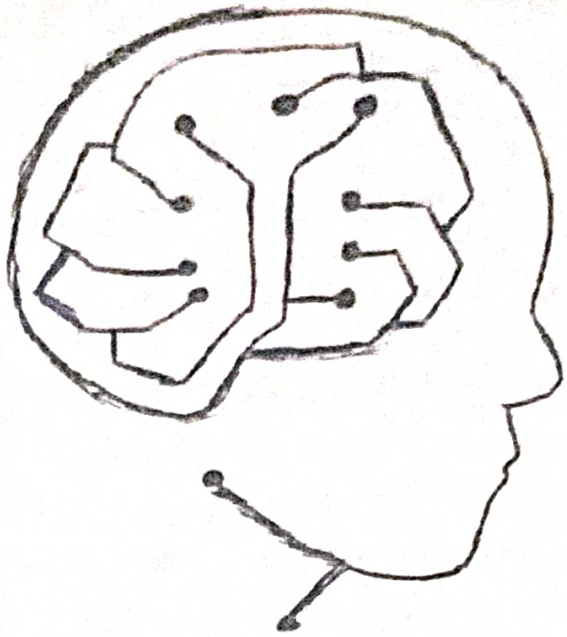


# Machine Learning

N. Likhith Reddy

Types

- Supervised Learning
  - Classification
    - K-Nearest Neighbor
    - Support vector machine
    - Decision Tree
    - Random Forest
    - Naive Bayes
    - Neural Network
    - Stochastic Gradient Descent
  - Regression
    - Linear Regression
    - Logistic Regression
    - polynomial Regression
- unsupervised Learning
  - Clustering
    - Principle Component Analysis
    - Hierarchical clustring
    - singular value Decomposition
    - Independent compo-nent Analysis.
    - k.Means clustering
  - Association
- Reinforcement Learning
  - 3-ways of implementation
    - Model Based
    - policy Based
    - value Based
  - Model
    - Q-learning
    - Markov Decision process.

Definition:-

* AI learn from Data to solve problems.

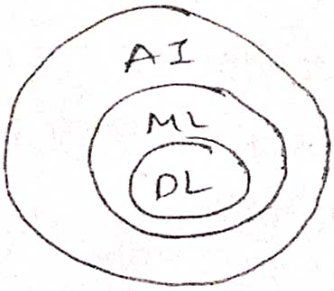
Types:-

# Machine Learning

It is a field of study that gives computers a capability to learn without being explicity programmed.

Example :- Online shopping.

→ Machine adapts to the user based on data.




**Well posed learning Problem :-**

An agent solves a problem or task T, performance P and gain some experience E If P is measured at T it can improve E.

Example :- 1. Hand written recognition problem.

2. Robo driving learning Problem

| Problem | Task(T) | Performance(P) | experience(E |
|---|---|---|---|
| 1. Hand writing recognition learning | classifying the Images & Text | Better classification | A database of home-work let |
| 2. Robot driving learning problem | Drive the car in a 4 lane highway | Source to dest. the avg distance travelled (long & safe) | Images, vehicles on Road. |
| 3. playing chess learning | playing against opponents to win game | make perfect moves to win game | It plays itself to improve |

## Perspectives of machine learning:-

prespective of machine learning involves searching very large space of possible hypotheses to determine one that best fits the observed data and any prior knowledge held by learner

* It is to enable computers to learn from data and make predictions without being explicitly programmed to perform task

Ex:-

→ Sorting Algorithms (merge sort)

## Issues in Machine learning:-

1. What algorithms should be used?
2. Which algorithms perform best for which types of problems?
3. How much trainins data is sufficient? and ~~typer~~ testins data.
4. What kind of methods should be used?
5. What methods should be used to reduce learnins overhead
6. For which type of data which methods should be used?

## Designins A learnins System:-

To get a successful learnins system, it should be designed for a proper design, servel steps should be followed

* To get perfect & efficient system.

4 steps

1. Choosing the training experience.
2. choosing the target function.
3. choosing a representation for target function.
4. choosing a learning algorithm for approximating the target function.

Step-1 :- choosing a training experience.

In choosing a training experience, 3 attributes are taken.

1. type of feedback.
2. degree.
3. distribution of examples.

1) whether the training experience provides direct or indirect feedback regarding the choices made by performance system.

Direct feedback

Indirect feedback

Ex:- checkers game, learning driving.

2) Degree to which learner will control the sequence of training.

Ex:- learning driving.

With trainers help

With trainers Partial help

Completely on your own.

3) How will it represent the distribution of examples over which the performance of final system is measured.

- more possible combinations.
- more situations, more examples.
- learning over distributed range of examples.

Ex:- learning driving.

## Step-2 :- Choosing the Target function :-

What type of knowledge is learnt and how it is used by the performance system.

Ex :- checkers game

→ while moving diagonally, set of all possible moves is called legal moves

- Travel only in forward dir
- only one move per chance.
- Only in diagonal direction.
- Jump over apponent

legal moves
↓
one move
↓
target move

Target function ⇒ V(b)
Board style ⇒ b
legal moves ⇒ B

1. b is final board state that is won, then V(b) = 100.
2. b is final board state that is lost, then (V(b) = -100).
3. b is final board state that is draw,

then $v(b) = 0$

4. If b is not final state then $v(b) = v(b')$

$b' \rightarrow$ best final state

Step-3 :- Choosing A Representation for target function :-

For any board state, we calculate function 'c' as linear combination of following board features c(b)

Features :-

$x_1$ - No. of black pieces on board

$x_2$ - No. of red pieces on board

$x_3$ - No. of black kings on board

$x_4$ - No. of red kings on board

$x_5$ - No. of black pieces threatened by red.
(blacks which can be beaten by red)

$x_6$ - No of red pices threatnd by black

$\hat{V}(b) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_3$

(representation of c(b))

$(w_0 - w_6)$ = numerical coefficients (or) weight of each features determine the importance / weightage of features

$w_0$ → additive constant.

Step-4 : choosing a learning Algorithm for approximating the target function

To learn a target function (f) we need a set of training examples.

(describe a particular board state(b) and training value $V_{train}(b)$)

ordered pair = $(b, V_{train}(b))$

(training example representation)

example: black won the game

(i.e $x_2 = 0$, which means no red

$$V_{train}(b) = +100$$

$$b = (x_1 = 3, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 0, x_6 = 0)$$

$$< (x_1 = 3, x_2 = 0, x_3 = 1, x_4 = 0, x_5 = 0, x_6 = 0) + 100 >$$

we need to do 2 steps in this phase

① Estimating training values:-

In every step, we consider successor (depending on the next step of opponent)

$$V_{train}(b) \leftarrow \hat{V}(successor(b))$$

↓ represent the next board state.

(estimating that this move will help/ destroy opponent)

$\hat{V}$ → represent approximation

2. Adjusting the weights:-

There are some algorithms to find weights of linear functions

Here, we are using LMS (least mean square) (used to minimise the error)

$$Error = (v(train(b) - v^n(b))^2$$

If error = 0, no need to change weights.

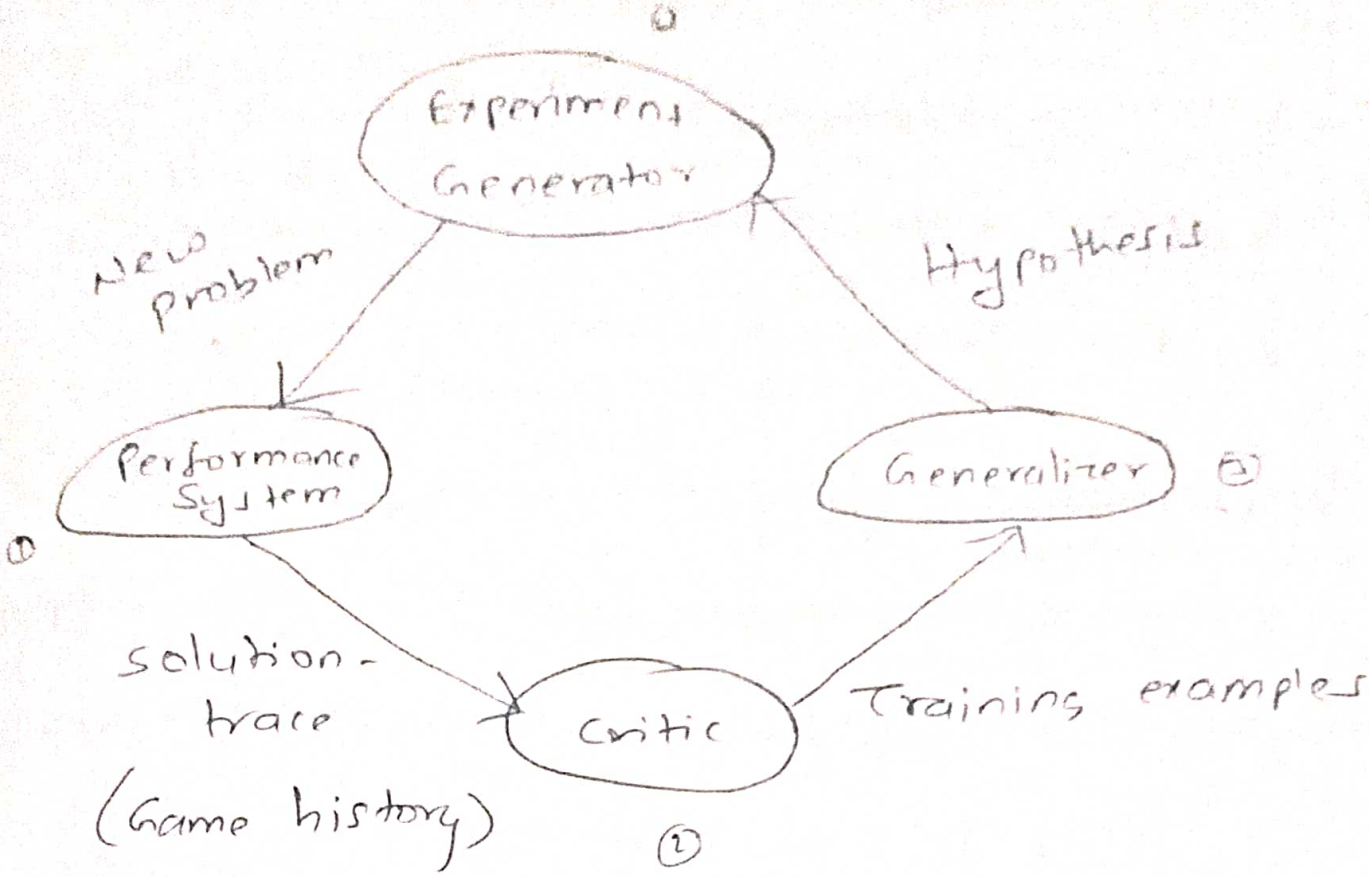
If error is positive, each weight is increased in proportion.

If error is negative, each weight is decreased in proportion.

Final Design:-

has 4 different modules.

1. performance system.
2. critic
3. Generalizer
4. Experiment Generator.

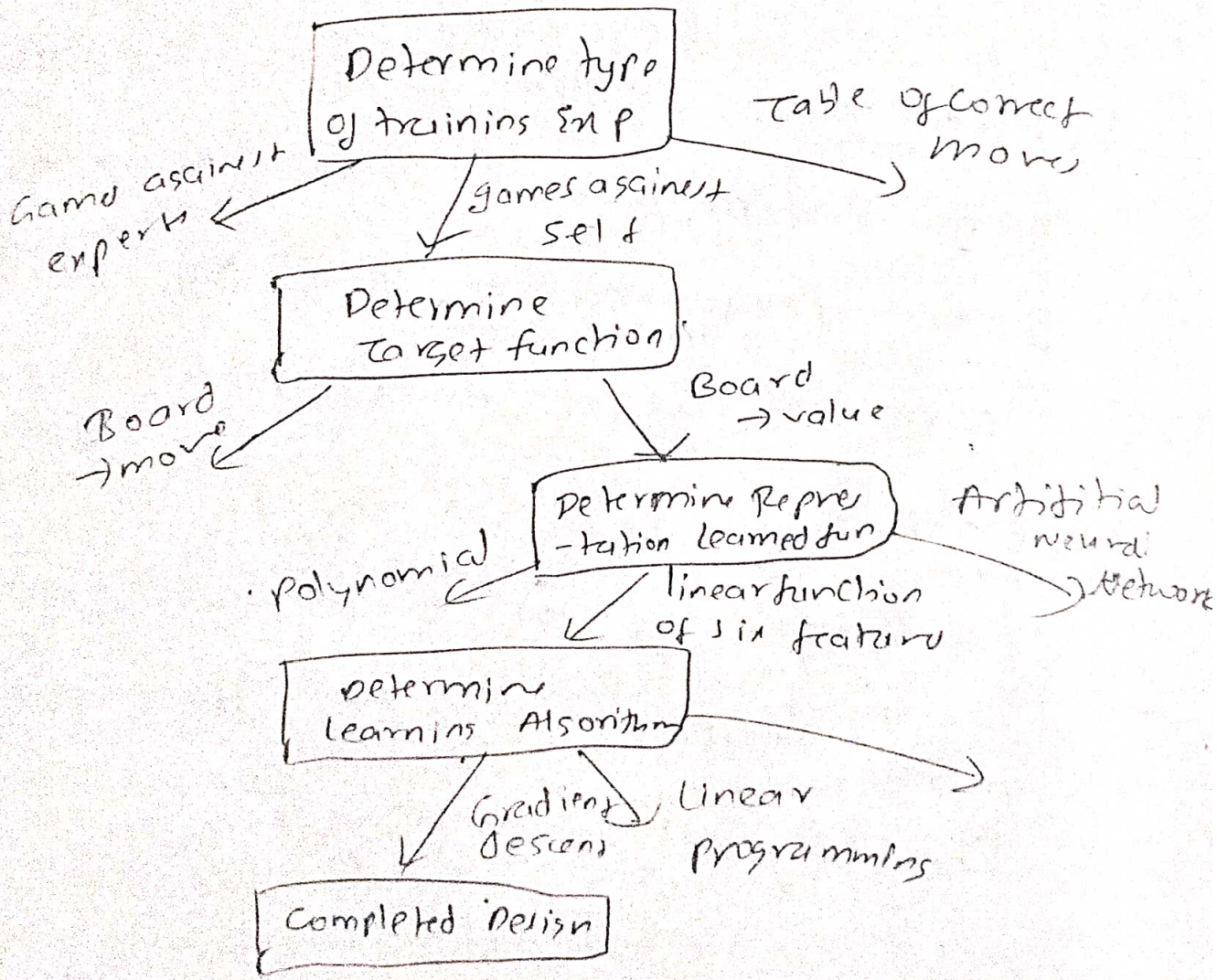



1. Performance System :- It is the module that perform and must solve the given performance task by usins the learned target functions.

2. Critic :- takes as input the history or trace of the game and produces as output a set of trainins examp of a traset function.

3. Generalizer :- takes as input the training examples and produces an output hypothesis that is its estimate of the target function.

4. Experiment Generator :- It takes as input the current hypothesis and outputs a new problem for the performance system to explore.



# Concept Learning :-

Acquiring general concepts from specific training examples.

1. Learning General concepts :-

* The learning process involves understanding general concepts from specific examples.
* like "bird", "car", "study more for the exam".

2. Concepts as subset Definitions :-

* General concepts represent subsets of objects or events within a larger set.
* Example :- "bird" is a subset of all animals

3. Concepts as Boolean functions :-

* Alternatively, concepts can be seen as boolean-valued functions.
* Example :- "A function could return "true" for birds and "false" for other animals.

4. Automatic interface of concepts :-

* The chapter's focus: Automatically deducing the general definition of a concept.

5. Concept learning :-

* It involves inferring boolean-valued functions.

* input :- Specific objects.
  output :- whether they belong to the concept.

* The process is about approximating a boolean-valued function using these examples.

## Concept Learning Task :-

1. Example Task :- "EnjoySport" concept

* Target concept :- "Days on which my friend Aldo enjoys his favorite water sport."

* Table provide example days, each with attributes and the "Enjoysport" value.

| Example | Sky | Air Temp | Humidity | Wind | Water | Forecast | Enjoysport |
|---|---|---|---|---|---|---|---|
| 1 | Sunny | warm | Normal | strong | warm | same | Yes |
| 2 | sunny | warm | High | strong | warm | same | Yes |
| 3 | Rainy | cold | High | strong | warm | change | No |
| 4 | sunny | warm | High | strong | cool | change | Yes |

Table 2.1

2. Hypothesis Representation

* Simple representation: Each hypothesis consists of a conjunction of constraints on six attributes.

Sky, Air Temp, Humidity, Wind, Water, Forecast.

* Constraints can be "?", a specific value or "0"

3. General Hypotheses

- Most general hypothesis: (?,?,?,?,?,?)
- Most specific hypothesis: (0,0,0,0,0,0)

4. Learning Task Summary

* Concept learning Task:- Learning the set of days for which EnjoySport = yes by describing this set through attribute constraints.

* Components of the task:- Instances, Target concept, Hypothesis, Training examples.

5. Notation

* instances: X: Possible days with attributes (Sky, Airtemp, Humidity, Wind, Water, Forecast)

* Target concept c:

  EnjoySport : X → {0, 1}.

* Hypotheses H: Conjuction of attribute constraints

* Training examples D: positive and negative examples of the target function.

6. Inductive Learning Hypothesis

* Assumption:- The best hypothesis for unseen instances is the one that best fits observed training data.

* Fundamental assumption of inductive learning.

## Concept Learning as Search (Section 2.3)

1. Concept Learning as a Search problem

* concept learning can be seen as a Search task in a space of hypothesis defined by the hypothesis representation.

* The goal is to find the hypothesis that best fits the provided training examples.

* The choice of hypothesis representation defines the space of hypotheses available to the learning algorithm.

* The example of the "EnjoySport" learning task involves a finite hypothesis space but real-world tasks can have larger or infinite spaces.

2. Strategies for Searching Hypothesis Space :-

* Learning algorithms need to efficiently Search through the hypothesis space.

* The text introduces the concept of a "general-to-specific ordering of hypothesis."

## 3. General to - Specific ordering of Hypotheses

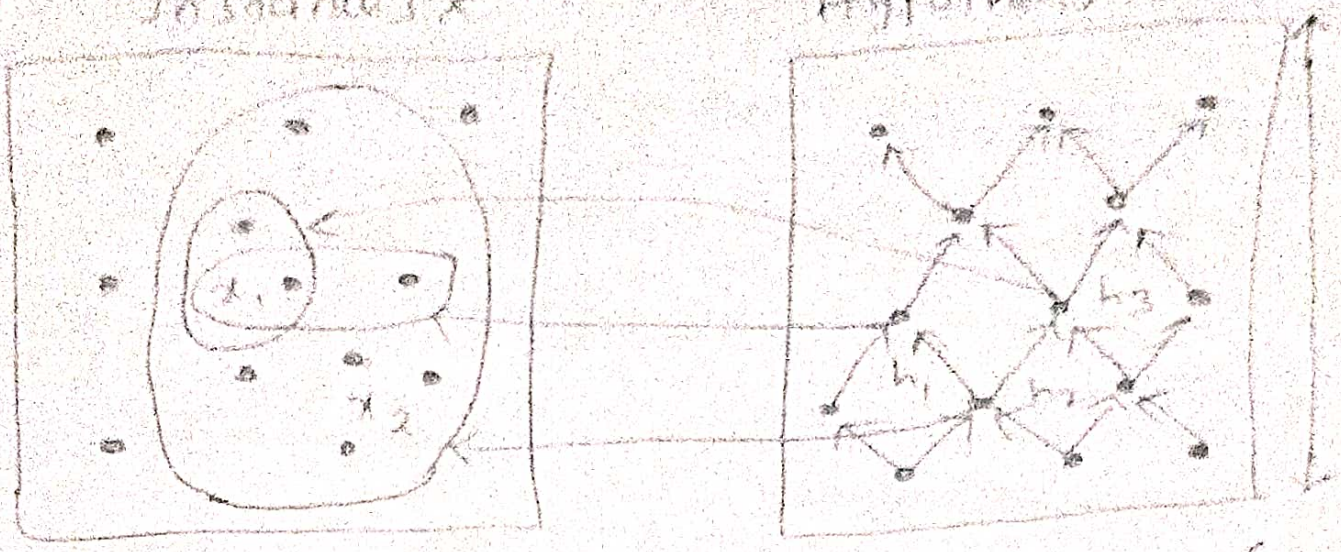
* Many concept learning algorithms leverage the structure of hypotheses in a general-to-specific order.
* Hypotheses can be compared in terms of being more general or more specific.
* A hypotheses is considered "more general" if it classifies at least as many instances as another hypothesis.
* The formal definition of the "More-general-than" relation is provided.
* The relation is reflexive, anti-symmetric, and transitive, making it a partial order over the hypothesis space.

4. Illustrating General-to-Specific ordering:

* Example hypotheses h1, h2, and h3 from the "Enjoy sport" task are shown in figure 2.1
* h2 is more general than h1 and h2 is more general than h3
* Neither h1 nor h3 is more general than the other, they dont subsume each other.

5. Importance of General-to-Specific ordering

* The partial order (PS) Provides structure in the hypothesis space for concept learning.
* Concept learning algorithm can be use this structure for more efficient hypothesis Search.




$x_1$ = <Sunny, Warm, Heigh, Strong, Cool, Same>

$x_2$ = <Sunny, Warm, High, Light, Warm, Same>

$h_1$ = <Sunny, ?, ?, Strong, ?, ?>

$h_2$ = <Sunny, ?, ?, ?, ?, ?>

$h_3$ = <Sunny, ?, ?, ?, Cool, ?>

Fig 2.1

## FIND-S Algorithm and its Application :-

1. Using the More-General-Than partial orderings :-

* Concept learning involves searching for a hypothesis consistent with training examples.

* The Search starts with most specific hypothesis and generalized when necessary to cover positive examples.
* The more-general-than partial ordering is employed to structure the search
* The "Find-S" algorithm (table 2.3) introduced for this purpose.

1. Initialize h to the most specific hypothesis in H
2. For each positive training instance x
   - For each attribute constraint $a_i$ in h
     If the constraint $a_i$ is satisfied by x
     Then do nothing
     Else replace $a_i$ in h by the next more general constraint that is satisfied by x.
3. Output hypothesis h.

Table 2.3
Find-S Algorithm.

2. Illustrates the Find-S Algorithm

* Assume a sequence of training example from the "EnjoySport" task table 2.1

* Start with the most specific hypothesis and generalize as needed to cover positive examples.

* The algorithm ignores negative examples as long as the hypothesis is consistent with them.

3. Guarantees and Questions Raised :-

* The Find-S algorithm guarantees the most specific hypothesis consistent with the trainins data in conjunction based hypothesis spaces

* important questions remain -

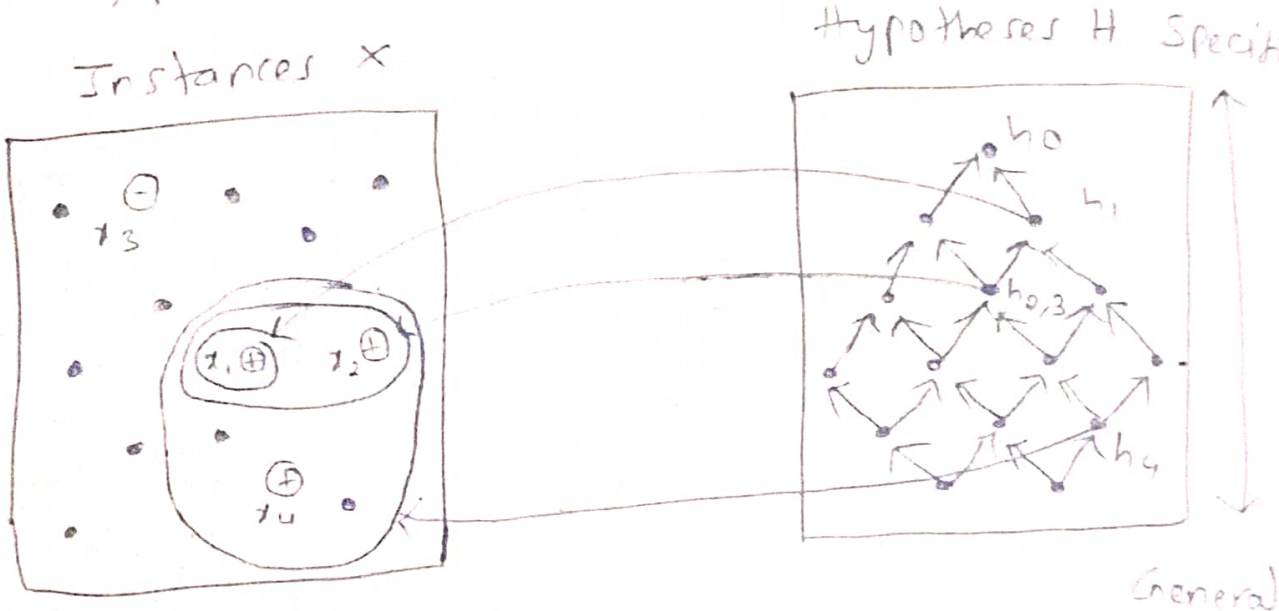
a. Has the learner converged to the correct target concept?

b. Why prefer the most specific hypothesis? is it always the best choice

c. Handling inconsistent training examples (errors or noise)

d. What if there are several maximally specific, consistent hypotheses in conjunction-based spaces?

e. Defining hypothesis spaces without a maximally specific consistent hypothesis.




$x_1$ = <Sunny, warm Normal Strong warm Same>, +

$x_2$ = <Sunny warm High Strong warm same>, +

$x_3$ = <Rainy Cold High Strong warm change>, −

$x_4$ = <Sunny warm High Strong cool change>, +

$h_0$ = <ϕ, ϕ, ϕ, ϕ, ϕ, ϕ>

$h_1$ = <Sunny warm Normal Strong warm Same>

$h_2$ = <Sunny warm ? Strong warm Same>

$h_3$ = <Sunny warm ? Strong warm same>

$h_4$ = <Sunny warm ? Strong ? ?>

The hypothesis space performed by find-S

The search begins (h0) with the most specific hypothesis in H, then considers increasingly general hypotheses (h1 through h4) as mandated by the training examples. In the instance space diagram, positive training examples are denoted by "+", negative by "-" and instances that have not been presented as training examples are denoted by a solid circle.

## Version spaces and the candidate-Elimination Algorithm:-

### Concept Learning and the candidate-Elimination Algorithm :-

1. Introduction :-

- The candidate-Elimination algorithm is an approach to concept learning.
- It addresses limitations of the FIND-algorithm by maintaining a set of consistent hypotheses.

2. Representation:-

- A hypothesis is consistent with training examples if it correctly classifies them.
- A hypothesis h is consistent with a set of trainins examples D if and only if $h(x) = c(x)$ for each example $(x, c(x))$ in D.
- The version space, denoted VS H VD, contains all plausible versions of the target concept.

3. The List-Then-Eliminate Algorithm:-

* It is a simple learnins Algorithm that list all hypotheses in H and eliminates inconsistent ones when processins trainins examples.
* However, it requires enumeratins all hypotheses in H, makins it unrealistic for non-trivial hypothesis space

4. A more compact Representation for version spaces

* The Candidate-Elimination algorithm represents the version space with the most general (G) and most specific (S) hypothesis.

* G and S delimit the version space within the partially ordered hypothesis space.

* The version space includes hypothesis that are more general or more specific than G and S, forming a compact representation

5 version space Representation Theorem

* The version space is the set of hypotheses contained in G, plus those contained in S, plus those that lie between G and S in the hypothesis space.

6. The candidate-elimination learning algorithm

* It initializes G and S with the most general and most specific hypotheses, respectively.
* It processes training examples and updates G and S based on the consistency of hypotheses.
* positive examples lead to the generalization of S, while negative examples lead to the specialization of G.
* The algorithm terminates with a version space containing consistent hypotheses.

7. An Illustrative Example :-

* Example trace of the candidate-elimination algorithm applied to training examples,
* positive examples lead to generalization of the S boundary, and negative examples lead to the specialization of the

G boundary.

* The S boundary summarizes past positive examples, and the G boundary summarizes past Negative examples.

* The algorithm converges to a version space containing all consistent hypotheses.

Example-1 :- Trainins with positive Examples :-

- Initial state:
  - Most general hypothesis: $h = (?, ?, ?, ?, ?)$
  - Most specific hypothesis: $h = (\langle$sunny, warm, Normal, strong, warm, same$\rangle)$
  - Training Example: $\langle$Sunny, warm, Normal, strong, warm, same$\rangle$
  - Action:-
  - The training example is positive.
  - The S boundary is revised to be more general : $S = \{\langle$sunny, warm, ?, strong, warm, same$\rangle\}$
  - No change in the G boundary

Example-2:- Training with a Negative Example.

- Current state:-
  - S boundary: 'S = {<Sunny, warm, ?, strong, ?, ?>}
  - G boundary: 'G = {<?, ?, ?, ?, ?, ?>}
- Training Example: '<Rainy, cold, high, strong, warm, change>
- Action:
  - The training example is negative.
  - The G boundary is revised to be more Specific:-
    - New hypothesis in G: 'G = {<Sunny, warm, ?, strong, ?, ?>, <?, warm, ?, ?, ?, ?>}
- No change in the S boundary.

## Remarks on version spaces and candidate elimination :-

### 1. Will the Candidate-Elimination Algorithm converge to the correct hypothesis :-

- The Candidate-Elimination algorithm will converge to the correct hypothesis if there are no errors in the training examples and if there is at least one hypothesis in the hypothesis space (H) that correctly describes the target concept.
- The version space can be monitored to determine when sufficient training examples have been observed to unambiguously identify the target concept.
- Convergence to the correct hypothesis is achieved when the S and G boundary sets converge to a single hypothesis that is identical

- The version space can be monitored

2. What Trainins Example should the learner Request Next?

- The learner can choose to request new trainins examples from an external oracle.
- An optimal query strategy is to generate instances that satisfy exactly half of the hypotheses in the current version space.
- The goal to evenly split the candidate hypothesinto sets that predict "yes" and "no" to determine the correct hypothesis.
- This Strategy minimizes that neimber of required experiments to identify the target concept

3. How can partially learned concepts be used?

- Even if the version space contains multiple hypotheses, the learner can classify certain instances with confidence.
- If an instance is classified as positive by every hypothesis in the version space, it can be classified a positive with confidence.
- If an instances is classified as negative by every hypothesis in the version space, it can be classified as negative with confidence.
- if an instance is classified differently by different hypothesis in the version space, further trainins examples are needed to make a confident classification.

# Inductive Bias

## 1. A Biased Hypothesis Space :-

- If the target concept is not contained in the initial hypothesis space (H), the candidate-Elimination algorithm cannot converge to the correct hypothesis.
- One way to address this is to enrich the hypothesis space to include every possible hypothesis, but this may not be practical.

## 2. An Unbiased Learner :-

- An unbiased learner would have a hypothesis space (H') capable of representing every possible subset of instances.
- The power set of X, containing all possible subsets, provides an unbiased

hypothesis space.

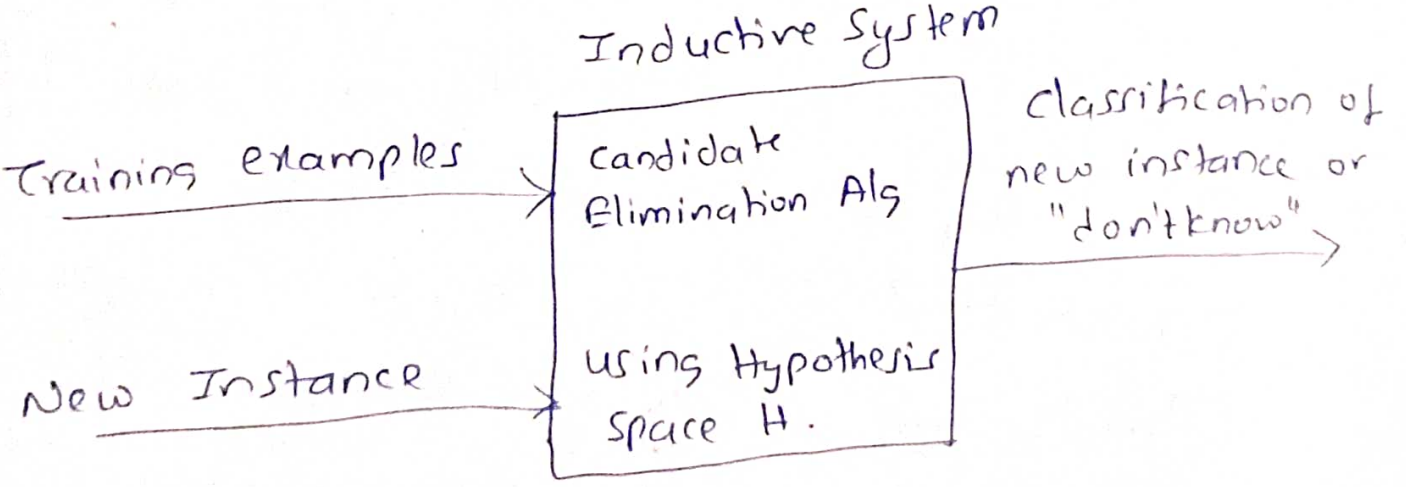
- with this highly expressive hypothesis space, the learner struggles to generalize beyound observed training examples, requiring all instances to be presented as training data

3. The Futility of Bias-free Learning :-

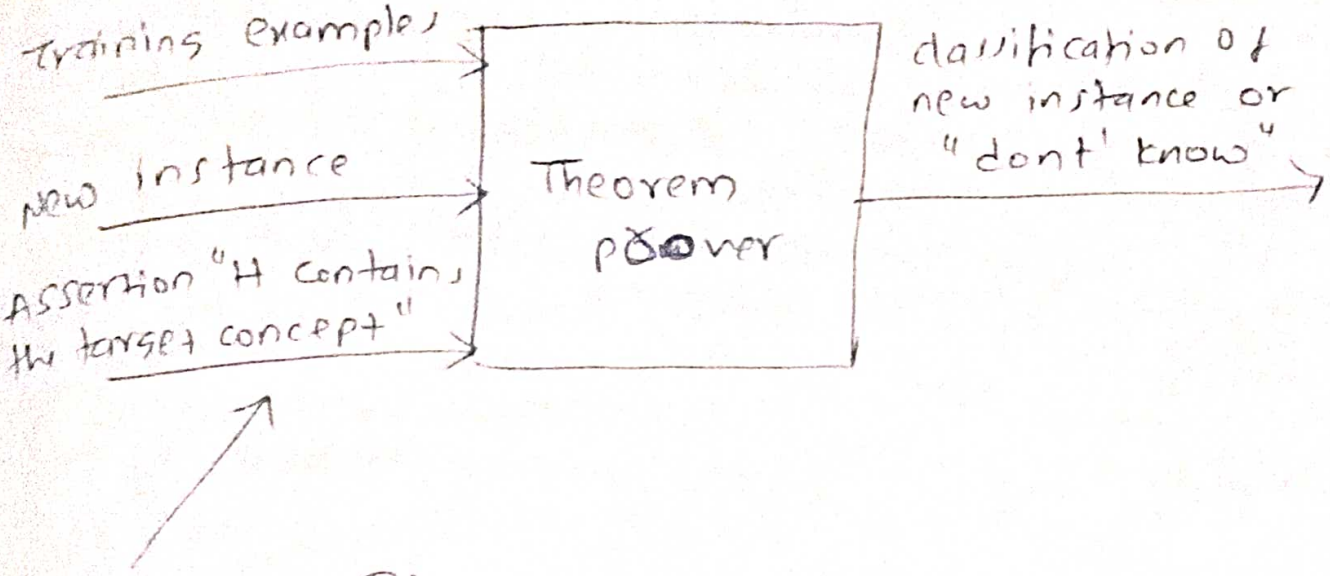
- An unbiased learner makes no a priori assumptions about the target concept and cannot generalize to unseen instances without extensive training data.
- The Candidate-Elimination algorithm, as originally defined, has a bias that the target concept can be represented in its hypothesis space (inductive bias).
- The concept learning algorithm's inductive bias is the set of additional assumptions required for justifying

in inductive interference as deductive interface

- By characterizing inductive systems by their inductive bias, we can improve, compare different learners based on the strength of their biases.
- Examples of learner with varying levels of baical include ROTE-LEARNER, CANDIDATE-ELIMINATION Algorithm, and FIND-S, each making different inductive leaps and classifications



Equivalent deductive system




Inductive Bias made explicit

## Decision Tree Learning :-

### Introduction :-

- Decision tree learning is a method for approximating discrete-valued target functions.
- ~~f~~ represents the learned function as a decision Tree.
- These methods are widely used for various tasks, from medical diagnosis to credit risk assessment.

## Decision Tree Representation:-

- Decision trees classify instances by traversing from the root to a leaf node.
- Each node tests an attribute of the instance.
- Branches correspond to possible attribute values.
- Instances are classified by following the path based on attribute tests.
- Decision trees represent a disjunction of conjuctions of attribute constraints
- Each path from root to leaf represent a conjunction of attribute tests
- The entire tree is a disjunction of these conjunctions.

Example:-

- Figure 3.1 shows a decision tree for classifying saturday mornings for playing tennis.
- An instance is classified by moving down the tree based on attribute tests.
- The tree represents disjunctions and conjunctions of attribute constraints, allowing it to capture complex decision boundaries in the data.

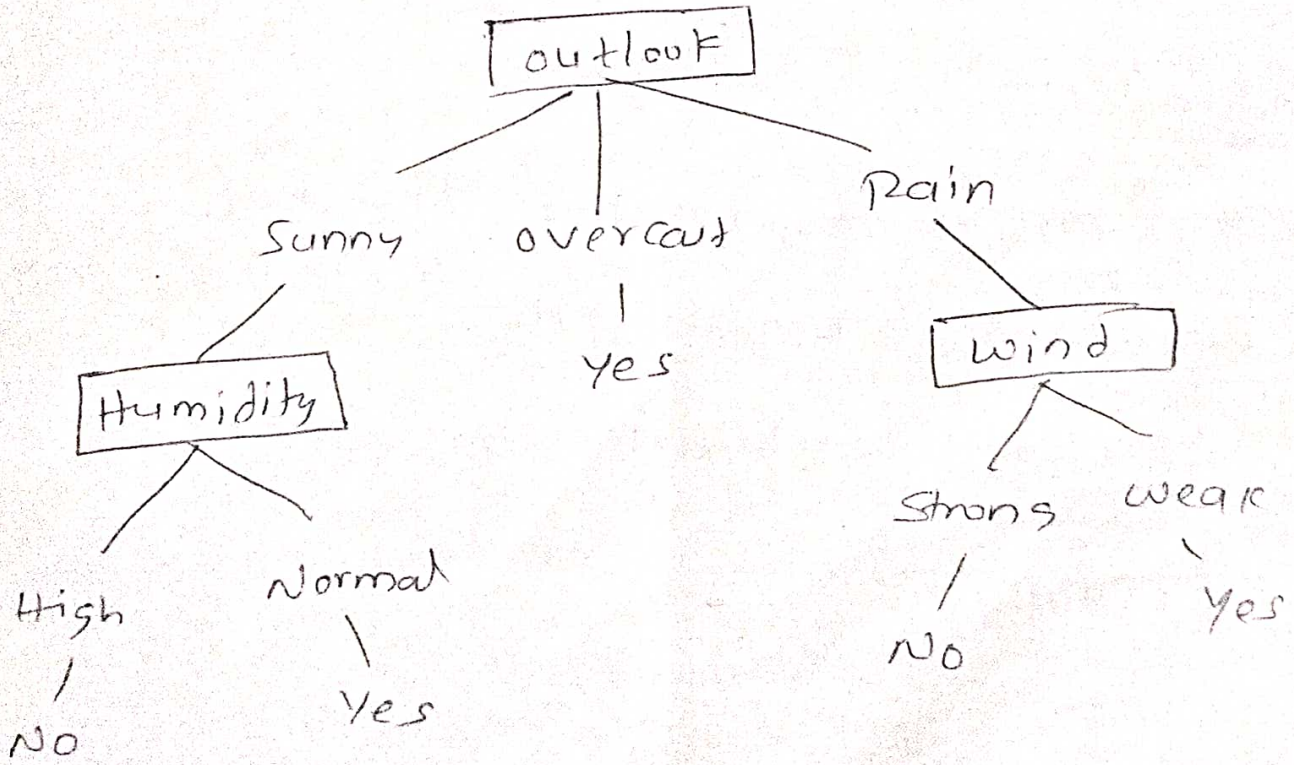



Figure 3.1

How would we represent :-

- $\wedge$, $\vee$, XOR
- $(A \wedge B) \vee (C \wedge \neg D \wedge E)$
- M of N

When to consider Decision Trees.

- Instances describable by attribute value pairs.
- Target function is discrete valued
- Disjunctive hypothesis may be required
- Possibly noisy trainins data.

Examples :-

- Equipment or medical diasnosis.
- Credit risk analysis
- Modelins calendar schedulins preferences.

For figure 3.1 the representing rule in DT's :-

if outlook = sunny AND humidity = normal

OR

if outlook = overcast

OR

if outlook = rain AND wind = weak

then play tennis

Rules can represent a decision tree :-

if item1 then subtree 1

else item2 then subtree 2

elseif ---

Advantages of rules over DT :-

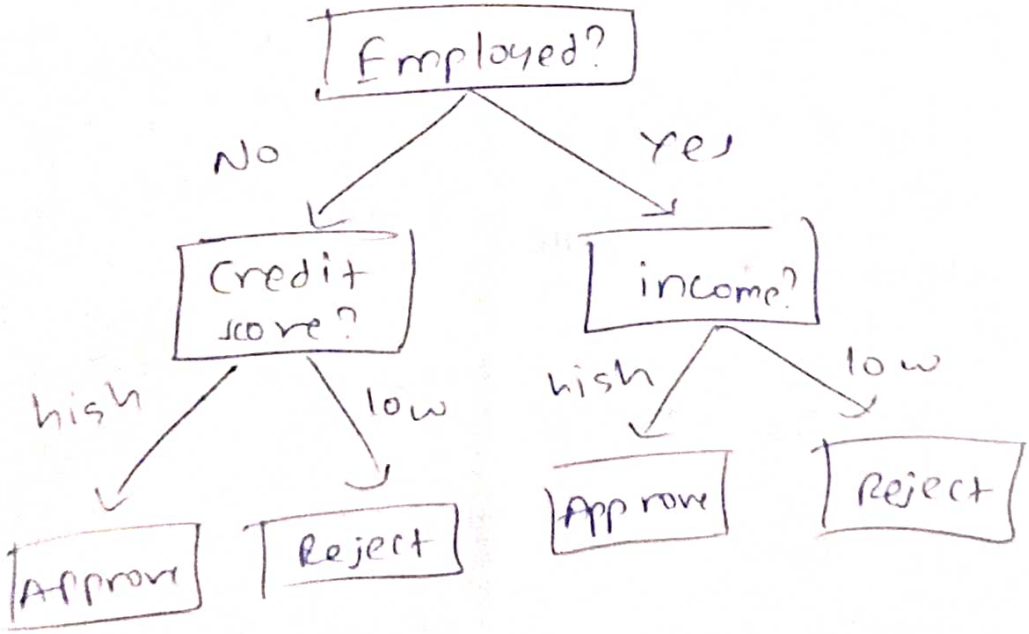
- Rules are easier, to understand, modify and combine.
- Rules can significantly improve classification performance by eliminating unnecessary tests.

- Rules make it possible to combine different decision trees for the same task.

## Advantages of Decision Trees

- Decision trees are simple to understand
- Decision trees have a clear evaluation strategy.
- Decision trees are able to handle both nominal and categorical data.

Example: whether to approve a loan

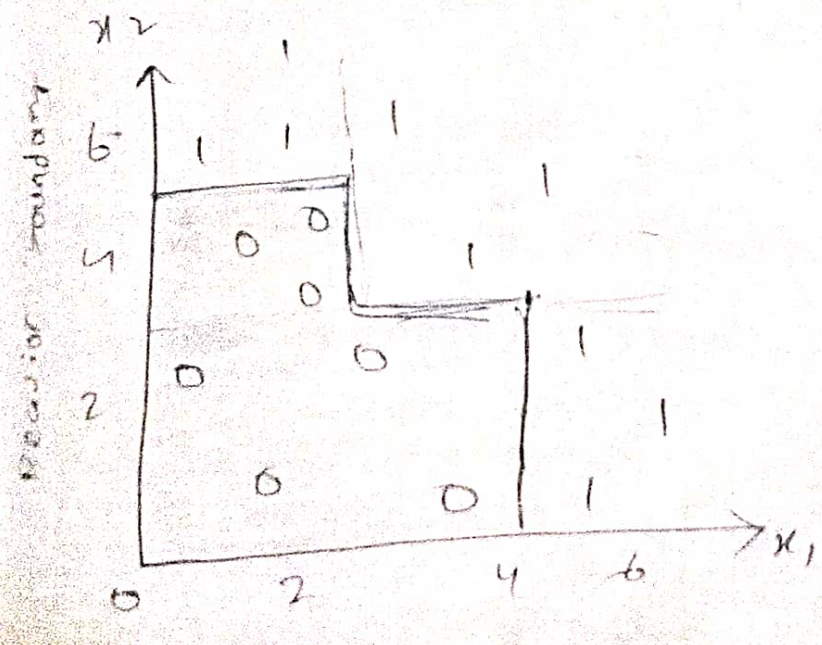


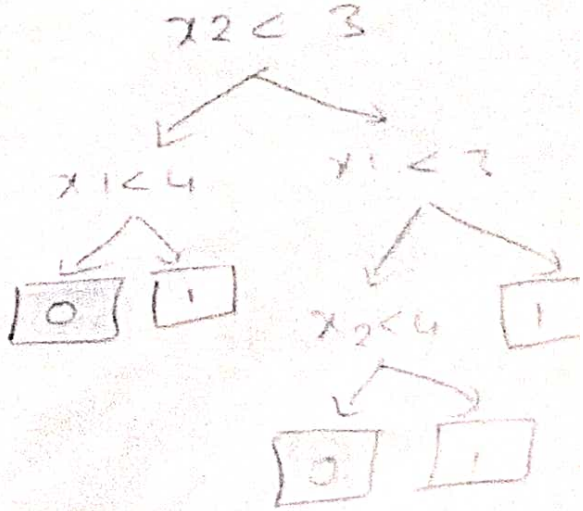
Applications:

- Instances describable by attribute-valued pairs
- Target function is discrete valued
- Disjunctive hypothesis may be required.
- possibly noisy training data.

## Decision Tree - Decision Boundary

- Decision trees divide the feature space into axis-parallel rectangles.
- Each rectangle region is labeled with one label - or a probability distribution over labels.



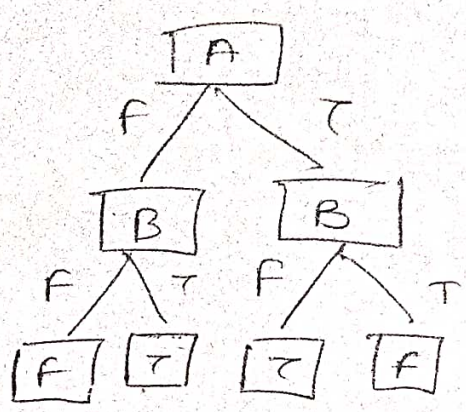



## Expressiveness :-

Decision trees can represent any function of the input attributes

- Boolean operations (and, or, xor, etc)
- All Boolean functions.

| A | B | A xor B |
|---|---|---|
| F | F | F |
| F | T | T |
| T | F | T |
| T | T | F |




## Searching for a good tree :-

- The space of decision trees is too big for systematic search.
- Stop and
  - return the a value for the target feature or
  - a distribution over target feature values

- choose a test (e.g an input feature) to split on.
- for each value of the test, build a subtree for those examples with this value for the test.

The Basic ID3 Algorithm :-

ID3 (Examples, Target_attribute, Attributes)

* Examples are training examples.
* Target_attribute is the attribute whose value to be predicted by the tree.
* Attributes is the list of other attributes that may be tested by the learned decision tree.

→ create a Root node for the tree.
→ if all Examples are Positive, Return the single-node tree Root, with label = +
→ if all Examples are negative, Return the single-node tree Root with label = -

→ if Attributes is empty, Return the single-node tree Root, with label = most common value of target-attribute in Examples.

→ otherwise Begin

- A ← the attribut from Attributes that best * classifies Examples.
- The decision attribute for Root ← A.
- for each possible value, $v_i$, of A
  - → Add a new tree branch below Root, corresponding to the test $A = v_i$
  - → ~~The decision attribute for Root ← A.~~
  - → ~~for each~~
  - → Let Examples, be the subset of Examples that have $v_i$ for A.
  - → if Examples $v_i$ is Empty.
    - Then below this new branch add a leaf node with label = most common value of Target-attribute in Examples

- Else below this new branch add the subtree
  ID3 (Examples$_{v_i}$, Target_Attribute, Attributes - {A}))

- End
- Return Root

Entropy :-

- A measure for
  - uncertainty
  - Purity
  - information content

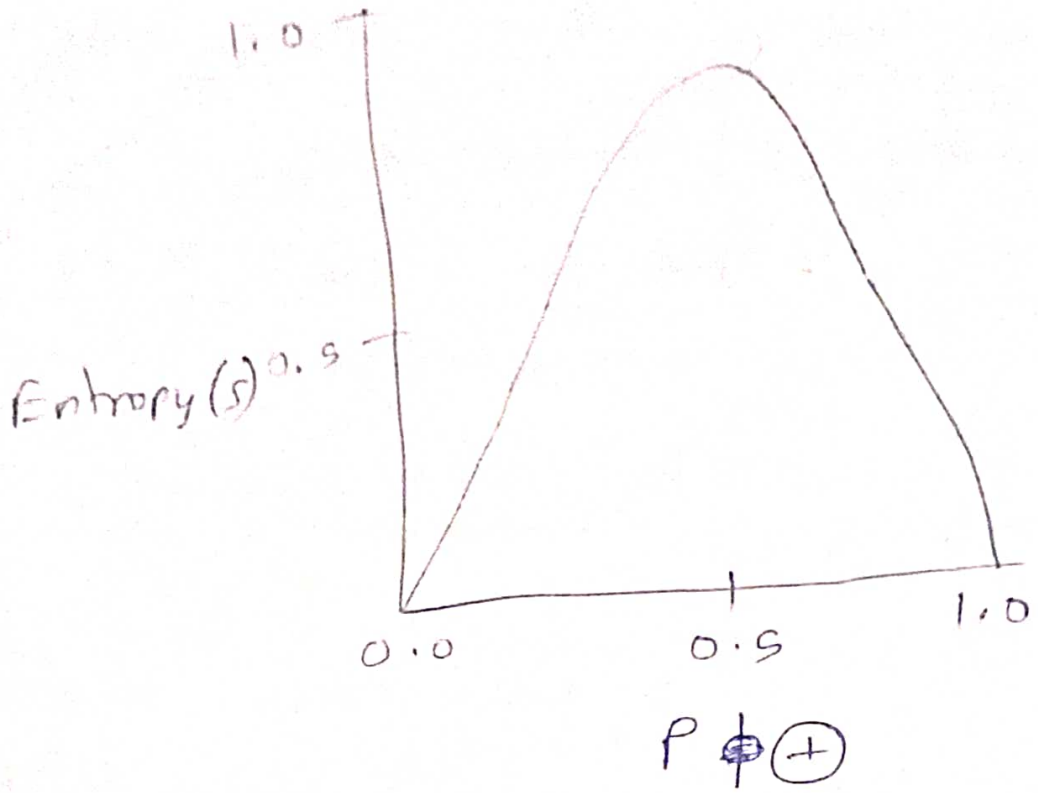
- The entropy is 0 if the outcome is "certain".
- The entropy is maximum if we have no knowledge of the system.

$$\text{Entropy} = \sum_{i=1}^{c} -p_i * \log_2(p_i)$$

info Gain = Entropy (Parent) - (Weighted Avg) Entropy (children)

Gain (S, A) = expected reduction in entropy due to sorting on A.

$$\text{Gain}(S, A) = \text{Entropy}(S) - \sum_{v \in \text{Values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$




- Entropy characterizes then (im) purity of an arbitrary collection of examples.
- Given a collection S (containing positive and negative examples), the entropy of S relative to this boolean classification is.
  - $P_+ / P_-$ is the proportion of positive/negative examples in S.
  - Entropy is 0 id all members of S belons to the same class; entropy is 1 when the collection

contains an equal number of positive and negative examples.

## Information Gain :-

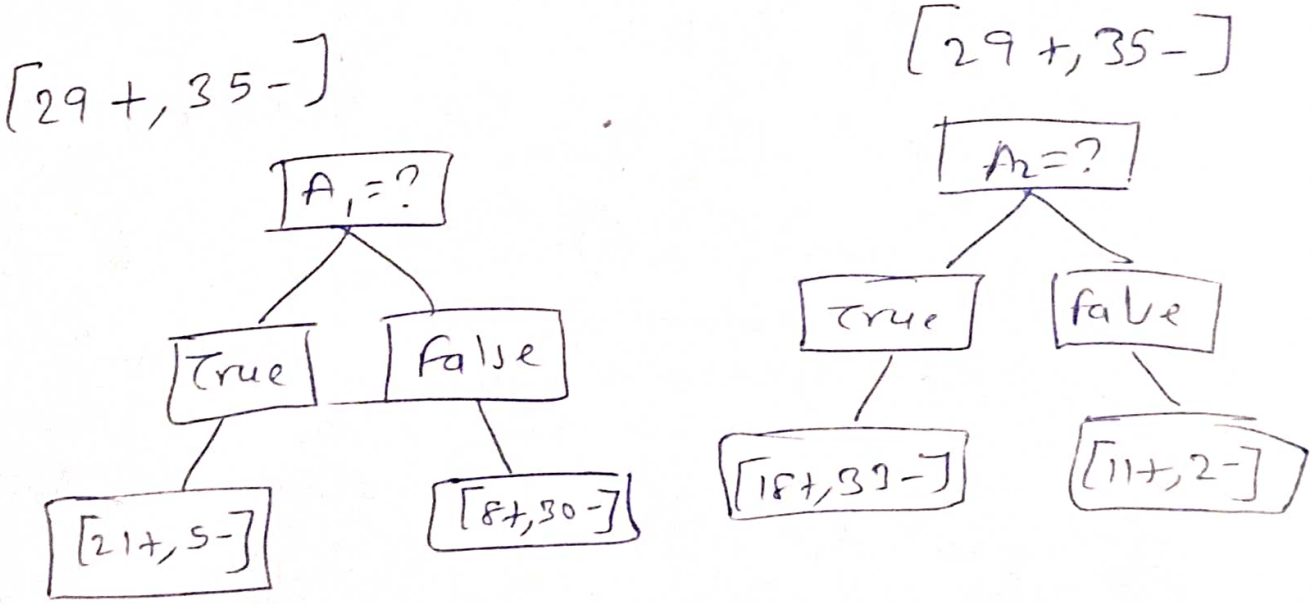
- Information gain measures the expected reduction in entropy caused by partitioning the examples according to an attribute.
- First term :- Entropy of the original collection S; Second term: expected value of entropy after S is partitioned using attribute A ($S_v$ subset of S).
- Gain (S, A): The expected reduction in entropy caused by knowing the value of attribute A.
- ID3 uses information gain to select the best attribute at each step in growing the tree.

Gain(S, A): expected reduction in entropy due to partitioning S on attribute A.

$$Gain(S, A) \equiv Entropy(S) - \sum_{v \in values(A)} |S_v|/|S| \; Entropy(S$$

$$Entropy([29+, 35-]) = -29/64 \log_2 29/64 - 35/64 \log_2 35/64$$

$$= 0.99$$



Entropy([21+,5-]) = 0.71

Entropy([8+,30-]) = 0.74

Gain (S, $A_1$) = Entropy(S)

-26/64 * Entropy([21+,5-])

-38/64 * Entropy ([8+,30-])

= 0.27

Entropy ([18+,33-]) = 0.94

Entropy ([8+,30-]) = 0.62

Gain (S, $A_2$) = Entropy(S)

-51/64 * Entropy ([18+,33-])

-13/64 * Entropy ([11+,2-])

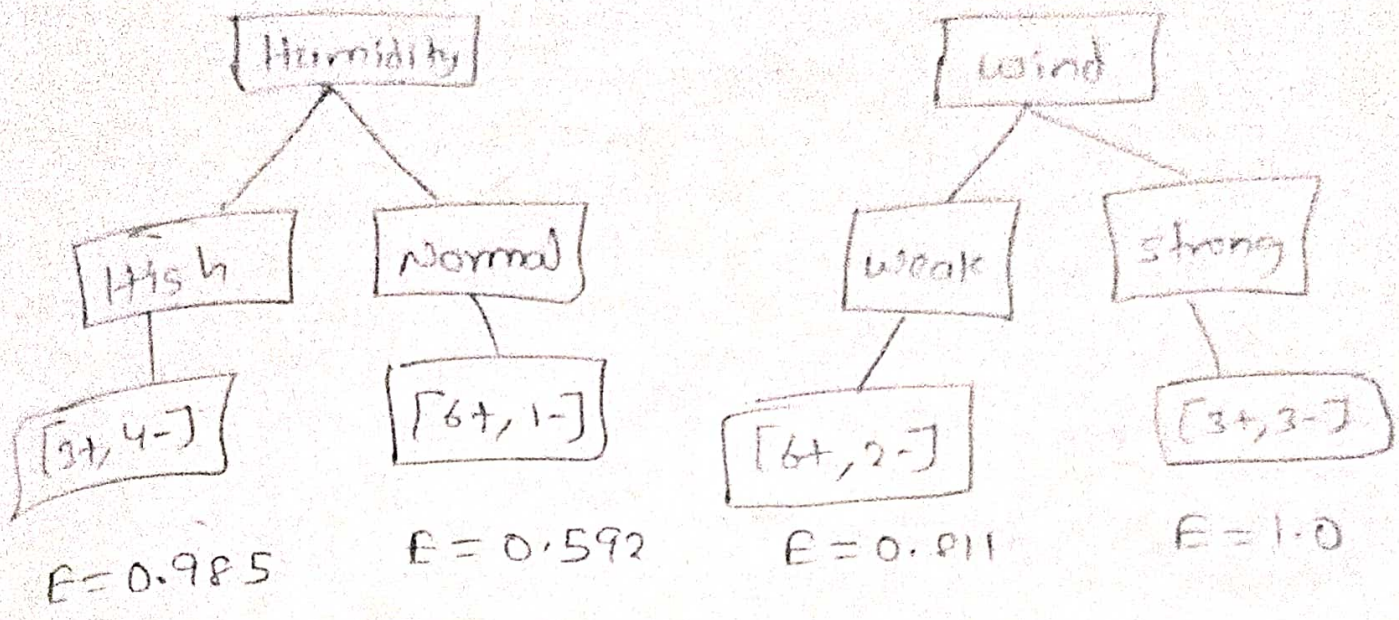
= 0.12

## Training Examples :-

| Day | outlook | Temp | Humidity | Wind | Termin |
|---|---|---|---|---|---|
| $D_1$ | Sunny | Hot | High | weak | No |
| $D_2$ | Sunny | Hot | High | Strong | No |
| D3 | overcast | Hot | High | weak | Yes |
| D4 | Rain | Mild | High | weak | Yes |
| D5 | Rain | cool | Normal | weak | Yes |
| D6 | Rain | cool | Normal | Strons | No |
| D7 | overcast | cool | Normal | strons | Yes |
| D8 | Sunny | Mild | High | weak | No |
| D9 | Sunny | cool | Normal | weak | Yes |
| D10 | Rain | Mild | Normal | weak | Yes |
| D11 | Sunny | Mild | Normal | strong | Yes |
| D12 | overcast | Mild | High | Strons | Yes |
| D13 | overcast | Hot | Normal | weak | Yes |
| D14 | Rain | Mild | High | strons | No |

S = [9+, 5-]
E = 0.940

S = [9+, 5-]
E = 0.940




Gain (S, Humidity)

$= 0.940 - (7/14) * 0.985 - (7/14) * 0.592$

$= 0.151$

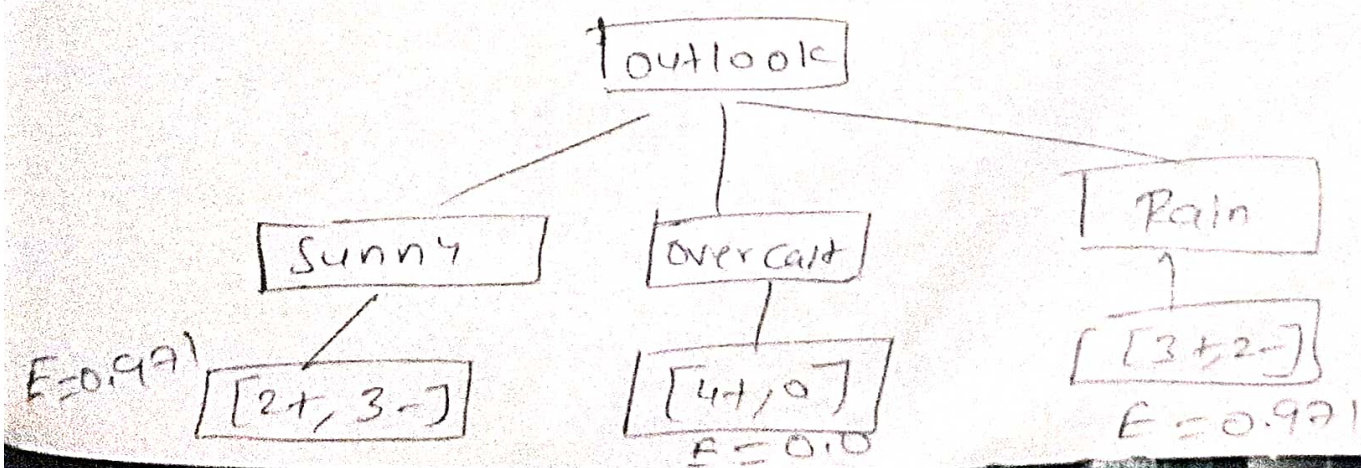
Gain (S, wind)

$= 0.940 - (8/14) * 0.811 - (6/14) * 1.0$

$= 0.048$

Humidity provides grater info. gain than wind, w.r.t target classification

S = [9+, 5-]
E = 0.940



Gain (S, Outlook)

$= (0.940 - (5/14) * 0.971$

$- (4/14) * 0.0 - (5/14) * 0.0971$

$= 0.247$

The information gain values for the 4 attributes are:

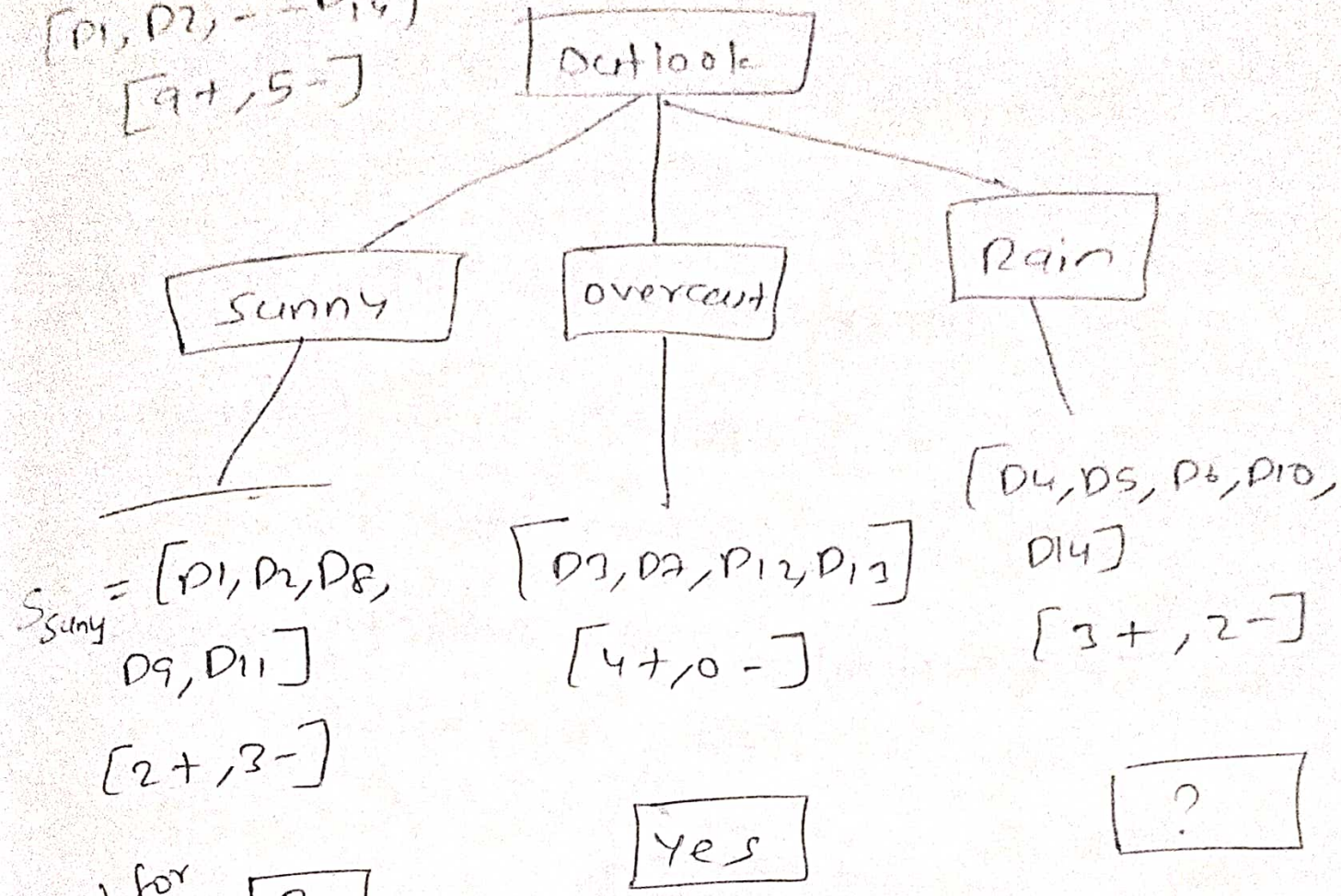
- Gain (S, Outlook) = 0.247.
- Gain (S, Humidity) = 0.151
- Gain (S, Wind) = 0.048
- Gain (S, Temperature) = 0.029

where S denotes collection of training examples.

Note :- $0 \log_2 0 = 0$

# ID3 Algorithm

[D1, D2, ... D14]
[9+, 5−]




$$\begin{cases} Gain(S_{sunny}, humidity) = 0.970 - (\frac{3}{5})0.0 - \frac{2}{5}(0.0) = 0.970 \\ Gain(S_{sunny}, temp) = 0.970 - (2/5)0.0 - 2/5(1.0) \\ \qquad\qquad - (1/5)\,0.0 = 0.570 \\ Gain(S_{sunny}, wind) = 0.970 - (2/5)1.0 - 3/5(0.918) \\ \qquad\qquad = 0.019 \end{cases}$$

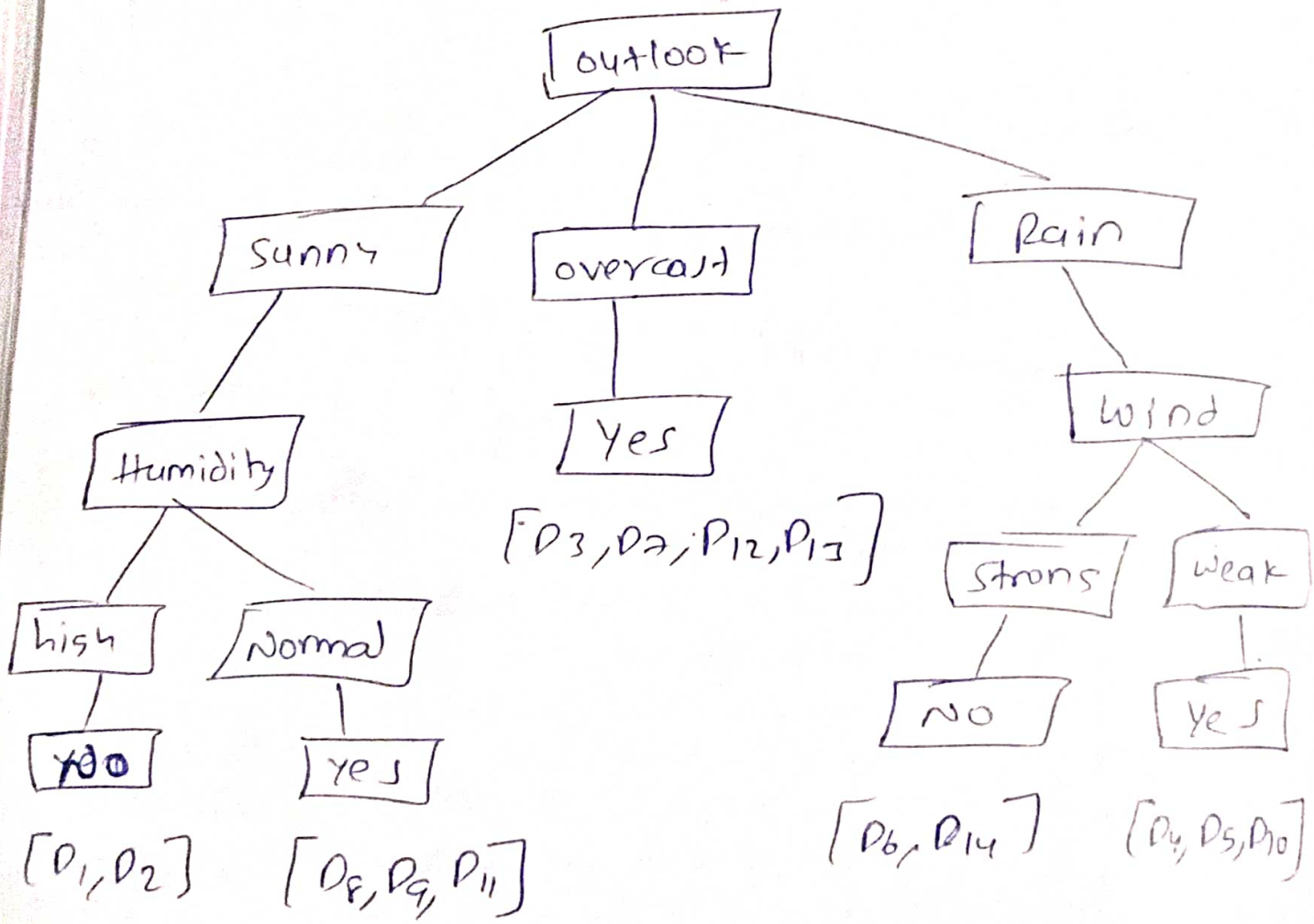
$R_1$: IF (outlook = Sunny) AND (windy = FALSE)
THEN Play = Yes

$R_2$: IF (outlook = sunny) AND (windy = TRUE)
THEN Play = No

$R_3$: IF (outlook = overCast) THEN Play = Yes

$R_4$: IF (outlook = Rainy) AND (Humidity
= High) THEN Play = No.

$R_5$: IF (outlook = Rain) AND (Humidity
= Normal) THEN Play = Yes.



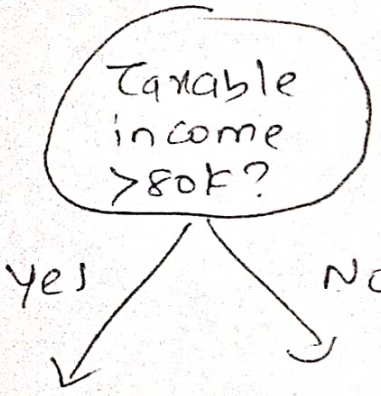
# Splitting Rule: GINI Index

- GINI index
  - Measure of node impurity

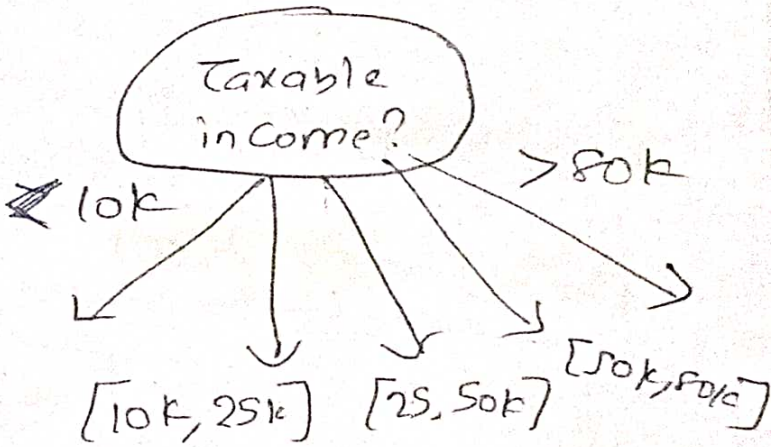
$$GINI_{node}(Node) = 1 - \sum_{c \in class} [p(c)]^2$$

$$GINI_{split}(A) = \sum_{v \in Values(A)} \frac{|S_v|}{|S|} GINI(N_v)$$

Splittins Based on Continuous Attributes




i, Binary split




ii, Multi-way split

## Binary Split

- For continous attribute
  - partition the continuous value of attribute A into a discrete set of intervals. Temperature = 82.5

- Create a new boolean attribute $A_c$ looking for a threshold c,

$$A_c = \begin{cases} \text{true} & \text{if } A_c < c \\ \text{false} & \text{otherwise} \end{cases}$$

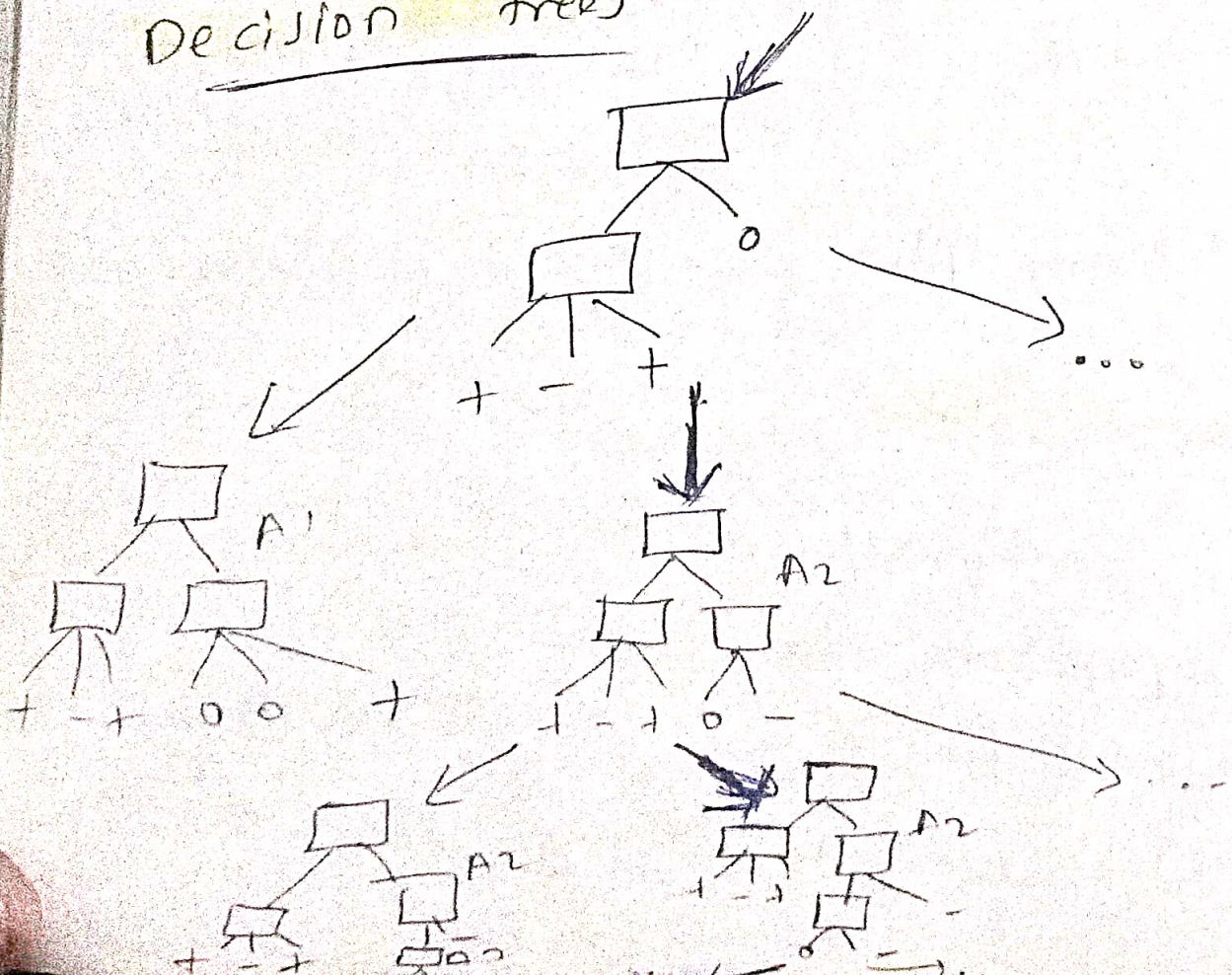
- Consider How to choose c?

find the best cut

| | | | | | | |
|---|---|---|---|---|---|---|
| Temperature: | 40 | 48 | 60 | 72 | 80 | 90 |
| PlayTennis: | No | No | Yes | Yes | Yes | No |

Hypothesis Space Search in

Decision trees



- ID3 Search a complete hypothesis space but does so incompletely since once it finds a good hypothesis it stops (cannot find other).

- Candidate-Elimination Search an incomplete hypothesis space (it can only represent some hypothesis) but does so completely.

- A preference bias is an inductive bias where some hypothesis are preferred over others.

- A restriction bias is an inductive bias where the set of hypotheses considered is restricted to a smaller set.

## Inductive Bias in Decision Tree learning:-

- inductive bias is the set of assumptions that, together with the training data.
- It choose the first acceptable tree it encounters in its simple-to-complex, hill climbing search through the space of possible trees.
- The ID3 search strategy.
  - selects in favor of shorter trees over longer ones.
  - selects trees that place the attributes with highest information gain closet to the root.

## Types of inductive bias

- preference bias or search bias
- restriction bias or language bias.

- The inductive bias of ID3 is thus a preference for certain hypotheses over others, with no hard restriction on the hypotheses that can be eventually enumerated. This form of bias is typically called a preference bias (or alternatively, a search bias).

- The bias of the candidate Elimination algorithm is in the form of a categorical restriction on the set of hypotheses considered. This form of bias is typically called restriction bias. (language bias).

a preference bias is more desirable than a restriction bias.

## Bias and Occam's Razor

**Bias:-** Searches a complete hypothesis space incompletely, inductive bias is solely a consequence of the orderings of hypotheses by its search strategy.

Inductive bias often goes by the name of occams razor

**Occam's Razor:-** prefer shorter hypotheses that fits the data

## Issues in Decision Tree learning:-

- Determining how deeply to grow the decision tree, underfittins and overfittins.
- Handlins continous attributes.
- Choosins an appropriate attribute selection measure.

- Handling training data with missing attribut values.
- Handling attributes with different costs.
- Improving computational efficiency.

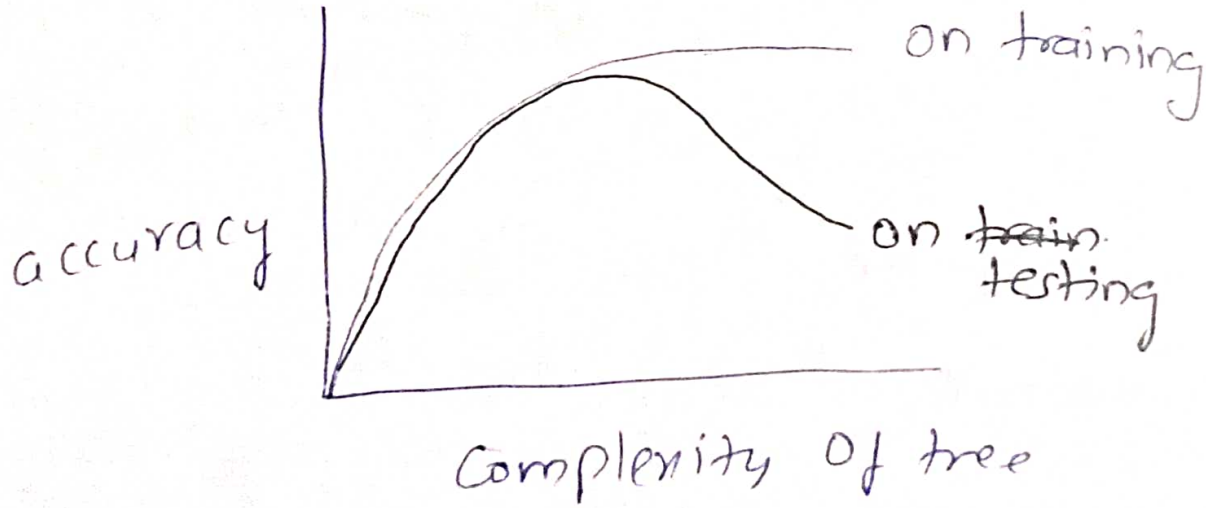
## Overfitting :-

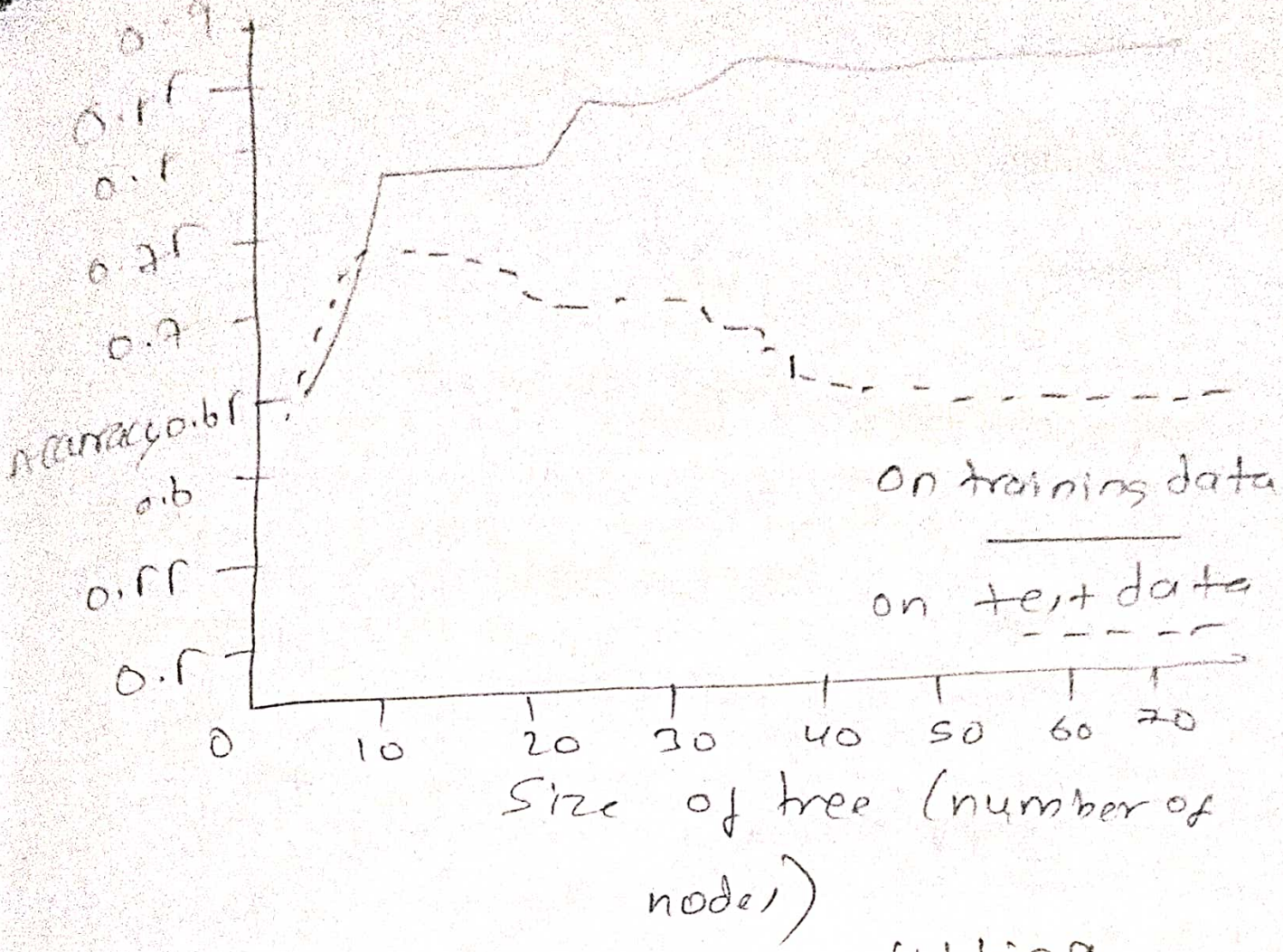
An algorithm can produce trees that overfit the training examples in the following two cases.

- There is noise in the data.
- The number of training examples is too small to produce a representative sample of the true target function.
- Learning a tree that classifies the training data perfectly may not lead to the tree with the best generalization performance.

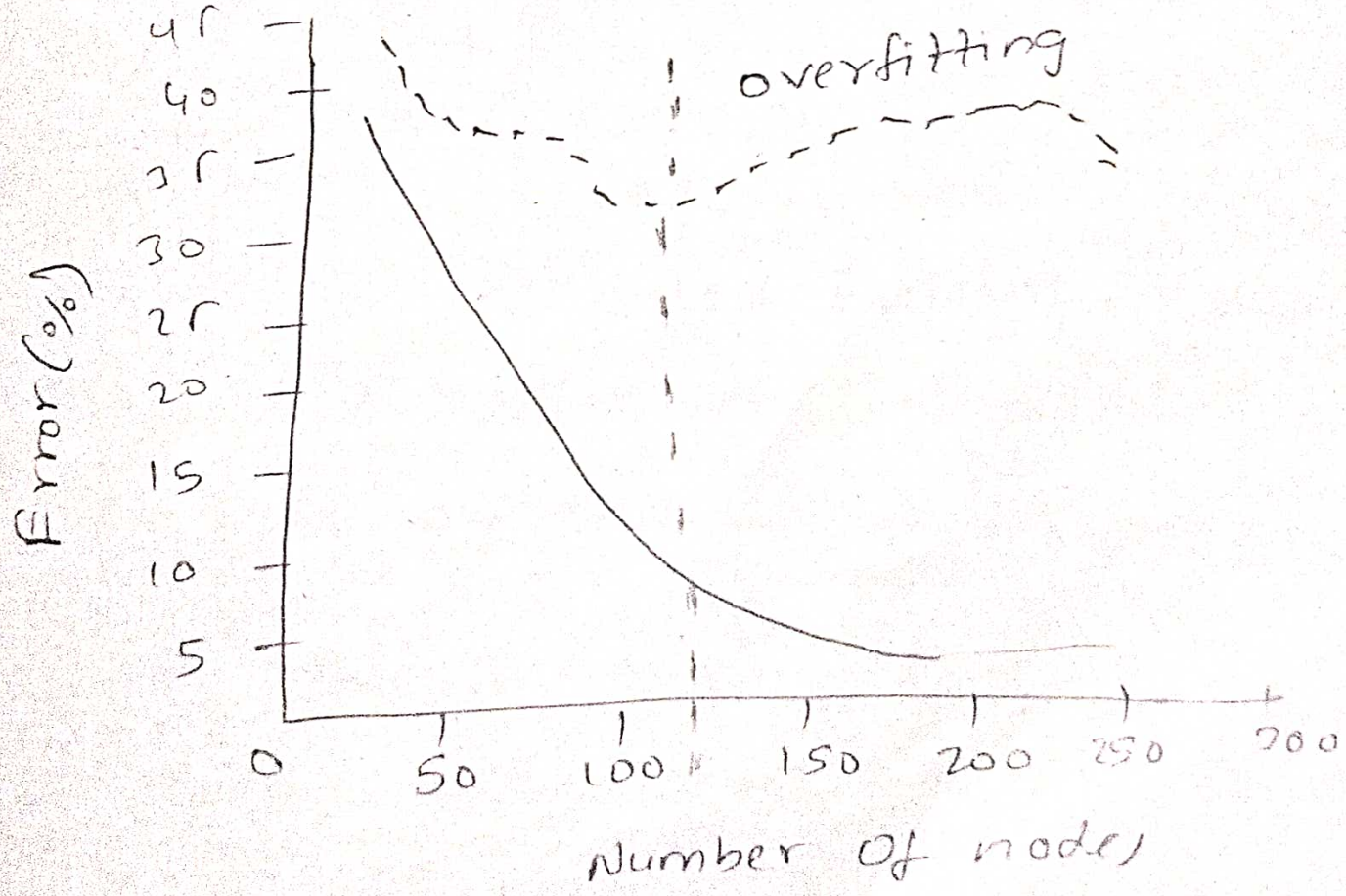
- There may be noise the training data.
- May be based on insufficient data.

- A hypothesis, h is said to overfit the trainins data if there is another hypothesis, h, such that h has smaller error than h' on the trainins data but h has larger error on the test data than h.



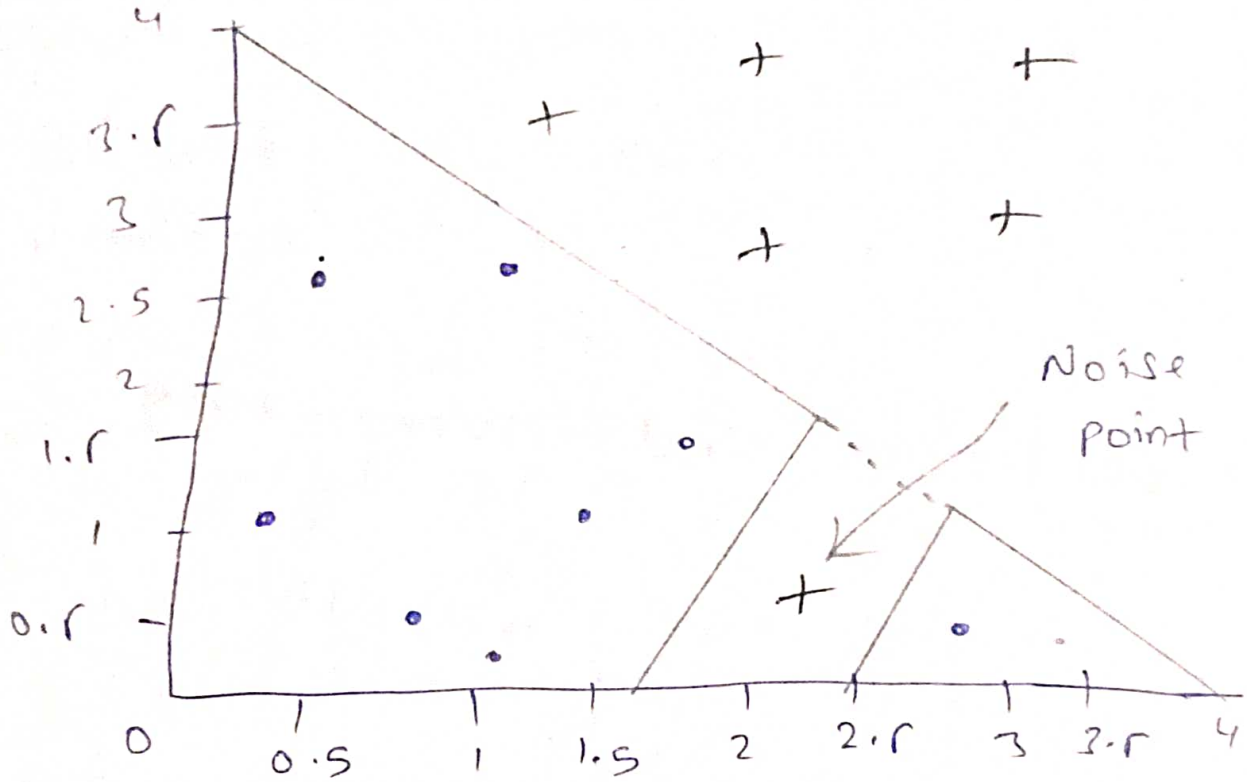



## Underfitting and overfitting



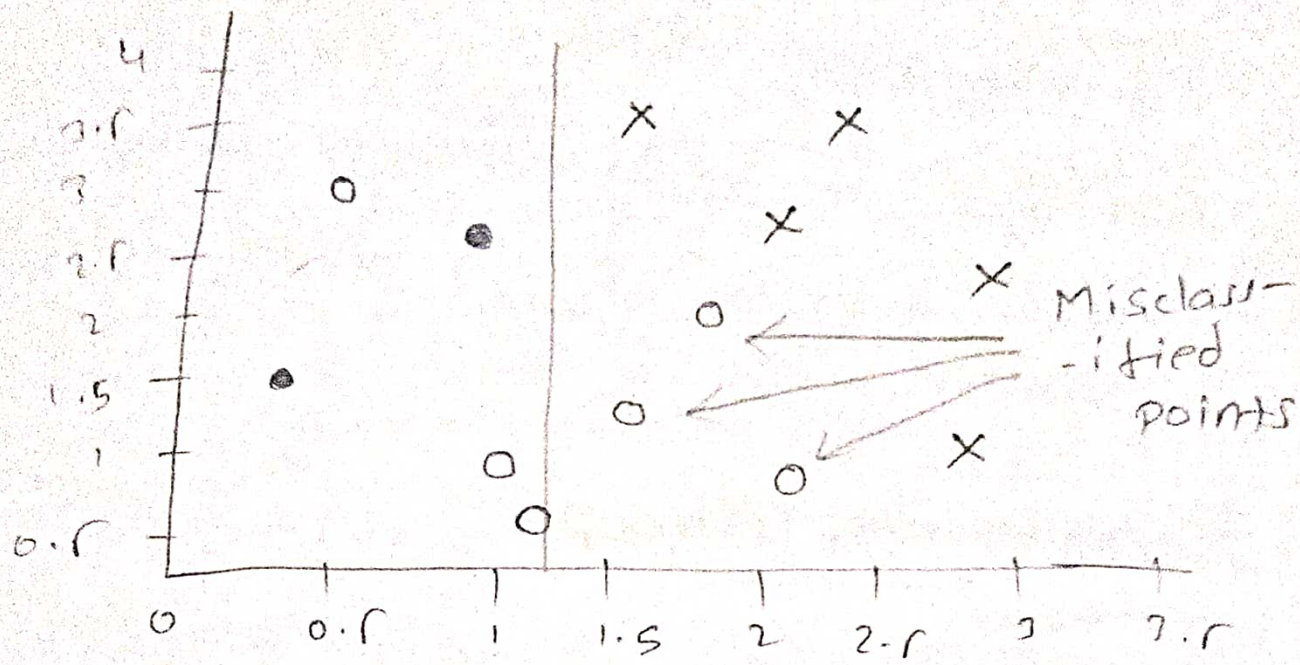
Underfitting :- when model is too simple, both trainins and test errors are large.

Overfitting due to noise :-




Decision boundary is distorted by noise point

## overfitting due to insufficient Examples :-




lack of data points make it difficult to predict correctly the class labels of the region.

* Overfitting results in decision trees that are more complex than necessary
* Training error no longer provides a good estimate of how well the tree will perform on previously unseen records.
* overfitting happens when a model is capturing idiosyncrasies of the data rather than generalities.

* often caused by too many parameters relative to the amount of training data.

* E.g. - an order-N polynomial can intersect any N+1 data points.

## Avoid overfitting:-

* There are several approaches to avoiding overfitting in decision tree learning. These can be grouped into two classes.

- prepruning :- stop growing when data split not statistically significant
- postpurning :- Grow full tree then remove nodes

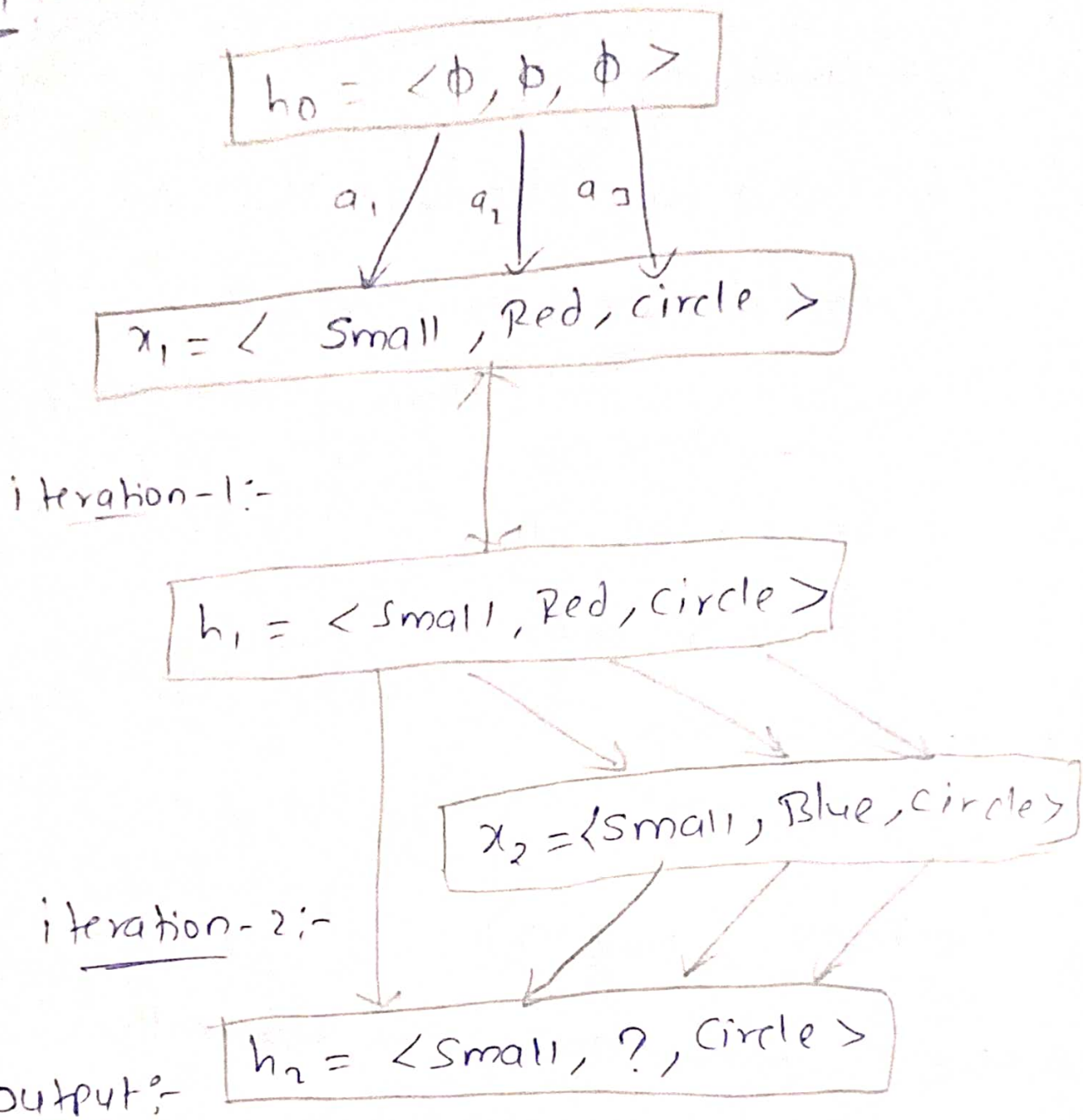
- methods for evaluating subtree to prune:-
  - Minimum description length(MDL)
    minimize: size(tree) + size (misclassifications(tree))
  - measure performance over training data.
- measure performance over separate validation data set

## Questions:- (Unit-I)

1) Find-S algorithm and find maximally specific hypothesis for the given training examples.

| size | color | shape | class |
|---|---|---|---|
| Big | Red | circle | No |
| Small | Red | triangle | No |
| Small | Red | circle | Yes |
| Big | Blue | circle | No |
| Small | Blue | Circle | Yes |

Sol:-




output:-

Note:- When there is No in logical and output values then no need to consider that Just check for only which has 'Yes'.

4) Find-S Algorithm

| example | citations | size | in library | price | editions | buy |
|---|---|---|---|---|---|---|
| 1 | some | small | no | affordable | many | no |
| 2 | many | big | no | expensive | one | yes |
| 3 | some | big | always | expensive | few | no |
| 4 | many | midum | no | expensive | many | yes |
| 5 | many | small | no | affordable | many | yes |

a) How many concepts are possible for the instance space?

b) How many hypotheses can be expressed by the hypothesis language

c) Apply the Find-S algorithm by hand on the given training set. Consider the examples in the specified order and write down your hypothesis each time after observing an example.

a) → 2 * 3 * 2 * 2 * 3

→ 72 concepts.

b) by adding null and ? in each attribute then the table becomes

→ 4 * 5 * 4 * 4 * 5

→ 1600

Symentically distint Hypothesis

not consider null. only ?

$= (3 * 4 * 3 * 3 * 4) + 1 = 433$

↑ all nulls as 1

c) step-1 :-

$h_0 = (\phi, \phi, \phi, \phi, \phi)$

step-2 :-

$x_1 = (some, small, no, expensive, many)$

-NO

Negative Example Hence ignore

$h_1 = (\phi, \phi, \phi, \phi, \phi)$

$h_1 = (\phi, \phi, \phi, \phi, \phi)$

$x_2 = (many, bis, no, expensive, one)$ - Yes

$h_2 = (many, bis, no, expensive, one)$

$x_3 = (some, bis, always, expensive, few)$
- No

Negative example hence ignore

$h_3 = (many, big, no, expensive, one)$

$x_4 = (many, medium, no, expensive, many)$
- Yes

$h_4 = (many, ?, no, expensive, ?)$

$x_5 = (many, small, no, affordable, many)$
- Yes

$h_5 = (many, ?, no, ?, ?)$

Final hypothesis

Φ)

| Ex | Eyes | nose | head | Color |
|---|---|---|---|---|
| 1 | Round | Triangle | Round | Purple |
| 2 | Square | Square | Square | Green |
| 3 | Square | Triangle | Round | Yellow |
| 4 | Round | Triangle | Round | Green |
| 5 | Square | Square | Round | Yellow |

| Hair | Smile |
|---|---|
| yes | yes |
| yes | No |
| yes | yes |
| No | No |
| yes | yes |

Do it yourself

Φ) Candidate algorithm

| Ex | Sky | AirTemp | Humidity | wind | water |
|---|---|---|---|---|---|
| 1 | Sunny | warm | normal | strong | warm |
| 2 | Sunny | warm | High | Strong | warm |
| 3 | Rainy | cold | High | strong | warm |
| 4 | Sunny | warm | High | strong | cool |

| Forecast | Enjoy Sport |
|---|---|
| Same | Yes |
| Same | Yes |
| Change | No |
| Change | Yes |

$S_0$: (ϕ, ϕ, ϕ, ϕ, ϕ, ϕ)

$S_1$: (sunny, warm, Normal, Strong, warm, same)

$S_2$ : $S_3$: (sunny, warm, ?, Strong, warm, same)

→ $S_4$: (sunny, warm, ?, Strong, ?, ?)

$G_4$: (sunny, ?, ?, ?, ?, ?) (?, warm, ?, ?, ?, ?)

(?, ?, ?, ?, ?, same)

(?, ?, ?, ?, Cool, ?)

$G_3$: (sunny, ?, ?, ?, ?, ?) (?, warm, ?, ?, ?, ?) (?, ?, Norm, ?, ?, ?)

$G_0$: $G_1$: $G_2$: (?, ?, ?, ?, ?, ?, ?)

---

S (sunny, warm, ?, Strong, ?, ?)

(sunny, ?, ?, Strong, ?, ?) (sunny, warm, ?, ?, ?, ?)

(?, warm, ?, Strong, ?, ?)

G (sunny, ?, ?, ?, ?, ?) (?, warm, ?, ?, ?, ?)

Q) Candidate Elimination Algs

| Size | Color | shape | label |
|---|---|---|---|
| Big | Red | circle | No |
| Big | Red | Triangle | No |
| Small | Red | circle | Yes |
| Small | Red | circle | ~~Yes~~ No |
| Big | Blue | circle | Yes |
| Small | Blue | circle | Yes |

Sol - $S_0$: ($\phi, \phi, \phi$)

$S_1$ : ($\phi, \phi, \phi$)

$S_2$ : ($\emptyset, \emptyset, \emptyset$)

$S_3$ : (Small, Red, circle)

$S_4$ : (Small, Red, circle)

$G_4$ : (Small, ?, circle)

$G_3$ : (Small, ?, circle)

(Big, ?, Triangle) (?, Blue, Triangle)

$G_2$ : (Small, Blue, ?) (Small, ?, circle) (?, Blue, ?)

$G_1$ : (Small, ?, ?) (?, Blue, ?) (?, ?, Triangle)

$G_0$ : (?, ?, ?)

S5 : (small, ?, circle)

G5 : (small, ?, circle)

S : G : (small, ?, circle)

It is perfectly learned ~~ally~~

φ)

| Example | citations | size | inLibrary |
|---|---|---|---|
| 1 | some | small | No |
| 2 | many | Big | No |
| 3 | many | medium | No |
| 4 | many | small | No |
| 5 | | | |
| 6 | | | |

| price | Editions | Buy |
|---|---|---|
| Affordable | one | No |
| ~~Affordable~~ | many | yes |
| Expensive | few | Yes |
| Expensive | many | Yes |
| Affordable | | |

Q)

| Ex | Eyes | Nose | Head | Fcolor | Hair |
|---|---|---|---|---|---|
| 1 | Round | Triangle | Round | purply | Yes |
| 2 | Square | Square | Square | Green | Yes |
| 3 | Square | Triangle | Round | yellow | Yes |
| 4 | Round | Triangle | Round | Green | No |
| 5 | Square | Square | Round | yellow | Yes |

Smile

Yes

No

Yes

No

Yes.

Q) Decision tree for Boolean functions.

a) $A \wedge \neg B$

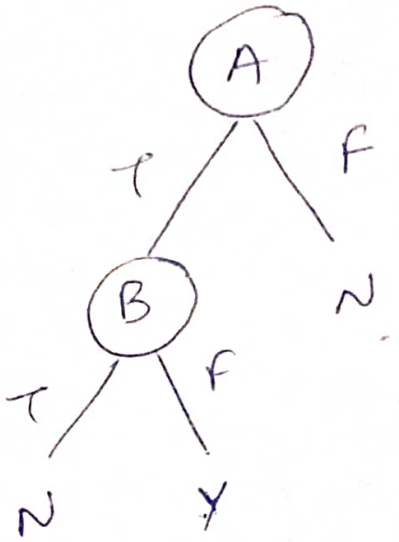
b) $A \vee [B \wedge C]$

c) A XOR B

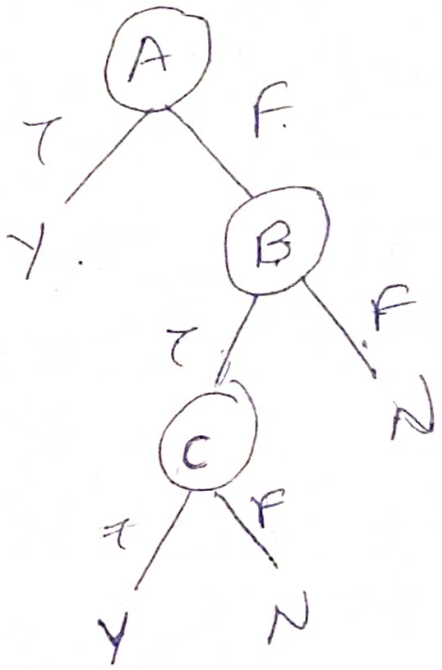
d) $[A \wedge B] \vee [C \wedge D]$

a) $A \wedge \neg B$

N → No
Y → Yes

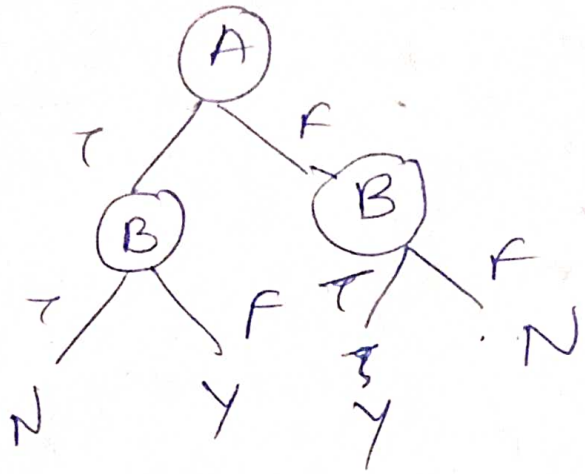



b) $A \vee [B \wedge C]$

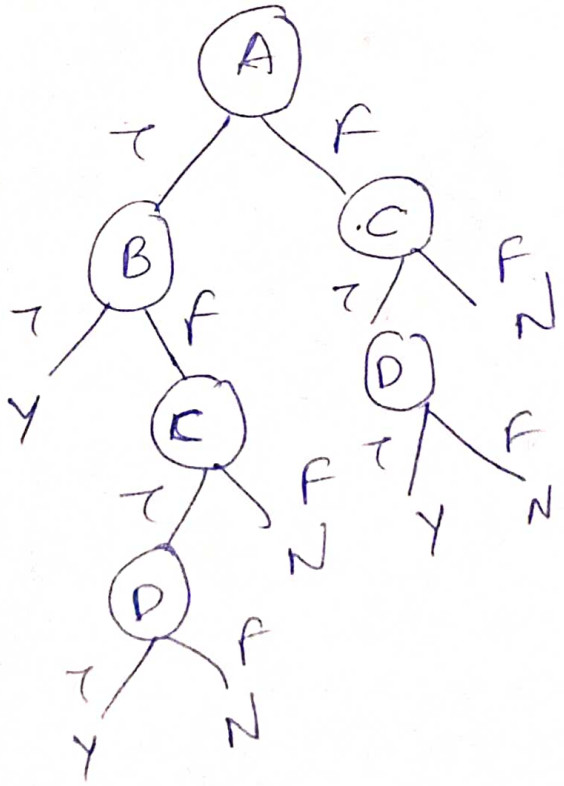


c) A XOR B

$(A \wedge \neg B) \vee (\neg A \wedge B)$




d) $[A \wedge B] \vee [C \wedge D]$



Q) Calculate Entropy and Information Gain

| Day | outlook | Temp | Humidity | wind | Play tennis |
|---|---|---|---|---|---|
| D1 | sunny | Hot | High | weak | No |
| D2 | sunny | Hot | High | strong | No |
| D3 | overcast | Hot | High | weak | Yes |
| D4 | Rain | mild | High | weak | Yes |
| D5 | Rain | cool | Normal | weak | Yes |
| D6 | Rain | cool | Normal | Strong | No |
| D7 | overcast | cool | Normal | strong | Yes |
| D8 | sunny | mild | High | weak | No |
| D9 | sunny | cool | Normal | weak | Yes |
| D10 | Rain | mild | Normal | weak | Yes |
| D11 | sunny | mild | Normal | strong | Yes |
| D12 | overcast | mild | High | strong | Yes |
| D13 | Overcast | Hot | Normal | weak | Yes |
| D14 | Rain | mild | High | strong | No |

Entropy :-

$$\text{Entropy}(S) \equiv -P_{\oplus} \log_2 P_{\oplus} - P_{\ominus} \log_2 P_{\ominus}$$

* if we have only positive examples and only negative examples then it is "0"

Ex :-

$$\text{Entropy}([14+, 0-]) = 0$$

* if there is equal no of positive and negative then "1"

$$\text{Entropy}([7+, 7-]) = 1$$

Information gain

$$\text{Gain}(S, A) \equiv \text{Entropy}(S) - \sum_{v \in \text{values}(A)} \frac{|S_v|}{|S|} \text{Entropy}(S_v)$$

Now for the table we find for wind

values (wind) = weak, strong

$S = [9+, 5-]$

$S_{weak} \leftarrow [6+, 2-]$

$S_{strong} \leftarrow [3+, 3-]$

$$Gain(S, wind) = Entropy(S) - \sum_{v \in (weak, strong)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$= Entropy(S) - \left(\frac{8}{14}\right) Entropy(S_{weak}) - \left(\frac{6}{14}\right) Entropy(S_{strong})$$

$$= 0.940 - \left(\frac{8}{14}\right) 0.811 - \left(\frac{6}{14}\right) 1.00$$

$$= 0.048$$

$Entropy([9+, 5-])$

$$= -\frac{9}{14} \log_2\left(\frac{9}{14}\right) - \frac{5}{14} \log_2\left(\frac{5}{14}\right)$$

$$= 0.94$$

Q) Given

$P_1 = 0.1, P_2 = 0.2, P_3 = 0.3, P_4 = 0.4$

Entropy ?

Sol:-

$$\text{Entropy} = -\sum_{i=1}^{n} P_i \log_2(P_i)$$

$$= -P_1 * \log_2(P_1) - P_2 * \log_2(P_2) - P_3 * \log_2(P_3) - P_4 * \log_2(P_4)$$

$$= -0.1 * \log_2(0.1) - 0.2 * \log_2(0.2) - 0.3 * \log_2(0.3) - 0.4 * \log_2(0.4)$$

$$= -0.1 * (-3.322) - 0.2 * (-2.322) - 0.3 * (-1.736) - 0.4 * (-1.322)$$

$$= 0.3322 + 0.4644 + 0.5208 + 0.5288$$

$$= 1.8462$$

$\log(0.1)$ / $\log(2)$

q) Findins Gini Index for Splittins Continuous Attribute

| Annual income | Label | split point |
|---|---|---|
| 60 | No | |
| 70 | No | |
| 75 | No | <80 / >=80 |
| 85 | Yes | |
| 90 | Yes | |
| 95 | Yes | <97.5 / >=97.5 |
| 100 | No | |
| 120 | No | |
| 125 | No | |
| 220 | No | |

if not in order arrange in order

| Annual income | Label | split point | Yes | No | Gini |
|---|---|---|---|---|---|
| 60 | No | | | | |
| 70 | No | | | | |
| 75 | No | <80 | 0 | 3 | 0.3427 |
| | | >=80 | 3 | 4 | |
| 85 | Yes | | | | |
| 90 | Yes | | | | |
| 95 | Yes | <97.5 | 3 | 3 | 0.30 |
| | | >=97.5 | 0 | 4 | |
| 100 | No | | | | |
| 120 | No | | | | |
| 125 | No | | | | |
| 220 | No | | | | |

- split point = 50
- $Gini(<50) = 1 - \sum_{i=1}^{c} (p_i)^2$

$= 1 - \left(\frac{0}{3}\right)^2 - \left(\frac{3}{3}\right)^2 = 0.0$

- $Gini(\geq 50) = 1 - \sum_{i=1}^{c} (p_i)^2$

$= 1 - \left(\frac{3}{7}\right)^2 - \left(\frac{4}{7}\right)^2 = 0.4897$

- Gini (50)

$= w_1 * Gini(<50) + w_2 * Gini(\geq 50)$

$= \frac{3}{10} * 0.0 + \frac{7}{10} * 0.4897$

$= 0.3427$

- split point = 97.5
- $Gini(<97.5) = 1 - \sum_{i=1}^{c} (p_i)^2$

$= 1 - \left(\frac{3}{6}\right)^2 - \left(\frac{3}{6}\right)^2 = 0.5$

- $Gini(\geq 97.5) = 1 - \left(\frac{0}{4}\right)^2 - \left(\frac{4}{4}\right)^2$

$= 0.0$

- $Gini(97.5) = w_1 * Gini(< 97.5) + w_2 * Gini(\geq 97.5)$

$= \left(\frac{6}{10}\right) * 0.5 + \frac{4}{10} * 0.0$

$= 0.30$

So for this continous splitting Attribute 97.5 is best point

Q) • Entropy
• Information Gain
• Gini Index
• Splitting Attribute

| Instance | $a_1$ | $a_2$ | $a_3$ | Target class |
|---|---|---|---|---|
| 1 | T | T | 1.0 | + |
| 2 | T | T | 6.0 | + |
| 3 | T | F | 5.0 | - |
| 4 | F | F | 4.0 | + |
| 5 | F | T | 7.0 | - |
| 6 | F | T | 3.0 | - |
| 7 | F | F | 8.0 | - |
| 8 | T | F | 7.0 | + |
| 9 | F | T | 5.0 | - |

(1) $\text{Entropy} = -\sum_{i=1}^{n} p_i (\log_2(p_i))$

$\text{Entropy}(S) = -\frac{4}{9}\log_2\left(\frac{4}{9}\right) - \frac{5}{9}\log_2\left(\frac{5}{9}\right)$

$= 0.9911$

① information gain of $a_1$

$$Entropy(S) = \sum_{i=1}^{n} p_i \log_2(p_i)$$

$$Entropy(S_T) = -\frac{3}{4}\log_2\left(\frac{3}{4}\right) - \frac{1}{4}\log_2\left(\frac{1}{4}\right)$$

$$= 0.311 + 0.5 = 0.811$$

$$Entropy(S_F) = -\frac{1}{5}\log_2\left(\frac{1}{5}\right) - \frac{4}{5}\log_2\left(\frac{4}{5}\right)$$

$$= 0.4644 + 0.2576$$

$$= 0.722$$

4 → 3

5 → ~~5~~

$$Gain(a_1) = Entropy(S) = \sum_{v \in (T,F)} \frac{|S_v|}{|S|} Entropy(S_v)$$

$$Gain(a_1) = Entropy(S) = \frac{4}{9} Entropy(S_T) - \frac{5}{9} Entropy(S_F)$$

$$Gain(a_1) = 0.2295$$

for information gain for $a_2$

$$\text{Entropy}(S_T) = -\frac{2}{5}\log_2\left(\frac{2}{5}\right) - \frac{3}{5}\log_2\left(\frac{3}{5}\right)$$

$$= 0.9709$$

$$\text{Entropy}(S_F) = -\frac{2}{4}\log_2\left(\frac{2}{4}\right) - \frac{2}{4}\log_2\left(\frac{2}{4}\right)$$

$$= 0.5 + 0.5 = 1.0$$

$$\text{Gain}(a_2) = \text{Entropy}(S) - \frac{5}{9}\text{Entropy}(S_T) - \frac{4}{9}\text{Entropy}(S_F)$$

$$= 0.0072$$

③ Gini index of attribute $a_1$

$$\text{Gini} = 1 - \sum_{i=1}^{n} (p_i)^2$$

$$\text{Gini}(T) = 1 - \left(\frac{3}{4}\right)^2 - \left(\frac{1}{4}\right)^2 = 0.375$$

$$\text{Gini}(F) = 1 - \left(\frac{1}{4}\right)^2 - \left(\frac{3}{4}\right)^2 = 0.375$$

$$\text{Gini index}(a_1) = \sum_{v \in \{T,F\}} \frac{|S_v|}{|S|} \text{Gini}(S_v)$$

$$= \left(\frac{4}{9}\right) * \text{Gini}(T) + \left(\frac{5}{9}\right) * \text{Gini}(F)$$

$$= 0.3444$$

for similarly for Gini index of $a_2$ also

(4) best splitting attribute between $a_1$ and $a_2$

$\text{Gain}(a_1) = 0.2295$

$\text{Gain}(a_2) = 0.0072$

should be high

Here $a_1$ is the best split attribute

Now

$\text{Gini index}(a_1) = 0.3444$

$\text{Gini index}(a_2) = 0.4889$

should be low

Here $a_1$ is the best split attribute