Intro

1. **Game Playing**
2. Search
3. Simulated Annealing
4. Constraint Satisfaction
5. Probability
6. Bayes Nets
7. Machine Learning
8. Pattern Recognition through Time
9. Logic and Planning
10. Planning under Uncertainty

Game Playing

1. adversarial search
2. minimax algorithm
3. alpha-beta pruning
4. evaluation f unctions
5. isolation game player

Isolation game

isolate the other user on a checker board
building a game tree:

1. shows all the moves possible in the game: example (we're o)
2. first moves matters! (e.g. when o goes to top middle, it leads to many losses
3. how to teach our computer player o to make better first move? --**minimax algorithm**

Minimax Algorithm

mark game tree with up/ down arrows: up (max level) -- we try to maximize our score; down (min level) -- opponent tries to minimize our score
top of minimax tree is always a max level

Propagating values up the tree
start from bottom nodes, if it's max level, take max of child nodes, vice versa. work our way up
[reading]: 5.1-5.2

Maximum num of nodes
25 * 24 * ... * 1
better estimate:
25 * 24 * 12 ^ 11 *12!
explanation:
1. middle of board has 16 moves but most cells have 12 or less moves (call it **branching factor**)
2. open moves: 25*24
3. last move when game ends 1, the one before 2, ... maximum choice before ending 12
4. moves before getting to 12 choices = 25-2-12 = 11
5. there are 11 moves with avg 12 choices after opening & before closing

[quiz] how many # nodes minimax algorithm needs to visit? (given b: branch factor, d: depth of tree)
ans: b^d

avg branch factor: 8 (from simulation) -- 8^25 nodes

Depth limited search
<u>if we can't search endgame, how deep can we search?</u>
assume we can search 10^9 nodes every second,
10^9*2 in 2 sec
8^x = 2 * 10^8

<u>Depth Limited Search</u>

10^9 nodes $\times 2$ sec $= 2 \times 10^9$ nodes

$8^x < 2 \times 10^9$

$\log_8 8^x < \log_8 2 \times 10^9$

$x < \frac{\log_{10} 2 \times 10^9}{\log_{10} 8}$

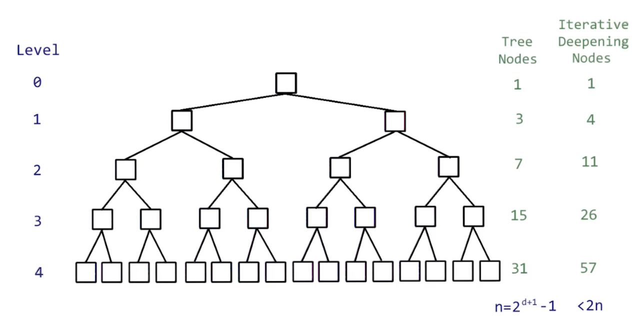
$x < 10.3$

$\log_a x = \frac{\log_b x}{\log_b a}$

Evaluation function
"maximize # moves player has left"
- when we search to leave nodes, we get good answer; but if we only search x levels deep (e.g. 2), we may get wrong answer

Quiescent Search
calculate recommended move with depth 1, 2, ... until it doesn't change much --> reached enough depth;
often gives us better results @ beginning or ending of the game

Iterative Deepening
at each level, compare & save result in case we run out of time. if time is up, return last saved result
-- it's not inefficient because in game tree search, computation time is dominated by last level searched



when branch == 2, iterative deepening node < 2n, when branch >2, it's lower

Bayes Nets

Replace denominator with constant

P(A|B) = P(B|A) * P(A) / P(B) = a * P(B|A) * P(A)
P(A'|B) = P(B|A') * P(A') / P(B) = a * P(B|A') * P(A')
A = NOT A' --> **P(A|B) + P(A'|B) = 1**
example: two test cancel

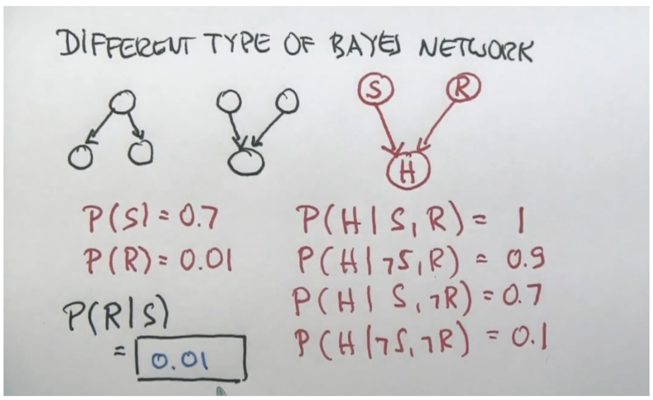
Conditional Independence

P(+ + | C) = P(+|C) * P(+|C)
conditional: only true if C is given
independent: two diagnoses don't rely on each other
foundation of bayes nets!

Q: Does conditional independence indicate absolute independence?
(does P(A,B|C)=P(A|C)*P(B|C) mean P(A,B) = P(A)P(B))?
A: NO.
P(+|+) = P(+|+, C)*P(C) + P(+|+, C')*P(C') != 0, hence not absolute independent
(note two +s represent second & first diagnose)

Q: Does absolute independence indicate conditional independence?
A: NO.

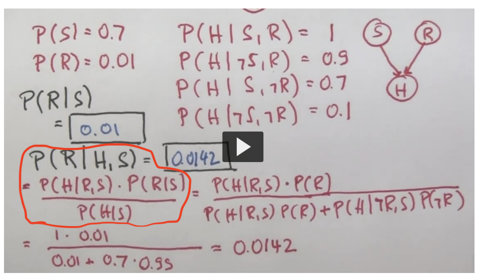
Confounding Cause



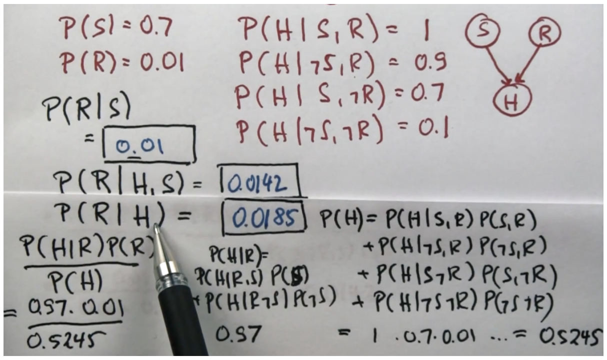
P(R|S) = P(R) because 2 causes S,R are independent

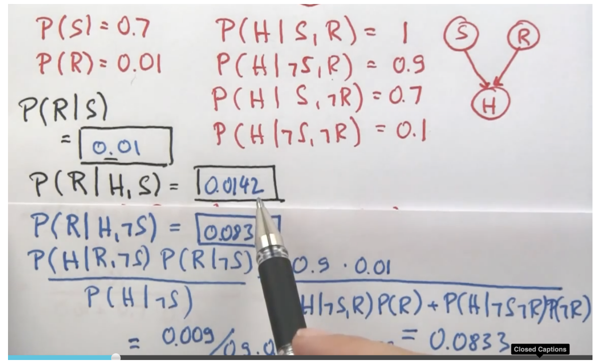
Explain away
observe: happy, also observe sunny -> P(raise) is lower

Q:



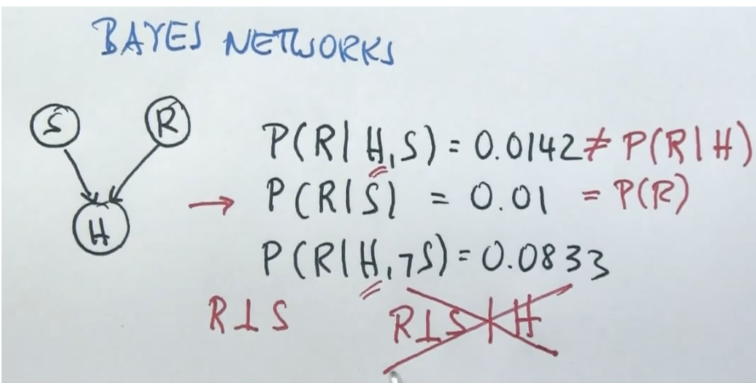
without knowing the weather (Sunny), chance of raise is higher given i'm happy





P(R|H,S) <<P(R|H,S'): when we know one cause is False & the output, the other cause prob significantly increases

Proof of independence doesn't imply conditional independence

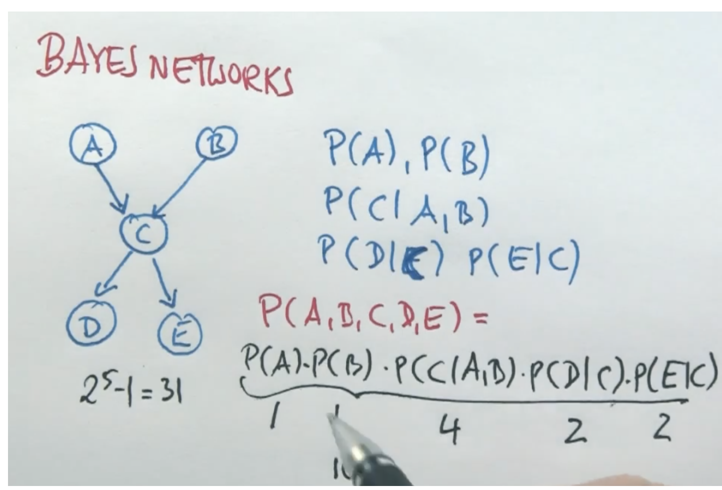


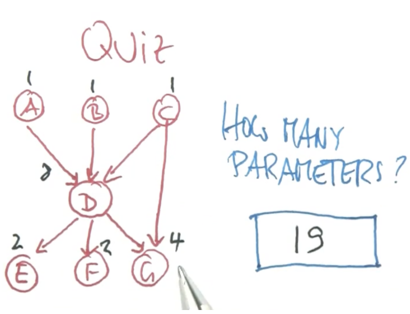
General Bayes Nets

enumerate all combo: 2^N-1
bayes nets representation: 10

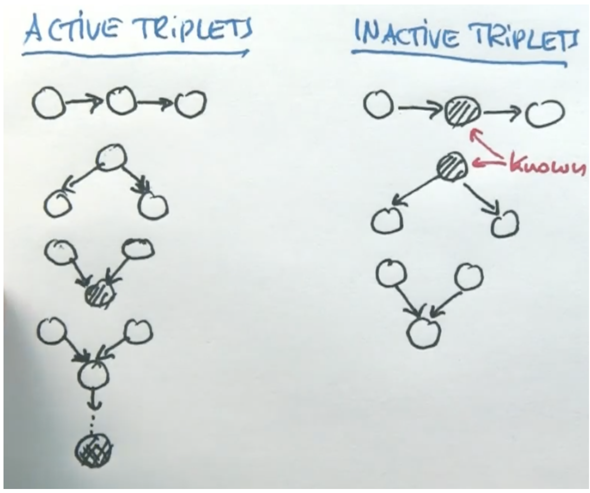
advantage: shrinks probability values required significantly

Quizes: count params



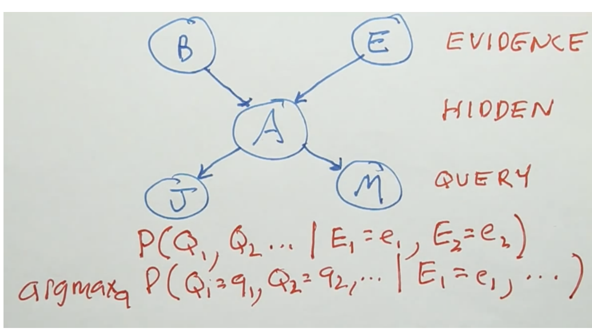


D-separation



streamline: dependent
cut middle node in streamline: independent
cut parent node: children become independent
cut children node: parents become dependent

Probabilistic Inference

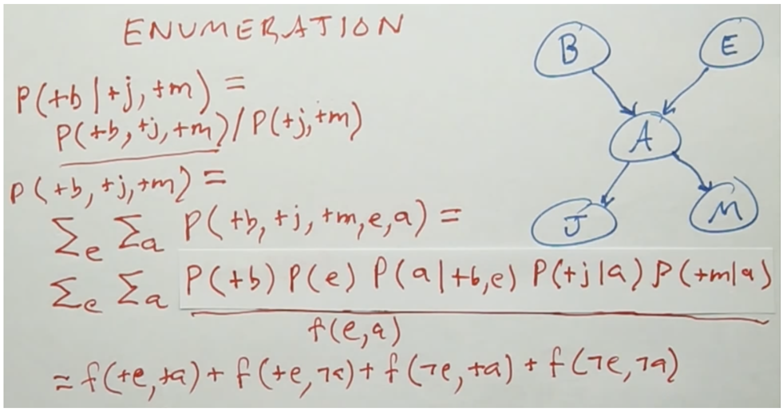


evidence: probabilities we know
query: probabilities we are interested to find out
hidden: everything else

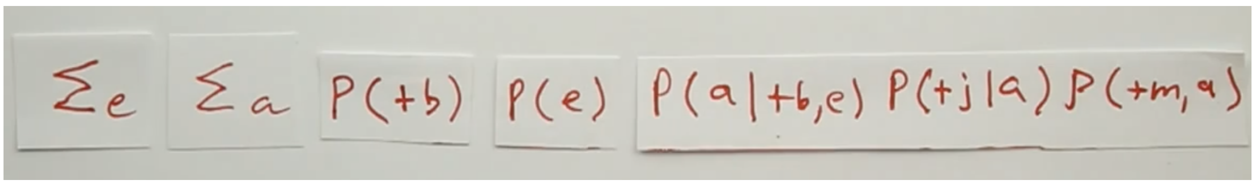
* bayes networks can go both ways (B,E can be query too)

Enumeration

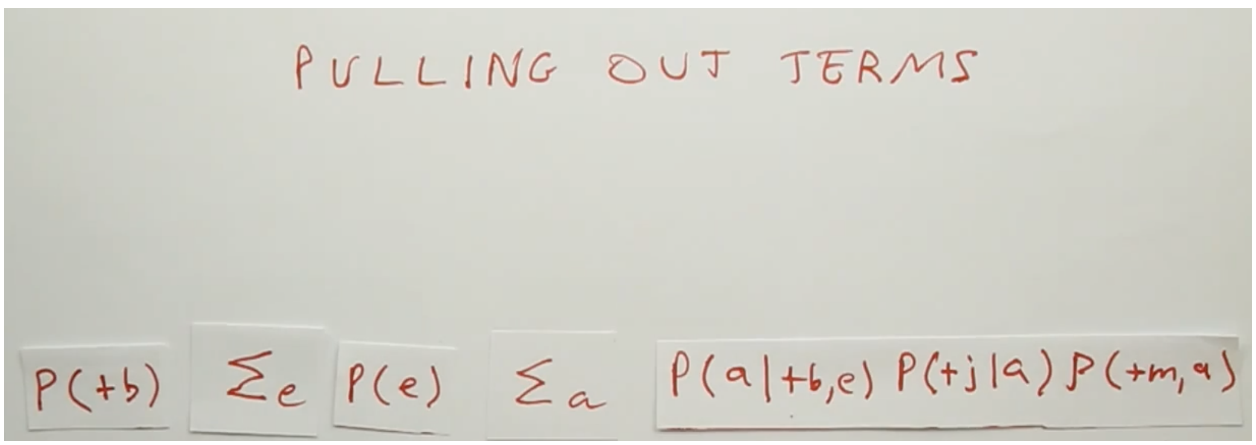
go through all possible probabilities, add them up and come up with the query probability



Speed up Enumeration
techniques:



1. pull out terms



2. maximize independence
for streamline network:
node0->node1->node2->...->noden : O(N)

for complete network (every pair nodes are connected)
O(2^N)

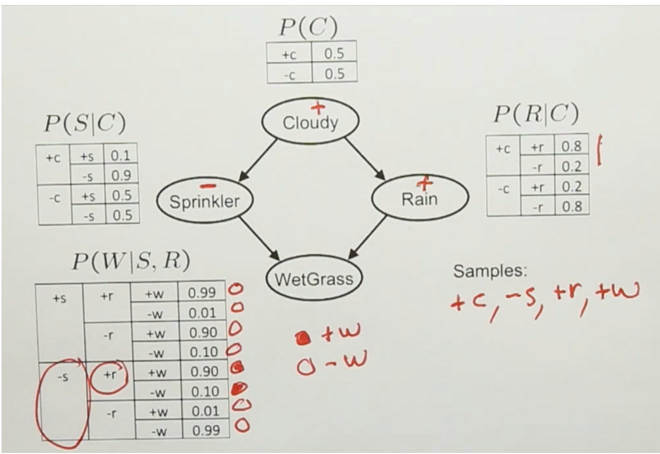
Variable Elimination

enumeration repeats many nodes

elimination: combine some variables, enumerate smaller networks & work on combined nodes

1. combine nodes into small network
2. sumout & eliminate small network

Approximate Inference (Sampling)



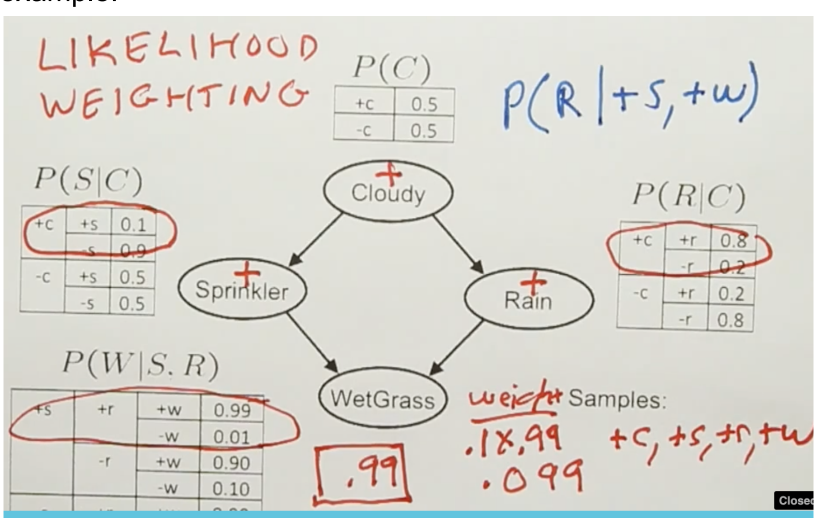
sample ancestor ->
sample next level conditional to ancestor output->
carry on->
until end of the network
-> repeat

sampling method is consistent
with inf samples, approach to true joint prob

rejection sampling
when compute conditional prob: remove samples that don't match with conditions and only keep those that do. e.g. when calculate P(+W | -C), the +c,-s, +r, +w sample is not useful -> remove
... but this end up generating lots unuseful samples

likelihood weighting
generate samples that we always can keep (e.g. C always negative, and generate S, R)
BUT, resulting samples are **inconsistent**

to fix it, assign likelihood to each sample & weighing correctly. it will make sampling **consistent**
example:



gibbs sampling
- takes all evidence (not just upstream evidence)
- uses MCMC
- **consistent**
1. generate random sample
2. pick ONE non-evidence variable, resample it ***based on*** other variables
3. repeat
(note variables are dependent in this gibbs sampling, which is opposite of rejection sampling)

monty hall
door 1: 1/3
door 2: 2/3
because we learned something new about second door, but nothing new about first door -- opening 1st door is never an option

Pattern Recognition Through Time

Dynamic Time Warping (DTW)

in time series analysis, DTW is an algorithm measuring similarity between two temporal sequences, which may vary in speed

Soe Chiba Bounds

won't allow match over certain bounds

HMMs

Markov Chain with self transition & out transition
expected time frame spent in a state = 1/(1-p) where p=self-transition prob
=> dummy state (initial X0, p =0, timeframe spent on X0: 1)

Q: what property does delta_y (observed sequence) help distinguish between two gestures ("we" & "i" in sign language)?
A: 1. probability distributions in middle states; 2. likely time spent in middle states

Viterbi Trellis

P(state s at t) = P(transition)*P(s)

Baum Welch

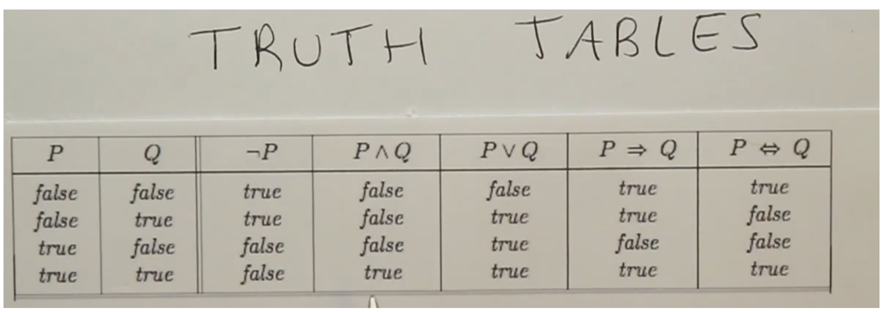
(EM)

Logic and Planning

Propositional Logic

(alarm example) Burglary, Earthquake, Alarm, MaryCalling, JohnCalling
use T/F
(E v B) => A (E or B is T implies A is T)
A => (J ^ M) (A implies J and M)
J <=> M (equivalent)
J <=> !M (opposite)
p.s. v: or; ^: and;

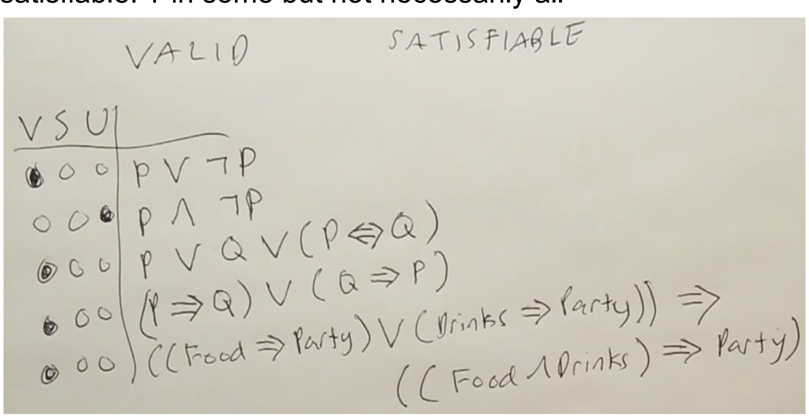
Truth Tables



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

true => true: true; false => false: true
different from ordinary english!

Terminology
valid: T in every model in every possible combination of symbols
satisfiable: T in some but not necessarily all



Propositional Logic Limitations

1. only T/F, no capability to handle uncertainty
2. can't tell objects or relationships b.w. objects
3. no shortcuts

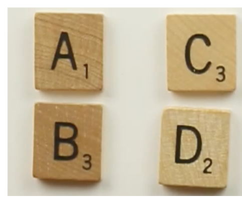
First Order Logic

types:

	world	beliefs
first-order logic:	relationship, objects, functions	T/F/unknown
propositional logic:	facts	T/F/unknown
probability theory:	facts	[0,1]

representation:
atomic: problem solving; transition b.w. A&B
factored:
structured:

models:



propositional: symbol {P: T, Q: F}
first-order:
- constants:{A,B,C,D,1,2,3,CEE}
- functions (mapping b.w. objects):
  - number of: {A->1, B->3, C->3, D->2}
- relations:
  - above: {[A,B], [C,D]}
  - vowel: {[A]}
  - rainy: {} not rainy {[]}