FIRST ORDER LOGIC:

- more expressive
- more succinct

Properties → Predicates with single argument.

SYNTAX:

Domain Specific ("Vocabulary"):
- Constant (objects)
- Predicates (Relations, Properties) : Return T/F
- Functions : Returns an object

Generic:
- Variables : $x, y, z, \ldots$
- Connectors : $\wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$
- Equality : $=$
- Quantifiers : $\forall, \exists$

Atomic Sentence:

Predicate (term$_1$, ..., term$_n$)

↳ constant /variable/ function(term$_1$, ..., term$_n$)

$\forall$: $\Rightarrow$

$\exists$: $\wedge$

PROPERTIES:

$\forall x \forall y \equiv \forall y \forall x$

$\exists x \exists y \equiv \exists y \exists x$

$\exists x \forall y \not\equiv \forall y \exists x$

$\neg \exists x \text{ likes}(A,B) \equiv \forall x \neg \text{likes}(A,B)$

$\exists! x \text{ king}(x)$: There is exactly one king.

eg: Spot has exactly 2 sisters

$\exists x \exists y \text{ sister}(x, \text{spot}) \wedge \text{sister}(y, \text{spot}) \wedge \neg(x=y) \wedge$

$[\forall z \text{ sister}(z, \text{spot}) \Rightarrow (z=x \vee z=y)]$

↓ same as $[\neg(\exists z \text{ sister}(z, \text{spot}) \wedge \neg(z=x) \wedge \neg(z=y))]$

$\forall z \neg \text{sister}(z, \text{spot}) \vee ((z=x) \vee (z=y))$

$\neg\alpha \vee \beta \quad \alpha \Rightarrow \beta$

$\forall z \text{ sister}(z, \text{spot}) \Rightarrow ((z=x) \vee (z=y))$

WELL-FORMED FORMULA: No free variables (without quantization)

BINDING LIST:

Answers that return True

Query : $\exists x$ Mammal(x)

$\{x/spot , x/Tuna\}$

ENCODE PROBLEM:

1. Vocabulary : Constants, Functions, Predicates
2. KB : i. Independent of case (Domain)
   ii. Specitic case (Instance)
3. Queries

INFERENCE:

1. Reducing FOL inference to propositional inference

i. Universal Instantiation (UI):

$$\frac{\forall x \quad \alpha}{\text{Substitute}(\{x/g\}, \alpha)}$$

g: ground term : Constants

function (constants)

=> If g finite => Equivalence is preserved.

ii. Existential Instantiation (EI):

$$\frac{\exists x \quad \alpha}{\text{Substitute}(\{x/k\}, \alpha)}$$

k: skolem constant: constant symbol that does not appear anywhere else in the KB.

=> Satisfiability is preserved. (not equivalence)

Entailment in FOL is semi decidable:

If α is entailed by FOL KB, it is entailed by finite subset of propositional KB.

For $n = 0$ to ∞ do

* Create propositional KB by instantiating depth-n terms.
* See if $KB \models \alpha$

FOL KB $\rightarrow$ Δ_0 (only depth 0, i.e. constants)

Δ

Check $\Delta_0 \models \alpha$: Yes $\Delta \models \alpha$

No : Δ_1 (only depth-1 functions + constants)

⋮

So if $\Delta \models \alpha$, some Δ_i will satisfy $\Delta_i \models \alpha$

if $\Delta \not\models \alpha$ => infinite search.

DEFINITE CLAUSE: Only one positive literal.

2. FORWARD CHAINING:

Start with ground terms and apply the KB, till we reach the answer.

"Datalog" : Definite Clause + No function symbols.

3. BACKWARD CHAINING:

Start with result and go back.

→ DFS

→ Logic Programming - Prolog.

4. RESOLUTION:

UNIFICATION:

α = Knows (John, x)

β = Knows (John, Jane)

Unifier : $\theta = \{x / Jane\}$ Unify $(\alpha, \beta) = \theta$

$\alpha\theta = \beta\theta$.

$\Delta \wedge \neg\alpha \rightarrow$ unsat

then $\Delta \models \alpha$

CONVERSION TO CNF:

1. Eliminate $\Rightarrow$, $\Leftrightarrow$

2. Move $\neg$ inwards

$$\neg \forall x\ \alpha \rightarrow \exists x\ \neg\alpha$$
$$\neg \exists x\ \alpha \rightarrow \forall x\ \neg\alpha$$

3. Standardize variables: each quantifier should use a different one.

4. Skolemize

If $\exists$ is inside $\forall x$

the replace variable with $F(x)$

If $\exists$ is on its own, use some Skolem constant, C_1.

5. Drop Universal Quantifiers

6. Distribute

$$(\alpha \wedge \beta) \vee \gamma \rightarrow (\alpha \vee \gamma) \wedge (\beta \vee \gamma).$$

PROBABILITY AND BELIEFS:

Classical logic is "monotonic."

↳ What is believed, remains to be believed even with new information.

$\Delta \models \alpha$ then $\Delta \cap \beta \models \alpha$

Non-classical logic: Δ + assumptions

"Degrees of belief": α not true/false

but belief(α) $\in [0,1]$

(PROBABILITY).

PROBABILITY AS A BASIS FOR REPRESENTING BELIEFS:

Propositional logic: $\Delta \models \alpha$

↳ Yes

↳ No

↳ Neither

Now: $Pr(\alpha) = \sum_{w \models \alpha} Pr(w)$

PROPERTIES:

- $Pr(\alpha \vee \beta) = Pr(\alpha) + Pr(\beta) - Pr(\alpha \wedge \beta)$
- if α, β are mutually exclusive

$$Pr(\alpha \wedge \beta) = 0$$

- BELIEF CHANGE: given evidence, β

$$Pr(\omega|\beta) = \begin{cases} 0 & \text{if } \omega \not\models \beta \\ \dfrac{Pr(\omega)}{Pr(\beta)} & \text{if } \omega \models \beta. \end{cases}$$

BAYES CONDITIONING:

$$Pr(\alpha|\beta) = \frac{Pr(\alpha \wedge \beta)}{Pr(\beta)}$$

* INDEPENDENCE:
  - → α is independent of β if
    - * $Pr(\alpha|\beta) = Pr(\alpha)$

$$\Rightarrow \frac{Pr(\alpha, \beta)}{Pr(\beta)} = Pr(\alpha) \Rightarrow Pr(\beta) = \frac{Pr(\alpha, \beta)}{Pr(\alpha)}$$

$$\Rightarrow Pr(\beta|\alpha) = Pr(\beta).$$

* $Pr(\alpha \wedge \beta) = Pr(\alpha) \cdot Pr(\beta)$

* $Pr(\alpha \mid \beta, \gamma)$ need not be $Pr(\alpha \mid \gamma)$

→ α is conditionally independent of β given γ if

* $Pr(\alpha \mid \beta \wedge \gamma) = Pr(\alpha \mid \gamma)$.

* $Pr(\alpha \wedge \beta \mid \gamma) = Pr(\alpha \mid \gamma) \cdot Pr(\beta \mid \gamma)$.

* CHAIN RULE:

$$Pr(\alpha_1 \wedge \alpha_2 \wedge \cdots \wedge \alpha_n) = Pr(\alpha_1 \mid \alpha_2 \wedge \alpha_3 \cdots \alpha_n) \cdot Pr(\alpha_2 \mid \alpha_3 \wedge \cdots \alpha_n) \cdots Pr(\alpha_n).$$

* CASE ANALYSIS (LAW OF TOTAL PROBABILITY):

$$Pr(\alpha) = \sum_{i=1}^{n} Pr(\alpha \wedge \beta_i) = \sum_{i=1}^{n} Pr(\alpha \mid \beta_i) Pr(\beta_i)$$

where $\beta_1, \beta_2 \cdots \beta_n$ are mutually exclusive and exhaustive!

* BAYES RULE:

$$Pr(\alpha \mid \beta) = \frac{Pr(\beta \mid \alpha) \cdot Pr(\alpha)}{Pr(\beta)}.$$

FALSE POSITIVE = $Pr(T \mid \neg D)$

FALSE NEGATIVE = $Pr(\neg T \mid D)$.

BAYESIAN NETWORKS:

SYNTAX

↳ * Directed Acyclic Graph (DAG) : Causality

↳ * Probabilities.

VARIABLE INDEPENDENCE:

Variable sets X, Y, Z

X and Y are independent given Z

$$I(X, Z, Y)$$

i.e. $Pr(X | Y, Z) = Pr(X | Z)$

eg: $X = \{A, B\}$ $Y = \{C\}$ $Z = \{D, E\}$

4 states 2 4

$I(X, Z, Y)$

32 Statements

↳ $A \wedge B$ independent of C given $D \wedge E$

$A \wedge \neg B$ ... C ... $D \wedge E$

⋮

$\neg A \wedge \neg B$ ... $\neg C$ ... $\neg D \wedge \neg E$

CAPTURING INDEPENDENCE GRAPHICALLY:

Non-Descendants(v) : All nodes - Descendants(v)
- v - Parents(v)

MARKOVIAN ASSUMPTIONS OF A BAYESIAN NETWORK:

* I(v, Parents(v), Non-Descendants(v))

Markov(G) → Apply this for all nodes(v) in G.

PARAMETERIZING THE STRUCTURE:

CONDITIONAL PROBABILITY TABLE (CPT):

CPT for variable C: Pr(C | Parents(C))

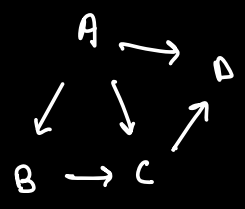
If C has 2 parents A, B

A	B	C	Pr(c|a,b)

} 8 parameters (4 independent)

$\theta_{\bar{c}|\bar{a}b} = Pr(\bar{c}|\bar{a}, b) = Pr(C = \text{false} \mid A = \text{false}, B = \text{true})$.

$Pr(a\bar{b}c\bar{d}) = Pr(a) \cdot Pr(\bar{b}|a) \cdot Pr(c|a\bar{b}) \cdot Pr(\bar{d}|ac)$



d - Separation:

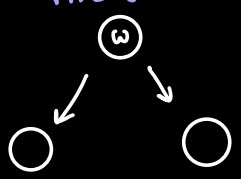
We know z

Sequential Valve:

Closed iff

$w \in z$

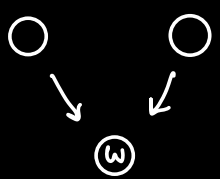
Divergent Valve:



Closed iff

$w \in z$

Convergent Valve:



Closed iff

neither w or any of its descendents are in z.

X and Y are d-separated given Z

dsep(X, Z, Y)

iff every path between X, Y is blocked by Z.

↓

Path is blocked by Z iff at least one valve on the path is closed given Z.

INFERENCE:

- Queries
- Algorithms

QUERIES:

1. PRIOR MARGINALS:

Probability of variable without any condition.

Pr(C) → C = yes 3.2%

C = no 96.8%

note: Deterministic → Pr(X) = 0/1.

2. POSTERIOR MARGINALS:

Pr(C | β)

3. MPE : Most Probable Explanation:

Say 5 variables

Evidence : One variable given

The most probable instantiation of other variables given evidences.

4. MAP : Maximum a Posteriori Hypothesis:

Most probable instantiation of a subset of variables given evidences.

MPE can be done using MAP.

COMPLEXITY OF INFERENCE:

- Variable elimination
- conditioning

Complexity depends on

↳ Topology

↳ Treewidth (connectedness)

n : #variables

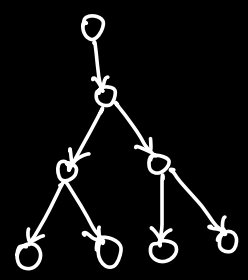
d : #values

w : treewidth

Marginals computed in $O(n \cdot d^w)$

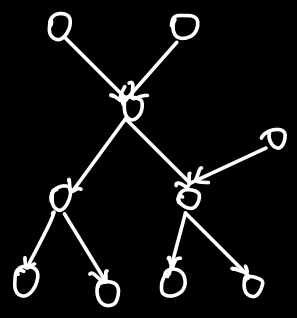
Topology:

* Tree:



* only 1 parent
* $w = 1$

* Polytree: (Singly - Connected Network)



* $w = k$
  ↳ maximum number of parents a node can have.
* Only single path between 2 nodes.

* Multiply - Connected (DAG).

Weighted Model Counting (WMC):

#SAT: model counting

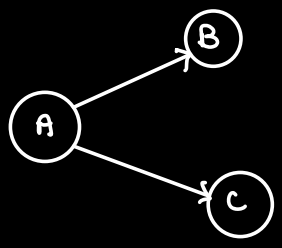
$WMC = \sum_{w \models \Delta} wt(w)$ if $\Delta = (A \vee B) \wedge \neg C$

↓ weight of w

Idea:

$\Delta \xrightarrow{\text{Compile}}$ d-DNNF Circuit + Smooth $\longrightarrow$ WMC in linear time.

Reducing Probabilistic Inference to WMC:



Probabilities: $\theta_A, \theta_{\bar{A}}$

$\theta_{B|A}, \theta_{\neg B|A}, \theta_{B|\neg A}, \theta_{\neg B|\neg A}$

$\theta_{C|A}, \theta_{\neg C|A}, \theta_{C|\neg A}, \theta_{\neg C|\neg A}$

Step 1: Take each node in the Bayesian network as a variable. Create a variable for each probability.

A, B, C, $\neg$A, $\neg$B, $\neg$C → "Indicator" variables (A, B, C)

P_1 θ_A

P_2 $\theta_{\neg A}$

P_3 $\theta_{B|A}$

P_4 $\theta_{\neg B|A}$

P_5 $\theta_{B|\neg A}$

P_6 $\theta_{\neg B|\neg A}$

P_7 $\theta_{C|A}$

P_8 $\theta_{\neg C|A}$

P_9 $\theta_{C|\neg A}$

P_{10} $\theta_{\neg C|\neg A}$

"Parameter" Variables (P_i)

Step 2 : Create KB

$\Delta =$ $A <=> P_1$

$\neg A <=> P_2$

$A \wedge B <=> P_3$

$A \wedge \neg B <=> P_4$

$\neg A \wedge B <=> P_5$

$\neg A \wedge \neg B <=> P_6$

$A \wedge C <=> P_7$

$A \wedge \neg C <=> P_8$

$\neg A \wedge C <=> P_9$

$\neg A \wedge \neg C <=> P_{10}$

Step 3 : Assign Weights

$w(A) = w(\neg A) = w(B) = w(\neg B) = w(C) = w(\neg C) = 1$

$w(P_i) = \theta_i$ i.e. $w(P_1) = \theta_A$, $w(P_7) = \theta_{C|A}$

$w(\neg P_i) = 1$

eg: One Model of Δ:

$A = T, B = F, C = T \Rightarrow A \; \neg B \; C \; P_1 \; P_4 \; P_7 \; \neg P_2 \cdots \neg P_{10}$

$w(\text{model}) = 1 \times 1 \times 1 \times \theta_A \times \theta_{\neg B|A} \times \theta_{C|A} \times 1 \cdots \times 1$

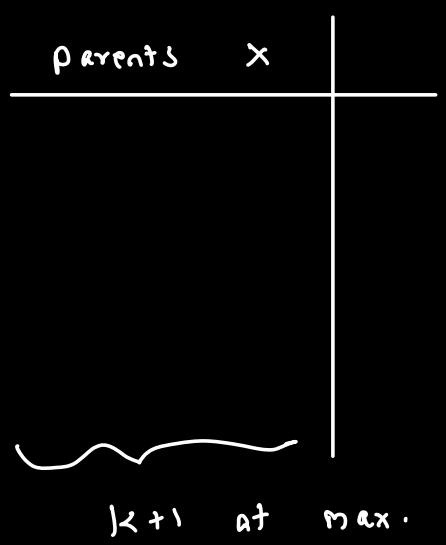
Step 4: $\Delta \xrightarrow{\text{Compile}}$ d-DNNF Circuit + Smooth $\longrightarrow$ WMC in linear time.

Step 5: Given Query α

$$Pr(\alpha) = WMC(\Delta \wedge \alpha).$$

ADVANTAGE: Not dependent on treewidth!

SIZE OF BAYESIAN NETWORK:



n: variables

k: maximum # parents per node.

d: maximum # values per variable

Size of CPT $\leq 2^{k+1}$ if 2 value per variable

generally

$\leq d^{k+1}$

There are n CPTs

Size of BN: $O(n \cdot d^{k+1})$. <<<

Joint Probability Table Size: d^n.

Modelling Bayesian Networks:

Step 1:

Variable, Values

Step 2:

Edge

Step 3:

CPT

→ Problem Statement

→ Subjective Beliefs

→ Learning from data.

Data:

Complete: All data present

Closed Form (Efficient)

Incomplete: Some data missing.

EM: Expectation Maximization.

Say we have 2 CPT_S - A, B

Which to choose.

Run the data through and get Pr(Data)

↳ $P_1, P_2 \dots P_n$ $P_i = Pr(\text{example } i)$

↳ this is the score

MAXIMUM LIKELIHOOD PRINCIPLE : $\arg\max_{CPT_S} Pr(Data \mid CPT_S)$

SENSITIVITY ANALYSIS:

Given we want Posterior Marginal to be some value, what changes can help achieve this.

Evidence : E

$Pr(R|E) = 10\%$

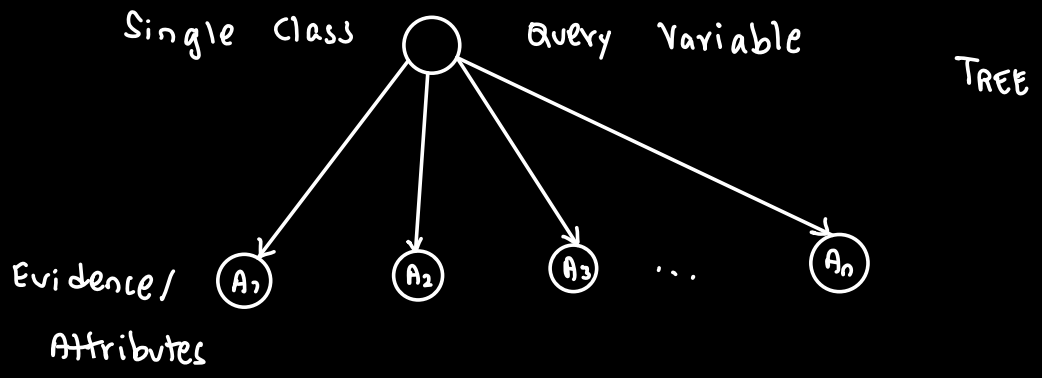
We want $Pr(R|E) \leq 5\%$

Run Sensitivity Analysis

eg: → False Negative for x

reduce from 10% to 4%.

NAIVE BAYES STRUCTURE:



Here we assume final Query variable is a single value. eg: Disease query can be Flu or Cold not both.

SINGLE FAULT ASSUMPTION!

LEARNING:

* Parameters
* Structure

PARAMETER LEARNING:

Structure Given.

1. Complete Dataset:

1. Take the dataset => Generate empirical distribution (Probability of each world).

A	B	C	$Pr_D(\cdot)$
T	T	T	2/16
⋮			
F	F	F	1/16

$Pr_D(\cdot) \longrightarrow$ Dataset

2. Compute Probabilities:

$$\theta_{\bar{a}|b} = Pr(\bar{a}|b) = \frac{Pr_D(\bar{a}, b)}{Pr_D(b)}$$

These are the MLE of them.

ONE ITERATION!

2. Incomplete Dataset:

EM [Expectation Maximization]

Generate random CPT_1.

* $CPT_1 \rightarrow BN_1 \rightarrow Pr_1(\cdot)$

Replace incomplete entries

A	B	C
T	F	?

eg:

T	F	T	x
T	F	F	y

T	F	?	
↓			
T	F	T	x
T	F	F	y

$x = Pr_1(C = T \mid A = T, B = F)$

$y = Pr_1(C = F \mid A = T, B = F)$

Now:

T	T	T	1
T	T	F	1
T	F	T	x
T	F	F	y

} Complete

* Use this to get CPT_2

* $CPT_2 \rightarrow BN_2 \rightarrow Pr_2(\cdot)$

⋮

converges.

[Local Search Algorithm]

$CPT_1 \cdots \xrightarrow{\text{Inference}} CPT_2 \xrightarrow{\text{Inference}} \cdots$

If n loops $\Rightarrow$ n inferences!

Learning Structure:

1. Local Search Methods:

   * Choose one structure.

   * Change and move to a neighboring structure with better score.

     * add, remove, reverse an edge

2. Systematic Search Methods:

   A* search.

What if we use MLE?

We end up with complete structure.

Overfitting. More parameters → Higher likelihood.

↳ Won't generalize.

Score : Likelihood + Penalty Term

eg: MDL Score

↳ α number of independent parameters and Size of data

Model-Oriented vs Query-Oriented Learning:

Unsupervised

[Till Now]

Supervised

Supervised Learning:

BN + Query => Boolean Formula + weights => NNF Circuit (compilation time $O(n \cdot d^w)$)

Convert NNF Circuit to AC

Parameters:

* $\theta_i = Pr(i)$

For evidence: (Indicators)

A	λ_a	$\lambda_{\bar{a}}$
T	1	0
F	0	1
?	1	1

Loss Function:

For each input, find $Pr(B)$ using AC, say P

True Output is: (Label)

B	
x	0
y	1
z	0

→ one-hot representation. Say Q

$$\text{Loss} = \frac{1}{n} \sum \text{Cross-Entropy}(P(x), Q(x))$$

Optimized using Gradient Descent.

CROSS-ENTROPY:

$P(x)$: Prediction

$Q(x)$: label

$$CE = \sum_{x} Q(x) \log_2 P(x)$$

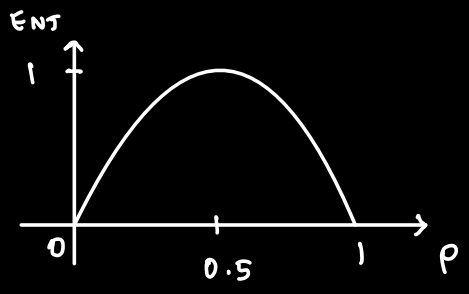
BACKGROUND KNOWLEDGE:

Can be added directly to BN.

DECISION TREES AND RANDOM FORESTS:

ENTROPY:

$$ENT(x) = -\sum_{x} Pr(x) \log_2 Pr(x)$$



$$ENT(x|y) = -\sum_{x} Pr(x|y) \log_2 Pr(x|y)$$

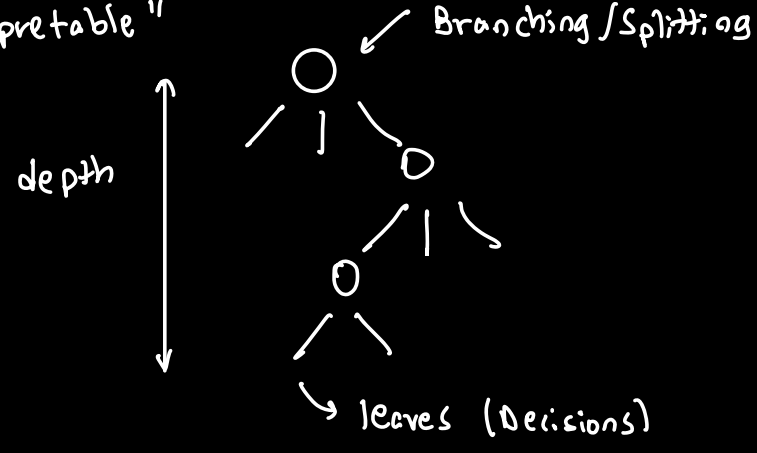
$$ENT(x|y) = \sum_{y} Pr(y)\, ENT(x|y)$$

$$\boxed{ENT(x|y) \leq ENT(x)}$$

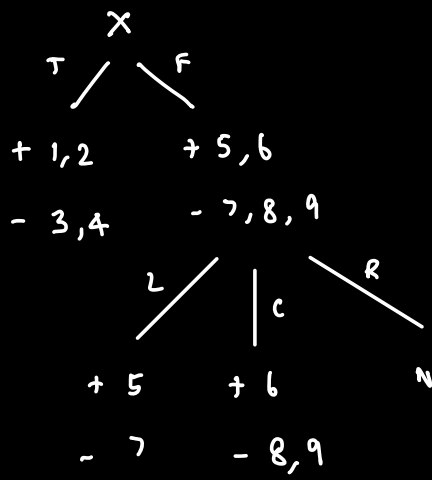
when independent

$$ENT(x|y) = ENT(x)$$

"Interpretable"



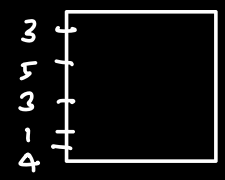
* If no attributes left, return the PLURALITY of left example (majority vote).
* No examples left, return majority of parent examples.



"Interpretable" : Decision Trees > Bayesian Networks > Neural Networks.

RANDOM FORESTS :

1. Ensemble learning methods: Majority vote from multiple Decision trees.
2. Bootstrapped Dataset: Randomly pick with replacement. So, each Decision tree has different Dataset.

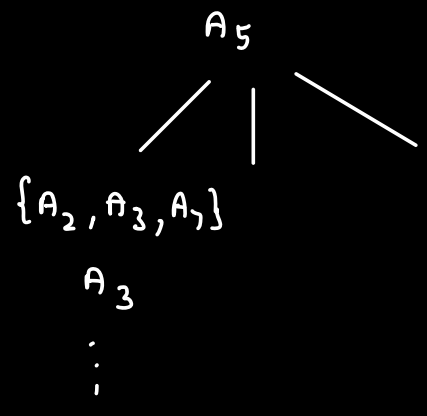


2 is "Out-of-bag" example.

3. Only a subset of attributes is considered at each point. At each point choose random 'm' attributes and select best of the m. Then next step choose random m again.

n = 10 attributes $\{A_1, A_5, A_7\}$

m = 3

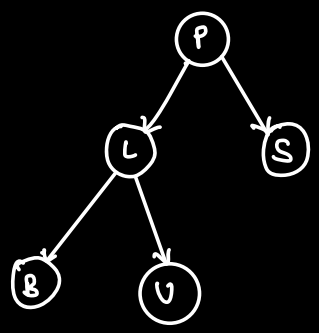


m can be $\sqrt{n}$

4. Accuracy tested on "Out-of-bag" examples.

BAYESIAN NETWORK CLASSIFIER:

Threshold, T



Instance

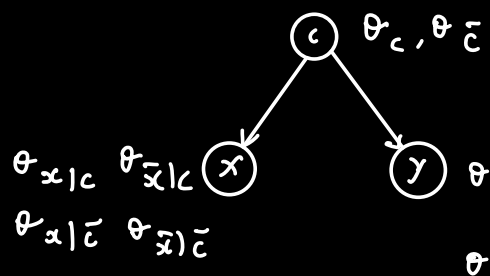
$S = +$, $B = -$, $U = -$

$Pr(P \mid S=+, B=-, U=-) \geq T$

↳ Return Pregnant

else Not Pregnant.

NAIVE BAYES CLASSIFIER:



$$Pr(c \mid x, y) = \frac{Pr(x, y \mid c)\, Pr(c)}{Pr(x, y)}$$

↳ $Pr(x, y \mid c) \cdot Pr(c) + Pr(x, y \mid \bar{c}) \cdot Pr(\bar{c})$

$$= \frac{\theta_{x,y|c} \cdot \theta_c}{\theta_{x,y|c} \cdot \theta_c + \theta_{x,y|\bar{c}} \cdot \theta_{\bar{c}}}$$

θ_c: CPT; $\theta_{x,y|c}$: CPT; $\theta_{\bar{c}}$: CPT

In Naive Bayes

x, y are independent given C

So $\theta_{x,y|c} = \theta_{x|c} \cdot \theta_{y|c}$

$\theta_{x,y|\bar{c}} = \theta_{x|\bar{c}} \cdot \theta_{y|\bar{c}}$ } → CPT

So, output Probability → directly from CPT!

Also, Time complexity for inference: $O(n \cdot d^w)$

here $w = 1$

So $O(n \cdot d)$ //

EXPLAINABLE AI:

Bayesian Network → Compiles into tractable circuit (NNF) with same input-output behavior.

↓ Numeric

↓ Symbolic

This circuit can help explain why we got an output.

"PI-Explanation": eg: $S=1, G=0, F=1, M=1$ is true

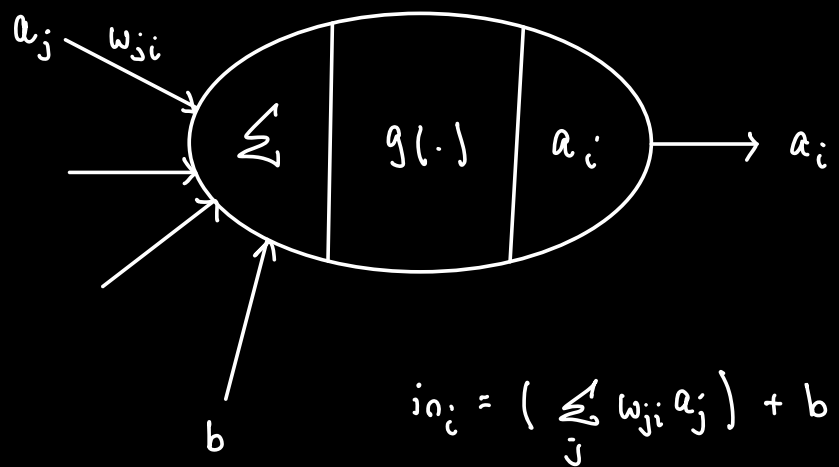
but we see all $F=M=1$ is true

So, only F, M contributed

NEURAL NETWORKS:

- Model-free approach.

NEURONS:



$$in_i = \left(\sum_j w_{ji} a_j \right) + b$$

$$a_i = g(in_i)$$

a_i, a_j : activations

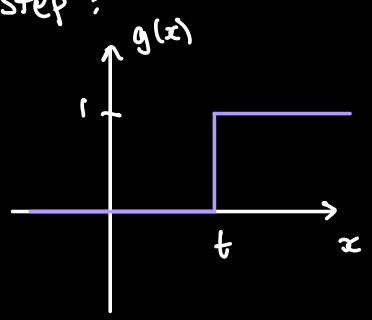
w_{ji} : weights

b : bias

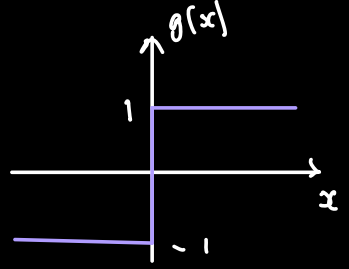
g : activation function

ACTIVATION FUNCTIONS:

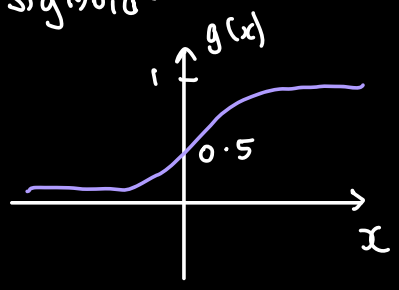
step :



Sign :



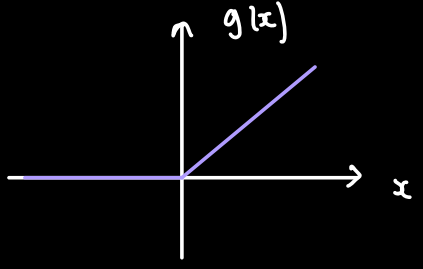
Sigmoid:



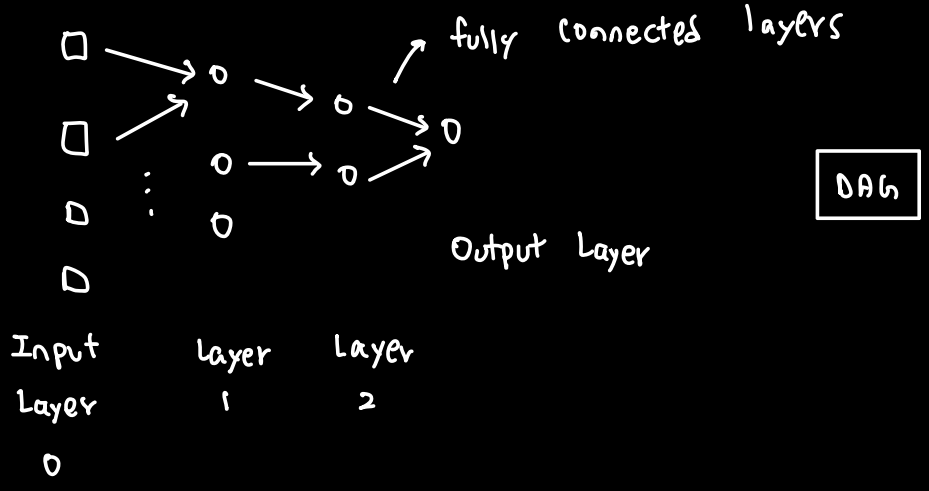
$$g(x) = \frac{1}{1+e^{-x}}$$

(Probabilistic)

ReLU (Rectified Linear Unit):



FEEDFORWARD NN:

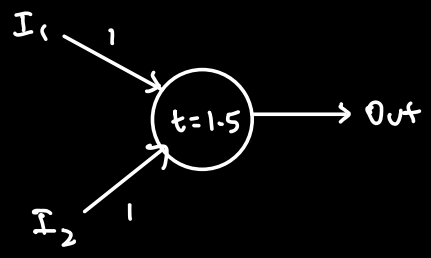


Depth: Number of layers

NN are "Universal Function Approximators".

NEURONS WITH STEP ACTIVATION FUNCTIONS:

→ AND



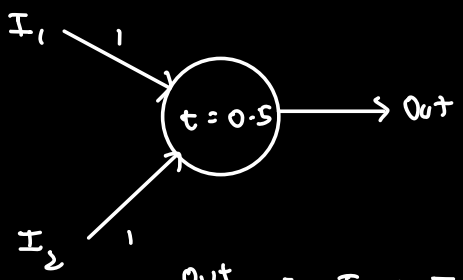
$Out = I_1 \wedge I_2$

So out is true only when $I_1 = I_2 =$ true

So if $w_1 = 1$ $w_2 = 1$, out = 2

$t = 1.5$

→ OR



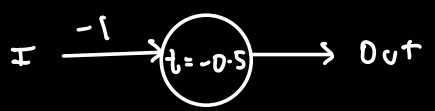
$Out = I_1 \vee I_2$

Out is false only when $I_1 = I_2 =$ false

If $w_1 = 1$, $w_2 = 1$, out = 0.

$t = 0.5$

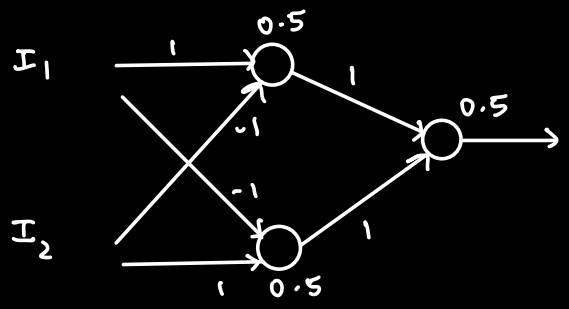
→ NOT



→ XOR :

$I_1 \oplus I_2$

$(I_1 \wedge \neg I_2) \vee (I_2 \wedge \neg I_1)$



$x = I_1 \wedge \neg I_2$

If $w_1 = 1$ $w_2 = -1$

$I_1 \wedge \neg I_2$ is true only when

I_1 = true, I_2 = false

which is $1 \times 1 + (-1 \times 0) = 1$

So $t = 0.5$

$y = I_2 \wedge \neg I_1$

$w_1 = -1$ $w_2 = 1$

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

$t = 0.5$

$out = x \vee y$

Neuron can only solve "linearly separable functions".

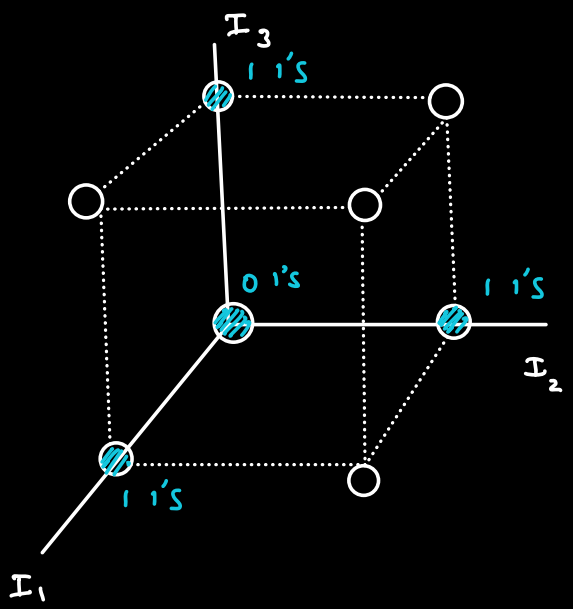
Not XOR.

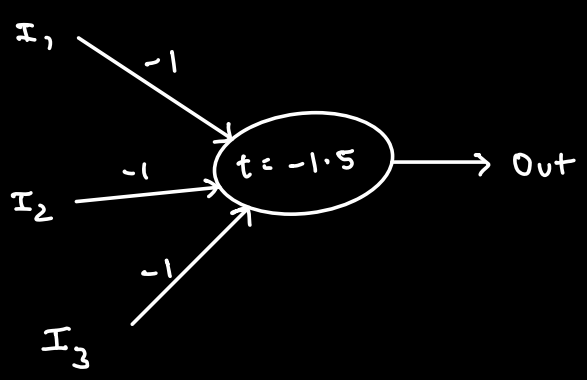
MINORITY:

0 0 1 → 1

0 1 1 → 0

Output = 1 if 1 is minority.





0 -1 -1 -1

000, 001, 010, 100 → 1

011, 110, 101, 111 → 0

-2 -2 -2 -3 ≥ -1.5

NEURAL NETWORK AS A FUNCTION:

$$out = f(\underbrace{in_1, in_2}_{\text{input}}, \underbrace{w_{13}, w_{14} \cdots w_{45}}_{\text{weight}})$$

[Complex non-linear function]

of inputs and weights.

Given an input I_i: in_1, in_2

$$out_i: f_i(\text{weights})$$

TRAINING NEURAL NETWORKS:

Loss function: [Optimize]

Cross Entropy (CE), Mean-Square Error (MSE).

	in_1	in_2	out	Label (l_i)	$NN(I_i)$
I_1					
I_2					
⋮					
I_N					

$$MSE: \frac{1}{N} \sum_{i=1}^{N} (\underbrace{NN(I_i)}_{\to f_i(w_1 \cdots w_k)} - l_i)^2$$

GRADIENT DESCENT:

$\mathcal{L} = f(w_1, w_2, \dots w_K)$ ← Loss function

- Initialize random values
- gradient: $\left(\partial f/\partial w_1, \partial f/\partial w_2, \dots \partial f/\partial w_K \right)$
- Update (Step: η)

OPTIMIZATION: Loss function [Training Data]

PERFORMANCE METRIC: Accuracy [Testing Data + Validation Data]

Epoch: One pass through data.

Batch: Divide training data into batches. → Parallel; → Randomization (SGD)

Epoch 1: Divide Training data into batches (eg: Batch size: 64 examples) [randomly]

↓

Gradient Descent over each batch ↺

↓

Output: Loss 1, Metric 1

Epoch 2: Divide ...

↓

Gradient Descent ...

↓

STOP

STOPPING CRITERION:

1. Epoch Count
2. Till Loss Decreases [Overditting]
3. VALIDATION: Training < Training 80%
   validation 20%

   After every epoch
   ↳ metric on validation data.

   Stop when accuracy (metric) stops increasing / stagnant.

   PATIENCE FACTOR: No improvement over some x epochs - STOP.

Epoch 1 loss 1 metric 1

Epoch 2 loss 2 metric 2

⋮

Epoch K Stop.

[SGD: Stochastic Gradient Descent].