CRYPTArimemeric Problem

↳ Type of constrain satisfaction Problem

↳ Constraints
- ↳ digits can be assigned a word/alphabet (Range 0–9)
- ↳ no 2 letters have same value
- ↳ sum of digits must be a shown in problem
- ↳ only 1 carry forward

↳ Start from left most digit
- ↳ if it is carry
  - ↳ assign it 1 (+8 → always 1)

θ)

$$\begin{array}{r} T_2\ O_1 \\ +\ G\ O_1 \\ \hline O\ U\ T_2 \end{array}$$

as there is carry

$2 + G = 7\ 10$

9 | 8 — will be 10 or >

U=1 ✗ as O=1 can't be same

U=0 ✓

Letter	Digit
T	2
O	1
G	8
U	0

θ)

$$\begin{array}{r} C_5\ C_4\ C_3\ C_2\ C_1 \rightarrow \text{Carry} \\ S_9\ E_5\ N_6\ D_7 \\ +\ M_1\ O_0\ R_8\ E_5 \\ \hline M_1\ O_0\ N_6\ E_5\ Y_2 \end{array}$$

L	D
S	9
E	5
N	6
D	7
M	1
O	0
R	8
Y	2

1. $S + M > 10$ (as carry)
   $S + 1 > 10$
   ↓
   $9 + 1 = 0$

2. $E + O = N$
   $E + O + C_2 = N$
   ↓ ↓
   $5 + 0 + 1 = 6$

   if E+O=N, E=N ✗ ↳ when means there should be a carry $C_2 = 1$

3. $N + R = E$
   $6 + R = 5$
   ↓ ↘
   9 8
   $6 + 9 = 15$
   ↓ carry
   $S = 9$ ✗

   $6 + 8 + C_1 = 15$ (carry from forward)
   $6 + 8 + 1 = 15$

4. $D + 5 = Y + 10$
   ↓ ↓ ↘
   4 5 6
   $4 + 5 = 9$ ✗ (E=5 ✗, 6 ✗)
   $S = 9$
   $7 + 5 = 12$
   $y = 2$

$C_5\ C_4\ C_3\ C_2\ C_1$

a)
$$\begin{array}{r} E_8\ A_1\ T_9 \\ +\ T_9\ H_2\ A_1\ T_9 \\ \hline A_1\ P_0\ P_0\ L_3\ E_8 \end{array}$$

L	D
A	1
P	0
L	3
E	8
T	9
H	2

2. $T \neq P$ $\quad 8+H = 0+10$

$T+C_3 = C_4+P$

$9+1 = 10$

$C_5\ C_4\ C_3\ C_2\ C_1$

b)
$$\begin{array}{r} S_1\ O_9\ M_3\ E_4 \\ +\ T_8\ I_5\ M_3\ E_4 \\ \hline S_1\ P_0\ E_4\ N_6\ T_8 \end{array}$$

$MM = N$, $EE = T$ → EVEN → 0, 2, 4, 6, 8

$T=8 \rightarrow E=4$

$N=6 \rightarrow M=3$

$1+T+C_3 = 10$

8

$O+1 = E+10$

$= 4+10$

$O+1 = 14$

7,7 × 6,8 × 9,5

$C\ \ 1$

c)
$$\begin{array}{r} B_1\ A_9\ S_8\ E_3 \\ +\ B_1\ A_4\ L_5\ L_5 \\ \hline G_1\ A_4\ M_9\ E_3\ S_8 \end{array}$$

$E+L = S \rightarrow C$

$E+L = S+10 \rightarrow C$

↳ $E = S-L+10$

$S+L = E$

$S-L+10 = S+L$

$2L = 10$

$L = 5$

1. $B+B = A+C$

5 6 7 8 9

X

2. $S-E = 5$

5,0 ×

6,1 ×

7,2

8,3

9,4

Taking $S,E\ (7,2)$ $B(6)$

$6+6 = 12$

C A ×

$S,E\ (8,3)$ $B(7)$

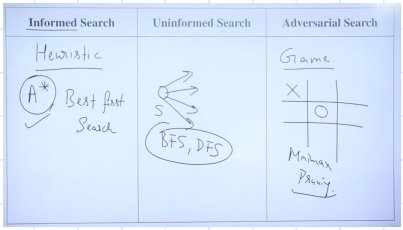
$7+7 = 14$

Adversial Search

↳ Enviornments
  ↳ determinishc
  ↳ fully observable

↳ competion b/w 2 people
↳ 1 persons loss, is me other gain
↳ both players have acess to complete info
↳ no chance → dice

e.g TIC TAK TOE
CHECKERS
CHESS



Evaluation function

- **Evaluation function** or **static evaluator** is used to evaluate the "goodness" of a game position
  - Contrast with heuristic search where evaluation function was a non-negative estimate of the cost from the start node to a goal and passing through the given node
- Zero-sum assumption lets us use a single evaluation function to describe goodness of a board wrt both players
  - **f(n) >> 0**: position n good for me and bad for you
  - **f(n) << 0**: position n bad for me and good for you
  - **f(n) near 0**: position n is a neutral position
  - **f(n) = +infinity**: win for me
  - **f(n) = -infinity**: win for you

Evaluation function examples

- Example of an evaluation function for Tic-Tac-Toe
  f(n) = [# of 3-lengths open for me] - [# of 3-lengths open for you]
  where a 3-length is a complete row, column, or diagonal
- Alan Turing's function for chess
  - **f(n) = w(n)/b(n)** where w(n) = sum of the point value of white's pieces and b(n) = sum of black's
- Most evaluation functions specified as a weighted sum of position features
  $f(n) = w_1*feat_1(n) + w_2*feat_2(n) + ... + w_n*feat_k(n)$
- Example features for chess are piece count, piece placement, squares controlled, etc.
- Deep Blue had >8K features in its evaluation function

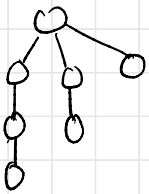
Introduction to Game Playing

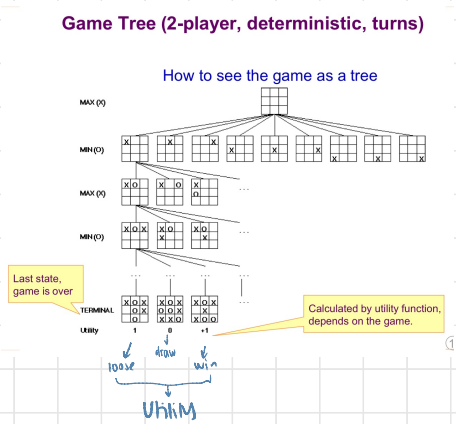
↳ Minimax Algo
↳ Alpha beta (α-β) Pruning

↳ use Depth First Search

Max: My turn
Min: Opponents turn
Utility: Winner Prize

b → branching factor =
m → max path length =





Minimax Algo → used in game play

↳ Backtracking Algo

↳ Best move strategy used

↳ Max will try to maximize its utility → Best move

↳ Min will try to minimize its utility → Worst move

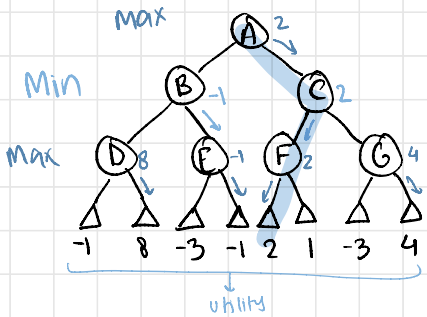
Time Complexity

↳ $O(b^d)$ — d: depth, b: branching factor

moves by max → always at top

counter moves by min

Game Tree

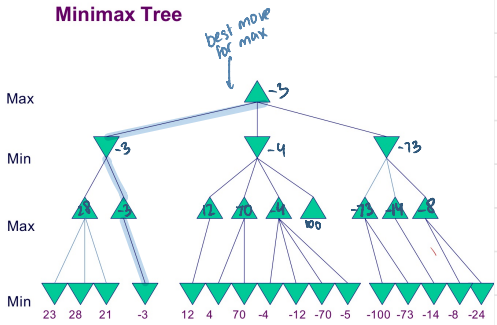


Minimax Tree Evaluation

For the MAX player

1. Generate the game as deep as time permits
2. Apply the evaluation function to the leaf states
3. Back-up values
   - At MIN assign minimum payoff move
   - At MAX assign maximum payoff move
4. At root, MAX chooses the operator that led to the highest payoff

Minimax Tree



Terminal nodes: values calculated from the utility function

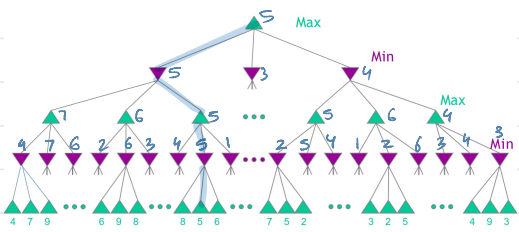
Properties

Complete	Yes, if tree is finite
Optimal	Yes, against an optimal opponent
Time Complexity	b^m
Space complexity	b^m

For chess, b ≈ 35, m ≈100 for "reasonable" games
→ exact solution completely infeasible, since it's too big

Instead, we limit the depth based on various factors, including time available.

MiniMax Example-2



Terminal nodes: values calculated from the utility function

Question No. 3 **CLO3 [Time: 20 Min] [Marks: 15 (10+5)]**

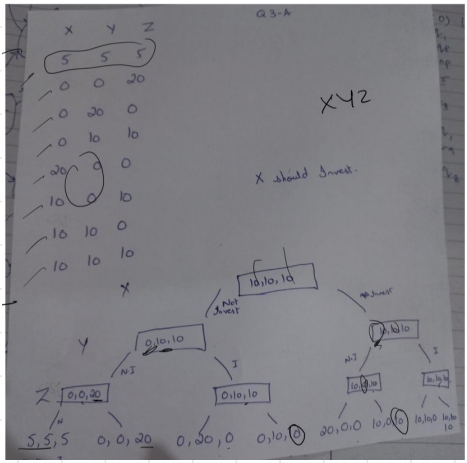
Part A) Under the Flag of FAST NUCES- IEEE an AI-Enabled Tech Exhibition 2022 project exhibition has been organized. This exhibition aims to provide students' work with broader visibility and make their work accessible and available for the interested people. With the policy of having 3 members in group and zero tolerance for plagiarism. If found it will lead to Disciplinary Action. So, you and two of your friends decided to participate (Consider yourself =X, Friend1=Y, Friend2= Z). X and Y sincerely did hard work, and both cooperated with each other to analyze and solve the occurring challenges. But Z always considered himself a mastermind and never participated in group activity. But suddenly he got stuck and copy pasted the code from internet without bringing this thing in his friends' knowledge.
During evaluation it was found that a particular chunk of code has been copied and now three of them will be called for DC. Where either each participant will Confess or Deny. As X and Y always worked as a team so they decided to give priority to co-operate (i.e. Confess or Deny) over opposite decision(X confess and Y Deny or vice versa).But always aiming for Z to Confess because he did wrong so only he will be liable for breaking rule. In case if in any circumstances both must take opposite decision still their aim would be same for Z to confess. Meanwhile Z realized that he did wrong and try to make sure his friends won't be suffering furthermore due to him.
So, using Min max draw a Game tree and Identify what decision should be taken by Z?
(Note: Tuple utilities to be assigned as X, Y, Z)

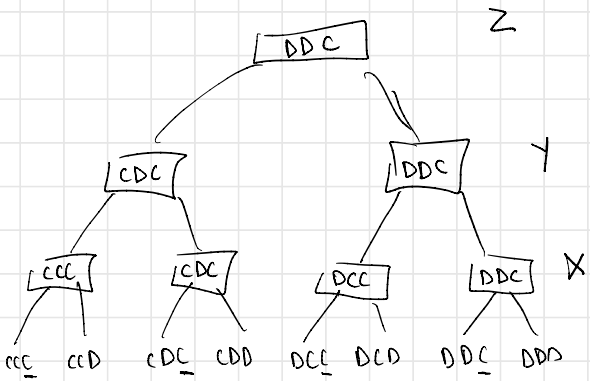
Part B) A game is played between max and min. Draw a tree considering MIN as a first player and branching factor is 2. Given terminal values below, show backed up values, best decision available at the root and branches that will get pruned (explain why they are getting pruned if any)?
5,6,4,10,7,8,4,5

A: Suppose that three companies, Company X, Company Y, and Company Z, are competing in the market for a new product. Each company can choose to invest in research and development (R&D) or not invest in R&D. The offer is:

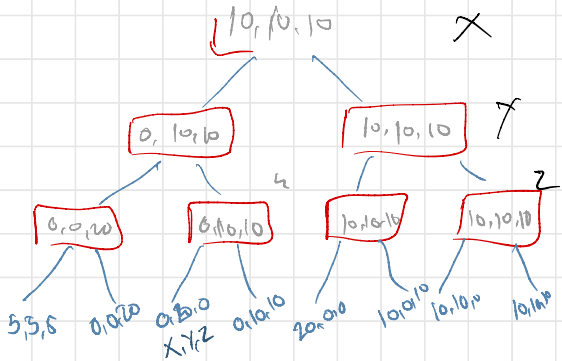
A. If all companies invest in R&D, they will all develop a high-quality product and receive a benefit of $10 million each. 1 1 1
B. If two companies invest in R&D, they will all develop a high-quality product and receive a benefit of $10 million each and the other companies receive nothing. 1 1 0
C. If only one company invests in R&D, that company will develop a superior product and receive a benefit of $20 million while the other companies receive nothing. 1 0 0
D. d)If no company invests in R&D, they will all develop a mediocre product and receive a benefit of $5 million each. 0 0 0

Now choose the best strategy for Company X using Min-Max concept and provide reasoning for your choice. Define properly how you generated utility values based on which the result/ strategy has been figured.





X	Y	Z
C	C	C
C	C	D
C	D	C
C	D	D
D	C	C
D	C	D
D	D	C
D	D	D



X	Y	Z
10	10	10
10	10	0
0	10	

P1 = Z confess
P2: Y, X opposi

Alpha - Beta Pruning (α-β)

(cut off)

→ improves performance of MinMax algo

↳ cut off search by exploring less no. of nodes

↳ if curr max > successor min
then dont explore that min subtree } → α cuts

↳ saves time

↳ avoids unnecessary computation

α → Max node

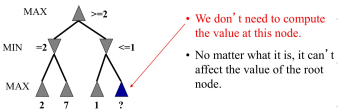
β → Min node

Time Complexity

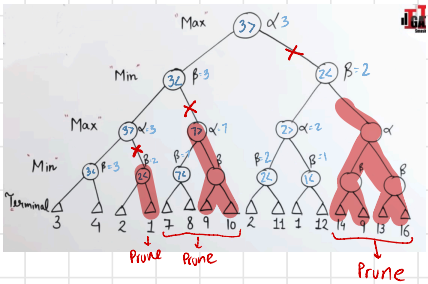
↳ $O(b^{d/2})$ → best/Avg case

↳ $O(b^d)$ → worst case

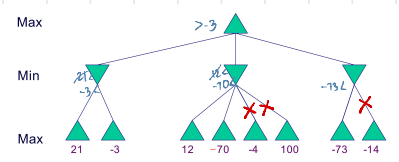
- Basic idea: *"If you have an idea that is surely bad, don't take the time to see how truly awful it is."* -- Pat Winston



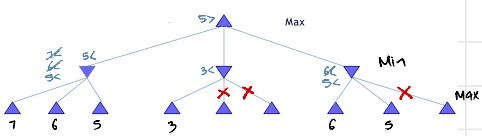
Pruning: ways of determining that certain branches will not be useful.



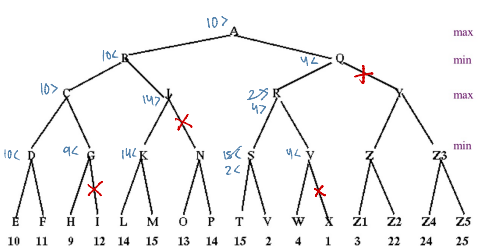
β Cut Example



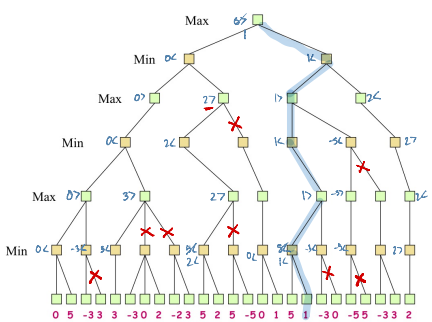
Alpha-Beta Example 2

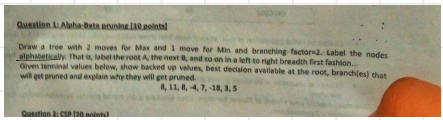


Exercise



Alpha-Beta Tic-Tac-Toe Example 2

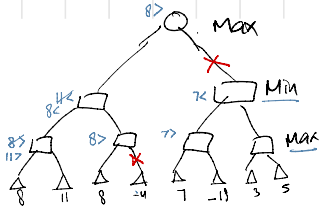



Question 1: Alpha-Beta pruning [10 points]

Draw a tree with 2 moves for Max and 1 move for Min and branching factor=2. Label the nodes alphabetically. That is, label the root A, the next B, and so on in a left to right breadth first fashion. Given terminal values below, show backed up values, best decision available at the root, branch(es) that will get pruned and explain why they will get pruned.

8, 11, 8, -4, 7, -11, 3, 5

Question 2: CSP [30 points]



Yes, BFS can be used instead of minmax in a gameplay to find all the game states that can be obtained in upto x moves from some starting game state using a queue.

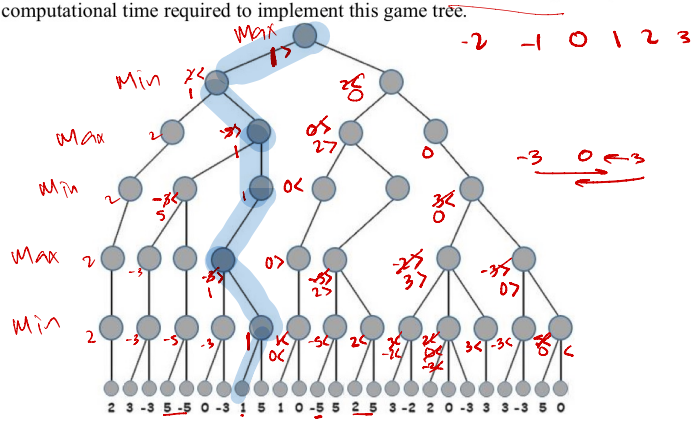
- Using BFS, all possible sequence of N moves can be easily figured out from starting position and situation after each sequence of N moves.
- In a BFS all reachable N moves can be easily found. The game states are reachable in N moves and will be at Nth-level of game state tree rooted at 's'(state). As BFS do level wise traversal.

& BFS can be used but there are some drawbacks i.e:

→ we have to store all the nodes (level wise) and their children nodes until you reach terminating search depth. consuming more memory as compared to minmax/DFS which require linear amount of memory.

A) Can Breadth First Search (BFS) be used instead of Min-Max Algorithm in game play? Justify your answer.

B) Use Min-Max algorithm on the following game tree to find the best possible path for Max player that ensures to maximize its score. Also, apply Alpha-Beta pruning to reduce the computational time required to implement this game tree.



α-β Pruning

Can store information along an entire *path*, not jus
at most recent levels!

Keep along the path:

α: best MAX value found on this path
(initialize to most negative utility value)

β: best MIN value found on this path
(initialize to most positive utility value)

Pruning at MAX node

α is possibly updated by the MAX of successors evaluated so far

If the value that would be returned is ever > β, then stop work on this branch

If all children are evaluated without pruning, return the MAX of their values

Pruning at MIN node

β is possibly updated by the MIN of successors evaluated so far

If the value that would be returned is ever < α, then stop work on this branch

If all children are evaluated without pruning, return the MIN of their values

Why is it called α-β?

- α is the value of the best (i.e., highest-value) choice found so far at any choice point along the path for *max*
- If *v* is worse than α, *max* will avoid it

  → prune that branch
- Define β similarly for *min*

Imperfect Decisions

Complete search is impractical for most games

Alternative: search the tree only to a certain depth

- Requires a cutoff-test to determine where to stop
  - Replaces the terminal test
  - The nodes at that level effectively become terminal leave nodes
- Uses a heuristics-based evaluation function to estimate the expected utility of the game from those leave nodes.

Utility Evaluation Function

Very game-specific

Take into account knowledge about game

"Stupid" utility
- 1 if player 1 wins
- -1 if player 0 wins
- 0 if tie (or unknown)
- Only works if we can evaluate complete tree
- But, should form a basis for other evaluations

Utility Evaluation

Need to assign a numerical value to the state
- Could assign a more complex utility value, but then the min/max determination becomes trickier.

Typically assign numerical values to lots of individual factors:
- a = # player 1's pieces - # player 2's pieces
- b = 1 if player 1 has queen and player 2 does not, -1 if the opposite, or 0 if the same
- c = 2 if player 1 has 2-rook advantage, 1 if a 1-rook advantage, etc.

Utility Evaluation

The individual factors are combined by some function

Usually a linear weighted combination is used:
- $u = \alpha a + \beta b + \chi c$
- Different ways to combine are also possible

Notice: quality of utility function is based on:
- What features are evaluated
- How those features are scored
- How the scores are weighted/combined

Absolute utility value doesn't matter – relative value does.

Evaluation Functions

If you had a perfect utility evaluation function, what would it mean about the minimax tree?

You would never have to evaluate more than one level deep!

Typically, you can't create such perfect utility evaluations, though.

Evaluation Functions for Ordering

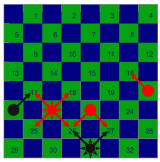
As mentioned earlier, order of branch evaluation can make a big difference in how well you can prune

A good evaluation function might help you order your available moves:

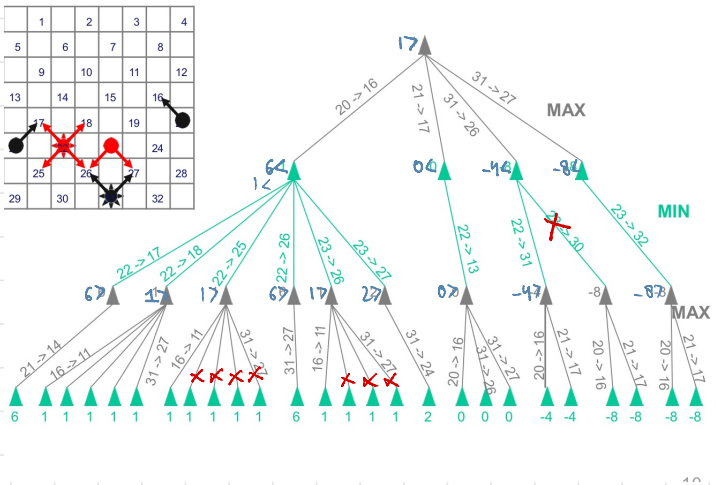
- Perform one move only
- Evaluate board at that level
- Recursively evaluate branches in order from best first move to worst first move (or vice-versa if at a MIN node)

Checkers Case Study

- Initial board configuration
  - Black: single on 20, single on 21, king on 31
  - Red: single on 23, king on 22
  - Evaluation function
    $E(s) = (5\,x_1 + x_2) - (5r_1 + r_2)$
    where
    x_1 = black king advantage,
    x_2 = black single advantage,
    r_1 = red king advantage,
    r_2 = red single advantage



Checkers MiniMax Example



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

S ← → B

ishma hafeez notes

Summary

- CSPs are a special kind of problem: states defined by values of a fixed set of variables, goal test defined by constraints on variable values
- Backtracking=depth-first search with one variable assigned per node
- Variable ordering and value selection heuristics help significantly
- Forward checking prevents assignments that lead to failure.
- Constraint propagation does additional work to constrain values and detect inconsistencies.
- The CSP representation allows analysis of problem structure. Tree
- structured CSPs can be solved in linear time.
- Iterative min-conflicts is usually effective in practice.

A) Explain the following in one or two statements.

1) **Zero-Sum game**: Agents have opposite utilities (values on outcomes), single value that one maximizes and the other minimizes
2) **Successor function** in min-max game: It defines what the legal moves a player can make are.
3) **Significance of game tree**: Game tree shows the best possible move of max and min player at each level.
4) **Types of constraints:** Unary Constraint and binary constraint.
5) **backtracking search** is used for a depth-first search that chooses values for one variable at a time and backtracks when a variable has no legal values left to assign.
6) **Binary CSP:** each constraint relates (at most) two variables
7) **Logic:** Logic is a formal system for manipulating facts so that true conclusions may be drawn
8) **Propositional Logic:** concrete statements that are either true or false and **Predicate Logic:** allows statements to contain variables, functions, and quantifiers
9) **Knowledge Base:** A set of sentences that encodes assertions about the world in a formal knowledge representation language.
10) Game theory is powerful tool for modeling cooperative behavior in many wireless networking applications such as cognitive radio networks, wireless system, physical layer security, virtual MIMO, among others.

Bayesian Network

↳ used when representing uncertainty

↳ uses a conditional probability table (CPT)

↳ uses Directed Acyclic Graph (DAG)

eg $P(b|e,m) = \frac{P(b,e,m)}{P(e,m)}$

ignored ←

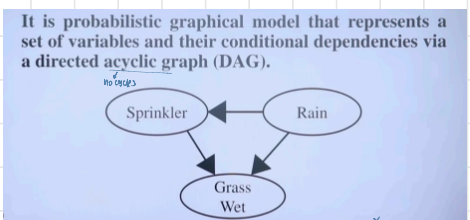
= P sab kay parents

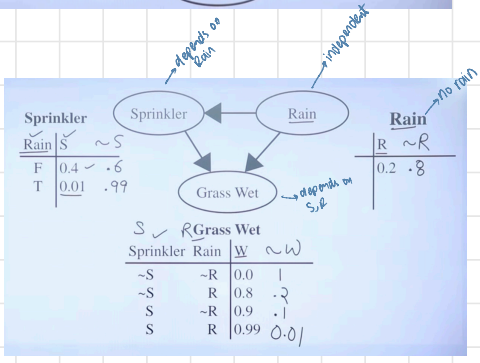
steps

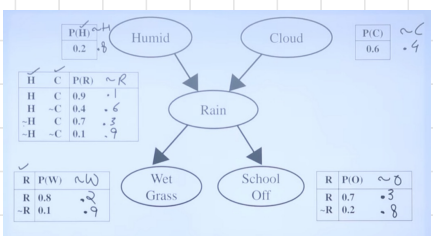
↳ Make CPT of each

↳ simple or independent variable

↳ for dependent variable







Q) P(W, S, R)

= P(W ∩ S ∩ R)

= P(W|S,R) × P(S|R) × P(R)

= 0.99 × 0.01 × 0.2

=

Q) P(W, S̄, R̄)

= P(W|S̄R̄) × P(S̄|R̄) × P(R̄)

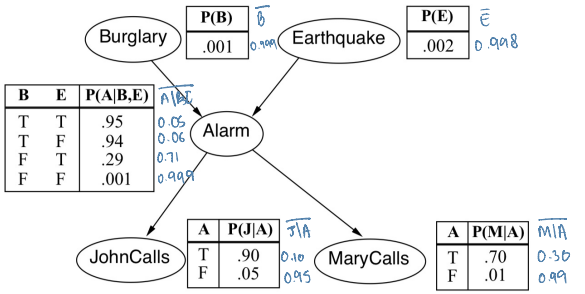
= 0.0 × 0.6 × 0.8

= 0

Q) P(O, R, H̄, C)

= P(O|R) × P(R|H̄, C) × P(H̄) × P(C)

= 0.7 × 0.1 × 0.8 × 0.6



8) $P(J, M, A, \bar{B}, \bar{E})$

- $P(J|A) \times P(M|A) \times P(A|\bar{B}, \bar{E}) \times P(\bar{B}) \times P(\bar{E})$
- $0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998$

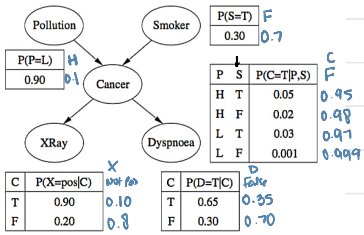
a) Draw the Bayesian Network that corresponds to the following conditional probability:

P (A | B,C,E) P(B | D,G) P(C | E,F,H) P(D | G) P(E| G,H) P(F | I) P(G | H) P(H | I) P(I)

b) Suppose we have a robot on a grid world with a noisy sensor and we know the distribution for P(Observation|Location) of sensor readings given the robot's grid square. We also have knowledge about the locations in the form of a prior, P(Location). However, we are really interested in the probability of being in a grid square given the sensor reading, P(Location|Observation). How could you compute this probability? Short answer.

c) Consider the Bayesian Network given below.

1. Calculate P(Cancer)
2. Calculate P(Pollution|Cancer)
3. Calculate P(Xray|Cancer)
4. Calculate P(Dyspnoea|Cancer)



P(P=L)
0.90

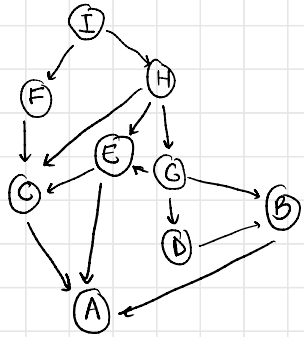
P(S=T)
0.30

P	S	P(C=T|P,S)
H	T	0.05
H	F	0.02
L	T	0.03
L	F	0.001

C	P(X=pos|C)
T	0.90
F	0.20

C	P(D=T|C)
T	0.65
F	0.30

a)



b) $P(L|O) = \frac{P(O|L) \times P(L)}{P(O)}$ (P(L) given)

ci) $P(\text{Cancer}) = P(\text{Cancer}|S,P) \times P(S) \times P(P)$

C	S	P	C=T|S,P	× S	× P	=
	0	0	0.001	0.7	0.9	0.00063
	0	1	0.07	0.7	0.1	0.0014
	1	0	0.03	0.3	0.9	0.0081
	1	1	0.05	0.3	0.1	0.0015
					+	0.01163

cii) $P(P|C) = \frac{P(C|P) \times P(P)}{P(C)}$ ∴ P(P=L) = 0.90, ∴ P(C=T) = 0.01163

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

S	C=T|P=L,S	S	
0	0.001	0.7	= 0.0007
1	0.03	0.3	= 0.009
			0.0097

$P(P|C) = \frac{0.0097 \times 0.9}{0.01163} = 0.7506$

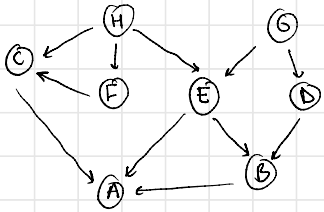
ciii) $P(X|C) = 0.9$

civ) $P(D|C) = 0.65$

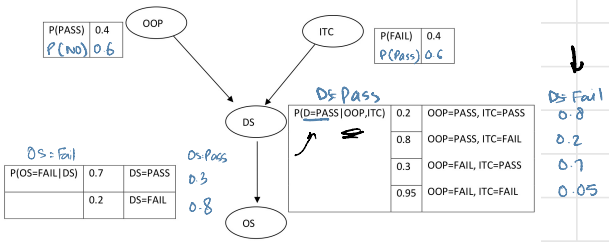
$P(X|C) = \frac{P(C|X)P(X)}{P(C)}$

$P(C|X) = \frac{P(C,X)}{P(X)} = P(C=T|P,S) \times P(P) \times P(S) \times P(X|C)$

a. Draw the **Bayesian Network** that corresponds to this conditional probability: P (A | B, C, E) P (B | D, E) P (C | F, H) P (D | G) P (E| G, H) P (F | H) P(G) P(H).



b. Calculate P(OS=PASS | ITC=PASS) using the network given below:



a) $P(OS=Pass \mid ITC=Pass)$

$$\frac{P(OS=Pass,\ P(ITC=Pass)}{P(ITC)=Pass}$$

$$\frac{P(O|DS)\times P(DS|ITC,OOP)\times P(\cancel{ITC})\times P(OOP)}{P(\cancel{ITC})}$$

2

DS	OOP	(O=Pass | DS)	DS | ITC=Pass, OPP	P(OOP)	
1	1	0.3	0.2	0.4	= 0.024
1	0	0.3	0.3	0.6	= 0.054
0	1	0.8	0.8	0.4	= 0.256
0	0	0.8	0.7	0.6	= 0.336

0.67

Semantics of Bayesian Networks

- Representing full joint probability distribution
  - method for constructing Bayesian network
    - compactness and node ordering
- conditional independence relation in Bayesian network

Construction of Bayesian Networks from Probabilities

1. No Parent

   $P(A)$

2. 1 Parent

   $P(A|B)$

3. 2 Parents

   $P(A|B,C)$

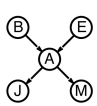
4. 3 Parents

   $P(A|B,C,D)$

Global semantics

"Global" semantics defines the full joint distribution as the product of the local conditional distributions:

$$P(x_1, \ldots, x_n) = \prod_{i=1}^{n} P(x_i | parents(X_i))$$



e.g., $P(j \wedge m \wedge a \wedge \neg b \wedge \neg e)$

$$= P(j|a)P(m|a)P(a|\neg b, \neg e)P(\neg b)P(\neg e)$$
$$= 0.9 \times 0.7 \times 0.001 \times 0.999 \times 0.998$$
$$\approx 0.00063$$

↓

How we solve it

$P(A=on) = 0.6$

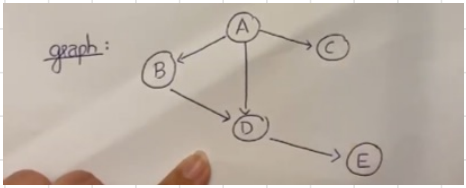
$$P(B=on \mid A) = \begin{cases} 0.1 & A=off \\ 0.95 & A=on \end{cases}$$

$$P(C=on \mid A) = \begin{cases} 0.8 & A=off \\ 0.5 & A=on \end{cases}$$

$$P(D=on \mid A,B) = \begin{cases} 0.1 & A=off,\ B=off \\ 0.9 & A=on,\ B=off \\ 0.3 & A=off,\ B=on \\ 0.95 & A=on,\ B=on \end{cases}$$

$$P(E=on \mid D) = \begin{cases} 0.8 & D=off \\ 0.1 & D=on \end{cases}$$

graph:



$$P(E=on \mid A=on) = \frac{P(E=on,\ A=on)}{P(A=on)}$$

$$= \frac{P(E=on \mid D) \times P(D \mid A=on, B) \times P(A=on) \times P(B \mid A=on)}{P(A=on)}$$

$$= P(E=on \mid D) \times P(D \mid A=on, B) \times P(B \mid A=on)$$

as we dont know whether these are on or off

D	B	$P(E=on \mid D)$	$P(D \mid A=on, B)$	$P(B \mid A=on)$	
off	on	0.8	0.05	0.95	=
on	off	0.1	0.9	0.05	=
on	on	0.1	0.95	0.95	=
off	off	0.8	0.1	0.05	=

sum

$(0.8 \times 0.05 \times 0.95) + (0.1 \times 0.9 \times 0.05) + (0.1 \times 0.95 \times 0.95) + (0.8 \times 0.1 \times 0.05) = 0.13675$

solved in previous question

(3) $P(A=on \mid E=on) = \frac{P(E=on \mid A=on) \times P(A=on)}{P(E=on)}$

$= \frac{0.13675 \times 0.6}{P(E=on)}$

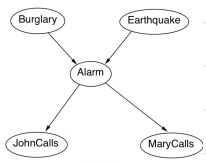
not taking A=on as we are only solving $P(E=on)$ which doesnt give us info on A

Denominator: $P(E=on) = P(E=on \mid D) \times P(D \mid A,B) \times P(A) \times P(B \mid A)$

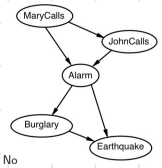
A	B	D	$P(E=on \mid D)$	$P(D \mid A,B)$	$P(A)$	$P(B \mid A)$	
on	on	off	0.8		0.6		0.13675 x 0.6
on	off	on			0.6		
on	on	on			0.6		
on	off	off			0.6		
off	on	off	0.8	0.7	0.4	0.1	
off	off	on	0.1	0.1	0.4	0.9	
off	on	on	0.1	0.3	0.4	0.1	
off	off	off	0.8	0.9	0.4	0.9	

Compactness

↳ a general property of locally structured (aka sparse) systems



A	A	B,E		
M	J	A	B	E
2^1	2^1	2^2	2^0	2^0
2	2	4	1	1 = 10



No

	M	M,J	A	B,A	
M	J	A	B	E	
2^0	2^1	2^2	2^1	2^2	
1	2	4	2	4	= 13

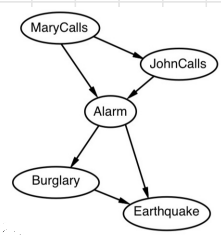
Network is less compact, less complex

Node ordering

Q) M, J, A, B, E

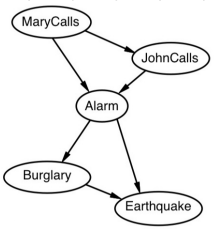
(J: root)

- Suppose we decide to add the nodes in the order MaryCalls, JohnCalls, Alarm, Burglary, Earthquake.
- Adding Mary Calls: No parents. M
- Adding JohnCalls: If Mary calls, that probably means the alarm has gone off, which of course would make it more likely that John calls. Therefore, JohnCalls needs Mary Calls as a parent — J|M
- Adding Alarm: Clearly, if both call, it is more likely that the alarm has gone off than if just one or neither call, so we need both MaryCalls and JohnCalls as parents. — A|M,J
- Adding Burglary: If we know the alarm state, then the call from John or Mary might give us information about our phone ringing or Mary's music, but not about burglary: — B|A
- **P(Burglary|Alarm, JohnCalls, MaryCalls) = P(Burglary|Alarm)**
- Hence we need just Alarm as parent.
- Adding *Earthquake*: if the alarm is on, it is more likely that there has been an earthquake. But if we know that there has been a burglary, then that explains the alarm, and the probability of an earthquake would be only slightly above normal. Hence, we need both *Alarm* and *Burglary* as parents. — E|B,A



$P(J|M) = P(J)$? No
$P(A|J,M) = P(A|J)$? $P(A|J,M) = P(A)$? No
$P(B|A,J,M) = P(B|A)$? Yes
$P(B|A,J,M) = P(B)$? No
$P(E|B,A,J,M) = P(E|A)$? No
$P(E|B,A,J,M) = P(E|A,B)$? Yes

↳ Conditional independence



↳ A node is conditionally independent of it non descendants, given its parents

$$X_i \perp \text{non descendants}(X_i) \mid \text{Parents}(X_i)$$

($\perp$: independent; $\mid$: given)

1. J independent of B, E given A
   ↳ J conditionaly independent of B, E
   ↳ J dependent on A

2. B independent of J, M given A
   ↳ B conditionaly independent of J, M
   ↳ B dependent on A, E

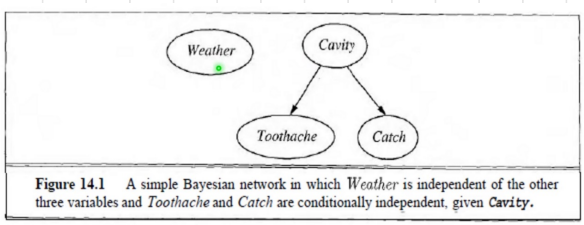


Figure 14.1 A simple Bayesian network in which *Weather* is independent of the other three variables and *Toothache* and *Catch* are conditionally independent, given **Cavity**.

W independent of other 3 variables

Toothache conditionaly independent of Catch, given cavity

Toothache and catch independent to each other but both depend on cavity

D seperation

X, Y = independent variable

E = set of observed variables

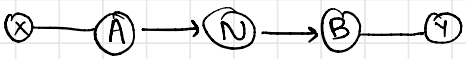
↳ E d-separates X and Y

if E blocks every un-directed Path b/w X and Y

(any arrow direction matter direction)

↳ If E d-separates X and Y

then X and Y are independent, given E

1) 

independent

if N is observed

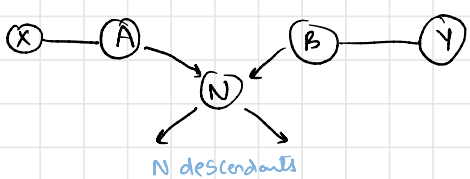
it blocks the path b/w X and Y

2) 

independent

if N is observed

it blocks the path b/w X and Y

3) 

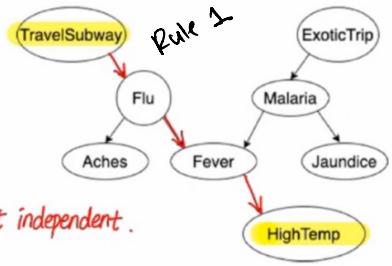
NOT independent if observed

if N and Ns descendants are NOT observed

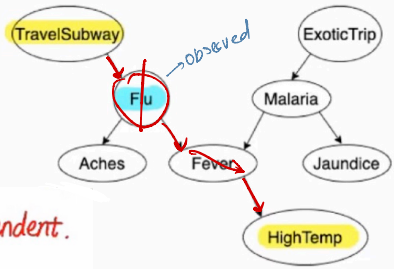
then they block the path b/w X and Y

CQ: Applying D-separation

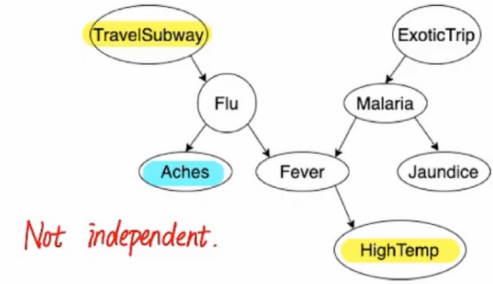
CQ 1a: Are **TravelSubway** and **HighTemp** independent?



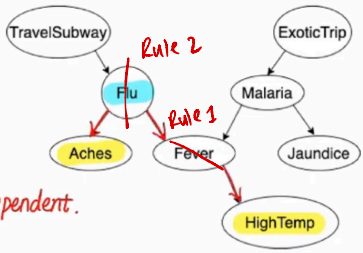
CQ 1b: Are **TravelSubway** and **HighTemp** conditionally independent given **Flu**?



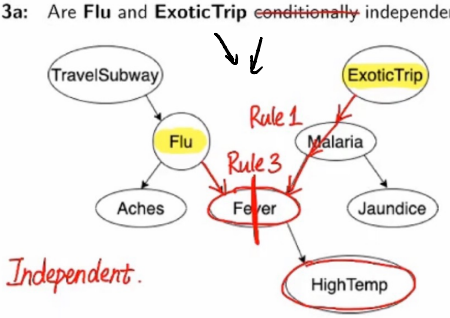
CQ 1c: Are **TravelSubway** and **HighTemp** conditionally independent given **Aches**?



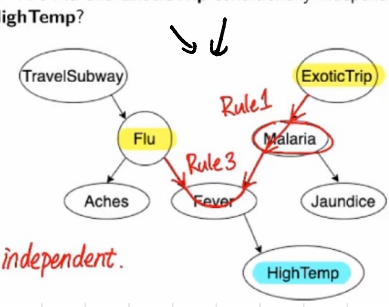
CQ 2b: Are **Aches** and **HighTemp** conditionally independent given **Flu**?



CQ 3a: Are **Flu** and **ExoticTrip** ~~conditionally~~ independent?



CQ 3b: Are **Flu** and **ExoticTrip** conditionally independent given **HighTemp**?



grey = dependent area

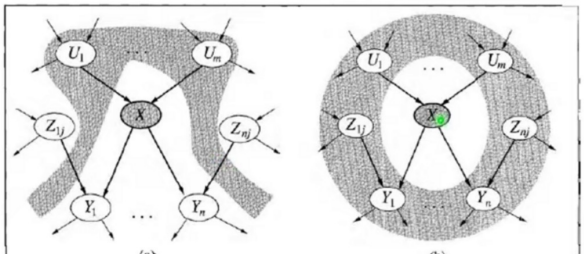


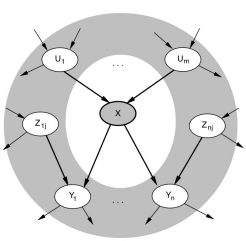
Figure 14.4 (a) A node X is conditionally independent of its non-descendants (e.g., the Z_{ij}s) given its parents (the U_is shown in the gray area). (b) A node X is conditionally independent of all other nodes in the network given its Markov blanket (the gray area).

a) X independent to Z_{ij} given U_i → its parents
- ↳ X conditionaly independent of Z_{ij}
- ↳ X dependent on U_i

b) X independent to all nodes
- ↳ X conditionaly independent of all nodes

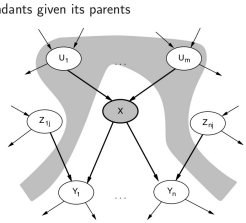
Markov blanket

Each node is conditionally independent of all others given its Markov blanket: parents + children + children's parents



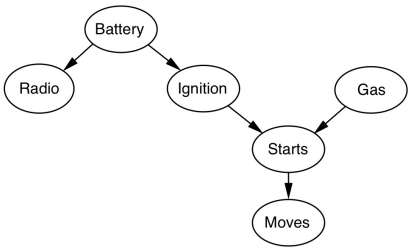
Local semantics

Local semantics: each node is conditionally independent of its nondescendants given its parents



Theorem: Local semantics ⇔ global semantics

Example



Are Gas and Radio independent? Given Battery? Ignition? Starts? Moves?

Compact conditional distributions

CPT grows exponentially with number of parents
CPT becomes infinite with continuous-valued parent or child

Solution: canonical distributions that are defined compactly

Deterministic nodes are the simplest case:
$X = f(Parents(X))$ for some function f

E.g., Boolean functions
$NorthAmerican \Leftrightarrow Canadian \vee US \vee Mexican$

E.g., numerical relationships among continuous variables

$$\frac{\partial Level}{\partial t} = \text{ inflow + precipitation - outflow - evaporation}$$

Compact conditional distributions contd.

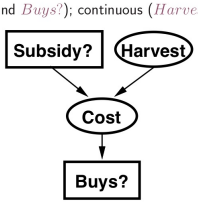
Noisy-OR distributions model multiple noninteracting causes
1) Parents $U_1 \ldots U_k$ include all causes (can add leak node)
2) Independent failure probability q_i for each cause alone
$\Rightarrow P(X|U_1 \ldots U_j, \neg U_{j+1} \ldots \neg U_k) = 1 - \prod_{i=1}^{j} q_i$

$Cold$	Flu	$Malaria$	$P(Fever)$	$P(\neg Fever)$
F	F	F	**0.0**	1.0
F	F	T	0.9	**0.1**
F	T	F	0.8	**0.2**
F	T	T	0.98	$0.02 = 0.2 \times 0.1$
T	F	F	0.4	**0.6**
T	F	T	0.94	$0.06 = 0.6 \times 0.1$
T	T	F	0.88	$0.12 = 0.6 \times 0.2$
T	T	T	0.988	$0.012 = 0.6 \times 0.2 \times 0.1$

Number of parameters **linear** in number of parents

Hybrid (discrete+continuous) networks

Discrete ($Subsidy?$ and $Buys?$); continuous ($Harvest$ and $Cost$)



Option 1: discretization—possibly large errors, large CPTs
Option 2: finitely parameterized canonical families

1) Continuous variable, discrete+continuous parents (e.g., $Cost$)
2) Discrete variable, continuous parents (e.g., $Buys?$)

Continuous child variables

Need one conditional density function for child variable given continuous parents, for each possible assignment to discrete parents

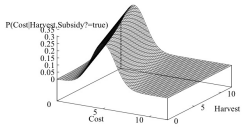
Most common is the linear Gaussian model, e.g.,:

$$\begin{aligned} &P(Cost = c | Harvest = h, Subsidy? = true) \\ &= N(a_t h + b_t, \sigma_t)(c) \\ &= \frac{1}{\sigma_t \sqrt{2\pi}} exp\left(-\frac{1}{2}\left(\frac{c - (a_t h + b_t)}{\sigma_t}\right)^2\right) \end{aligned}$$

Mean $Cost$ varies linearly with $Harvest$, variance is fixed

Linear variation is unreasonable over the full range
but works OK if the **likely** range of $Harvest$ is narrow

Continuous child variables



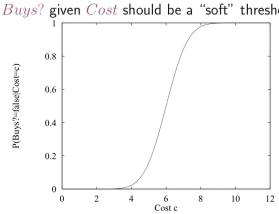
All-continuous network with LG distributions

$\Rightarrow$ full joint distribution is a multivariate Gaussian

Discrete+continuous LG network is a conditional Gaussian network i.e., a multivariate Gaussian over all continuous variables for each combination of discrete variable values

Discrete variable w/ continuous parents

Probability of $Buys?$ given $Cost$ should be a "soft" threshold:



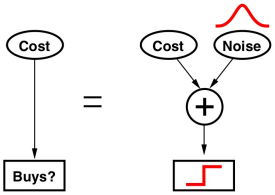
Probit distribution uses integral of Gaussian:

$\Phi(x) = \int_{-\infty}^{x} N(0,1)(x)dx$

$P(Buys? = true \mid Cost = c) = \Phi((-c + \mu)/\sigma)$

Why the probit?

1. It's sort of the right shape
2. Can view as hard threshold whose location is subject to noise

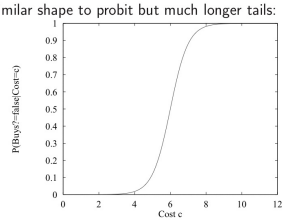


Discrete variable contd.

Sigmoid (or logit) distribution also used in neural networks:

$$P(Buys? = true \mid Cost = c) = \frac{1}{1 + exp(-2\frac{-c+\mu}{\sigma})}$$

Sigmoid has similar shape to probit but much longer tails:



Summary

Bayes nets provide a natural representation for (causally induced) conditional independence

Topology + CPTs = compact representation of joint distribution

Generally easy for (non)experts to construct

Canonical distributions (e.g., noisy-OR) = compact representation of CPTs

Continuous variables $\Rightarrow$ parameterized distributions (e.g., linear Gaussian)

BAYES THEOREM

Conditional Probability

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

A will occur given that B already

BAYES RULE

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

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

$$\hookrightarrow P(A|B)\,P(B) = P(B|A)\,P(A)$$

Examples

Suppose there are three bowls B1, B2, B3 and bowl B1 has 2 red and 4 blue coins; bowl B2 has 1 red and 2 blue coins; bowl B3 contains 5 red and 4 blue coins.Suppose the probabilities for selecting the bowls is not the same but are:-

- P(B1) = 1/3
- P(B2) = 1/6
- P(B3) = 1/2

Now, let us compute, assuming that a red coin was drawn what will be the probability that it came from bowl B1.

a) $P(R) = P(B1 \cap R) + P(B2 \cap R) + P(B3 \cap R)$

= P(selecting B1) * P(Number of Red coins / total number of coins in B1) + P(selecting B2) * P(Number of Red coins in B2 / total number of coins in B2) + P(selecting B3) * P(Number of Red coins in B3/ total number of coins in B3)

$$= \frac{1}{3} \times \frac{2}{6} + \frac{1}{6} \times \frac{1}{3} + \frac{1}{2} \times \frac{5}{9}$$

$$= \frac{4}{9}$$

b) $P(R|B1) = \frac{P(B1 \cap R)}{P(B1)} = \frac{\left(\frac{1}{3} \times \frac{2}{6}\right)}{4/9}$

$$= \frac{2}{8}$$

NAIVE BAYES CLASSIFIER

↳ supervised learning

↳ based on applying Bayes Theorem

↳ assumes no dependencies b/w features

BAYES RULE

$$P(Y|X) = \frac{P(X|Y) \times P(Y)}{P(X)}$$

$$P(Y|X_1, X_2, \ldots X_n) = P(X_1|Y) \times P(X_2|Y) \ldots P(X_3|Y) \times P(Y)$$

Person	X_1 COVID (Yes/No)	X_2 Flu (Yes/No)	Fever (Yes/No)
1	Yes	No	Yes
2	No	Yes	Yes
3	Yes	Yes	Yes
4	No	No	No
5	Yes	No	Yes
6	No	No	Yes
7	Yes	No	Yes
8	Yes	No	No
9	No	Yes	Yes
10	No	Yes	No

Step-1: Prior Probability:

P(fever = yes) = 7/10

P(fever = no) = 3/10

Step-2: Conditional Probability:

	Yes	No
Covid	Covid, Fever 4/7 (fever yes)	Covid, fever 2/3 (fever NO)
Flu	Flu, Fever 3/7	Flu, Fever 2/3

Given Person(flu, Covid)

P(yes| flu, Covid)= P(flu|yes)*P(covid|yes)* P(yes)

P(NO|flu, Covid)= P(flu|NO)*P(covid|No)*P(NO)

Q) Given Person(Flu, Covid), is fever yes OR NO

$P(\text{yes}|\text{Flu, Covid}) = P(\text{Flu}|\text{yes}) \times P(\text{covid}|\text{yes}) \times P(\text{yes})$

$= \frac{3}{7} \times \frac{4}{7} \times \frac{7}{10}$

$= 0.17$

↓

higher prob, so Prediction is yes fever

$P(\text{NO}|\text{Flu, Covid}) = P(\text{Flu}|\text{NO}) \times P(\text{covid}|\text{NO}) \times P(\text{NO})$

$= \frac{2}{3} \times \frac{2}{3} \times \frac{3}{10}$

$= 0.13$

fruit = {Yellow, Sweet, long}

Fruit	Yellow	Sweet	Long	Total
Orange	350	450	0	650
Banana	400	300	350	400
Others	50	100	50	150
Total	800	850	400	1200

$$P(\text{Yellow}|\text{orange}) = \frac{P(\text{orange}|\text{Yellow})\,P(\text{Yellow})}{P(\text{orange})}$$

$$= \frac{\frac{350}{800} \times \frac{800}{1200}}{\frac{650}{1200}} = 0.53$$

$$P(\text{sweet}|\text{orange}) = \frac{P(O|S)\,P(S)}{P(O)} = 0.69$$

$P(L|O) = 0$

$P(\text{fruit}|\text{orange}) = P(Y|O) \times P(S|O) \times P(L|O)$
$= 0.53 \times 0.69 \times 0 = 0$

$P(\text{fruit}|\text{Banana}) = P(Y|B) \times P(S|B) \times P(L|B)$
$= 1 \times 0.75 \times 0.89 = 0.65$

$P(\text{fruit}|\text{others}) = P(Y|O) \times P(S|O) \times P(L|O)$
$= 0.33 \times 0.66 \times 0.33 = 0.072$

TYPES OF NAIVE BAYES A190

1. Gaussian

↳ used in classification

↳ assumes features follow a normal distribution

2. Bernoulli

↳ binomial model

↳ used when features are binary

e.g. text classification with a bag of words

zeros and ones). One application would be text classification with 'bag of words' model where the 1s & 0s are "word occurs in the document" and "word does not occur in the document" respectively.

3. Multinomial Naive Bayes

↳ used on multinomially distributed data

For example, let's say, we have a text classification problem. Here we can consider bernoulli trials which is one step further and instead of "word occurring in the document", we have "count how often word occurs in the document", you can think of it as "number of times outcome number x_i is observed over the n trials".

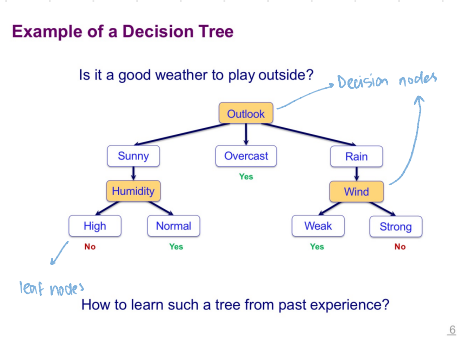
What to Do About Errors

- Problem: there's still spam in your inbox
- Need more features – words aren't enough!
  - Have you emailed the sender before?
  - Have 1M other people just gotten the same email?
  - Is the sending information consistent?
  - Is the email in ALL CAPS?
  - Do inline URLs point where they say they point?
  - Does the email address you by (your) name?
- Naïve Bayes models can incorporate a variety of features, but tend to do best in homogeneous cases (e.g. all features are word occurrences)

Decision Trees

Decision Tree

↳ It is a tree structure classifier
↳ a graphical representation
of all possible decisions related to a problem
↳ used in
↳ classification → using decision tree classifiers
↳ regression

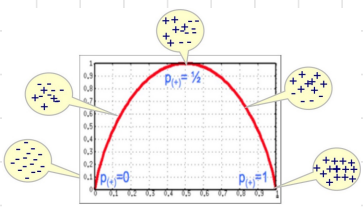


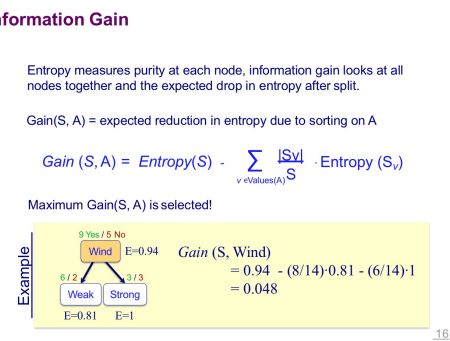
Entropy

↳ tells how pure/impure a data set is

$$E\{+X,-Y\} = -\left(\frac{X}{N}\log_2\frac{X}{N}\right)-\left(\frac{Y}{N}\log_2\frac{Y}{N}\right)$$

(N → no. of rows)





Pros

↳ Interpretable
↳ humans can understand decisions
↳ Easily handles irrelevant attributes e.g. G=0
↳ very compact
↳ no. of nodes << D after pruning
↳ fast at testing time
↳ O(Depth)

Cons

↳ Greedy Approach
↳ may not find best tree
↳ Instances are represented by attribute pairs → Outlook: sunny, Wind: strong
↳ what if we have discrete input values
↳ target function has discrete output values → Yes, NO
↳ hence cant have continuous no. output values
↳ training data may contain errors / miss attributes
↳ Uncertainty in the data

Uncertainty in the data (e.g., suppose we have two exact days/features, one with "yes" and one with "no". ➔ no classifier can help in such totally Uncertain data.

Day	Weather	Temperature	Humidity	Wind	Play Football?
Day 1	Sunny	Hot	High	Weak	No
Day 2	Sunny	Hot	High	Strong	No
Day 3	Cloudy	Hot	High	Weak	Yes
Day 4	Rain	Mild	High	Weak	Yes
Day 5	Rain	Cool	Normal	Weak	Yes
Day 6	Rain	Cool	Normal	Strong	No
Day 7	Cloudy	Cool	Normal	Strong	Yes
Day 8	Sunny	Mild	High	Weak	No
Day 9	Sunny	Cool	Normal	Weak	Yes
Day 10	Rain	Mild	Normal	Weak	Yes
Day 11	Sunny	Mild	Normal	Strong	Yes
Day 12	Cloudy	Mild	High	Strong	Yes
Day 13	Cloudy	Hot	Normal	Weak	Yes
14	Rain	Mild	High	Strong	No

STEPS

- IG (Information gain) of all variables
  - Entropy (E) of entire data set
  - Entropy of all attributes
  - IG
- Find max and repeat

dont forget to put $\log_2$

IG of weather

- Entropy of entire data set

$$S\{+9,-5\} = -\left(\frac{9}{14}\log_2\frac{9}{14}\right) - \left(\frac{5}{14}\log_2\frac{5}{14}\right) = 0.94$$

(9 = yes, 5 = no, 14 = no of rows)

- Entropy of all attributes

$$\text{Sunny}\{+2,-3\} = -\left(\frac{2}{5}\log_2\frac{2}{5}\right) - \left(\frac{3}{5}\log_2\frac{3}{5}\right) = 0.97$$

(2 = yes sunny, 3 = no sunny, 5 = no of rows of sunny)

$$\text{Cloudy}\{+4,-0\} = -\left(\frac{4}{4}\log_2\frac{4}{4}\right) - \left(\frac{0}{4}\log_2\frac{0}{4}\right) = 0$$

(if either 0 ans = 0)

$$\text{Rain}\{+3,-2\} = -\left(\frac{3}{5}\log_2\frac{3}{5}\right) - \left(\frac{2}{5}\log_2\frac{2}{5}\right) = 0.97$$

- IG = E(whole data) $- \frac{5}{14}E(S) - \frac{4}{14}E(C) - \frac{5}{14}E(R)$

(5 = total sunny, 14 = total rows)

$= 0.246$

Calculate IG of Temperature

- Step1: Entropy of entire dataset

$$S\{+9,-5\} = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$$

- Step2: Entropy of all attributes:
- Entropy of Hot $\{+2,-2\} = -\frac{2}{4}\log_2\frac{2}{4} - \frac{2}{4}\log_2\frac{2}{4} = 1.0$
- Entropy of Mild$\{+4,-2\} = -\frac{4}{6}\log_2\frac{4}{6} - \frac{2}{6}\log_2\frac{2}{6} = 0.91$
- Entropy of Cold$\{+3,-1\} = -\frac{3}{4}\log_2\frac{3}{4} - \frac{1}{3}\log_2\frac{1}{3} = 0.81$
- Information Gain= Entropy(whole data) $-\frac{4}{14}$Ent(H) $-\frac{6}{14}$Ent(M) $-\frac{4}{14}$Ent(C) $= 0.029$

Calculate IG of Humidity

- Step1: Entropy of entire dataset

$$S\{+9,-5\} = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$$

- Step2: Entropy of all attributes:
- Entropy of High $\{+3,-4\} = -\frac{3}{7}\log_2\frac{3}{7} - \frac{4}{7}\log_2\frac{4}{7} = 0.98$
- Entropy of Normal $\{+6,-1\} = -\frac{6}{7}\log_2\frac{6}{7} - \frac{1}{7}\log_2\frac{1}{7} = 0.59$
- Information Gain= Entropy(whole data) $-\frac{7}{14}$Ent(H) $-\frac{7}{14}$Ent(N) $= 0.15$

Calculate IG of Wind

Step1: Entropy of entire dataset

$$S\{+9,-5\} = -\frac{9}{14}\log_2\frac{9}{14} - \frac{5}{14}\log_2\frac{5}{14} = 0.94$$

Step2: Entropy of all attributes:

Entropy of Strong $\{+3,-3\} = -\frac{3}{6}\log_2\frac{3}{6} - \frac{3}{6}\log_2\frac{3}{6} = 1.0$

Entropy of Normal $\{+6,-2\} = -\frac{6}{8}\log_2\frac{6}{8} - \frac{2}{8}\log_2\frac{2}{8} = 0.81$

Information Gain= Entropy(whole data) $-\frac{6}{14}$Ent(S) $-\frac{8}{14}$Ent(W) $= 0.0478$

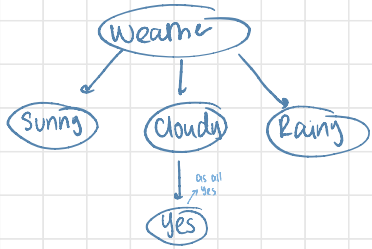
IG of entire dataset (S)

(S, Weather) = 0.246 → MAX → Hence ROOT NODE

(S, Temp) = 0.029

(S, Humi) = 0.15

(S, Wind) = 0.0478



FOR SUNNY

Day	Weather	Temperature	Humidity	Wind	Play Football?
Day 1	Sunny	Hot	High	Weak	No
Day 2	Sunny	Hot	High	Strong	No
Day 8	Sunny	Mild	High	Weak	No
Day 9	Sunny	Cool	Normal	Weak	Yes
Day 11	Sunny	Mild	Normal	Strong	Yes

Calculate IG of Temperature

Step1: Entropy of Sunny $\{+2,-3\} = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.97$

- Step2: Entropy of all attributes:
- Entropy of Hot $\{+0,-2\} = -\frac{0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2} = 0$
- Entropy of Mild $\{+1,-1\} = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1.0$
- Entropy of Cool $\{+1,-0\} = -\frac{1}{1}\log_2\frac{1}{1} - \frac{0}{1}\log_2\frac{0}{1} = 0$
- Information Gain = Entropy(Sunny) $-\frac{2}{5}$Ent(H) $-\frac{2}{5}$Ent(M) $-\frac{1}{5}$Ent(C)
  $= 0.57$

Calculate IG of Humidity

Step1: Entropy of Sunny $\{+2,-3\} = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.97$

Step2: Entropy of all attributes:

Entropy of High $\{+0,-3\} = -\frac{0}{3}\log_2\frac{0}{3} - \frac{3}{3}\log_2\frac{3}{3} = 0$

Entropy of Normal $\{+2,-0\} = -\frac{2}{2}\log_2\frac{2}{2} - \frac{0}{2}\log_2\frac{0}{2} = 0$

Information Gain = Entropy(Sunny) $-\frac{3}{5}$Ent(H) $-\frac{2}{5}$Ent(M)
$= 0.97$

Calculate IG of Wind

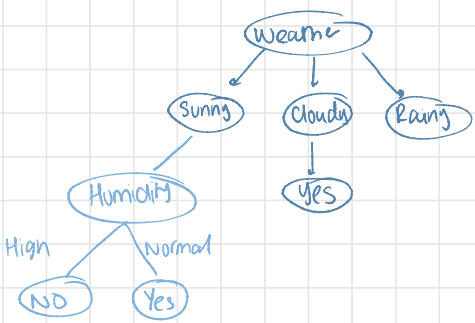
- Step1: Entropy of Sunny $\{+2,-3\} = -\frac{2}{5}\log_2\frac{2}{5} - \frac{3}{5}\log_2\frac{3}{5} = 0.97$
- [Step2: Entropy of all attributes:
- Entropy of Strong $\{+1,-1\} = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$
- Entropy of Weak $\{+1,-2\} = -\frac{1}{3}\log_2\frac{1}{3} - \frac{2}{3}\log_2\frac{2}{3} = 0.918$
- Information Gain = Entropy(Sunny) $-\frac{2}{5}$Ent(S) $-\frac{3}{5}$Ent(W)
  $= 0.019$

IG Sunny

(Sunny, Temp) = 0.57

(Sunny, Humid) = 0.97 → MAX

(Sunny, Wind) = 0.019



FOR RAIN

Day	Weather	Temperature	Humidity	Wind	Play Football?
Day 4	Rain	Mild	High	Weak	Yes
	Rain	Cool	Normal	Weak	Yes
Day 6	Rain	Cool	Normal	Strong	No
Day 10	Rain	Mild	Normal	Weak	Yes
	Rain	Mild	High	Strong	No

Calculate IG of Temperature

Step1: Entropy of Rain $\{+3,-2\} = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.97$

Step2: Entropy of all attributes:

- Entropy of Hot $\{+0,-0\} = -\frac{0}{2}\log_2\frac{0}{2} - \frac{0}{2}\log_2\frac{0}{2} = 0$
- Entropy of Mild $\{+2,-1\} = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.918$
- Entropy of Cool $\{+1,-1\} = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1.0$

Information Gain= Entropy(Rain) $-\frac{0}{5}$Ent(H) $-\frac{3}{5}$Ent(M) $-\frac{2}{5}$Ent(C)
= 0.019

Calculate IG of Humidity

Step1: Entropy of Rain $\{+3,-2\} = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.97$

Step2: Entropy of all attributes:

Entropy of High $\{+1,-1\} = -\frac{1}{2}\log_2\frac{1}{2} - \frac{1}{2}\log_2\frac{1}{2} = 1$

Entropy of Normal $\{+2,-1\} = -\frac{2}{3}\log_2\frac{2}{3} - \frac{1}{3}\log_2\frac{1}{3} = 0.918$

Information Gain= Entropy(Rain) $-\frac{2}{5}$Ent(S) $-\frac{3}{5}$Ent(W)
= 0.019

Calculate IG of Wind

- Step1: Entropy of Rain $\{+3,-2\} = -\frac{3}{5}\log_2\frac{3}{5} - \frac{2}{5}\log_2\frac{2}{5} = 0.97$
- Step2: Entropy of all attributes:

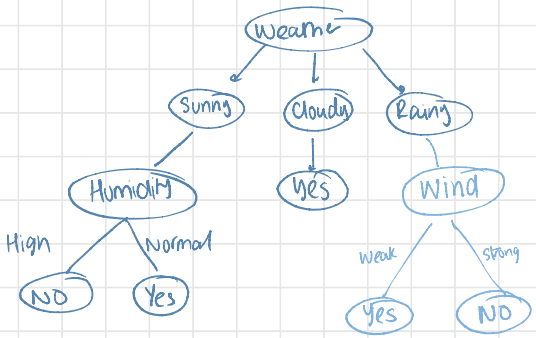
Entropy of Strong $\{+0,-2\} = -\frac{0}{2}\log_2\frac{0}{2} - \frac{2}{2}\log_2\frac{2}{2} = 0$

Entropy of Weak $\{+3,-0\} = -\frac{3}{3}\log_2\frac{3}{3} - \frac{0}{3}\log_2\frac{0}{3} = 0$

Information Gain= Entropy(Rain) $-\frac{2}{5}$Ent(S) $-\frac{3}{5}$Ent(W)
= 0.97

IG Rains

(Rain, Temp) = 0.019
(Rain, Humid) = 0.019
(Rain, Wind) = 0.97 → Max



In this problem, the following dataset has been used to learn a decision tree which predicts that if students pass **Artificial Intelligence** course (Yes or No), based on their Study (**Yes, No**) and previous Grade (High, Medium, or Low).

Study	Grade	Pass
Yes	Low	Yes
No	Medium	No
No	Low	No
Yes	Medium	Yes
No	High	No
Yes	High	Yes

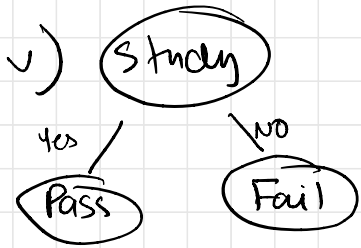
i. What is the entropy H(Pass)? = 1

ii. What is the entropy H(Pass | Grade)? = 1

iii. What is the entropy H (Pass | Study)? = 0

iv. Which attribute would you consider as the root node? What was the information gain of the attribute you chose as the root node? Study

v. Draw the decision tree.



↳ entire dataset entropy

$S=\{+3,-3\} = -\left(\frac{3}{6}\log_2\frac{3}{6}\right)-\left(\frac{3}{6}\log_2\frac{3}{6}\right) = 1$

IG Grade

↳ entropy of all variables

High: $\{+1,-1\} = -\left(\frac{1}{2}\log_2\frac{1}{2}\right)-\left(\frac{1}{2}\log_2\frac{1}{2}\right) = 1$

Medium: $\{+1,-1\} = -\left(\frac{1}{2}\log_2\frac{1}{2}\right)-\left(\frac{1}{2}\log_2\frac{1}{2}\right) = 1$

LOW: $\{+1,-1\} = -\left(\frac{1}{2}\log_2\frac{1}{2}\right)-\left(\frac{1}{2}\log_2\frac{1}{2}\right) = 1$

↳ Weighted entropy

$\frac{2}{6}(1)+\frac{2}{6}(1)+\frac{2}{6}(1) = 1$

↳ $IG = 1-\frac{2}{6}(1)-\frac{2}{6}(1)-\frac{2}{6}(1) = 0$

IG Study

↳ entropy of each variable

Yes $\{+3,0\} = 0$

NO $\{0,-3\} = 0$

↳ Weighted entropy

$0+0 = 0$

↳ $IG = 1-0 = 1$

↓

Root node

Question No. 4 [Marks: 3+7]

A) What is Machine Learning? Name its types and discuss the difference between classification and clustering.

B) Following data is being provided from 10 patients, having some symptoms similar to that of Covid-19. Use this data to design and draw a decision tree for an automated system which can decide if a patient should be tested for covid-19 or not. Also, show the calculations of Entropy and Gain.

Patient ID	Headache	Cough	Fever	Flu	Pain in body	Diagnosis
1	Yes	Yes	Yes	Yes	Yes	Covid-19
2	No	No	No	Yes	Yes	Allergy
3	Yes	Yes	No	Yes	No	Cold
4	Yes	No	Yes	No	No	Covid-19
5	No	Yes	No	Yes	No	Cold
6	No	No	No	Yes	No	Allergy
7	No	No	Yes	No	No	Covid-19
8	Yes	No	No	Yes	Yes	Allergy
9	No	Yes	No	Yes	Yes	Cold
10	Yes	Yes	No	Yes	Yes	Cold

Entropy(all values) $= S\{+3,-7\} = -\left(\frac{3}{10}\log_2\frac{3}{10}\right)-\left(\frac{7}{10}\log_2\frac{7}{10}\right) = 088$

IG Headache

Yes $\{+2,-3\} = -\left(\frac{2}{5}\log_2\frac{2}{5}\right)-\left(\frac{3}{5}\log_2\frac{3}{5}\right) = 0.97$

No $\{+1,-4\} = -\left(\frac{1}{5}\log_2\frac{1}{5}\right)-\left(\frac{4}{5}\log_2\frac{4}{5}\right) = 0.72$

$IG = 0.88 - \frac{5}{10}(0.97) - \frac{5}{10}(0.72) = 0.035$

IG Cough

Yes $\{+1,-4\} = -\left(\frac{1}{5}\log_2\frac{1}{5}\right)-\left(\frac{4}{5}\log_2\frac{4}{5}\right) = 0.72$

No $\{+2,-3\} = -\left(\frac{2}{5}\log_2\frac{2}{5}\right)-\left(\frac{3}{5}\log_2\frac{3}{5}\right) = 0.97$

$IG = 0.88 - \frac{5}{10}(0.72) - \frac{5}{10}(0.97) = 0.035$

IG Fever

DO Rest

MACHINE LEARNING ALGOS

1. Supervised learning

↳ learns mapping b/w input and target variables

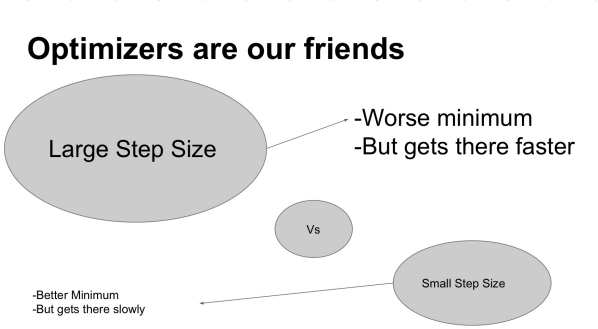
↳ Classification
  ↳ Predicts a class label

↳ Regression
  ↳ Predicts a numerical label

2. Unsupervised learning

↳ It learns on its own

↳ no input given



LINEAR REGRESSION

Regression

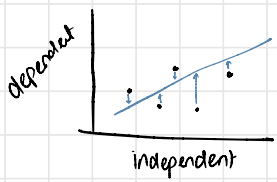
↳ Predicts a continous outcome variable

- Predicting the value of a company's future stock price using its pat and existing financial info
- Predicting the amount of rainfall
- Predicting ...

- Key difference from classification
  - We measure prediction errors differently
  - This leads us to quite different learning models and algorithms

Linear Regression

↳ Predicts by plotting a straight line that fits well to the dataset



↳ $y = mx + c$ (slope, intercept → value of y when x=0; dependent variable, independent variable)

1. $\bar{X} = \frac{\text{Sum } X}{N}$ (mean X)

2. $\bar{Y} = \frac{\text{Sum } Y}{N}$ (mean Y)

3.

X	Y	$X-\bar{X}$	$Y-\bar{Y}$	$D(x) * D(y)$	$(D(x))^2$	Y-Predict	$(Y-\text{Predict})^2$
		Deviation (x) D(x)	Deviation (y) D(y)	Product of Deviation	Square of Deviation of X	mx+c (To find ERRORS)	
				Product D	SUM D		SUM → error

4. $m = \frac{\text{Product D}}{\text{Sum}^2 \text{ D}}$ (m → gradient)

5. $C = \bar{y} - m\bar{x}$

Diameter(X) In Inches	Price(Y) In Dollars	Mean(X)	Mean(Y)	Deviations(X)	Deviations(Y)	Product of Deviations	Sum of Product of Deviations	Square of Deviations for X
8	10	10	13	-2	-3	6	12	4
10	13			0	0	0		0
12	16			2	3	6		4
						12		8

Calculate m= Sum of product of deviations/ Sum of square of deviation for X

Calculate b = Mean of Y - (m * Mean of X)

$m = \frac{12}{8} = 1.5$

$C = 13 - (1.5 \times 10) = -2$

$Y = 1.5x - 2$

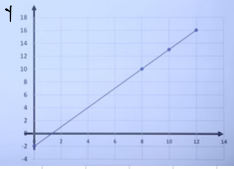


Table. B.

[Use $x = r\cos\emptyset$ *and* $y = r\sin\emptyset$ to transform angles into the x and y coordinates, and radius = 10].

Table.B Channels and angle values

Channels	Angles (in degree)
A	34
B	45
C	55
D	48
E	49

			D(x)	D(y)		
	X	Y	$X-\bar{X}$	$Y-\bar{Y}$	D(x)×D(y)	$(D(x))^2$
$10\cos(34)$ ←	8.29	5.59	1.43	-1.57	-2.2451	2.0449
	7.07	7.07	0.21	-0.094	-0.01974	0.0441
	5.73	8.19	-1.13	1.026	-1.159	1.276
	6.69	7.43	-0.17	0.266	-0.045	0.0289
	6.56	7.54	-0.3	0.376	-0.1128	0.09
					-3.58164	3.4839

$$\bar{X} = \frac{8.29+7.07+5.73+6.69+656}{5} = 6.86$$

$$\bar{y} = \frac{5.59+7.07+8.19+7.43+7.54}{5} = 7.164$$

$$m = \frac{-3.58164}{3.4839} = -1.028$$

$$c = \bar{y} - m\bar{x}$$

$$c = 7.164 - (-1.028)(6.86) = 14.216$$

$$y = -1.028x + 14.216$$

21K4638

Date:

no of weeks X	sold cars Y	$(x-\bar{x})$	$(y-\bar{y})$		Y-Predict	$(Y-Y_{predict})^2$
168	272	-29.33	[illegible]	42.6	212.272	3567.43
428	300	230.67	193.6	70.6	363.592	4013.47
299	311	98.67	66.6	81.6	289.814	505.6
392	365	194.67	162.6	135.6	342.64	499.9
80	167	-117.33	-149.4	-62.4	161.056	35.3
56	149	-141.33	-173.4	-80.4	147.088	3.655
352	366	154.67	122.6	-136.6	319.36	2175.28
444	310	246.67	214.6	80.6	372.904	395691
168	192	-29.33	-63.4	-37.4	212.272	410.9
200	229	2.67	-29.4	-0.4	230.896	3.59
4	88	-193.33	-229.4	-141.4	116.824	830.8
52	118	-145.33	-179.4	-111.4	144.76	716.09
20	62	-177.33	-209.4	-167.4	126.136	4112.6
228	319	30.67	-1.4	89.6	247.192	5156.38
72	193	-125.33	-157.4	-36.4	156.4	1339.56

Sum: 27587

↓ error

$\bar{x} = 197.33$ $\bar{y} = 229.4$

$\bar{x} = \frac{2960}{45}$ $\bar{y} = \frac{3441}{45}$

$y = a + bx$

$a = 114.496$

$b = 0.582$ $y = 114.496 + 0.582x$

$r = 0.895$

???

Select

1. Gradient Descent Algo

↳ optimises y-intercept and m → for Linear Regression

Repeat until convergence {

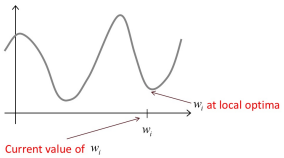
$$w_i := w_i - \alpha \frac{\partial}{\partial w_i} J(w_0, w_1) \quad \text{for } i = 0 \text{ and } i = 1$$

}

Learning rate

Partial derivative

What if you initialize w_i at a minimum



↳ If α too small

↳ gradient descent can be slow

↳ If α too large

↳ gradient descent can overshoot the minimum

↳ It may fail to converge/diverge

Gradient descent algorithm for Linear regression

Repeat until convergence {

$$w_0 := w_0 - \alpha \sum_n [y_n - (w_0 + w_1 x_n)]$$

$$w_1 := w_1 - \alpha 2 \sum_n [y_n - (w_0 + w_1 x_n)].x_n$$

}

This particular version is called "**Batch**" gradient descent

https://youtu.be/LzXakYdtAOo?si=-58VyX5Ko310W8Bd

2. Least squared method

↳ to find optimal

1. mean(x) = $\bar{x}$
2. mean(y) = $\bar{y}$
3. mean(xy) = $\overline{xy}$
4. mean(x^2) = $\overline{x^2}$
5. $m = \dfrac{(\bar{x} \times \bar{y}) - \overline{xy}}{(\bar{x})^2 - \overline{x^2}}$ → gradient
6. $c = \bar{y} - (m \times \bar{x})$ → y-intercept

$y = mn + c$

find

Are there other methods to find optimal **w**

- Closed form solution exists using Linear Algebra
  $y = w^T X - b$
- The method is called **Least squares** method

↳ Co-relation co-efficient → R

- ↳ −1 −vely corelated
- ↳ 0 not corelated at all
- ↳ 1 +vely corelated → linear

Q)

x	y
1	2
2	1
4	3

$\bar{x} = \dfrac{4+2+1}{3} =$

$\bar{y} = \dfrac{2+1+3}{3} =$

$\overline{xy} = \dfrac{(4\times2)+(2\times1)+(3\times1)}{3},$

$\overline{x^2} = \dfrac{1^2+2^2+4^2}{3},$

$m = 0.428$

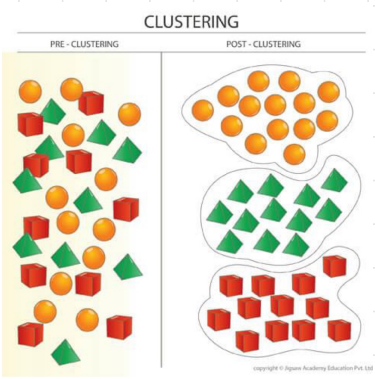
$c = 1$

$y = 0.428n + 1$

CLUSTERING

Clustering

- Unsupervised ML technique
- Clusters are inferred from the data without human input
- groups similar instances
  - Euclidean distance



Unsupervised

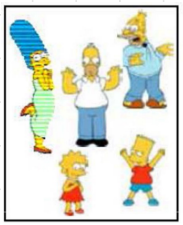
- It learns on its own how to classify the data by implicitly learned features
- requires data but no labels

– Detect patterns e.g. in
  - Group emails or search results
  - Customer shopping patterns
  - Regions of images
– Useful when don't know what you're looking for
– But: can get gibberish

CLUSTERING ALGOS

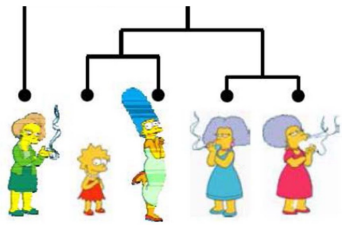
Partition Algos → Flat

- k-means
- Mixture of Gaussian
- Spectral Clustering



Hierarchical Algos

- Agglomerative → bottoms up
- Divisive → Top down



K-means / KNN

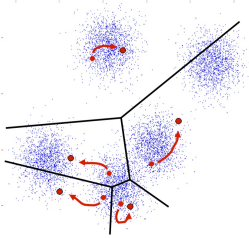
↳ an iterative clustering algo

↳ guarantees convergence in a finite no of iterations

– Initialize: Pick *K* random points as cluster centers

– Alternate:
1. Assign data points to closest cluster center
2. Change the cluster center to the average of its assigned points

– Stop when no points' assignments change



Euclidean Distance $= \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$

(x_2: observed value; x_1: centroid value)

Running time per iteration:

1. Assign data points to closest cluster center

   O(KN) time

2. Change the cluster center to the average of its assigned points

   O(N)

Steps

↳ choose any 2 points for 2 clusters (taking K = 2) → C1, C2

↳ find euclidean distance b/w next point, C1 and next point, C2

↳ which ever cluster gives smallest distance

↳ Get Avg of that cluster and point

↳ Update cluster

REPEAT

Q)

sr_no	age	amount
C1	20	500
C2	40	1000
C3	30	800
C4	18	300
C5	28	1200
C6	35	1400
C7	45	1800

take any seed randomly

$K1 = (C1) = (20, 500)$

$d(K1, C3) = \sqrt{(30-20)^2 + (800-500)^2} = 300.0$

$K2 = (C2) = (40, 1000)$

$d(K2, C3) = \sqrt{(30-40)^2 + (800-1000)^2} = 200.$

smaller so update this cluster

$K1 + C3$

$\left(\frac{40+30}{2}, \frac{1000+800}{2}\right) = 35, 900$

$K1 = (C1) = (20, 500)$

$d(K1, C4) = \sqrt{(18-20)+(300-500)^2} =$

smaller

$K1 + C4$

$\left(\frac{18+20}{2}, \frac{300+500}{2}\right) =$

$K2 = (C2, C3) = (35, 900)$

$d(K2, C4) = \sqrt{(18-35)^2 + (300-900)^2} =$

Repeat till all points done

Assume that the control unit (CU) inside a mobile phone of a user A receives signal values from the BS in terms of distance and power as shown in **Table. C** between ***11:00 to 3:00 PM***. The CU is intelligent and can classify the signals based on distance and power levels. Use ***KNN algorithm*** with k = 3 to classify the signal **5**.

Table.C Mobile signal with respect to power and distance

Signals	*Distance*	*Power*	Class (Normal and Abnormal)
1	8	20	**Normal**
2	4	18	**Normal**
3	5	16	**Abnormal**
4	10	15	**Normal**
5	32	10	??

a)

$d(1,5) = \sqrt{(32-8)^2 + (10-20)^2} = 26$ — N

$d(2,5) = \sqrt{(32-4)^2 + (10-18)^2} = 29.19$ — N

→ NORMAL

$d(3,5) = \sqrt{(32-5)^2 + (10-16)^2} = 27.65$ — A

$d(4,5) = \sqrt{(32-10)^2 + (10-15)^2} = 22.56$

b)

Table. D Patient dataset

	Patient Name	Address	Age	Body temperature	Height	Blood pressure	Cough	Respiratory Problem	Weight	Admitted to the hospital
1	Aslam	DHA	46	102	6	109	Yes	Severe	60	Yes
2	Umair	Malir	44	97	6.11	98	No	Less severe	55	No
3	Zain	Johar	36	99	5.9	85	No	Less severe	56	No
4	Umer	Gulshan	37	100	5.8	83	No	Severe	59	Yes
5	Ali	Saddar	39	97	5.3	95	No	Less severe	76	No
6	Ziauddin	Johar	40	98	5.6	120	Yes	Severe	78	Yes
7	Waqar	DHA	42	99	5.5	180	Yes	Less severe	80	No
8	**Saleem**	**Bahria town**	**44**	**102**	**5.8**	**110**	**Yes**	**Severe**	**??**	**Yes**

$\frac{P1 + P2 + P7}{3} = \frac{60 + 55 + 80}{3} = 65$

check endidean distance

A) The outbreak of the coronavirus pandemic has completely affected the educational system worldwide, leading to the near-total closures of academia impacting about 98.5 percent of the world's student population. So, as an ingenuity action HEC allowed universities and degree awarding institutes to continue online education and conduct examinations by making a new policy without compromising the health and safety factors. For which FAST NUCES has decided a policy of S/U and Letter grade for final examination. But in an online meeting with classmates you noticed that it was difficult for everyone to choose which policy will be more effective.

So, you are required to make a classifier using K-Means Algorithm to divide the observation into TWO clusters **(C1=S/U Policy) and (C2= Letter Grade)** using the following dataset rows according to your roll number digits.

For Example as shown in Figure 1 below, For 19k-2858 choose all relevant rows even if a digit is repeated twice or thrice (choose the 1st, 9th, 2nd, 8th, 5th and 8th row again) and make a new table.

07th July 2020, 9:00 am – 12:30 pm

Digit	A	B
0	3	3
1	8	5
2	4	4
3	2	4
4	7	7
5	5	8
6	3	5
7	4	8
8	6	9
9	9	6

Digit	A	B
1	8	5
9	9	6
2	4	4
8	6	9
5	5	8
8	6	9

Figure 1: Example data selection for Roll No. 19k-2858

17 3730

	Digit	A	B
→	1	8	5
	7	4	8
	3	2	4
	7	4	8
→	3	2	4
→	0	3	3

Taking 3 and 7 as random seeds

$C_1 = (3)(2,4)$

$1\,(8,5)\ (C_1,1) = \sqrt{(4-2)^2 + (5-4)^2} = 6.08$

$C_2\,(7)\,(4,8)$

$(C_2,1) = \sqrt{(8-4)^2 + (5-8)^2} = 5$ ✓

$\frac{8+4}{2}, \frac{5+8}{2}$

$C_2\,(7,1) = (6, 6.5)$

2,4

$7\,(4,8)\quad \sqrt{(4-2)^2 + (8-4)^2} = 4.47$

6, 6.5

$\sqrt{(4-6)^2 + (8-6.5)^2} = 2.5$ ✓

$C_1\,(3) = (2,4)$

$C_2\,(7,1,7) = (5, 7.25)$

$3\,(2,4)\quad \sqrt{(2-2)^2 + (4-0)} = 0$

$C_1\,(3,3) = (2,4)$

$C_2\,(7,1,7) = (5, 7.25)$

$0\,(3,3)\quad \sqrt{(3-2)^2 + (3-4)^2} = 1.414$

$\sqrt{(3-5)^2 + (3-7.25)^2} = 4.69$

$C_1\,(3,3,0) = (2.5, 3.5)$

$C_2\,(7,1,7) = (5, 7.25)$

Figure 1: Example data selection for Roll No: 19K-2658

B) Consider a fictitious document collection. There are 6 documents in this collection and these are represented as 6 points in two-dimensional vector space as follow:

D1	(2,1)
D2	(1,1)
D3	(4,1)
D4	(1,2)
D5	(2,2)
D6	(4,2)

Suppose that the distance between a pair of documents is measured by the Euclidean distance between their corresponding points. Show that how the k-means algorithm (with k=2) clusters these documents, using D2 and D3 as seeds.

	C1 (D2) (1,1)	C2 (D3) (4,1)
D1(2,1)	$\sqrt{(2-1)^2+(1-1)^2} = 1$ ✓	$\sqrt{(2-4)^2+(1-1)^2} = 2$
	C1(D2,D1) = (1.5,1)	
D4(1,2)	$\sqrt{(1-1.5)^2+(2-1)^2} = 1.1$ ✓	$\sqrt{(1-4)^2+(2-1)^2} = 3.16$
	C1(D2,D1,D4) = (1.25,1.5)	
D5(2,2)	$\sqrt{(2-1.25)^2+(2-1.5)^2} = 0.901$ ✓	$\sqrt{(2-4)^2+(2-1)^2} = 2.23$
	C1(D2,D1,D4,D5) = (1.625,1.75)	
D6(4,2)	$\sqrt{(4-1.625)^2+(2-1.75)^2} = 2.38$	$\sqrt{(4-4)^2+(2-1)^2} = 1$ ✓
	C1(D2,D1,D4,D5)	C2(D3,D6)

NEURAL NETWORKS

$$y_{in} = \sum_{i=1}^{n} w_i x_i + b$$ ← biasness

$$y_{in} = w_1x_1 + w_2x_2 \dots w_nx_n + b$$

Activation Functions → so ans b/w (0,1)

↳ Sigmodial $\delta(y) = \frac{1}{1+e^{-y}}$

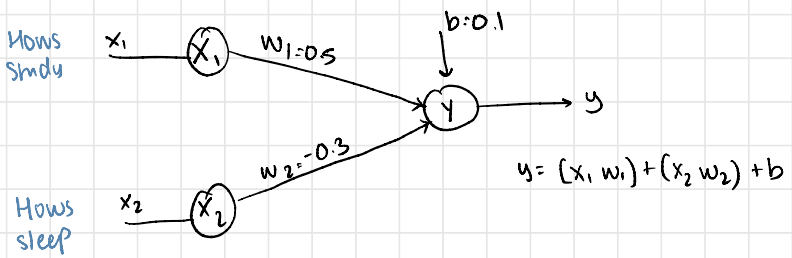
↳ RELU = $\max(y, 0)$

↳ Step function $y = \begin{cases} 1 & \text{if } n > 0 \\ 0 & \text{if } n \leq 0 \end{cases}$

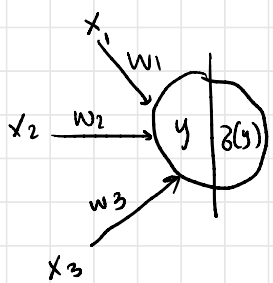
$w1 = 0.5$

$w2 = -0.3$

$b1 = 0.1$ ↓ biasness



$y = (x_1 w_1) + (x_2 w_2) + b$



$y = w_1x_1 + w_2x_2 + w_3x_3 + b$

$\delta(y) = \frac{1}{1+e^{-y}}$

Weight Update

learning rate ↑ target ↑ observed ↑

$$w_{new} = w_{old} + L(t-O)x_i$$

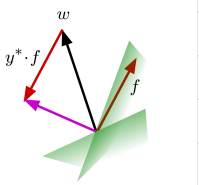
L=0.5 Threshold :1

Problem: Why can't a perceptron represent XOR?

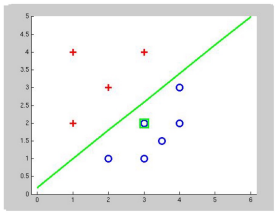
Solution: Intuitively, a perceptron is a linear classifier (given that we use the step function as the activation function), but XOR is not linearly separable. In the drawing below, you can see that AND and OR are linearly separable, but there is no way to linearly separate the shaded outputs of XOR.

Binary Perceptron

- inputs → feature values
- each feature has a weight
- sum is the activation
  - if activation is
    - +ve → output +1
    - −ve → output −1



- Separable Case



- Start with $w=0$
- classify with current weights

$$y: \begin{cases} +1 & \text{if } w \cdot f(x) \geq 0 \\ -1 & \text{if } w \cdot f(x) \leq 0 \end{cases}$$

- if correct (e.g. $y = y^*$) → no change
- if wrong → adjust the weight vector by adding/subtracting the feature vector

$$w = w + y^* \cdot f$$

Multiclass Perceptron

↳ If we have multiple classes

↳ a weight vector for each class w_y

↳ score activation of class y $w_y \cdot f(x)$

↳ Prediction highest score wins $y = \arg_y \max w_y \cdot f(x)$

↳ Start with all $w=0$

↳ Pick up training examples 1 by 1

↳ Predict with current weights

$y = \arg\max_y w_y \cdot f(x)$

↳ If correct → no change

↳ If wrong → lower score of wrong ans $w_y = w_y - f(n)$

raise score of right ans $w_y = w_y + f(n)$

"win the vote"

"win the election"

"win the game"

w_{SPORTS}

BIAS	: 1
win	: 0
game	: 0
vote	: 0
the	: 0
...	

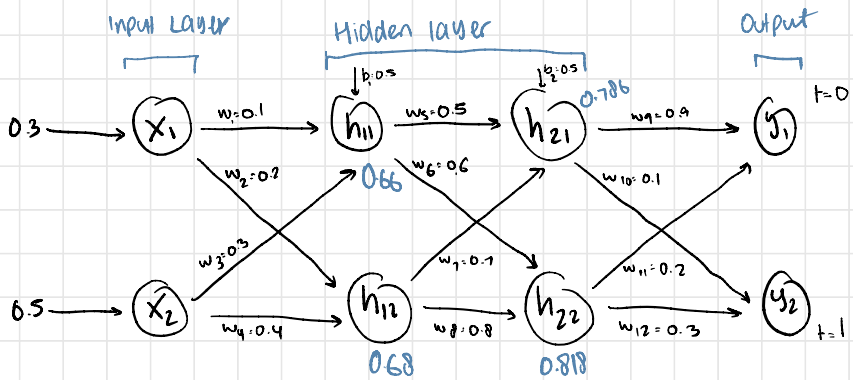
$w_{POLITICS}$

BIAS	: 0
win	: 0
game	: 0
vote	: 0
the	: 0
...	

w_{TECH}

BIAS	: 0
win	: 0
game	: 0
vote	: 0
the	: 0
...	

Feed Forward Neural Network



$h_{11} = w_1x_1 + w_3x_2 + b_1$

$(0.1 \times 0.3) + (0.3 \times 0.5) + 0.5 = 0.68$

$\delta(h_{11}) = \frac{1}{1+e^{-0.68}} = 0.66$

$h_{12} = x_2w_4 + x_1w_2 + b_1$

$= (0.5 \times 0.4) + (0.3 \times 0.2) + 0.5 = 0.76$

$\delta(h_{12}) = \frac{1}{1+e^{-0.76}} = 0.68$

$h_{21} = \delta(h_{11})w_5 + \delta(h_{12})w_7 + b_2$

$= (0.66 \times 0.5) + (0.68 \times 0.7) + 0.5$

$= 1.306$

$\delta(h_{21}) = \frac{1}{1+e^{-1.306}} = 0.786$

$h_{22} = \delta(h_{11})w_6 + \delta(h_{12})w_8 + b_2$

$= (0.66 \times 0.6) + (0.68 \times 0.8) + 0.5$

$= 1.504$

$\delta(h_{22}) = \frac{1}{1+e^{-1.504}} = 0.818$

$y_1 = w_9\,\delta(h_{21}) + w_{11}\,\delta(h_{22})$

$= 0.1636$

$\delta(y_1) = 0.54$

$y_2 = w_{10}\,\delta(h_{21}) + w_{12}\,\delta(h_{22})$

$= 0.324$

$\delta(y_2) = 0.58$

Error = mean squared error $= \frac{1}{2}\left[(y_0 - y_T)^2 + (y_0 - y_T)^2\right]$

(y_1, y_2)

$= \frac{1}{2}\left[(0.54-0)^2 + (0.58-1)^2\right] = 0.234$

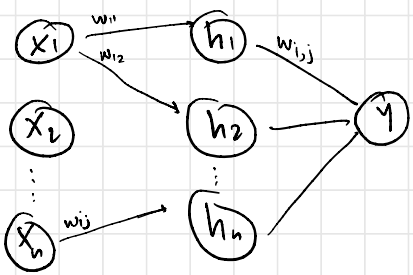
get corrected weight for net iteration

NOW back propogation

↳ change weights and bias

NOT DONE

A) Suppose you had a neural network with linear activation functions. That is, for each unit the output is some constant **c** times the weighted sum of the inputs. Assume that the network has one hidden layer. For a given assignment to the weights **w**, write down equations for the value of the units in the output layer as a function of **w** and the input layer **x**, without any explicit mention of the output of the hidden layer.



$$H: c\left(\sum_{k}^{n} w_{ij} x_k\right) + b_{hn}$$

$$O: \left(\sum w_{ij} \times H_i\right) = c\left[\sum_{i} w_{ij} \times c\left(\sum_{k}^{n} w_{ij} x_k\right) + b_{hn}\right] + b_o$$

B). The spread spectrum (SS) has been a very popular technique to encode messages in military communication. In SS, a message signal is combined with the secret chip sequence using ***XOR*** operation. The resultant message is unable to decode by the unknown receiver and transmitted through wireless channel. Draw a simple **Neural Network Architecture** which can achieve the ***XOR*** operation of the following given data shown in **Table. E**.

Table.E Input and output values

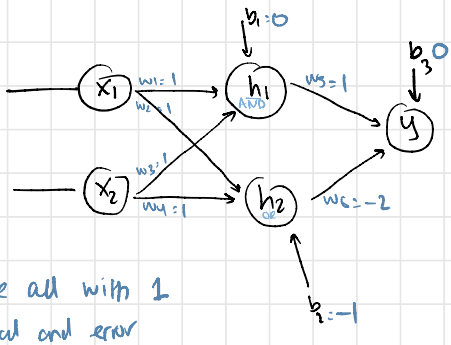
Message signal: x1	Secret chip sequence: x2	Resultant message: y
0	0	0
1	1	0
1	0	1
0	1	1

Problem: Why can't a perceptron represent XOR?

Solution: Intuitively, a perceptron is a linear classifier (given that we use the step function as the activation function), but XOR is not linearly separable. In the drawing below, you can see that AND and OR are linearly separable, but there is no way to linearly separate the shaded outputs of XOR.

Activation function

RELU = max(y, 0)



↳ Initialize all with 1

↳ do trial and error

$h_1 = x_1 w_1 + x_2 w_3 + b_1 = 1$

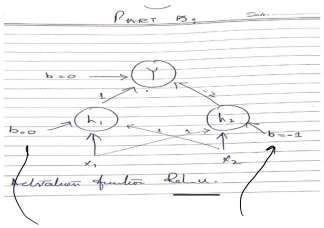
1 1 0 1 0

$h_2 = x_1 w_2 + x_2 w_4 + b_2 = 0$

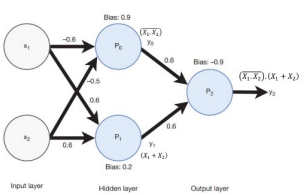
1 1 0 1 −1

$y = h_1 w_6 + h_2 w_5 + b_3 = 0$

1 −2 1 0 0



Combining Multiple Perceptrons Cont'd



Weight Update

learning rate, target, observed

$w_{new} = w_{old} + L(t - o) x_i$

L=0.5 Threshold :1

AND Gate - PERCEPTRON TRAINING RULE

w1 = 1.2, w2 = 0.6 Threshold = 1 and Learning Rate n = 0.5

A	B	A ^ B
0	0	0
0	1	0
1	0	0
1	1	1

3. A=1, B=0 and Target = 0
   - *wi.xi = 1*1.2 + 0*0.6 = 1.2*
   - This is greater than the threshold of 1, so the output = 1

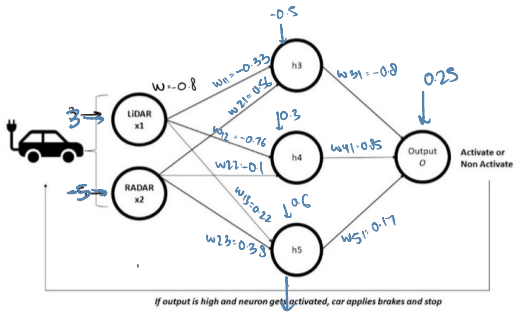
$$wi = wi + n(t - o)xi$$

$$w1 = 1.2 + 0.5(0 - 1)1 = 0.7$$

$$w2 = 0.6 + 0.5(0 - 1)0 = 0.6$$

of output neuron ***O*** based on the given data as follows using only feedforward propagation and for 1 epoch. Use Sigmoid function = $f(x) = \frac{1}{1-e^{-x}}$ for all the activation functions.

LiDAR x1 = 3	RADAR x2 = -5	w11 =-0.33	w12 = -0.76	w13 = 0.22	w21 = 0.56	w22 = -0.1	w23 = 0.38
W31 = -0.8	W41 = 0.85	w51 = 0.17	b3 = -0.5	b4 = 0.3	b5= 0.6	b6= 0.25	b7= 0.1



hidden layer

$h_3 = (3 \times -0.33) + (-5 \times 0.56) + (-0.5)$

$= -4.29$

$\delta(h_3) = \frac{1}{1 - e^{-(-4.29)}} = -0.0139$

$h_4 = (3 \times -0.76) + (-5 \times -0.1) + 0.3$

$= -1.48$

$\delta(h_4) = -0.295$

$h_5 = (3 \times 0.22) + (-5 \times 0.38) + 0.6$

$= -0.64$

$\delta(h_5) = -1.1155$

$\delta(h_3) = -0.0139$

$\delta(h_4) = -0.295$

$\delta(h_5) = -1.1155$

$y = (-0.0139 \times -0.8) + (-0.295 \times 0.85) + (-1.115 \times 0.17) + 0.25$

$= 0.32232$

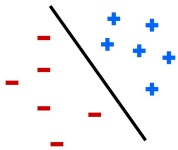
$\delta(y) = 3.629$

$O = 3.629$

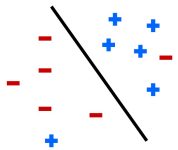
Properties of Perceptron

- Separatability
  - true if some parameter get the training set perfectly correct

Separable



Non-Separable

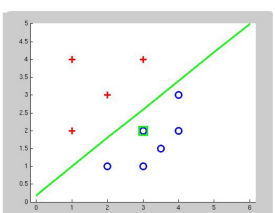


- Convergence
  - If the training set is seperable perceptron will eventually converge → binary case

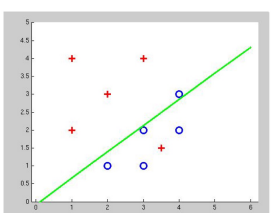
- Mistake bound
  - the max no. of mistakes related to the margin / degree of separatability → binary case

$$\text{mistakes} < \frac{k}{\delta^2}$$

- Separable Case



- Non-Separable Case



Problems with Perceptrons

1. Noise
   - ↳ If the data isn't seperable, weights might thrash

SOL → Avg weight vectors over time can help → average perceptron

2. Mediocre generalization
   - ↳ finds a 'barely' seperating solution

3. Overtraining
   - ↳ tests accuracy usually rises, then falls
   - ↳ Overtraing is a kind of overfitting

Fixing the Perceptrons

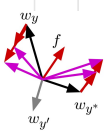
- Idea: adjust the weight update to mitigate these effects
- MIRA*: choose an update size that fixes the current mistake…
- … but, minimizes the change to w

$$\min_w \frac{1}{2}\sum_y ||w_y - w'_y||^2$$

$$w_{y^*} \cdot f(x) \geq w_y \cdot f(x) + 1$$

- The +1 helps to generalize

* Margin Infused Relaxed Algorithm



Guessed y instead of y^* on example x with features $f(x)$

$$w_y = w'_y - \tau f(x)$$
$$w_{y^*} = w'_{y^*} + \tau f(x)$$

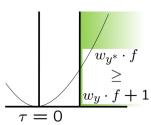
Minimum Correcting Update

$$\min_w \frac{1}{2}\sum_y ||w_y - w'_y||^2$$
$$w_{y^*} \cdot f \geq w_y \cdot f + 1$$

$$\min_\tau ||\tau f||^2$$
$$w_{y^*} \cdot f \geq w_y \cdot f + 1$$

$$(w'_{y^*} + \tau f) \cdot f = (w'_y - \tau f) \cdot f + 1$$
$$\tau = \frac{(w'_y - w'_{y^*}) \cdot f + 1}{2f \cdot f}$$

$$w_y = w'_y - \tau f(x)$$
$$w_{y^*} = w'_{y^*} + \tau f(x)$$



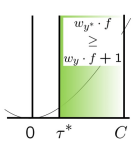
min not T=0, or would not have made an error, so min will be where equality holds

Maximum Step Size

- In practice, it's also bad to make updates that are too large
  - Example may be labeled incorrectly
  - You may not have enough features
  - Solution: cap the maximum possible value of τ with some constant C

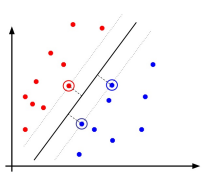
$$\tau^* = \min\left(\frac{(w'_y - w'_{y^*}) \cdot f + 1}{2f \cdot f}, C\right)$$

- Corresponds to an optimization that assumes non-separable data
- Usually converges faster than perceptron
- Usually better, especially on noisy data



Support Vector Machines

- Maximizing the margin: good according to intuition, theory, practice
- Only support vectors matter; other training examples are ignorable
- Support vector machines (SVMs) find the separator with max margin
- Basically, SVMs are MIRA where you optimize over all examples at once



MIRA

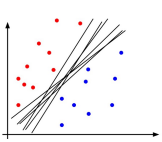
$$\min_w \frac{1}{2}||w - w'||^2$$
$$w_{y^*} \cdot f(x_i) \geq w_y \cdot f(x_i) + 1$$

SVM

$$\min_w \frac{1}{2}||w||^2$$
$$\forall i, y \; w_{y^*} \cdot f(x_i) \geq w_y \cdot f(x_i) + 1$$

Linear Separators

- Which of these linear separators is optimal?



Classification: Comparison

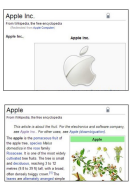
- Naïve Bayes
  - Builds a model training data
  - Gives prediction probabilities
  - Strong assumptions about feature independence
  - One pass through data (counting)
- Perceptrons / MIRA:
  - Makes less assumptions about data
  - Mistake-driven learning
  - Multiple passes through data (prediction)
  - Often more accurate

Web Search

Extension: Web Search

- Information retrieval:
  - Given information needs, produce information
  - Includes, e.g. web search, question answering, and classic IR
- Web search: not exactly classification, but rather ranking

x = "Apple Computers"



Feature-Based Ranking

x = "Apple Computer"

$f(x,$ $) = [0.3\ 5\ 0\ 0\ \ldots]$

$f(x,$ $) = [0.8\ 4\ 2\ 1\ \ldots]$

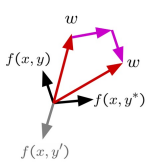
Perceptron for Ranking

- Inputs x
- Candidates y
- Many feature vectors: $f(x, y)$
- One weight vector: w
  - Prediction:

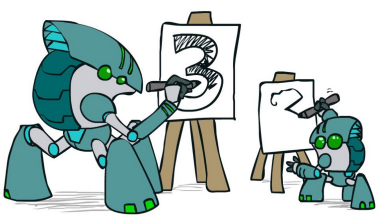
$$y = \arg\max_y \ w \cdot f(x, y)$$

  - Update (if wrong):

$$w = w + f(x, y^*) - f(x, y)$$



Apprenticeship



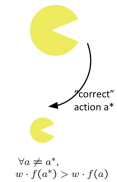
Pacman Apprenticeship!

- Examples are states s

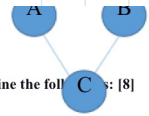
- Candidates are pairs (s,a)
- "Correct" actions: those taken by expert
- Features defined over (s,a) pairs: f(s,a)
- Score of a q-state (s,a) given by:

$$w \cdot f(s, a)$$

- How is this VERY different from reinforcement learning?



[Demo: Pacman Apprentice (L22[



A) Define the foll s: [8]

1- Define AI agent.

An agent perceives its environment through sensors and acts upon it through actuators (or effectors, depending on whom you ask). The Agent takes its action(s), which changes the environment, and process repeats.

2- What is back propagation and why do we use it?

Algorithm for training neural networks with hidden layers. Calculate error for output layer and propagate error back layer wise and update weights.

3- How supervised and reinforcement learning are related?

Both Supervised and RL get feedback:
Supervise Learning get feedback in form of Labeled Class
RL get feedback in form Reward.

4- Define the role of activation function and give the name of popular activation function.

Activation function is non-linear function which is used activate/ trigger particular neuron. Sigmoid, tanh, RELU are mostly commonly used activation functions.

5- Differentiate between Markov decision process and reinforcement learning.

In MDP transition probability and rewards are known. In RL transitions and rewards are unknown and it is estimated through interacting with Environment

6- Given the figure below, determine the followings:

a) Whether A and B are conditionally independent given C is observable (NO)
No. If C is observed than probability A can effect the probability of B so they are not independent.

not independent

b) Whether A and B are conditionally independent given C is unobservable (YES)
Yes. They are conditional independent if C is unobserved. Probability of A does not affect probability of B.

independent

7- What is the concept of gradient descent in neural network?

Gradient descent is used to minimize the loss function of neural network. We choose any starting point in weight space and then move to a neighboring point that is downhill, repeating until we converge on the minimum possible.

Reasoning under uncertainty in AI

↳ It refers to situations where there is a lack of complete info/knowledge about a particular aspect leading to ambiguity and unpredictability

↳ Reasoning strategy to handle uncertainty

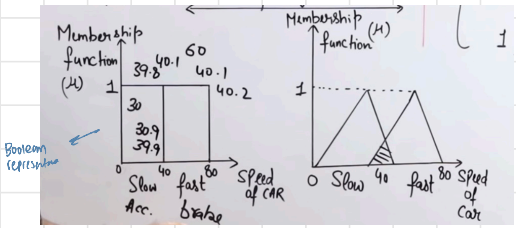
1. Probabilistic Models
2. Bayesian Networks
3. Monte Carlo Methods
4. Decision Theory
5. Fuzzy Logic
6. Qualitative Reasoning

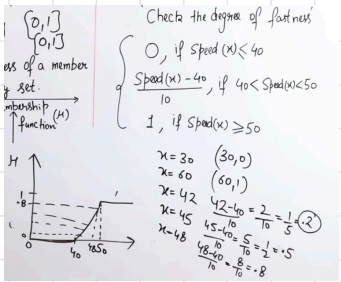
Fuzzy Logic

↳represents uncertainty

↳represent with degree

↳represent the belongings of a member → μ
of a crisp set to fuzzy set





↳if we use

if speed becomes 40.1
it will break
then with speed = 39.9
it will accelerate
so it will keep oscillating b/w the two
no flexibility

'Operations in fuzzy logic'

→ Union: $\text{Max}\{\mu_A(x), \mu_B(x)\}, x \in U$

→ Intersection: $\text{Min}\{\mu_A(x), \mu_B(x)\}, x \in U$

→ Complement: $\mu_{\bar{A}}(x) = [1 - \mu_A(x)], x \in U$

→ Bold Union: $\mu_{A \oplus B} = \min[1, \mu_A(x) + \mu_B(x)]$

→ Bold Intersection: $\mu_{A \odot B}(x) = \text{Max}[0, \mu_A(x) + \mu_B(x) - 1]$

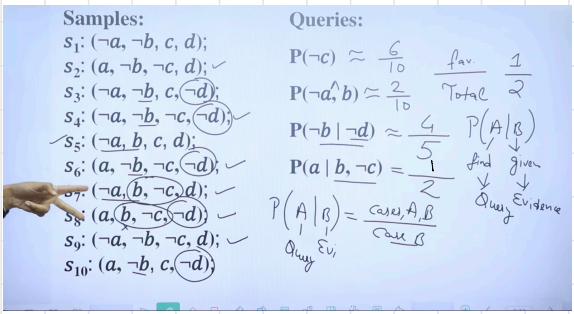
→ Equality $A = B$ if $\mu_A(x) = \mu_B(x) \; \forall x \in S$

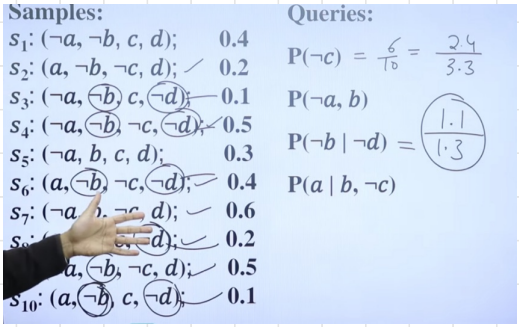
$U = \{5, 10, 20, 25, 30, 40\}$

$A = \{(10, 0.2), (20, 0.4), (25, 0.7), (30, 0.9), (40, 1)\}$

$B = \{(10, 0.4), (20, 0.1), (25, 0.9), (30, 0.2), (40, 0.6)\}$

Probalistic Interference





Fahad Sherwani
20 May

Dear Students,

The pattern for final exam will be around 8
• 10 questions. 80 • 85% content to be included is studied after MID 2 while 15 • 20% before MID 2.

The Syllabus includes:

1) Content before MID 2.
2) Bayesian Networks
3) Naive Bayes
4) Decision Trees
5) Linear Regression
6) Neural Networks
7) K-means/KNN
8) Performance matrices (Confusion matrix, I will share a tutorial about it tomorrow)
9) Some part of Reinforcement learning/Q-Learning (Not confirmed yet, but i will share a tutorial about it too).

Thank you.

Conditional Probability

Conditional probability is the probability of an event A occ
Itis denoted as P(A|B), where P represents probability. The

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

Probability Theorems
BN
HMM
} Lab 7
} → Lab 8

Lab 9,10

Supervised

- Regression → output Real value
  - Linear
  - Polynomial
  - SVM
  - Decision Tree
  - Random Forest
- Classification → output Category
  - Logistic
  - SVM
  - Decision Tree
  - Random Forest
  - Naive Bayes

Unsupervised

- Clustering → group similar data together
  - exclusive Partitioning
  - Agglomerative
  - Overlapping
  - Probabilistic
  - Hierachical
  - Kmeans
  - Principal Component Analysis
  - Gaussian Mixture Models
  - DBSCAN
- Association → identify patterns
  - Apriorr
  - FP Growth

Reinforcement

- Q Learning MDP

SUPERVISED

Regression

Classification

Code Implementations

Linear Regression

Definition- Linear regression is a statistical method used to model the relationship between a dependent variable (target) and one or more independent variables (features) by fitting a linear equation to the observed data.

Use Cases

- Predicting house prices based on features like area, number of bedrooms, etc.
- Forecasting sales revenue based on advertising spend.
- Estimating the demand for a product based on historical sales data.

```
from sklearn.linear_model import LinearRegression
import numpy as np

# Sample data (house areas in sq. ft. and prices in USD)
X = np.array([[1500], [1800], [2200], [2500]])  # Input features (house areas)
y = np.array([300000, 350000, 400000, 450000])  # Target variable (house prices)

# Create and fit the linear regression model
model = LinearRegression()
model.fit(X, y)

# Predict price for a new house with area 2000 sq. ft.
new_area = np.array([[2000]])
predicted_price = model.predict(new_area)
print("Predicted Price:", predicted_price)  # Output: [383333.33333333]
```

Polynomial Regression

Definition- Polynomial regression extends linear regression by adding polynomial terms to the model, allowing it to capture non-linear relationships between features and target.

Use Cases

- Modeling physical phenomena where the relationship between variables is non-linear.
- Predicting the growth rate of organisms based on time and environmental factors.
- Capturing the effect of interactions between variables in the model.

```
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
import numpy as np

# Sample data (input features and target variable)
X = np.array([[1], [2], [3], [4]])
y = np.array([1, 4, 9, 16])  # Quadratic relationship (y = x^2)

# Transform features to polynomial features
poly_features = PolynomialFeatures(degree=2)
X_poly = poly_features.fit_transform(X)

# Create and fit the polynomial regression model
model_poly = LinearRegression()
model_poly.fit(X_poly, y)

# Predict for new values
X_new = np.array([[5]])
X_new_poly = poly_features.transform(X_new)
predicted_value = model_poly.predict(X_new_poly)
print("Predicted Value (Polynomial Regression):", predicted_value)
# Output: [25.]
```

Support Vector Machine (SVM) Regression

Definition- Support Vector Machine (SVM) regression finds a hyperplane that best fits the data by maximizing the margin between support vectors and the hyperplane.

Use Cases

- Predicting stock prices based on historical data.
- Forecasting future demand for products or services.
- Estimating the price of real estate based on property features.

```
from sklearn.svm import SVR
import numpy as np

# Sample data (input features and target variable)
X = np.array([[1], [2], [3], [4]])
y = np.array([2, 3, 4, 5])  # Linear relationship (y = x + 1)

# Create and fit the SVM regression model
svm_regressor = SVR(kernel='linear')
svm_regressor.fit(X, y)

# Predict for new values
X_new = np.array([[5]])
predicted_value = svm_regressor.predict(X_new)
print("Predicted Value (SVM Regression):", predicted_value)  # Output: [6.]
```

Decision Tree Regression

Definition- Decision tree regression divides the feature space into regions and predicts the average target value within each region.

Use Cases

- Predicting the price of a used car based on its age, mileage, and other features.
- Estimating the energy consumption of a building based on weather conditions.
- Forecasting the sales volume of a product based on marketing campaigns.

```
from sklearn.tree import DecisionTreeRegressor
import numpy as np

# Sample data (input features and target variable)
X = np.array([[1], [2], [3], [4]])
y = np.array([2, 3, 4, 5])  # Linear relationship (y = x + 1)

# Create and fit the decision tree regression model
tree_regressor = DecisionTreeRegressor()
tree_regressor.fit(X, y)

# Predict for new values
X_new = np.array([[5]])
predicted_value = tree_regressor.predict(X_new)
print("Predicted Value (Decision Tree Regression):", predicted_value)  # Output: [6.]
```

Random Forest Regression

Definition- Random forest regression is an ensemble learning method that builds multiple decision trees and averages their predictions to improve accuracy and reduce overfitting.

Use Cases

- Predicting the yield of a crop based on weather conditions and soil properties.
- Estimating the price of a product based on customer demographics and market data.
- Forecasting the sales revenue of a retail store based on historical data.

```
from sklearn.ensemble import RandomForestRegressor
import numpy as np

# Sample data (input features and target variable)
X = np.array([[1], [2], [3], [4]])
y = np.array([2, 3, 4, 5])  # Linear relationship (y = x + 1)

# Create and fit the random forest regression model
forest_regressor = RandomForestRegressor()
forest_regressor.fit(X, y)

# Predict for new values
X_new = np.array([[5]])
predicted_value = forest_regressor.predict(X_new)
print("Predicted Value (Random Forest Regression):", predicted_value)  # Output: [6.]
```

Code Implementations

Logistic Regression

Definition- Logistic regression is a statistical method used for binary or multiclass classification by modeling the probability of a categorical outcome using a logistic (sigmoid) function.

Use Cases

- Predicting whether an email is spam or not.
- Classifying customer churn (yes/no).
- Identifying the type of iris flower based on petal and sepal measurements.

```
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load the Iris dataset
iris = load_iris()
X, y = iris.data, iris.target

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and fit the logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy (Logistic Regression):", accuracy)
```

Support Vector Machines (SVM)

Definition- Support Vector Machines (SVM) are supervised learning models used for classification and regression tasks by finding the optimal hyperplane that best separates classes in a high-dimensional space.

Use Cases

- Text classification (e.g., sentiment analysis).
- Image classification.
- Anomaly detection in network traffic.

```
from sklearn.svm import SVC
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load the Iris dataset
iris = load_iris()
X, y = iris.data, iris.target

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and fit the SVM classifier
model = SVC(kernel='linear')
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy (SVM):", accuracy)
```

Decision Trees

Definition- Decision trees are hierarchical models that partition the feature space into regions and assign class labels to each region based on majority voting.

Use Cases

- Customer segmentation based on purchasing behavior.
- Medical diagnosis based on patient symptoms.
- Credit scoring for loan approval.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load the Iris dataset
iris = load_iris()
X, y = iris.data, iris.target

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and fit the decision tree classifier
model = DecisionTreeClassifier()
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy (Decision Tree):", accuracy)
```

Random Forests

Definition- Random forests are ensemble learning methods that build multiple decision trees and aggregate their predictions to improve accuracy and reduce overfitting.

Use Cases

- Predicting the species of plants based on botanical features.
- Detecting fraudulent transactions in financial transactions.
- Recommender systems based on user preferences.

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load the Iris dataset
iris = load_iris()
X, y = iris.data, iris.target

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and fit the random forest classifier
model = RandomForestClassifier()
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy (Random Forest):", accuracy)
```

Naive Bayes

Definition- Naive Bayes classifiers apply Bayes' theorem with strong (naive) independence assumptions between features.

Use Cases

- Text classification tasks such as spam detection and sentiment analysis.
- Document categorization based on content.
- Medical diagnosis based on symptoms and test results.

```
from sklearn.naive_bayes import GaussianNB
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score

# Load the Iris dataset
iris = load_iris()
X, y = iris.data, iris.target

# Split data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create and fit the Naive Bayes classifier
model = GaussianNB()
model.fit(X_train, y_train)

# Predict on the test set
y_pred = model.predict(X_test)

# Evaluate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy (Naive Bayes):", accuracy)
```

UNSUPERVISED

CLUSTERING

Hierarchical Clustering
Definition- Hierarchical clustering is a method of cluster analysis that builds a hierarchy of clusters by either recursively merging or splitting clusters based on distance criteria.

Use Cases
- Grouping similar documents or articles in text mining.
- Creating taxonomies or dendrograms in biological sciences.
- Customer segmentation based on purchasing behavior.

```
from sklearn.cluster import AgglomerativeClustering
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt
# Generate synthetic data
X, _ = make_blobs(n_samples=300, centers=4, cluster_std=0.60,random_state=0)

# Perform Hierarchical Clustering
agg_clustering = AgglomerativeClustering(n_clusters=4)
labels = agg_clustering.fit_predict(X)

# Visualize the clusters
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
plt.title("Hierarchical Clustering")
plt.show()
```

K-means Clustering
Definition: K-means clustering partitions data into K clusters by iteratively assigning data points to the nearest cluster centroid and updating centroids based on the mean of points assigned to each cluster.

Use Cases:
- Market segmentation based on customer demographics.
- Image segmentation for object recognition.
- Analyzing clickstream data for user behavior patterns.

```
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt

# Generate synthetic data
X, _ = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)

# Perform K-means Clustering
kmeans = KMeans(n_clusters=4)
labels = kmeans.fit_predict(X)

# Visualize the clusters
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
plt.scatter(kmeans.cluster_centers_[:, 0],  kmeans.cluster_centers_[:, 1],  s=200, c='red', marker='X', label='Centroids')
plt.title("K-means Clustering")
plt.legend()
plt.show()
```

Principal Component Analysis (PCA)
Definition: Principal Component Analysis (PCA) is a dimensionality reduction technique that projects data onto a lower-dimensional space while preserving the maximum variance.

Use Cases:
- Feature extraction and data visualization.
- Noise reduction in signal processing.
- Enhancing the performance of machine learning models by reducing multicollinearity.

```
from sklearn.decomposition import PCA
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt

# Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

# Perform PCA for visualization (2D projection)
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X)

# Visualize PCA-transformed data
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=y, cmap='viridis')
plt.title("PCA Visualization of Iris Dataset")
plt.xlabel("Principal Component 1")
plt.ylabel("Principal Component 2")
plt.show()
```

Gaussian Mixture Models (GMMs)
Definition: Gaussian Mixture Models (GMMs) assume that the data is generated from a mixture of several Gaussian distributions and estimate the parameters of these distributions to perform probabilistic clustering.

Use Cases:
- Image segmentation in computer vision.
- Density estimation for anomaly detection.
- Modeling complex data distributions in unsupervised learning.

```
from sklearn.mixture import GaussianMixture
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt

# Generate synthetic data
X, _ = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)

# Perform Gaussian Mixture Model (GMM) clustering
gmm = GaussianMixture(n_components=4)
labels = gmm.fit_predict(X)

# Visualize the clusters
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
plt.title("Gaussian Mixture Model (GMM) Clustering")
plt.show()
```

Density-Based Spatial Clustering of Applications with Noise (DBSCAN)
Definition: DBSCAN is a density-based clustering algorithm that groups together closely packed points based on a density threshold.

Use Cases:
- Identifying spatial hotspots in geographic data.
- Anomaly detection in cybersecurity.
- Clustering large datasets with irregular shapes and noise.

```
from sklearn.cluster import DBSCAN
from sklearn.datasets import make_blobs
import matplotlib.pyplot as plt

# Generate synthetic data
X, _ = make_blobs(n_samples=300, centers=4, cluster_std=0.60, random_state=0)

# Perform DBSCAN clustering
dbscan = DBSCAN(eps=0.5, min_samples=5)
labels = dbscan.fit_predict(X)

# Visualize the clusters (including noise points)
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis')
plt.title("DBSCAN Clustering")
plt.show()
```

ASSOCIATE LEARNING

- Apriori Algorithm
- FP-Growth Algorithm

Apriori Algorithm
Definition: The Apriori algorithm is an association rule mining algorithm that discovers frequent itemsets in a transaction database. It uses an iterative approach to generate candidate itemsets and prune the search space by leveraging the "apriori" property.

Use Cases:
- Market basket analysis to identify frequently co-occurring items in customer transactions.
- Recommender systems to suggest related products based on purchase history.
- Web usage mining to understand sequential patterns in user behavior.

```
from mlxtend.frequent_patterns import apriori
from mlxtend.preprocessing import TransactionEncoder
import pandas as pd

# Sample transaction dataset (list of lists)
dataset = [['bread', 'milk', 'eggs'],
           ['bread', 'butter'],
           ['milk', 'butter'],
           ['bread', 'milk', 'butter'],
           ['bread', 'milk']]

# One-hot encode the dataset
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)

# Apply Apriori algorithm to find frequent itemsets
frequent_itemsets = apriori(df, min_support=0.4, use_colnames=True)
print(frequent_itemsets)
```

FP-Growth Algorithm
Definition: The FP-Growth (Frequent Pattern Growth) algorithm is a frequent itemset mining algorithm that constructs a compact data structure called the FP-tree to efficiently mine frequent itemsets without generating candidate itemsets.

Use Cases:
- Market basket analysis for identifying association rules.
- Bioinformatics for identifying frequent sequence patterns in DNA sequences.
- Telecommunication for analyzing call patterns and customer behaviors.

```
from mlxtend.frequent_patterns import fpgrowth
from mlxtend.preprocessing import TransactionEncoder
import pandas as pd

# Sample transaction dataset (list of lists)
dataset = [['bread', 'milk', 'eggs'],
           ['bread', 'butter'],
           ['milk', 'butter'],
           ['bread', 'milk', 'butter'],
           ['bread', 'milk']]

# One-hot encode the dataset
te = TransactionEncoder()
te_ary = te.fit(dataset).transform(dataset)
df = pd.DataFrame(te_ary, columns=te.columns_)

# Apply FP-Growth algorithm to find frequent itemsets
frequent_itemsets = fpgrowth(df, min_support=0.4, use_colnames=True)
print(frequent_itemsets)
```

Evaluating Non-Supervised Learning Models
Evaluating non-supervised learning models is an important step in ensuring that the model is effective and useful. However, it can be more