U_i corresponds to the noise variable associated with X_i .

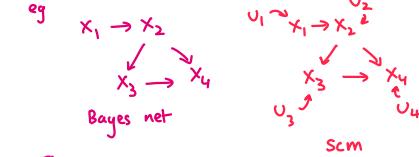
eg



$$X_1 = f_1(U_1), \quad X_2 = f_2(X_1, U_2).$$

$$X_3 = f_3(X_2, U_3), \quad X_4 = f_4(X_3, X_2, U_4)$$

- We can convert SCMs to causal Bayesian networks, but not v.v.



Then

$$P(X_1) = \sum_{U_1} P(U_1) f(X_1 | U_1)$$

$$P(X_2 | X_1) = \sum_{U_2} P(U_2) f(X_2 | X_1, U_2)$$

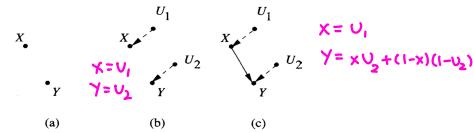
- SCMs are more descriptive since they separate causal relations from noise.

METHOD

For a causal model M , to find $P(Y=y | e, \text{do}(X=x))$:

- ① Update $P(u)$ to find $P(u|e)$ (abduction);
 - u = noise variables
- ② Replace the equations corresponding to variables in set X by the equations $X=x$ (action); &
- ③ Use the modified model to calculate $P(Y=y)$.

EXAMPLE



Model 1

	$u_2 = 0$	$u_2 = 1$	Marginal
$x = 1$	$x = 0$	$x = 1$	$x = 0$
$y = 1$ (death)	0	0.25	0.25
$y = 0$ (recovery)	0.25	0	0.25

Model 2

	$u_2 = 0$	$u_2 = 1$	Marginal
$x = 1$	$x = 0$	$x = 1$	$x = 0$
$y = 1$ (death)	0	0.25	0.25
$y = 0$ (recovery)	0.25	0	0.25

model B:

$$\begin{array}{ccc} X & \leftarrow & U_1 \\ & \downarrow & \\ Y & \leftarrow & U_2 \end{array}$$

evidence: $X=\text{true}, Y=\text{true}$
 $\Rightarrow P(U_2=1 | \text{evidence}) = 1.$
 Then
 $Y = U_2 = 1.$

model C:

$$\begin{array}{ccc} X & \leftarrow & U_1 \\ & \downarrow & \\ Y & \leftarrow & U_2 \end{array}$$

evidence: $X=\text{true}, Y=\text{true}$
 $\Rightarrow P(U_2=1 | \text{evidence}) = 1.$
 Then

$$Y = XU_2 + (1-X)(1-U_2)$$

$$= 0(1) + (1-0)(1-1)$$

$$= 0.$$

Chapter 8: Reasoning Over Time

STATIC VS DYNAMIC INFERENCE

Q: So far, we have assumed "static inference"; ie the world does not change.

θ_2 However, we need to perform "dynamic inference" in the real world since the world evolves over time.

In particular, we need

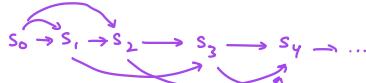
- ① A set of all possible states/worlds;
 - ② A set of time-slices/snapshots;
 - ③ Different probability distributions for each state at each time-slice; &
 - ④ Dynamics encoding how distributions change over time.

STOCHASTIC PROCESS

Q A "stochastic process" is defined by

- ① a set of states S ; &
 - ② some stochastic dynamics $P(s_t | s_{t-1}, \dots, s_0)$.

eg



Q2 This is a Bayes net with 1 r.v. per time slice.

 **Problems:**

- ① We may have infinite variables;
and so
 - ② We may have infinitely large
conditional probability tables.

To solve this, we will assume

 - ① "Stationary process": dynamics do not
change over time;
ie the CPT is the same regardless of
the time step.
 - ② "Markov assumption": current state depends
only on a finite history of past
states.

K-ORDER MARKOV PROCESS

 Idea: The last k states are sufficient for inference.

- e.g. - first-order: $P(s_t | s_{t-1}, \dots, s_0) = P(s_t | s_{t-1})$

$$s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$$

- $$\begin{aligned} \text{- second-order: } & P(s_t | s_{t-1}, \dots, s_0) \\ & = P(s_t | s_{t-1}, s_{t-2}) \end{aligned}$$

$$s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow \dots$$

 Q₂ Advantage: we can specify the entire process with finitely many time slices.

eq for 1st order: $s_{t-1} \rightarrow s_t$

- dynamics: $P(s_t | s_{t-1})$
 - prior: $P(s_0)$

HIDDEN MARKOV MODELS

 Motivation: In general,

- ① States are not directly observable;
 - ② Uncertain dynamics increase state uncertainty; but
 - ③ Observations made from sensors reduce state uncertainty.

 A "Hidden Markov model" encapsulates this and includes

- ① a set of states S ;
 - ② a set of observations O ;
 - ③ a transition model $P(s_t | s_{t-1}, \dots, s_0)$;
 - ④ an observation model $P(o_t | s_{t-1}, \dots, s_0)$;
&
 - ⑤ a prior $P(s_0)$.

eq 1st order HMM

$$s_0 \rightarrow s_1 \rightarrow s_2 \rightarrow s_3 \rightarrow s_4$$

\downarrow \downarrow \downarrow \downarrow
 s_1 s_2 s_3 s_4

- $P(s_t|s_{t-1})$: State transition with uncertainty
 - $P(o_t|s_t)$: uncertainty in measurements from sensors

INFERENCE IN TEMPORAL MODELS

Q₁: We have 4 common tasks:

- ① "Monitoring": $P(s_t | o_t, \dots, o_1)$
- ② "Prediction": $P(s_{t+k} | o_t, \dots, o_1)$
- ③ "Hindsight": $P(s_k | o_t, \dots, o_1)$, $k < t$
- ④ "Most likely explanation": $\underset{s_0, \dots, s_t}{\operatorname{argmax}} P(s_0, \dots, s_t | o_t, \dots, o_1)$

Q₂: We can solve ①-③ using variable elimination & ④ with a variant.

MONITORING

Q₁: Idea: We want to compute

$$P(s_t | o_t, \dots, o_1).$$

i.e. the distribution of the current state given observations.

Q₂: We can solve this using the "forward algorithm", which corresponds to variable elimination:

1. Factors: $P(s_0)$, $P(s_i | s_{i-1})$, $P(o_i | s_i)$, $1 \leq i \leq t$
2. Restrict o_1, \dots, o_t to observations made
3. Sumout s_0, \dots, s_{t-1} ; i.e.

PREDICTION

Q₁: Goal: we want to compute

$$P(s_{t+k} | o_t, \dots, o_1);$$

i.e. the distribution over future state given observations.

Q₂: We can also use the forward algorithm:

1. Factors: $P(s_0)$, $P(s_i | s_{i-1})$, $P(o_i | s_i)$, $1 \leq i \leq t+k$
2. Restrict o_1, \dots, o_t to observations made
3. Sumout s_0, \dots, s_{t+k-1} , o_{t+1}, \dots, o_{t+k}

HINDSIGHT

Q₁: Goal: we want to compute

$$P(s_k | o_t, \dots, o_1)$$

Q₂: We can use "forward-backward algorithm" to solve this:

1. Factors: $P(s_0)$, $P(s_i | s_{i-1})$, $P(o_i | s_i)$, $1 \leq i \leq t+k$
2. Restrict o_1, \dots, o_t
3. Sumout s_0, \dots, s_{k-1} , s_{k+1}, \dots, s_t

MOST LIKELY EXPLANATION

Q₁: Goal: We want to compute

$$\underset{s_0, \dots, s_t}{\operatorname{argmax}} P(s_0, \dots, s_t | o_t, \dots, o_1).$$

Q₂: We can use the "Viterbi algorithm" to solve this:

1. Factors: $P(s_0)$, $P(s_i | s_{i-1})$, $P(o_i | s_i)$, $1 \leq i \leq t$
2. Restrict o_1, \dots, o_t
3. "Maxout" s_0, \dots, s_t

COMPLEXITY OF TEMPORAL INFERENCE

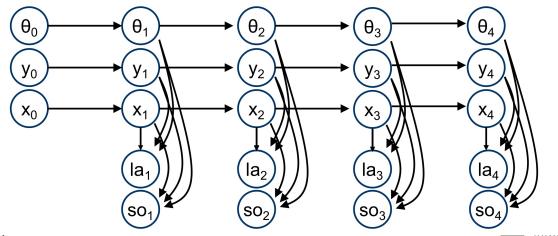
Q₁: HMMs are Bayes nets with a polytree structure.

Q₂: Thus, variable elimination is

- ① Linear wrt # of time slices;
- ② Linear wrt the largest CPT.

DYNAMIC BAYESIAN NETWORKS

Idea: Encode states & observations with several random variables, and exploit conditional independence to save time & space.



This allows us to write the transition and observation models very compactly.

NON-STATIONARY PROCESS

If the process is not stationary, we can add new state components until dynamics are stationary.

NON-MARKOVIAN PROCESS

If the process is not Markovian, we can add new state components until dynamics are Markovian.