DECISION TREE INDUCTION

Team 1 - CS 3203N

Dayata, Wayne Matthew
Izumi, Sana
Monzales, Kathleen Iza
Tejada, Jay
Woogue, Ivan Ric

April 17, 2023

TOPIC OUTLINE

1. Concept of Decision Tree
2. Basic algorithm for building Decision Tree
3. Concept of Entropy
4. Decision Tree induction algorithms (ID3, CART, C4.5)

1. Concept of Decision Tree

Basic Concept

- A Decision Tree is an important data structure known to solve many computational problems
  - Specifically on classification of data

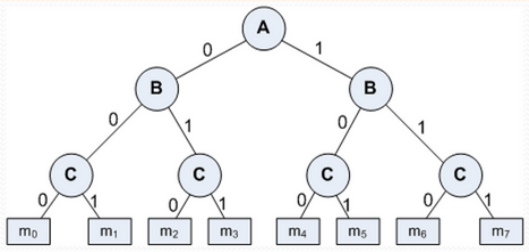
- Attributes of a Decision Tree can be of:
  - Discrete type (binary/n-ary)
  - Continuous type

- Such a classification is, in fact, made by posing questions starting from the root node down to each terminal node.

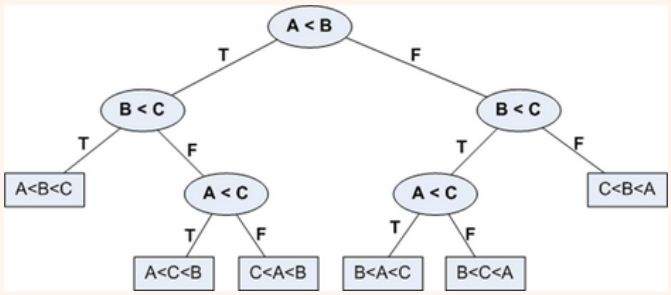
Basic Concept

Binary decision tree

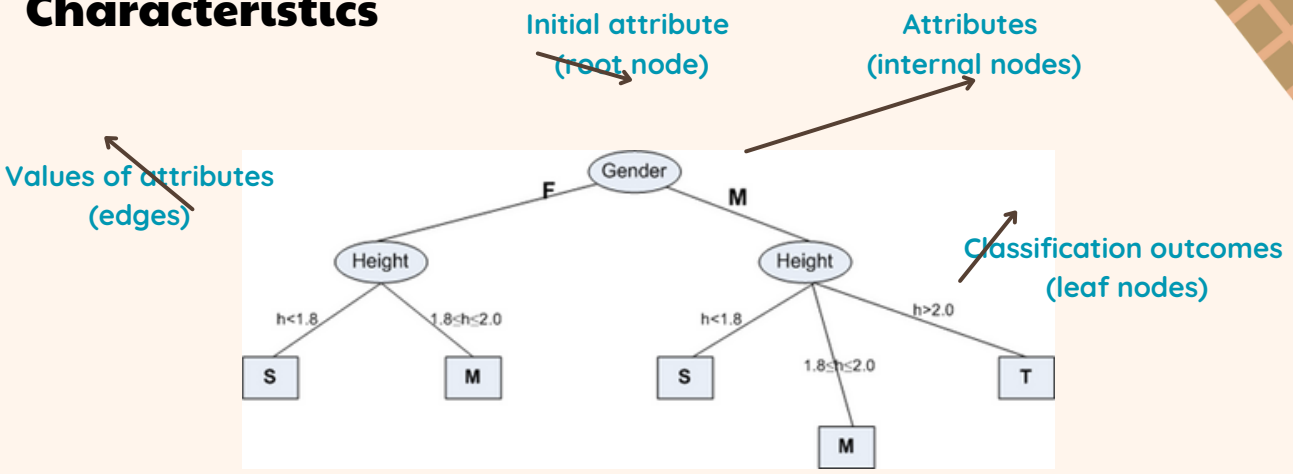
A	B	C	f
0	0	0	m_0
0	0	1	m_1
0	1	0	m_2
0	1	1	m_3
1	0	0	m_4
1	0	1	m_5
1	1	0	m_6
1	1	1	m_7



Decision tree with numeric data



Characteristics



- Decision tree **may be n-ary**, n ≥ 2.
- In a path, a node with same label is never repeated.
- Decision tree is **not unique**, as different ordering of internal nodes can give different decision tree.

Example: Classifying Vertebrates

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class
Human	Warm	hair	yes	no	no	yes	no	Mammal
Python	Cold	scales	no	no	no	no	yes	Reptile
Salmon	Cold	scales	no	yes	no	no	no	Fish
Whale	Warm	hair	yes	yes	no	no	no	Mammal
Frog	Cold	none	no	semi	no	yes	yes	Amphibian
Komodo	Cold	scales	no	no	no	yes	no	Reptile
Bat	Warm	hair	yes	no	yes	yes	yes	Mammal
Pigeon	Warm	feathers	no	no	yes	yes	no	Bird
Cat	Warm	fur	yes	no	no	yes	no	Mammal
Leopard	Cold	scales	yes	yes	no	no	no	Fish
Turtle	Cold	scales	no	semi	no	yes	no	Reptile
Penguin	Warm	feathers	no	semi	no	yes	no	Bird
Porcupine	Warm	quills	yes	no	no	yes	yes	Mammal
Eel	Cold	scales	no	yes	no	no	no	Fish
Salamander	Cold	none	no	semi	no	yes	yes	Amphibian

- What are the class labels of Dragon and Shark?

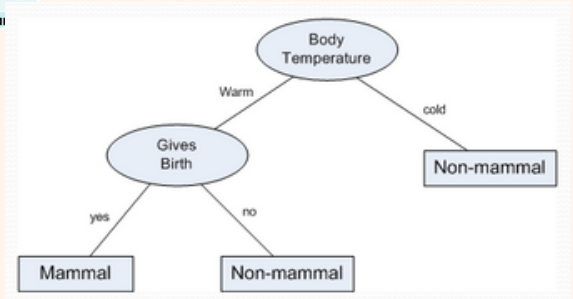
Example: Classifying Vertebrates

Name	Body Temperature	Skin Cover	Gives Birth	Aquatic Creature	Aerial Creature	Has Legs	Hibernates	Class
Human	Warm	hair	yes	no	no	yes	no	Mammal
Python	Cold	scales	no	no	no	no	yes	Reptile
Salmon	Cold	scales	no	yes	no	no	no	Fish
Whale	Warm	hair	yes	yes	no	no	no	Mammal
Frog	Cold	none	no	semi	no	yes	yes	Amphibian
Komodo	Cold	scales	no	no	no	yes	no	Reptile
Bat	Warm	hair	yes	no	yes	yes	yes	Mammal
Pigeon	Warm	feathers	no	no	yes	yes	no	Bird
Cat	Warm	fur	yes	no	no	yes	no	Mammal
Leopard	Cold	scales	yes	yes	no	no	no	Fish
Turtle	Cold	scales	no	semi	no	yes	no	Reptile
Penguin	Warm	feathers	no	semi	no	yes	no	Bird
Porcupine	Warm	quills	yes	no	no	yes	yes	Mammal
Eel	Cold	scales	no	yes	no	no	no	Fish
Salamander	Cold	none	no	semi	no	yes	yes	Amphibian

<--- Dataset

Size: 15
Attributes: 7
Classes: 5

Possible Decision Tree Induction --->



Decision Tree and Classification Task

- The series of questions and their answers can be organized in the form of a decision tree as a hierarchical structure consisting of nodes and edges
- Each time we receive an answer, a follow-up question is asked until we reach a conclusion about the class-label of the test.
- Once a decision tree is built, it is applied to any test to classify it.

Review: What form of learning is applied in Decision Trees?

Why are Decision Trees popular?

- No domain knowledge nor parameter setting needed, therefore appropriate for **exploratory knowledge discovery**.
- Decision trees can handle **multidimensional data**. Their representation of acquired knowledge in tree form is intuitive and generally easy to assimilate by humans.
- The learning and classification steps of decision tree induction are **simple and fast**. In general, decision tree classifiers have **good accuracy**. However, successful use may depend on the data at hand.
- Decision tree induction algorithms have been used for classification in many application areas such as medicine, manufacturing and production, financial analysis, astronomy, and molecular biology, making decision trees the basis of several commercial rule induction systems.

Source: Data Mining: Concepts and Techniques, (3rd Edn.), Jiawei Han, MichelineKamber, Morgan Kaufmann, 2015. (p.335)

Decision Tree Definition

Definition 9.1: Decision Tree

Given a database $D = \{t_1, t_2, \ldots\ldots, t_n\}$, where t_i denotes a tuple, which is defined by a set of attribute $A = \{A_1, A_2, \ldots\ldots, A_m\}$. Also, given a set of classes $C = \{c_1, c_2, \ldots\ldots, c_k\}$.

A decision tree ***T*** is a tree associated with D that has the following properties:

- Each internal node is labeled with an attribute A_i
- Each edges is labeled with predicate that can be applied to the attribute associated with the parent node of it
- Each leaf node is labeled with class c_j

2. Building Decision Trees

Building Decision Trees

Exponentially many decision tree can be constructed from a given database (also called training data).

- Some may not be optimum
- Some may give inaccurate result

Two approaches:

- **Greedy strategy** - A top-down recursive divide-and-conquer
- **Modification of greedy strategy** - ID3, C4.5, CART, etc.

In principle

Iterative Dichotomiser 3
Classification And Regression Trees

Node Splitting

For Each Attribute Type

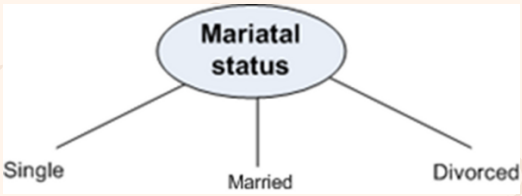
1. attribute test condition
2. corresponding outcome

Binary Attribute

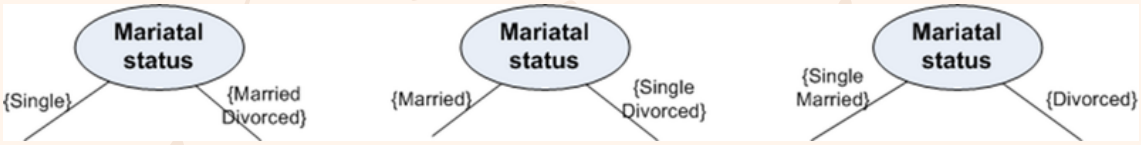
- simplest case
- test condition generates only two outcomes

Nominal Attribute

Multi-way split



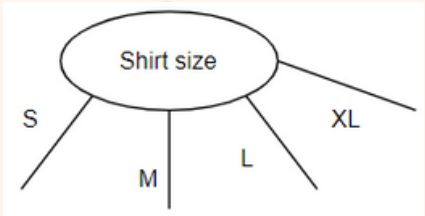
Binary splitting



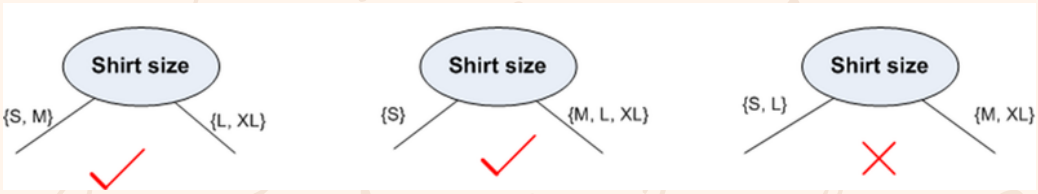
test condition can be expressed

Ordinal Attribute

Multi-way split



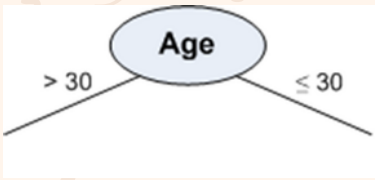
Binary splitting



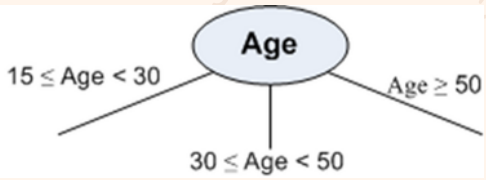
Numerical Attribute

- Discrete or continuous values
- Can be expressed as a comparison set

Binary outcome



Range query



must consider all possible split positions

Illustration

Person	Gender	Height	Class
1	F	1.6	S
2	M	2.0	M
3	F	1.9	M
4	F	1.88	M
5	F	1.7	S
6	M	1.85	M
7	F	1.6	S
8	M	1.7	S
9	M	2.2	T
10	M	2.1	T
11	F	1.8	M
12	M	1.95	M
13	F	1.9	M
14	F	1.8	M
15	F	1.75	S

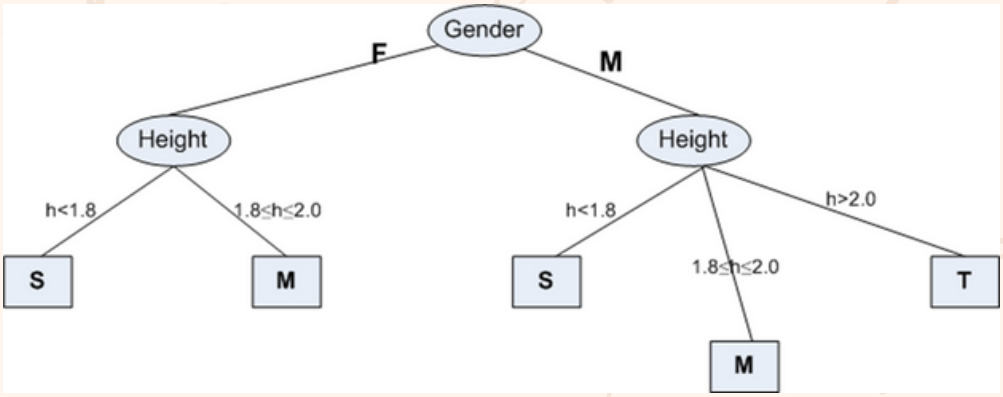
Attributes

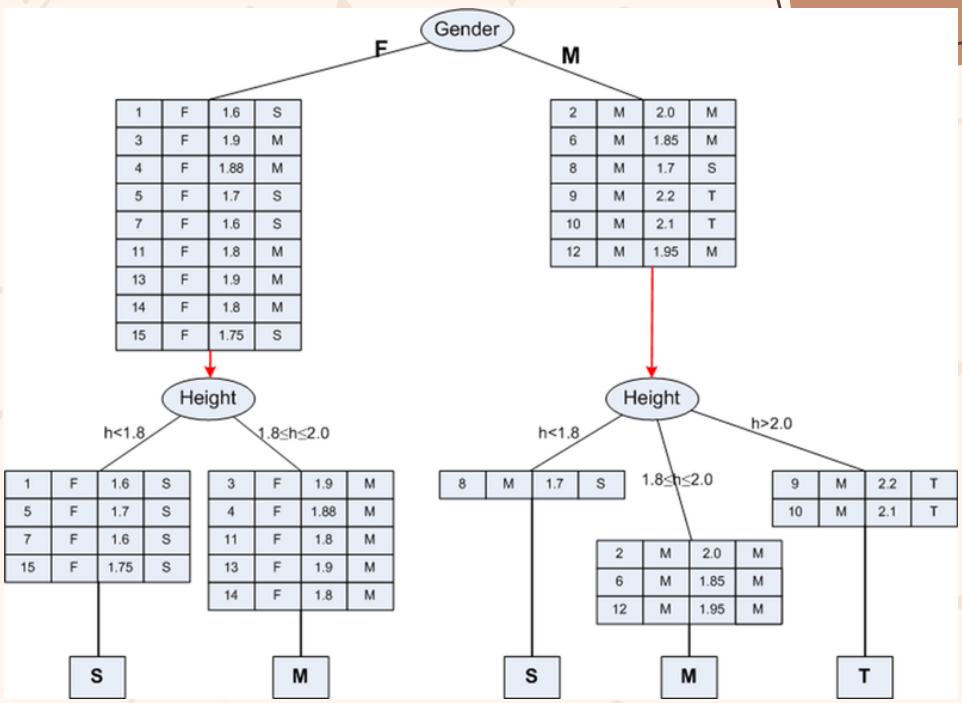
1. **Gender - M or F (binary attribute)**
2. **Height - 1.5 to 2.5 (continuous attribute)**

Class - S, M, T (short, medium, tall)

Different Orderings

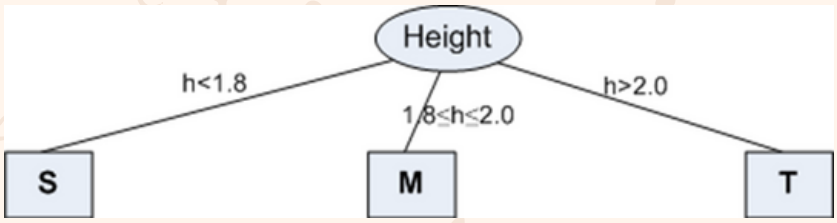
Approach 1: <Gender, Height>

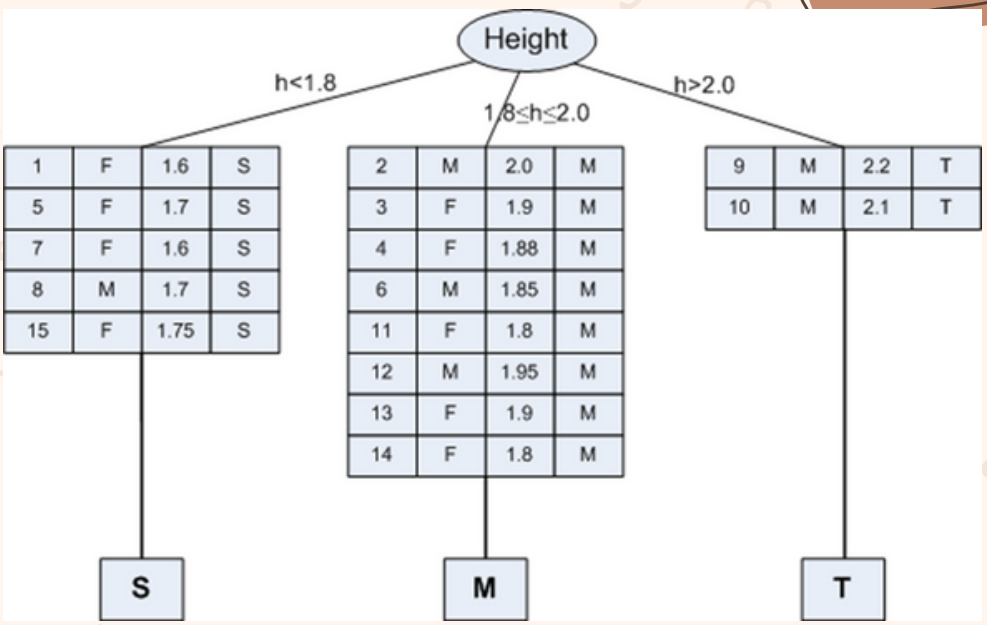



Gender
F
M
1 F 1.6 S
3 F 1.9 M
4 F 1.88 M
5 F 1.7 S
7 F 1.6 S
11 F 1.8 M
13 F 1.9 M
14 F 1.8 M
15 F 1.75 S
2 M 2.0 M
6 M 1.85 M
8 M 1.7 S
9 M 2.2 T
10 M 2.1 T
12 M 1.95 M
Height
h<1.8
1.8≤h≤2.0
Height
h<1.8
1.8≤h≤2.0
h>2.0
1 F 1.6 S
5 F 1.7 S
7 F 1.6 S
15 F 1.75 S
3 F 1.9 M
4 F 1.88 M
11 F 1.8 M
13 F 1.9 M
14 F 1.8 M
8 M 1.7 S
2 M 2.0 M
6 M 1.85 M
12 M 1.95 M
9 M 2.2 T
10 M 2.1 T
S
M
S
M
T

Different Orderings

Approach 2: <Height, Gender>




Height
h<1.8
1.8≤h≤2.0
h>2.0
1 F 1.6 S
5 F 1.7 S
7 F 1.6 S
8 M 1.7 S
15 F 1.75 S
2 M 2.0 M
3 F 1.9 M
4 F 1.88 M
6 M 1.85 M
11 F 1.8 M
12 M 1.95 M
13 F 1.9 M
14 F 1.8 M
9 M 2.2 T
10 M 2.1 T
S
M
T

Numerical Attribute

- Decision Tree Induction is a combinational optimization problem
- Examples of combinational optimization problem
  - Travelling Salesman Problem
  - Minimum Spanning Tree
- Make the size of decision tree smaller

Combinatorial optimization problems involve finding the best solution from a finite set of possible solutions.
So say in Minimum Spanning Tree, given a set of discrete number of nodes, the goal is to determine the best combination of those nodes that satisfies a certain set of constraints like no cycles and achieves the objective which is to generate a tree with the least possible cost that connects all the nodes.

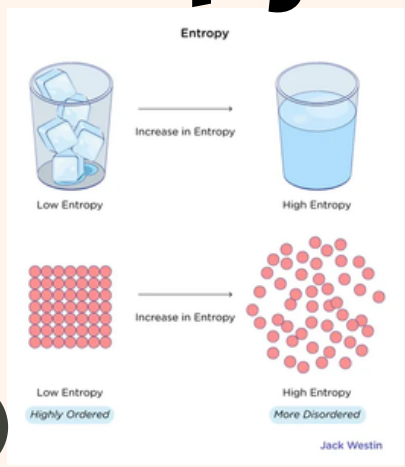
Decision Tree Algorithm

Algorithm (Function name) - BuildDT
Input (I) - Training data set Output (O) - Decision tree

1. If all tuples in input I belongs to the same class C_j
   a. Add a leaf node labeled as C_j
   b. Return // Termination condition
2. Select an attribute A_i (so that it is not selected twice in the same branch)
3. Partition I = { I_1 , I_2, …, I_p } based on p different values of A_i in I
4. For each $I_k \epsilon$ I Create a node and add an edge between I and Ik with label as the A_i 's attribute value in I
5. For each $I_k \epsilon$ I
   a. BuildTD(I_k) // Recursive call

3. Concept of Entropy

Entropy

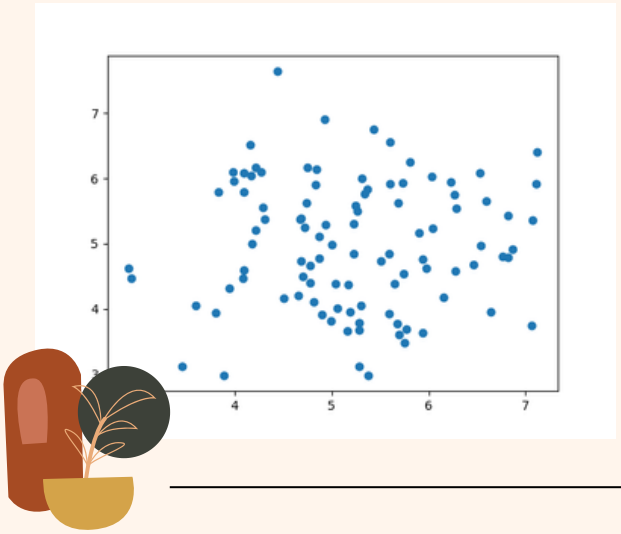


Entropy is a measure of the molecular disorder, or randomness, of a system.

More ordered, less entropy

Less ordered, more entropy

Entropy



Entropy is an information-theoretic measure of the "uncertainty" contained in a training data.

Less uncertainty, less entropy	More uncertainty, more entropy

Due to the presence of more than 1 class

Entropy in Information Theory

Information theory is the mathematical study of the quantification, storage, and communication of information.

The first time it was used to measure the “information content” in messages

Entropy, nowadays, is a way of representing messages for efficient transmission by Telecommunication Systems

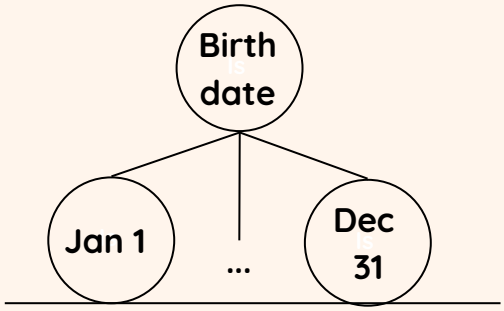
Measure of Information Content

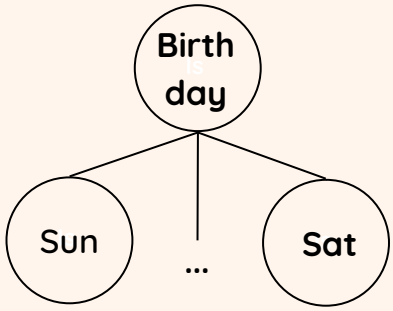
We may note that information gathering may be with certainty or uncertainty.

In fact, fundamental thing is that we gather information by asking questions (and decision tree induction is no exception).

Examples

Guessing a birthday of your classmate
It is with uncertainty ~ 1/365
Whereas guessing the day of his/her birthday is 1/7.
This uncertainty, we may say varies between 0 to 1, both inclusive





Examples

As another example, a question related to event with eventuality (or impossibility) will be answered with 0 or 1 uncertainty.

Does sun rises in the East?	(answer is with 0 uncertainty)
Will mother give birth to male baby?	(answer is with ½ uncertainty)
Is there a planet like earth in the galaxy?	(answer is with an extreme uncertainty)

Definition of Entropy

Suppose there are m distinct objects, which we want to identify by asking a series of Yes/No questions. Further, we assume that m is an exact power of 2, say m=2^n, where n≥1.

Definition 9.2: Entropy

The entropy of a set of m distinct values is the minimum number of yes/no questions needed to determine an unknown values from these m possibilities.

Entropy Calculation

Entropy is the number of yes/no questions to be asked to determine the unknown values from a set m of possibilities.

Brute-force

Clever / Binary Search

Examples

Suppose, Thee is a quiz relating to guess a city out of 8 cities, which are as follows:

Bangalore, Bhopal, Bhubaneshwar, Delhi,
Hyderabad, Kolkata, Madras, Mumbai

The question is, “Which city is called city of joy”?

- Brute force approach
  - We can ask "Is it city *X*?",
  - if *yes* stop, else ask next …

In this approach, we can ask such questions randomly choosing one city at a time. As a matter of randomness, let us ask the questions, not necessarily in the order, as they are in the list.

Q.1:	Is the city Bangalore?	No
Q.2:	Is the city Bhubaneswar?	No
Q.3:	Is the city Bhopal?	No
Q.4:	Is the city Delhi?	No
Q.5:	Is the city Hyderabad?	No
Q.6:	Is the city Madras?	No
Q.7:	Is the city Mumbai?	No

No need to ask further question! Answer is already out by the Q.7. If asked randomly, each of these possibilities is equally likely with probability $\frac{1}{8}$. Hence on the average, we need

$$\frac{(1+2+3+4+5+6+7+7)}{8} = 4.375 \text{ questions.}$$

The question is, "Which city is called city of joy"?
Bangalore, Bhopal, Bhubaneshwar, Delhi, Hyderabad, Kolkata, Madras, Mumbai

- Clever approach (binary search)
  - In this approach, we divide the list into two halves, pose a question for a half
  - Repeat the same recursively until we get *yes* answer for the unknown.

Q.1:	Is it Bangalore, Bhopal, Bhubaneswar or Delhi?	No
Q.2:	Is it Madras or Mumbai?	No
Q.3:	Is it Hyderabad?	No

So after fixing 3 questions, we are able to crack the answer.

Note:
Approach 2 is considered to be the best strategy because it will invariably find the answer and will do so with a minimum number of questions on the average than any other strategy.
Approach 1 occasionally do better (when you are lucky enough!)

- It is no coincidence that $8 = 2^3$, and the minimum number of yes/no questions needed is 3.
- If $m = 16$, then $16 = 2^4$, and we can argue that we need 4 questions to solve the problem. If $m = 32$, then 5 questions, $m = 256$, then 8 questions and so on.

Entropy Calculation

Lemma 9.1: Entropy calculation

The minimum number of *yes/no* questions needed to identify an unknown object from $m = 2^n$ equally likely possible object is n.

If m is not a power of 2, then the entropy of a set of m distinct objects that are equally likely is $\log_2 m$

Entropy in Messages

- We know that the most conventional way to code information is using binary bits, that is, using 0s and 1s.
- The answer to a question that can only be answered *yes/no* (with equal probability) can be considered as containing one **unit of information**, that is, one bit.
- In other words, the unit of information can also be looked at as the amount of information that can be **coded** using only 0s and 1s.

Example 9.7: Information coding

- If we have **two** possible objects say **male** and **female,** then we use the coding
  0 = female
  1 = male
- We can encode **four** possible objects say **East**, **West**, **North**, **South** using two bits, for example
  00 : North
  01 : East
  10 : West
  11 : South
- We can encode **eight** values say eight different colours, we need to use **three** bits, such as
  000 : Violet
  001 : Indigo
  010 : Blue
  011 : Green
  100 : Yellow
  101 : Orange
  110 : Red
  111 : White

Thus, in general, to code m values, each in a distinct manner, we need n bits such that $m = 2^n$.

- In this point, we can note that to identify an object, if it is encoded with bits, then we have to ask questions in an alternative way. For example
  - Is the first bit 0?
  - Is the second bit 0?
  - Is the third bit 0? and so on
- Thus, we need n questions, if m objects are there such that $m = 2^n$.
- The above leads to (an alternative) and equivalent definition of entropy

Definition 9.3: Entropy

The entropy of a set of m distinct values is the number of bits needed to encode all the values in the most efficient way.

Messages when ($m \neq 2^n$)

Previously, we assumed that m, number of distinct objects is exactly 2^n, where $m = 2^n$ and m objects are equally likely.

We will now redefine entropy in a more general case, where $m \neq 2^n$ and m are not necessarily equally probable.

Name Game

Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday

We will identify a sequence of k values

Rules:

$k \geq 1$
Each day is chosen independently
Repetitions are allowed

E_k^m = number of questions required (entropy)

Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday

Name Game

$m = 7 \qquad k = 6$

{Tue, Thu, Tue, Mon, Sun, Tue}

$m^k = 7^6 = 117649$ possible sequences

To get the minimum number of questions:

$\log_2 m^k = \log_2 117649 = 16.8443$

$$E_k^m = \lceil \log_2 m^k \rceil$$
$$= \lceil 16.8443 \rceil$$
$$= 17$$

E_k^m / k = average number of questions needed

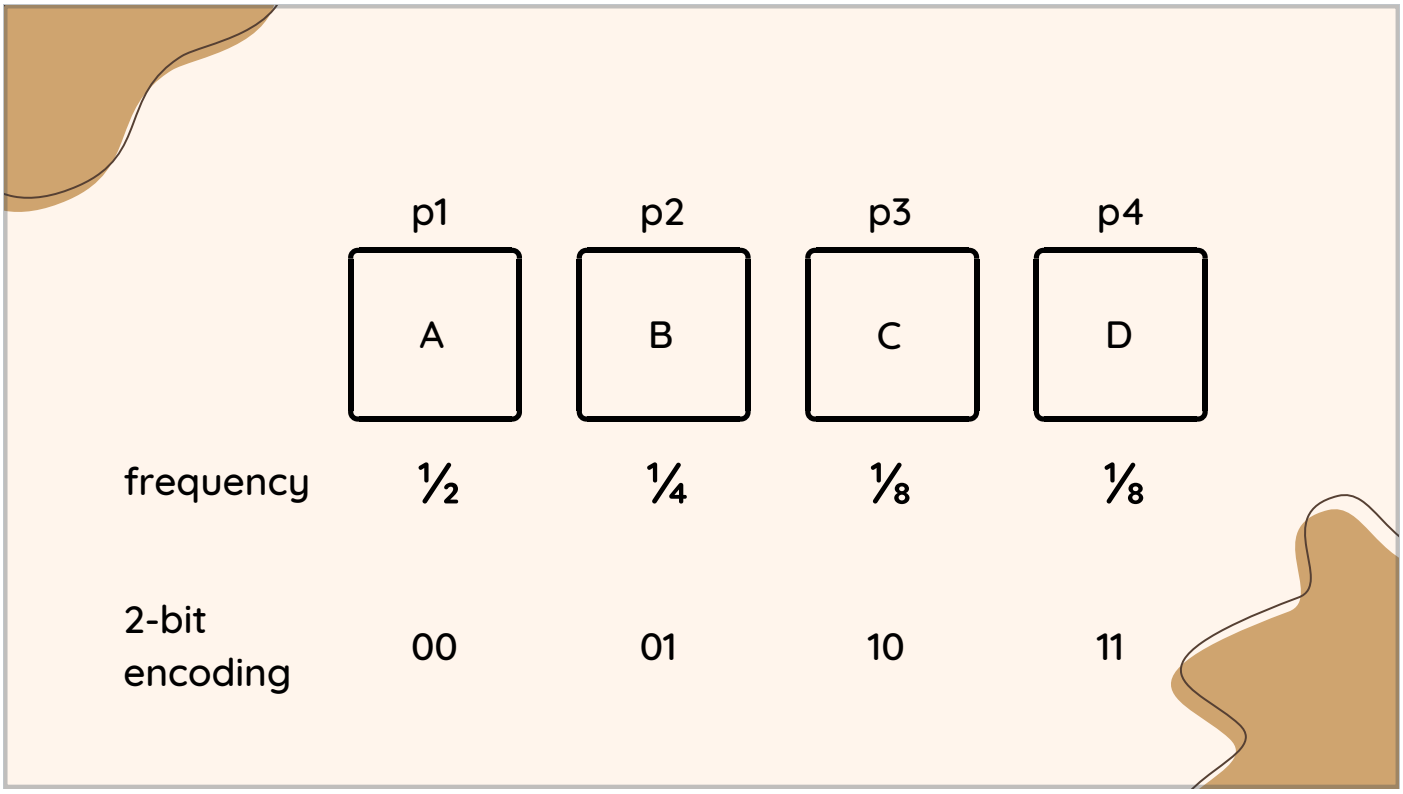
k	m^k	$\log_2 m^k\rceil$	No. Q	$\frac{No.\,Q}{k}$
6	117649	16.84413	17	2.8333
21		58.95445	59	2.8095
1000		2807.3549	2808	2.8080
.....	.....	.....	.....	.....

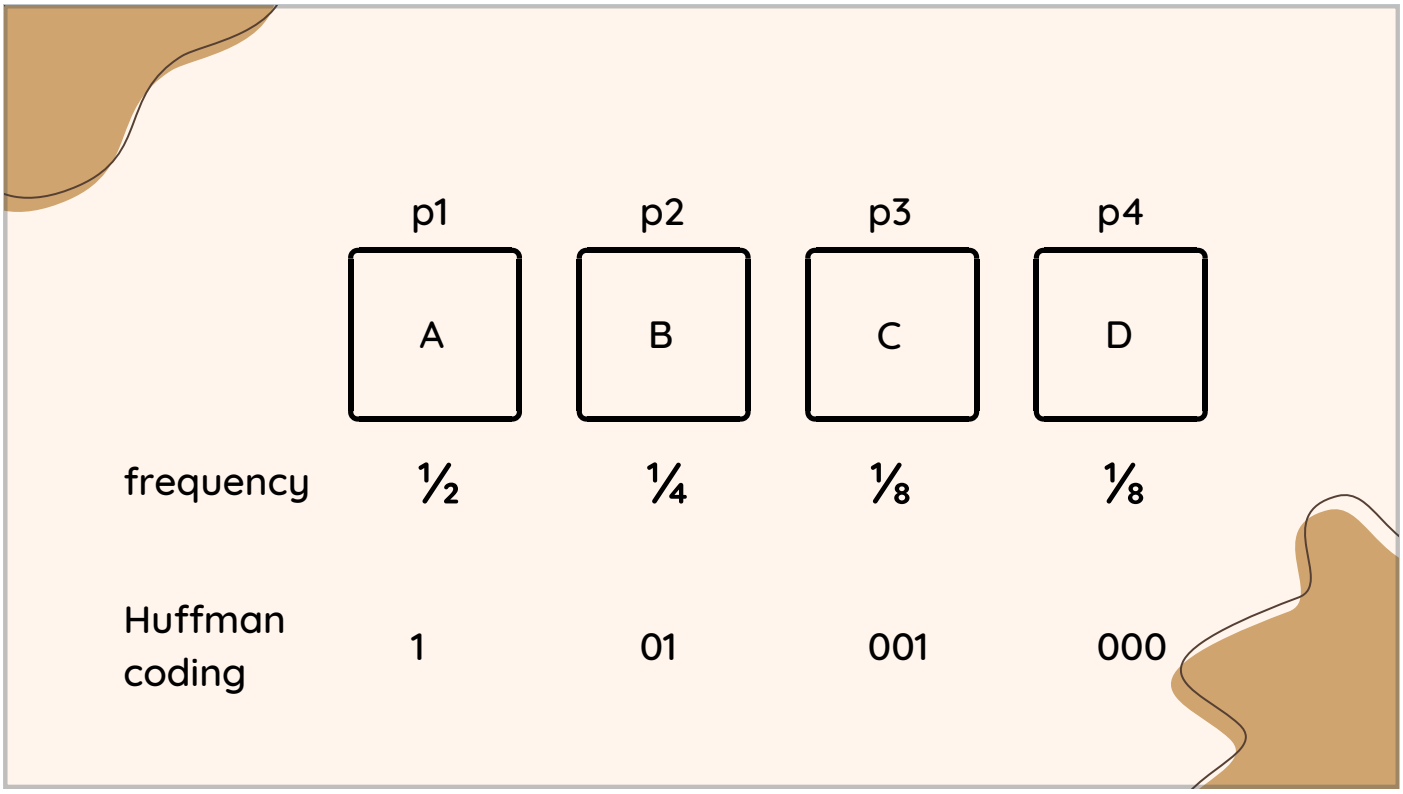
What if all m objects are not equally probable?

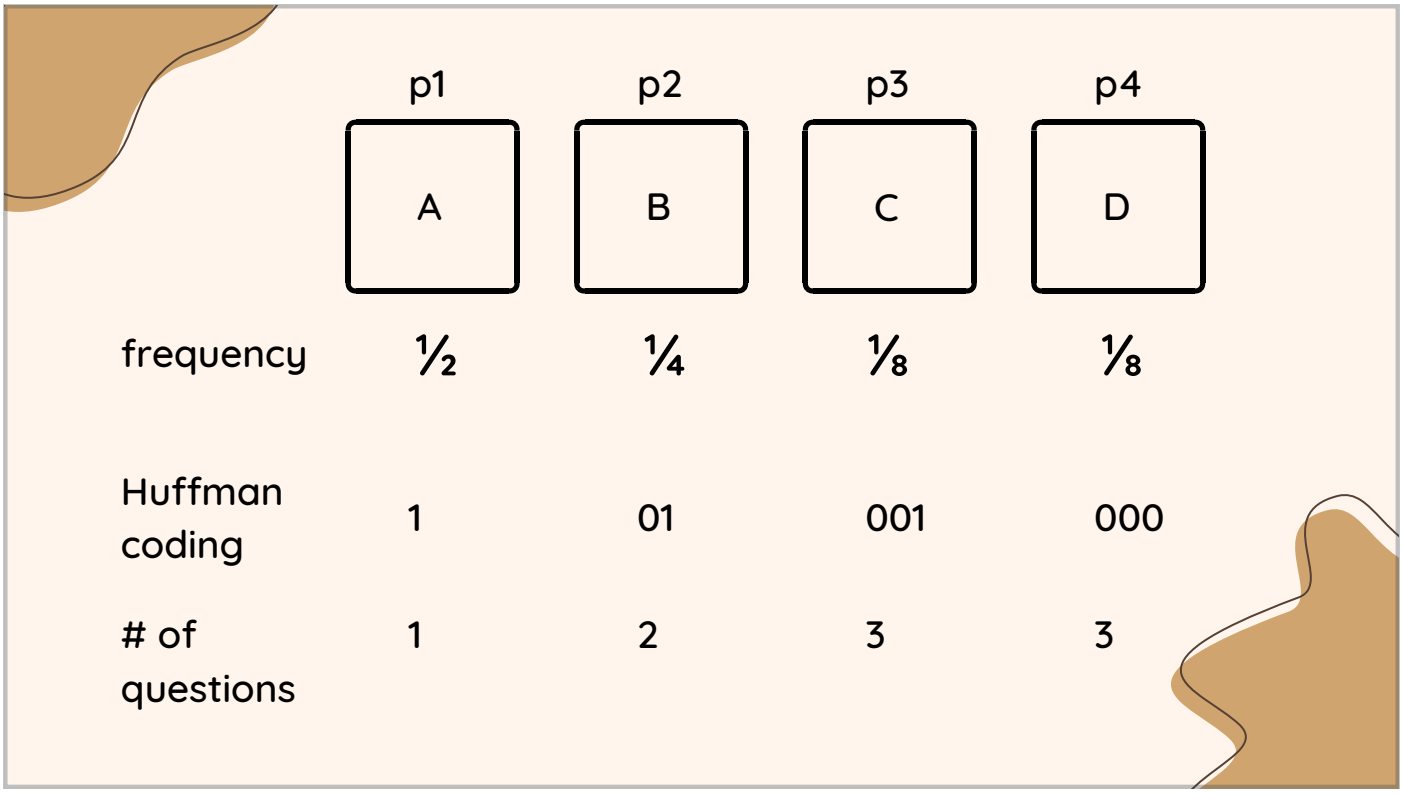
Suppose, p_i denotes the frequency with which i^{th} of the m object occurs

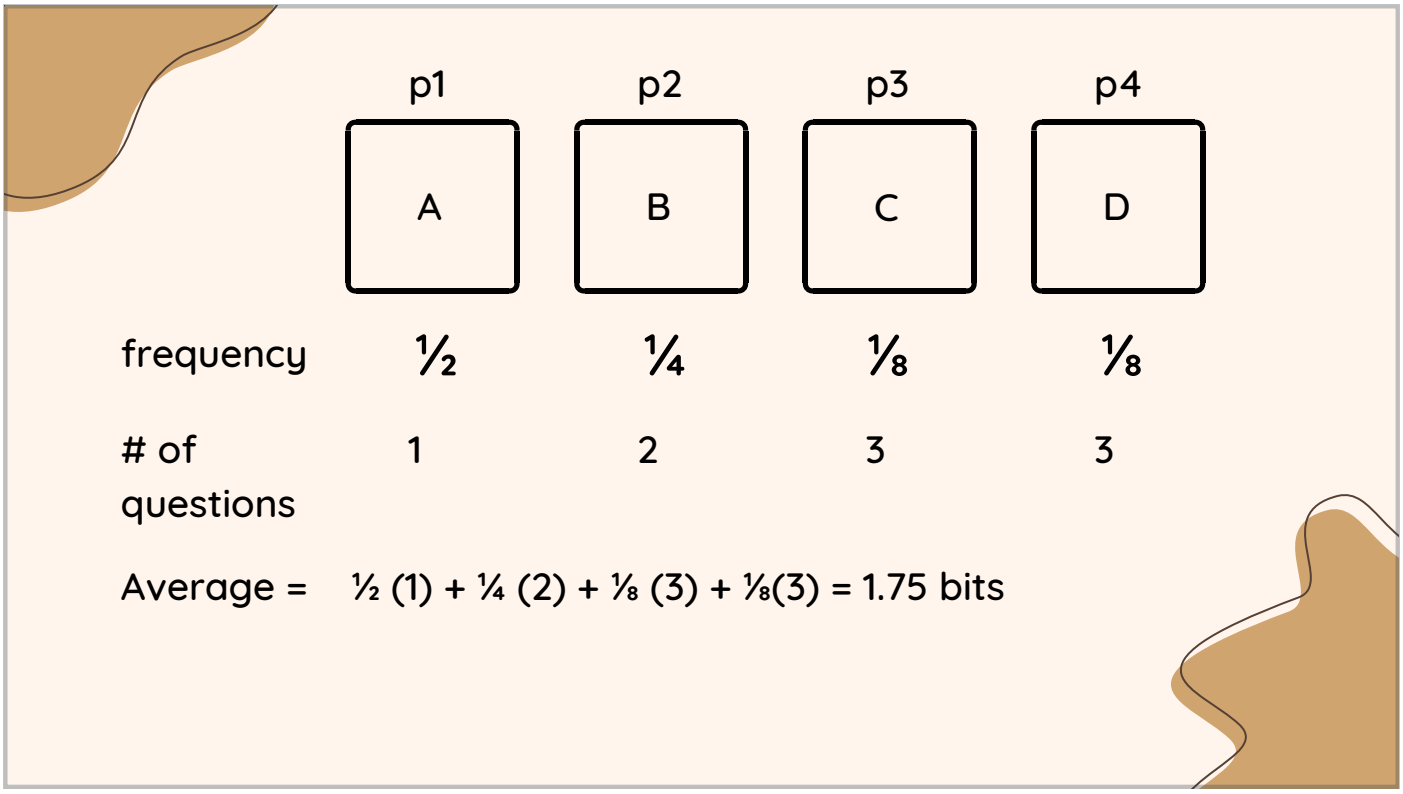
where $0 \leq p_i \leq 1$ for p_i such that:

$$\sum_{i=1}^{m} p_i = 1$$


p1
p2
p3
p4
A
B
C
D
frequency
½
¼
⅛
⅛
2-bit encoding
00
01
10
11


p1
p2
p3
p4
A
B
C
D
frequency
1/2
1/4
1/8
1/8
Huffman coding
1
01
001
000


p1
p2
p3
p4
A
B
C
D
frequency
1/2
1/4
1/8
1/8
Huffman coding
1
01
001
000
# of questions
1
2
3
3


p1
p2
p3
p4
A
B
C
D
frequency
½
¼
⅛
⅛
# of questions
1
2
3
3
Average = ½ (1) + ¼ (2) + ⅛ (3) + ⅛(3) = 1.75 bits

Theorem 9.4: Entropy calculation

If p_i denotes the frequencies of occurrences of m distinct objects, then the entropy E is

$$E = \sum_{i=1}^{m} p_i \log(1/p_i) \text{ and } \sum_{i=1}^{m} p_i = 1$$

A = 01
B = 1
C = 001
D = 000

A = 101
B = 001
C = 10011
D = 100001

A = 01
B = 1
C = 001
D = 000

E = 2

A = 101
B = 001
C = 10011
D = 100001

A = 01
B = 1
C = 001
D = 000

E = 2

A = 101
B = 001
C = 10011
D = 100001

E = 3.875

A = 01
B = 1
C = 001
D = 000

E = 2

A = 101
B = 001
C = 10011
D = 100001

E = 3.875

Assign the smallest number of bits to the object with highest frequency

Based on the previous discussion we can easily prove the following lemma

Lemma 9.3: Information content

If an object occurs with frequency p, then the most efficient way to represent it with $\log_2(1/p)$ bits.

A which occurs with frequency 1/2 is represented by 1-bit
B which occurs with frequency 1/4 represented by 2-bits
Both C and D which occurs with frequency 1/8 are represented by 3 bits each.

Entropy of a Training Set

if there are k classes ($c_1, c_2, \ldots, c_k$) and p_i for i = 1 to k denotes the number of occurrences of class c_i divided by the total number of instances in the training set.

$$E = -\sum_{i=1}^{m} p_i \log_2 p_i$$

Notes:

- Only include non-empty class
- E is **always** a positive quantity
- E take it's minimum value(zero) if and only if all the instances have the same class

Consider the OTPH data shown in the following table with total 24 instances in it.

Age	Eye sight	Astigmatic	Use Type	Class
1	1	1	1	3
1	1	1	2	2
1	1	2	1	3
1	1	2	2	1
1	2	1	1	3
1	2	1	2	2
1	2	2	1	3
1	2	2	2	1
2	1	1	1	3
2	1	1	2	2
2	1	2	1	3
2	1	2	2	1
2	2	1	1	3
2	2	1	2	2
2	2	2	1	3
2	2	2	2	3
3	1	1	1	3
3	1	1	2	3
3	1	2	1	1
3	1	2	2	3
3	2	1	1	3
3	2	1	2	2
3	2	2	1	3
3	2	2	2	3

Age	Eye Sight	Astigmatic	Use Type
1: Young	1: Myopia	1: No	1: Frequent
2: Middle-aged	2: Hypermetropia	2: Yes	2: Less
3: Old			

Classes:

1 - Contact Lens
2 - Normal Glasses
3 - Nothing

Consider the OTPH data shown in the following table with total 24 instances in it.

Instances:
Class 1 - 4 instances
Class 2 - 5 instances
Class 3 - 15 instances

$$E = -\sum_{i=1}^{m} p_i \log_2 p_i$$

$$E = -4/24 \log_2 (4/24) + -5/24 \log_2 (5/24) + -15/24 \log_2 (15/24)$$
$$= 1.3261$$

4. Decision Tree Induction Techniques

Decision Tree Induction Techniques

- Decision tree induction is a top-down, recursive and divide-and-conquer approach.
- The procedure is to choose an attribute and split it into from a larger training set into smaller training sets.
- Different algorithms have been proposed to take a good control over
  - Choosing the best attribute to be splitted, and
  - Splitting criteria
- Several algorithms have been proposed for the above tasks.
  - Examples: ID3, C 4.5, CART

ID3 algorithm

- In 1986, Quinlan introduced the ID3, a popular short form of Iterative Dichotomizer 3 for decision trees from a set of training data.
- In ID3, each node corresponds to a splitting attribute and each arc is a possible value of that attribute.
- At each node, the splitting attribute is selected to be the most informative among the attributes not yet considered in the path starting from the root.

ID3 algorithm

- Entropy - measures how informative a node is.
  - Splitting on any attribute has the property that average entropy of the resulting training subsets <= entropy of the previous training set.

- Information Gain - determines the goodness of a split.
  - Splitting attribute: attribute w/ largest value of information gain
  - it partitions into a number of smaller training sets based on the distinct attribute values under split.

Computing Information Gain

General Procedure

1. Get entropy of entire dataset $E(D)$
2. Identify an attribute A
3. Split the dataset to m partitions, each grouped by a distinct value of the attribute A
4. For each partition (training set) D_i
   a. Get entropy of partition $E(D_i)$
   b. Get weighted entropy of the partition $E_A(D_i)$
5. Get the total weighted entropy with respect to A -> $E_A(D)$
6. Obtain the information gain for the attribute A -> $\alpha(A,D)$

Computing Information Gain

Step 1: Entropy of training set

Number of classes

$$E(D) = -\sum_{i=1}^{k} p_i \log_2(p_i)$$

Probability (of tuple in D belonging to class I)

Entropy

$$p_i = \frac{|C_{i,D}|}{|D|}$$

Number of class i tuples in training set

Size of training set

Computing Information Gain

Step 2: Weighted entropy of partition

Number of partitions

Entropy of partitions

$$E_A(D) = \sum_{j=1}^{m} \frac{|D_j|}{|D|} E(D_j)$$

Weighted Entropy

Weight of partition

$\frac{|D_j|}{|D|}$

Number of tuples in partition j

Size of dataset

Computing Information Gain

Step 3: Determining information gain

$$\alpha(A, D) = E(D) - E_A(D)$$

Information Gain Entropy Weighted Entropy

- Higher E (D) -> more impure (multiple classes)
- Lower E (D) -> more power of partitions
- Attribute A with highest α(A,D) -> splitting attribute for D

Example: Computing Information Gain

Age	Eye sight	Astigmatic	Use Type	Class
1	1	1	1	3
1	1	1	2	2
1	1	2	1	3
1	1	2	2	1
1	2	1	1	3
1	2	1	2	2
1	2	2	1	3
1	2	2	2	1
2	1	1	1	3
2	1	1	2	2
2	1	2	1	3
2	1	2	2	1
2	2	1	1	3
2	2	1	2	2
2	2	2	1	3
2	2	2	2	3
3	1	1	1	3
3	1	1	2	3
3	1	2	1	3
3	1	2	2	1
3	2	1	1	3
3	2	1	2	2
3	2	2	1	3
3	2	2	2	3

1. Get entropy of entire dataset $E(D)$
2. Identify an attribute A
3. Split the dataset to m partitions, each grouped by a distinct value of the attribute A
4. For each partition (training set) D_i
   a. Get entropy of partition $E(D_i)$
   b. Get weighted entropy of the partition $E_A(D_i)$
5. Get the total weighted entropy with respect to A -> $E_A(D)$
6. Obtain the information gain for the attribute A -> $\alpha(A,D)$

Example: Computing Information Gain

Age	Eye sight	Astigmatic	Use Type	Class
1	1	1	1	3
1	1	1	2	2
1	1	2	1	3
1	1	2	2	1
1	2	1	1	3
1	2	1	2	2
1	2	2	1	3
1	2	2	2	1
2	1	1	1	3
2	1	1	2	2
2	1	2	1	3
2	1	2	2	1
2	2	1	1	3
2	2	1	2	2
2	2	2	1	3
2	2	2	2	3
3	1	1	1	3
3	1	1	2	3
3	1	2	1	3
3	1	2	2	1
3	2	1	1	3
3	2	1	2	2
3	2	2	1	3
3	2	2	2	3

Dataset entropy

$$E(D) = -\frac{4}{24}\log_2\frac{4}{24} - \frac{5}{24}\log_2\frac{5}{24} - \frac{15}{24}\log_2\frac{15}{24} = \mathbf{1.3261}$$

Class 1: **4** Class 2: **5** Class 3: **15**

1. Get entropy of entire dataset E(D)
2. Identify an attribute A
3. Split the dataset to m partitions, each grouped by a distinct value of the attribute A
4. For each partition (training set) D_i
   a. Get entropy of partition $E(D_i)$
   b. Get weighted entropy of the partition $E_A(D_i)$
5. Get the total weighted entropy with respect to A -> $E_A(D)$
6. Obtain the information gain for the attribute A -> $\alpha(A,D)$

Example: Computing Information Gain

Age	Eye sight	Astigmatic	Use Type	Class
1	1	1	1	3
1	1	1	2	2
1	1	2	1	3
1	1	2	2	1
1	2	1	1	3
1	2	1	2	2
1	2	2	1	3
1	2	2	2	1
2	1	1	1	3
2	1	1	2	2
2	1	2	1	3
2	1	2	2	1
2	2	1	1	3
2	2	1	2	2
2	2	2	1	3
2	2	2	2	3
3	1	1	1	3
3	1	1	2	3
3	1	2	1	3
3	1	2	2	1
3	2	1	1	3
3	2	1	2	2
3	2	2	1	3
3	2	2	2	3

Pick an attribute: Age

Number of partitions: 3 (Why?)

→

1.3261

1. Get entropy of entire dataset E(D)
2. Identify an attribute A
3. Split the dataset to m partitions, each grouped by a distinct value of the attribute A
4. For each partition (training set) D_i
   a. Get entropy of partition $E(D_i)$
   b. Get weighted entropy of the partition $E_A(D_i)$
5. Get the total weighted entropy with respect to A -> $E_A(D)$
6. Obtain the information gain for the attribute A -> $\alpha(A,D)$

Example: Computing Information Gain

Training set: D_1(Age = 1)

Age	Eye-sight	Astigmatism	Use type	Class
1	1	1	1	3
1	1	1	2	2
1	1	2	1	3
1	1	2	2	1
1	2	1	1	3
1	2	1	2	2
1	2	2	1	3
1	2	2	2	1

Training set: D_2(Age = 2)

Age	Eye-sight	Astigmatism	Use type	Class
2	1	1	1	3
2	1	1	2	2
2	1	2	1	3
2	1	2	2	1
2	2	1	1	3
2	2	1	2	2
2	2	2	1	3
2	2	2	2	3

Training set: D_3(Age = 3)

Age	Eye-sight	Astigmatism	Use type	Class
3	1	1	1	3
3	1	1	2	3
3	1	2	1	3
3	1	2	2	1
3	2	1	1	3
3	2	1	2	2
3	2	2	1	3
3	2	2	2	3

1.3261
Age
m=3

1. Get entropy of entire dataset E(D)
2. Identify an attribute A
3. Split the dataset to m partitions, each grouped by a distinct value of the attribute A
4. For each partition (training set) D_j
   a. Get entropy of partition $E(D_j)$
   b. Get weighted entropy of the partition $E_A(D_j)$
5. Get the total weighted entropy with respect to A -> $E_A(D)$
6. Obtain the information gain for the attribute A -> $\alpha(A,D)$

Example: Computing Information Gain

Training set: D_1(Age = 1)

Age	Eye-sight	Astigmatism	Use type	Class
1	1	1	1	3
1	1	1	2	2
1	1	2	1	3
1	1	2	2	1
1	2	1	1	3
1	2	1	2	2
1	2	2	1	3
1	2	2	2	1

$$E(D_1) = -\frac{2}{8}log_2\left(\frac{2}{8}\right) - \frac{2}{8}log_2\left(\frac{2}{8}\right) - \frac{4}{8}log_2\left(\frac{4}{8}\right) = 1.5$$

$$E_{Age}(D_1) = \frac{8}{24} \times 1.5 = 0.5000$$

→ 1.3261 Age m=3

1. Get entropy of entire dataset E(D)
2. Identify an attribute A
3. Split the dataset to m partitions, each grouped by a distinct value of the attribute A
4. For each partition (training set) D_i
   a. Get entropy of partition $E(D_i)$
   b. Get weighted entropy of the partition $E_A(D_i)$
5. Get the total weighted entropy with respect to A -> $E_A(D)$
6. Obtain the information gain for the attribute A -> $\alpha(A,D)$

Example: Computing Information Gain

Training set: D_2(Age = 2)

Age	Eye-sight	Astigmatism	Use type	Class
2	1	1	1	3
2	1	1	2	2
2	1	2	1	3
2	1	2	2	1
2	2	1	1	3
2	2	1	2	2
2	2	2	1	3
2	2	2	2	3

$$E(D_2) = -\frac{1}{8}log_2\left(\frac{1}{8}\right) - \frac{2}{8}log_2\left(\frac{2}{8}\right) - \frac{5}{8}log_2\left(\frac{5}{8}\right) = 1.2988$$

$$E_{Age}(D_2) = \frac{8}{24} \times 1.2988 = 0.4329$$

$$E_{Age}(D_1) = 0.5000$$

1.3261
Age

m=3

1. Get entropy of entire dataset E(D)
2. Identify an attribute A
3. Split the dataset to m partitions, each grouped by a distinct value of the attribute A
4. For each partition (training set) D_j
   a. Get entropy of partition $E(D_j)$
   b. Get weighted entropy of the partition $E_A(D_j)$
5. Get the total weighted entropy with respect to A -> $E_A(D)$
6. Obtain the information gain for the attribute A -> $\alpha(A,D)$

Example: Computing Information Gain

Training set: D_3(Age = 3)

Age	Eye-sight	Astigmatism	Use type	Class
3	1	1	1	3
3	1	1	2	3
3	1	2	1	3
3	1	2	2	1
3	2	1	1	3
3	2	1	2	2
3	2	2	1	3
3	2	2	2	3

$$E(D_3) = -\frac{1}{8}log_2\left(\frac{1}{8}\right) - \frac{1}{8}log_2\left(\frac{1}{8}\right) - \frac{6}{8}log_2\left(\frac{6}{8}\right) = 1.0613$$

$$E_{Age}(D_3) = \frac{8}{24} \times 1.0613 = 0.3504$$

$$E_{Age}(D_1) = 0.5000$$

$$E_{Age}(D_2) = 0.4329$$

→

1.3261
Age

m=3

1. Get entropy of entire dataset E(D)
2. Identify an attribute A
3. Split the dataset to m partitions, each grouped by a distinct value of the attribute A
4. For each partition (training set) D_i
   a. Get entropy of partition $E(D_i)$
   b. Get weighted entropy of the partition $E_A(D_i)$
5. Get the total weighted entropy with respect to A -> $E_A(D)$
6. Obtain the information gain for the attribute A -> $\alpha(A,D)$

Example: Computing Information Gain

Weighted Entropy (Age) $E_A(D) = 0.5000 + 0.4329 + 0.3504 = \mathbf{1.2833}$

Information Gain (Age) $\alpha\,(Age, D) = 1.3261 - \mathbf{1.2833} = \mathbf{0.0394}$

$E_{Age}(D_1) = \mathbf{0.5000}$

$E_{Age}(D_2) = \mathbf{0.4329}$

$E_{Age}(D_3) = \mathbf{0.3504}$

1.3261
Age
m=3

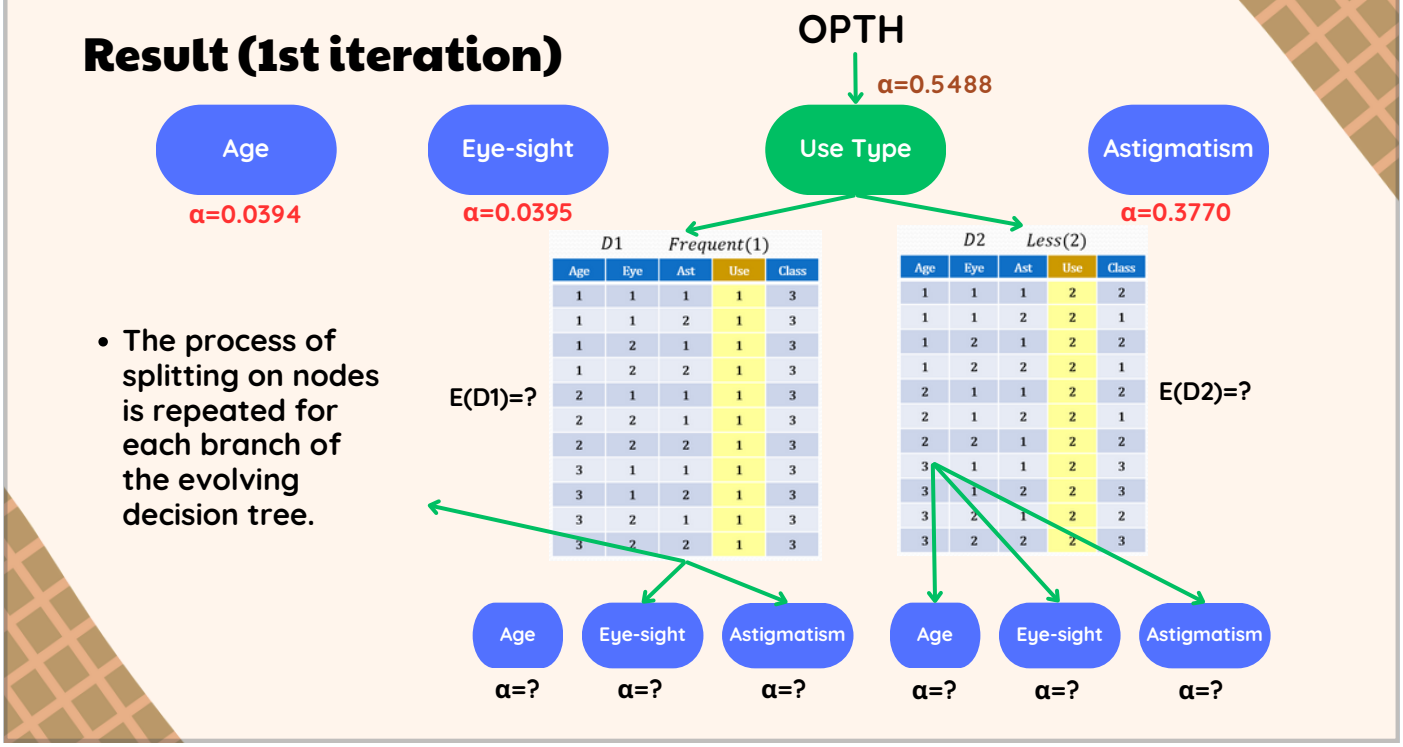
1. Get entropy of entire dataset E(D)
2. Identify an attribute A
3. Split the dataset to m partitions, each grouped by a distinct value of the attribute A
4. For each partition (training set) D_j
   a. Get entropy of partition $E(D_j)$
   b. Get weighted entropy of the partition $E_A(D_j)$
5. Get the total weighted entropy with respect to A -> $E_A(D)$
6. Obtain the information gain for the attribute A -> $\alpha(A,D)$

Information gains per attribute

Splitting attribute A	Information gain α(A,D)
Age	0.0394
Eye-sight	0.0395
Astigmatic	0.3770
Use Type	0.5488

- The attribute Use Type gave the greatest reduction in the weighted average entropy; thus, chosen as the splitting attribute.

Result (1st iteration)



- The process of splitting on nodes is repeated for each branch of the evolving decision tree.


THANK YOU
SO MUCH!