Week	Monday	Wednesday
1	Mar. 28: Introductions	Mar. 30: Agents (HW1 out)
2	Apr. 4: Uninformed Search (PA1 out)	Apr. 6: Informed Search (HW1 due)
3	Apr. 11: Local Search	Apr. 13: Adversarial Search (PA2 out)
4	Apr. 18: Logic 1 (PA1 due)	Apr. 20: Logic 2 (HW2 out)
5	Apr. 25: Flex Time/Catch-up	Apr. 27: Guest Lecture: AlphaGo Prof. Alan Fern (PA2 due)
6	May 2: Midterm Review (HW2 due)	May 4: Midterm
7	May 9: Probability (HW3 out)	May 11: Bayes Nets 1
8	May 16: Bayes Nets 2 (HW3 due, HW4 out, PA3 out)	May 18: Game Theory (HW5 out)
9	May 23: Guest Lecture: Safety in AI Prof. Sandhya Saisubramanian (HW4 due)	May 25: Guest Lecture: Ethics in AI Prof. Prasad Tadepalli (HW5 due)
10	*May 30: Cancelled (univ. holiday)*	Jun. 1: Final Exam Review (PA3 due)

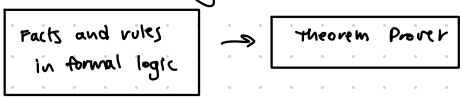
Mon Mar 28

Turing test (acting like humans)
- natural language processing
- knowledge representation
- machine learning

Problems:
- not reproducible
- no mathematical analysis
- focus on human-like errors
- not useful programs

Thinking Humanly (Cognitive Modeling)

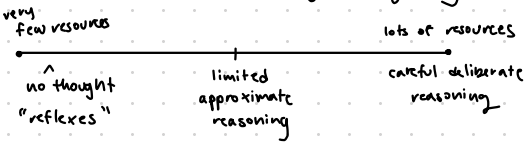
Thinking rationally (Laws of Thought)



Acting Rationally (Rational Agents)

"agent" something that acts

- "rational" more than just logically defined → "doing the right thing"



Uninformed Search
Informed Search

Bayes Nets
- probability
- learning to classify emails as spam or not spam

Wed Mar 30

- Spring 6-7 pm Johnson 102

Agents

information gathering : define → taking input?
stochastic, partially observable, multi-agent → raises chances
↓
doing actions in order to modify future percepts

simple → respond directly to percepts

model → maintains an internal state to track world not seen to it

goal → tries to reach its goal

utility → maximizes its "happiness" at each state

fully observable	vs	partially observable		
deterministic	vs	stochastic	vs	strategic
episodic	vs	sequential		
static	vs	dynamic		
discrete	vs	continuous		

A1:
stay consistent w/ agent design

In class Exercise

PEAS description

Performance measure : rating system, customer satisfaction, # of movie purchases, recommended vs. elsewhere, speed

Environment : list of movies available, user watch history, user profile, user ratings

Actuators : what it recommends

Sensors : lookup capability

PEAS

P: 1-5 given

E: runs on a server

A: place a movie in 'rec' list of web interface

S: access ratings provided by users

fully observable	vs	partially observable			
deterministic	vs	stochastic	vs	strategic	
episodic	vs	sequential			
static	vs	dynamic			
discrete	vs	continuous			

AI Search Problems (unobservable)

Assumptions About Our Environment

- Fully Observable
- Deterministic
- Sequential
- Static
- Discrete
- Single-Agent

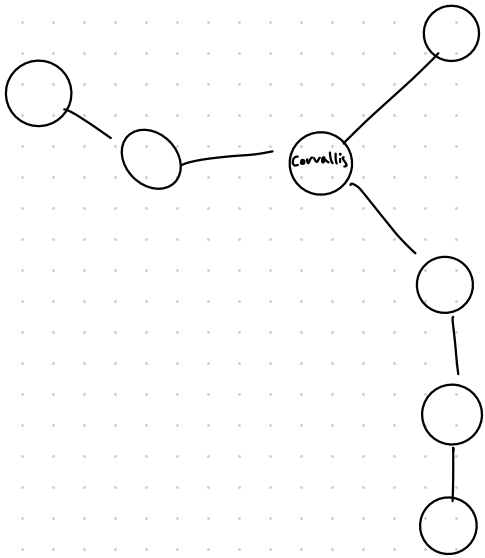
Search Problem Formulation

Components of a Search Problem:

1. A state space S
2. An initial state $I \in S$
3. A non-empty set of goal states $G \subseteq S$ (or a goal test function that tests whether a state is a goal)
4. A description of the actions available in state S, actions(s)
5. A transition model describing the result of action "a" applied in state "s": result(s,a)
6. A cost function cost(s,a,s') which returns the non-negative one-step cost of travelling from state s to s' by applying a. Cost function is only defined if s' is a successor state of s

* When actions are deterministic, combine actions(s) and results(s,a) into a successor function succ(s) that returns the states reachable in one step from s.

cost(s,a,s')
as
cost(s,s')



Search Tree

1. Start w/ initial state. Is initial the goal?
   - yes → return
   - No → expand
2. Apply Successor() function
3. Make queue from "frontier"
4. Remove node from queue. If it's a goal, return solution. Else, call Successor() and put results on the queue. Repeat.

Note:
- order in which you expand nodes
- avoid repeated states

Avoid Repeated States

- trade off between space and time
- "explored" list, stores every expanded
- if current node matches a node on closed list, discard

Graph-Search

Uninformed Search

- no info about states other than generating successors and recognizing goal states
- informed search can tell if a nongoal state is more promising than another

Evaluate Uniformed Search

- Completeness
- Optimality
- Time Complexity
- Space Complexity

Complexity

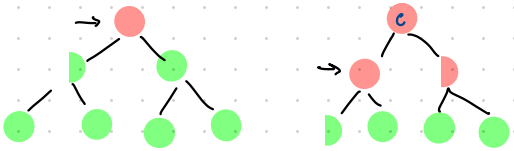
1. Branching Factor — max number of successors of any node
2. Depth of shallowest goal node
3. Maximum length

Time Complexity: number of nodes generated during search
Space Complexity: maximum number of nodes stored in memory

Uninformed Search Algorithms

Breadth-First Search

- expand all nodes at given depth before any at next lvl expand
- FIFO queue



Evaluating BFS

Complete? yes, as long as the branching factor is finite and contains the goal

Optimal? yes (shallowest goal node) if step costs are identical

Time Complexity $b + b^2 + b^3 + \ldots + b^d + (b^{d+1} - b) = O(b^{d+1})$

Space Complexity $O(b^{d+1})$

* Exponential time and space makes BFS impractical

Uniform-cost Search

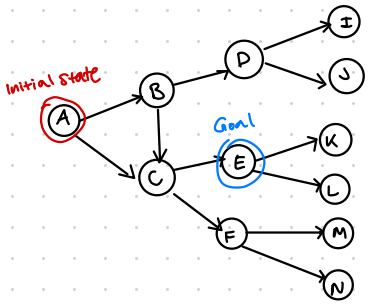
- step costs aren't equal?
- BFS expands shallowest node

Complete? yes if branching is fin

Optimal? yes

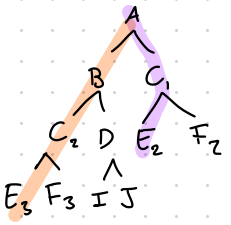
Time Complexity $O(b^{1+\text{floor}(C^*/\varepsilon)})$ where C^* is the cost of optimal solution

Space Complexity $O(b^{1+\text{floor}(C^*/\varepsilon)})$ where C^* is the cost of optimal solution



Arrows describe actions, results, successors.

All step costs = 1



State space S

How to set up backtracing? 3.3.1

BFS-tree FIFO

FRONTIER:

A ↺ A
A ↺ B C1
B ↺ C1 C2 D
C1 ↺ C2 D E F
C2 ↺ D E2 F2 E3 F3
D ↺ E2 F2 E3 F3 I J
E2 ↺

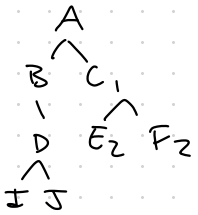
DFS-tree LIFO

FRONTIER:

A ↺ A
A ↺ B C1
B ↺ C2 D C1
C2 ↺ E3 F3 D C1
E3 ↺

node
- state
- parent
- action

reconstruct w/ parent



BFS-graph

FRONTIER	EXPLORED
A ↺ A	A
B ↺ B C1	A B
C1 ↺ C1 D	A B C1
D ↺ D E2 F2	A B C1 D
E2 ↺ E2 F2 I J	

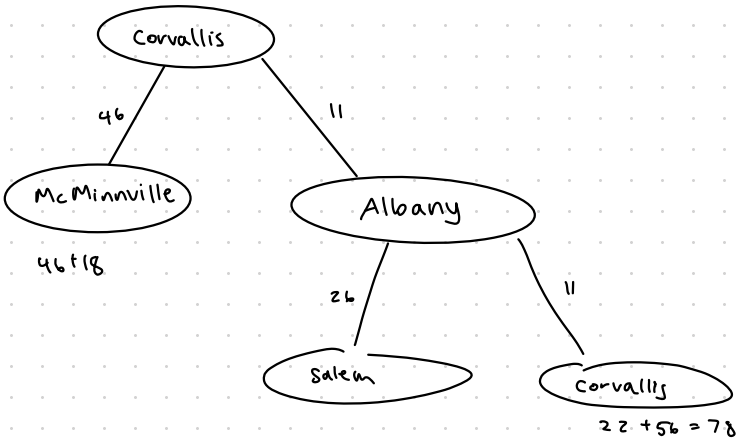
DFS-graph

FRONTIER	EXPLORED
A ↺ A	A
B ↺ B C1	A B
D ↺ D C1	A B D
I ↺ I J C1	A B D I
J ↺ J C1	A B D I J
C1 ↺ C1 E2 F2	A B D I J C1
E2 ↺	

h(n): straight line distance

A* Search

h(n): straight line distance to Wilsonville



* Don't stop when you get a goal state on the queue. Stop when you pop a goal state from the priority queue

Proof that A* with Tree-Search is optimal if h(n) is admissible

Suppose A* returns suboptimal goal node G_2

G_2 must be least cost node in frontier. Let cost of optimal solution be C^*

Because G_2 is suboptimal

$$f(G_2) = g(G_2) + h(G_2) = g(G_2) > C^*$$

If h(n) is admissible,

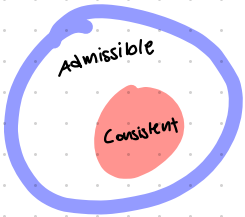
$$f(n) = g(n) + h(n) \leq C^*$$

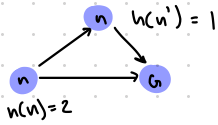
We show that $f(n) \leq C^* < f(G_2)$ hence contradiction

consistent: if every node n and every successor n' of n generated by any action a

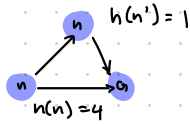
$$h(n) \leq c(n, a, n') + h(n')$$

- form of triangle inequality





consistent



inconsistent

If h(n) is consistent, then the values of f(n) along any path are non-decreasing

* write search algo
  Heuristic

C^* cost to optimal goal

h_2 dominates h_1
↑
prefer this!

Inventing Admissible Heuristics

- relaxed problem: problem with fewer restrictions on the action
- cost of optimal solution to a relaxed problem is admissible heuristic to the original problem.

* A* graph for assignment
  ↓
  admissible → consistent

Sun Apr 10

Local Search Algorithm Recipe

1. start with initial configuration X
2. Evaluate its neighbors i.e. the set of all states reachable in one move from X — neighborhood
3. Select one of its neighbors X^* — who to select
4. Move to X^* and repeat until the current configuration is satisfactory — stopping criteria
   - # of iteration
   - time
   - uphill

① Hill-Climbing

- starting at initial state X, keep moving to the neighbor with the highest objective function value greater than X's

```
X ← Initial configuration
Iterate:
    E ← Eval(X)
    N ← Neighbors(X)
    For each Xi in N
        Ei ← Eval(Xi)
    E* ← highest Ei
    X* ← Xi with highest Ei
    If E* > E
        X ← X*
    Else
        Return X
```

- greedy local search
- Problems: gets stuck in local maxima, or flat local maximum
- Neighborhood:
  - large → lots of evaluations, better chance of good max.
  - small → fewer evaluations, can get stuck

- stochastic hill climbing
  - choose at random

- first choice hill climbing
  - generates successors randomly until one is generated that is better than current

- random restart hill climbing
  - good for dealing with local maxima
  - series of searches from random initial states

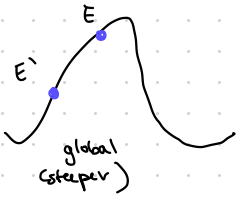
② Simulated Annealing

- add some random moves to hill-climbing to help it get out of local max.?

```
X ← Initial configuration
Iterate:
    E ← Eval(X)
    X` ← Randomly selected neighbor of X
    E` ← Eval(X`)
    If E` ≥ E
        X ← X`
        E ← E`
    Else with probability p        } different from hillclimbing
        X ← X`
        E ← E`
```

How do you set p?
- Too low: not many downhill moves
- Too high: too many suboptimal moves
- const: too random near global max

Setting p
- Decrease p as iterations progress
- Decrease p as $E - E'$ increases



③ Beam Search

Local Beam Search
- keep track of k states instead of just 1, start w/ k randomly generated states.
- generate all successors of all the k states
- continue
- Select the best k successors from complete list.
- Repeat the process until goal found

useful info is passed among k parallel search thread

④ Genetic Algorithms 4.1.4

- like natural selection
- variant of stochastic beam search

- fitness function: evaluation function in GA terminology

- population: k randomly generated states

- individual: string over finite alphabet

- selection: two random individuals for reproduction

- crossover: point at mix parent string.

- mutation: randomly change a location in an individual's string w/ small probability

Variant 1: Culling

Variant 2:

$$P(X \text{ selected}) = \frac{Eval(X)}{\sum_{Y \in Pop.} Eval(Y)}$$

initially, population is diverse but overtime, cross over doesn't produce a big change.

Crossover is an advantage.

Schema
- substring which some of the positions can be left unspecified

⑤ Gradient Descent

local minimum

$$x \leftarrow x - \alpha \nabla f(x)$$

local maximum

$$x \leftarrow x + \alpha \nabla f(x)$$

Multivariate

$$\nabla f(x_1, x_2, x_3) = \left(\frac{\partial f}{\partial x_1}, \frac{\partial f}{\partial x_2}, \frac{\partial f}{\partial x_3} \right)$$

$$x_1 \leftarrow x_1 - \alpha \frac{\partial f}{\partial x_1}$$

Weaknesses

1. Can be slow to converge to local optimum
2. Depends on learning rate α
3. What if $f(x)$ isn't differentiable at x?

Wed Apr 13

minimax → Prog 2

Office Hours

evaluation function: heuristic, estimate value of intermediate node
cutoff

what is better than others?
as close as possible

A2: Simple Othello
1 algorithm

Blue > Purple

Human first
We're playing second (computer)
always win or tie

terminal output

bookkeeping

1. utility
2. successor
3. minimax

C++, Python

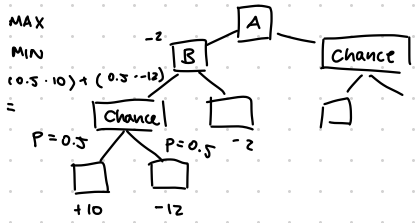
skeleton code

If computer cannot be beaten, full credit

10:00 AM

Finite stochastic games

Expect minimax



$(0.8 \cdot 100) + (0.2 \cdot 0)$

80 + 0

$(15 \cdot 0.5) + (5 \cdot 0.5)$

0.1

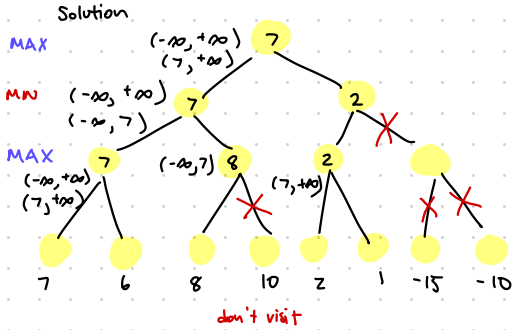
Don't take minimum

8 + 9 > 17

Chance node

chance chance

Solution



MAX updates alpha based on beta

MIN updates beta based on alpha.

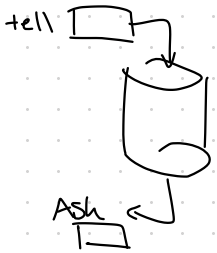
Which if any, are pruned?

Logic

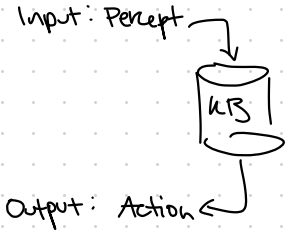
Knowledge Based Agents

- the agent has an encoded base of knowledge
- accept new tasks in the form of explicitly described goals
- achieve competence quickly by being told or learning new knowledge about the environment
- adapt to changes

KB
- set of "sentences"



Inference
- derive new sentences from old ones



1. Tell the KB what it percieves
2. Ask the KB what action it should perform
3. Tell the KB that the action was executed

Wumpus World

- Wumpus eats anyone entering its room
- Wumpus can be shot by an agent, agent has 1 arrow

Performance

Environment

Actuators

Sensors

[stench, breeze, glitter, bump, scream]

Agent is ignorant of configuration of 3x3 world

Logic:
1. Syntax
   - Symbols, rules, legal configs
2. Semantics
   - "meaning"
   - truth of each sentence
   - everything is either true or false, no in between

Mathematical abstractions which fix the truth or falsehood of every relevant sent.

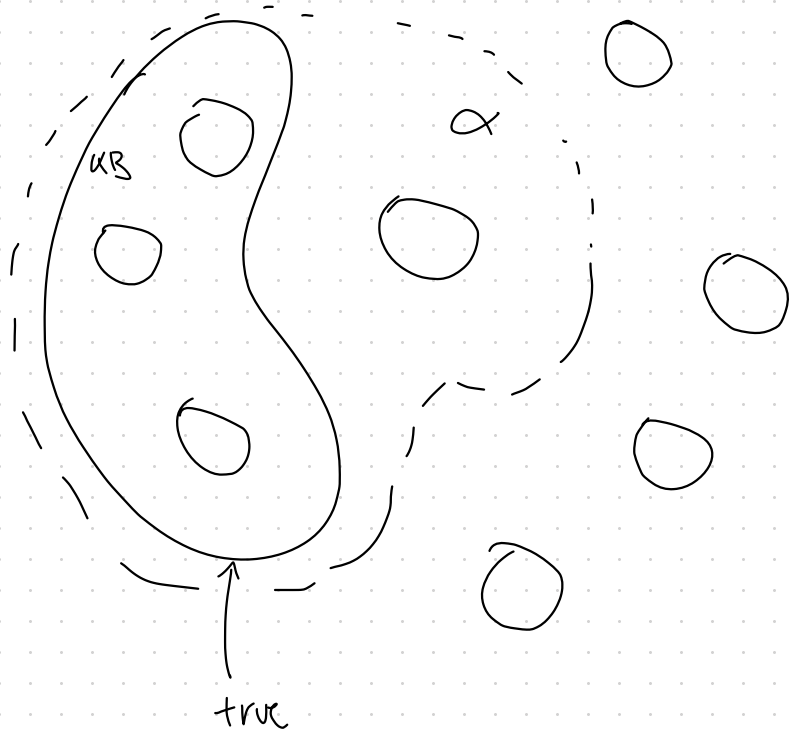
M is a model of α

Entailment

$\alpha \models \beta$ means α entails β i.e. β follows logically from α, where α and β are sentences

if α is true, then B MUST be true

Entailment



Sound

sound truth-preserving

complete

derive any sentence that is entailed.

Soundness: i is sound if whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$
the things we derive are true

Completeness: i is complete if whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$
we can derive all the true things

Atomic Sentences

- indivisible syntactic elements

P Q false

literal

true

False

$\neg$	not
$\wedge$	and
$\vee$	or
$\rightarrow$	implies

highest to lowest
$\neg, \wedge, \vee, \rightarrow, \leftrightarrow$

$\neg P \vee Q \wedge R \rightarrow S$

$((\neg P) \vee (Q \wedge R)) \rightarrow S$

P = true
Q = false
R = true

$P \wedge Q \wedge R$

false

Survey

Question

Rephrase

Recall

Repetition

Inference: enumerate the models, check that α is true in every model in which KB is true

$KB \models \alpha$

look in the knowledge base

Every known algo inference is exponential

Entailment: $\alpha \models \beta$ means α entails β i.e. β follows logically from α, where α and β are sentences

Logical equivalence:

two sentences are logically equivalent if they are true in the same set of models

Formally: $\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

Prove with TT

Satisfiable: true in some model

Unsatisfiable: true in no models

$\alpha \models \beta$ iff the $\alpha \rightarrow \beta$ is valid

Validity

For any sentences α and β, $\alpha \models \beta$ iff the sentence $(\alpha \Rightarrow \beta)$ is valid

valid: always true

Inference Rules

1. Modus Ponens

$$\frac{\alpha \Rightarrow \beta, \quad \alpha}{}$$

both sound correct

2.

complete: by applying, I can come up with everything that is entailed

Resolution

an inference rule that is sound and complete

resolution can refute or confirm the truth of any sentence w/ respect to the KB

Resolution

complementary literals

$$\frac{l_1 \vee l_2 \qquad \neg l_2 \vee l_3}{l_1 \vee l_3}$$

you need to remove multiple copies

$$\frac{l_1 \vee l_2, \quad \neg l_2 \vee l_1}{l_1}$$

If l_i and m_j are complementary literals

$$\frac{l_1 \vee \dots \vee l_k, \quad m_1 \vee \dots \vee m_n}{l_1 \vee \dots \vee l_{i-1}}$$

Conjunctive Normal Form

To prove $KB \models \alpha$, $(KB \wedge \neg\alpha)$ is unsatisfiable

Deduction Theorem

$\alpha \models \beta$ iff $\alpha \Rightarrow \beta$ is valid

$KB \models \beta$

α	β	$\alpha \Rightarrow \beta$	
F	F	T	
F	T	T	
T	F	F	←
T	T	T	

$\alpha \models \beta$ iff $\alpha \Rightarrow \beta$ is valid

$\alpha \models \beta$ iff $\neg\alpha \vee \beta$ is valid

$\alpha \models \beta$ iff $\neg(\neg\alpha \vee \beta)$ is unsatisfiable

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

KB

To prove $KB \models \alpha$, we show that $(KB \wedge \neg\alpha)$

The algorithm

must add $\neg\alpha$ to KB, it is not there before

yes to entailment, then add to KB

pseudocode
notes
sections

what you should know / takeaways
> 60 min
> 40+ q's (2019 midterm

P(Pass = true)
P(study = true)
P(Pass = true | Study = true)
P(Study = true | Pass = true) = $\frac{P(\text{Pass} = \text{true} \mid \text{Study} = \text{true}) \cdot P(\text{Study} = \text{true})}{P(\text{Pass} = \text{true})}$

$P(P,S) = P(S)P(P|S) = P(P)P(S|P)$

$P(P|S) = \frac{P(P,S)}{P(S)}$ $\quad$ $P(S|P) = \frac{P(P|S)P(S)}{P(P)}$

DATA

Coin $\in \{H,T\}$	Card $\in \{R,B\}$	Candy $\in \{1,2,3\}$
T	B	3
T	R	3

Joint probability
$2 \cdot 2 \cdot 3 = 12$

P(candy = 2) 0.25

P(candy = 1 | card = red)

$P(C=1 \mid C=r) = \frac{P(C=1, C=r)}{P(\text{Card} = \text{red})}$

$= \frac{P(\text{tails, red, 1}) + P(\text{head, red, 1})}{}$

Coin and Card independent

P(Coin) = $\langle \underset{t}{0.5}, \underset{h}{0.5} \rangle$

P(Card) = $\langle \underset{b}{0.45}, \underset{r}{0.55} \rangle$

P(Coin, Card) =

	b	r
t	0.3	0.2
h	0.15	0.35

If independent, then
P(Coin, Card) = P(Coin) P(Card)
t r t r
0.2 $\neq$ $0.5 \cdot 0.55 = 0.275$

not independent

$P(C = \text{heads}, \text{Card} = \text{red}, \text{candy} = 3)$
$P(\text{Coin} = \text{heads}) \cdot P(\text{red} \mid \text{heads}) \cdot P(3 \mid \text{red})$

$P(\text{Coin} = \text{tails} \mid \text{Card} = \text{red})$

$P(\text{Coin} = \text{tails}) \cdot P(\text{red} \mid \text{tails}) = P(\text{red})$

$0.5 \cdot (0.4 + 0.7) = 0.5 \cdot 1.1 = 0.55$

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

$$= \frac{P(\text{red} \mid \text{tails})}{\text{Card} = \text{red}} \text{Coin} = \text{tails}$$

$$= \frac{0.4 \cdot 0.5}{0.55} = 0.364$$

$$P(\text{red}) = \sum_{c \in \{h,t\}} P(\text{Coin} = c, \text{Card} = \text{red})$$

$$\sum_{c \in \{h,t\}} P(\text{Card} = \text{red} \mid \text{Coin} = c) P(\text{Coin} = c)$$

$$= \underset{h}{0.7} \cdot 0.5 + \underset{t}{0.4} \cdot 0.5$$

$$= 0.55$$

C = loc. of prize $\in \{1, 2, 3\}$

D = door initially chosen $\in \{1, 2, 3\}$

H = door host opens $\in \{1, 2, 3\}$

$P(C) = \langle \frac{1}{3}, \frac{1}{3}, \frac{1}{3} \rangle$

$P(D) = \langle \frac{1}{3}, \frac{1}{3}, \frac{1}{3} \rangle$?

$P(C \mid D=2) = \langle \frac{1}{3}, \frac{1}{3}, \frac{1}{3} \rangle$ C and D are independent $P(C \mid D) = P(C)$

$P(C \mid D, H) = \langle \underbrace{?}_{\text{choose highest val}} \rangle$ ← query

$P(C, D, H) = P(H \mid C, D)\, P(C, D)$ 27 full joint

$= P(H \mid C, D)\, P(C \mid D) P(D)$ $P(C, D) = P(C \mid D) P(D)$

$= \underline{P(H \mid C, D)}\, P(C) P(D)$ ← factored

$D=3$	C	H	$P(H \mid C, D=3)$
	1	1	0
	1	2	1
	1	3	0
	2	1	1
	2	2	0
	2	3	0
	3	1	1/2
	3	2	1/2
	3	3	0

$$P(C \mid D=3, H=2) = \frac{P(C, D=3, H=2)}{P(D=3, H=2)}$$

$$= \frac{P(H=2 \mid C, D=3)\, P(C)\, P(D=3)}{\sum_c P(C=c, D=3, H=2)}$$

$$= \frac{P(H=2 \mid C, D=3)\, P(C)\, P(D=3)}{\sum_c \left[P(H=2 \mid C=c, D=3)\, P(D=3)\, P(C=c)\right]}$$

$$= \frac{P(H=2 \mid C, D=3)\, P(C)}{\sum_c P(H=2 \mid C=c, D=3)\, P(C=c)} = P(H=2 \mid C=1, D=3) P(C=1) + P(H=2 \mid C=2, D=3) P(C=2) + P(H=2 \mid C=3, D=3) P(C=3)$$

$= \alpha \langle 1 \cdot \frac{1}{3}, 0 \cdot \frac{1}{3}, \frac{1}{2} \cdot \frac{1}{3} \rangle$ (c=1, c=2, c=3)

$= \alpha \langle \frac{1}{3}, 0, \frac{1}{6} \rangle$

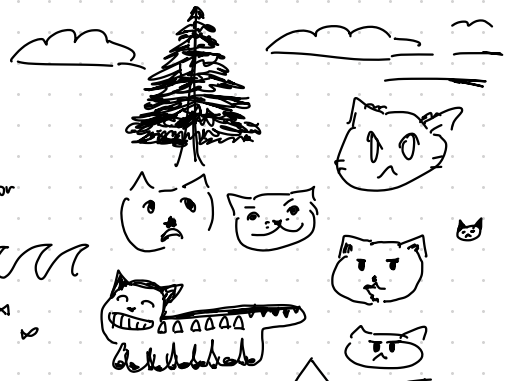
$P(C|H,D) = \langle \frac{2}{3}, 0, \frac{1}{3} \rangle$ $D = 3$ $H = 2$

C=1 switch C=3 orig

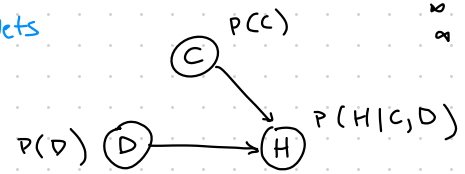
$\frac{1}{3} + 0 + \frac{1}{6} = \frac{1}{2} = 2$

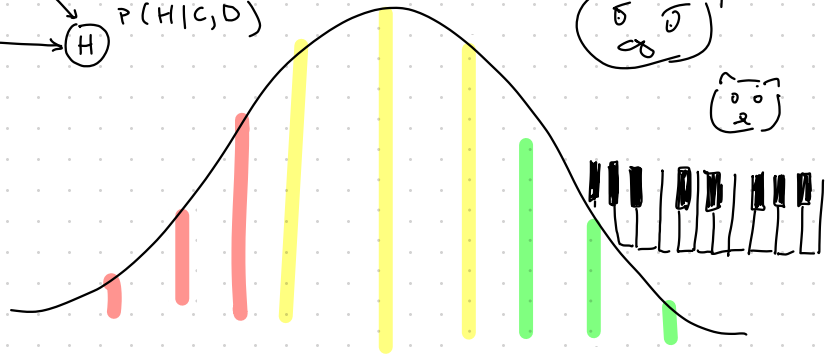
$\alpha = 2$

$\frac{1}{\text{denominator}}$

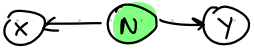


Bayes Nets



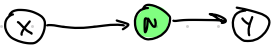


Case 1



blocked

Case 2

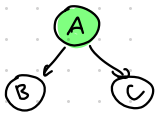


blocked

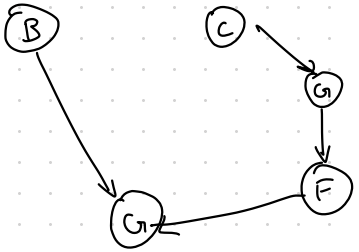
Case 3

cannot be any descendants in evidence set

1. Ignore arrows, find all possible paths between X and Y
2. Then look at arrows
3. If all paths are blocked then d-separated



case 1



case 3 due to G