Intro to ML

Thursday, February 16, 2023 11:33 PM

Supervised learning setup:

Training data, D comes in pair (x, y) where $x \in R^d$ is input instance and y is its label.

$$D = \{(x_1, y_1), (x_2, y_2), \ldots\ldots (x_n, y_n)\} \subseteq R^d \times C$$

Where,

R^d = d dimensional feature space

x_i = input vector of i^{th} sample

y_i = Output vector of i^{th} sample

C = label space

Different types of label space

Binary classification $\Rightarrow$ $C = \{0, 1\}$ or $\{-1, 1\}$

Multiclass classification $\Rightarrow$ $C = \{0, 1 \cdots K\}$

Regression $\Rightarrow$ $C = R$, Real number set

Hypothesis class

Before we find the function 'h' such that $h(x) \approx y$ we have to find to which class this fn belongs to. Different classes are, AI, Neural networks, Decision tree, Regression, etc.

tree, Regression, etc.

The set of all hypothesis functions is known as hypothesis class.

No Free lunch theorem -

This theorem states that every successful ML algo. must make assumptions. This also implies that there is no single ML algorithm that can solve every problem.

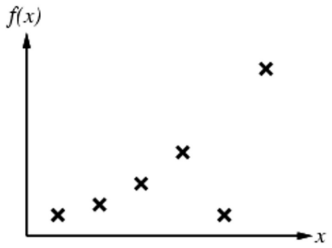
pure induction task is Supervised (Inductive) Learning

- Given a set of hypothesis function $\boxed{h}$ find such h that approximate the function f.

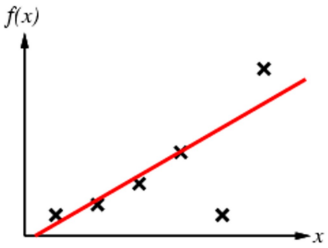
i.e if we are given Training data, D as

$$D = \{ (x_i, f(x_i)) \mid f(x_i) = y_i \}$$

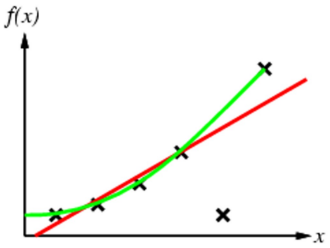
what we have to do is that find a 'h' such that $h \approx f$ with high probability.



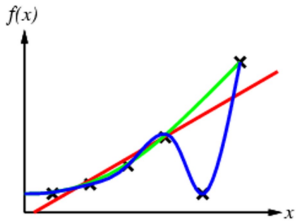
Given D



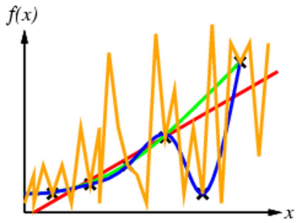
one possible h



Another possible h



One more h



another one.

So, here we have different h's but we have to take One which approximate f. i.e $h \approx f$

Ockham's razor: prefer the simplest hypothesis consistent with data

Supervised Learning

Regression

Learning a Continues function f

Classification

Learning a discrete function f

linear regression

- Simple $h_\theta(x) = \theta_0 + \theta_1 x$
- Multiple $h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots \theta_K x_K$

Simple linear Regression :

Here, our hypothesis's h is,

$$h(x) = \theta_0 + \theta_1 x \quad , \quad \theta = (\theta_0, \theta_1)$$

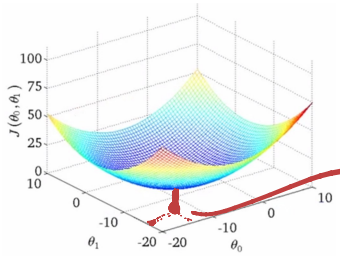
Now, we have to adjust 'h' means adjust θ such

Now, we have to adjust 'h' means adjust θ such that $h \approx f$ with high probability.

parameters are θ_0, θ_1.

Cost function, $J(\theta_0, \theta_1) = \frac{1}{2m} \sum_{i=1}^{m} \underbrace{(h_\theta(x_i) - y_i)^2}_{\text{Squared error}}$

Goal : Minimise $J(\theta_0, \theta_1)$



will be minimise somewhat here

Gradient Descendent :

Let some function $J(\theta_0, \theta_1)$

- we have to find $\min_{\theta_0, \theta_1} J(\theta_0, \theta_1)$
- Start with some θ_0 & θ_1
- Change θ_0, θ_1 to reduce $J(\theta_0, \theta_1)$ until we hopefully reach minimum.

Algorithm:

While (not converged)

{ $\theta_j := \theta_j - \alpha \frac{\partial}{\partial \theta_j} J(\theta_0, \theta_1)$ for $j = 0, 1$

⇓ this shows simultaneous update

}

Simultaneous update:

— Right —

$temp_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

$temp_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

} first calculate new values

$\theta_0 = temp_0$

$\theta_1 = temp_1$

} then update

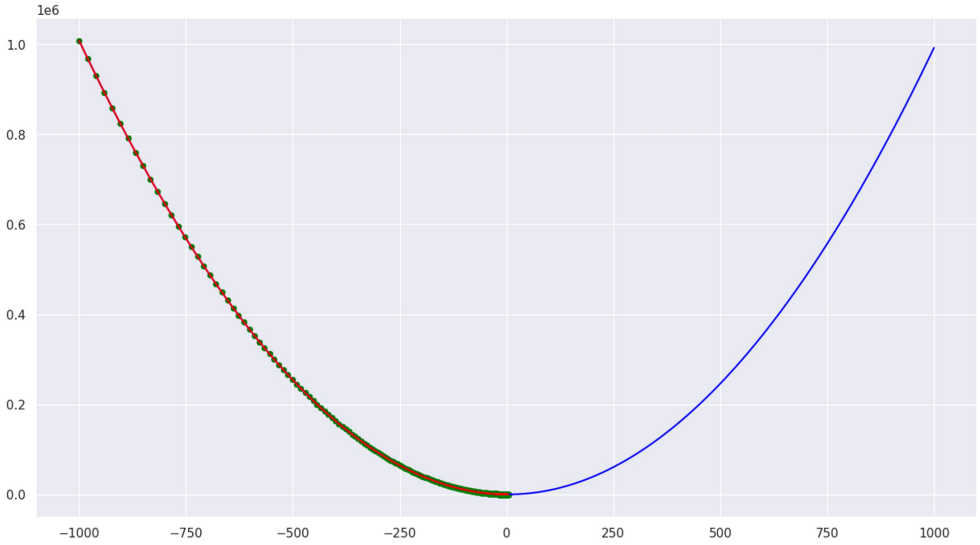
— Wrong —

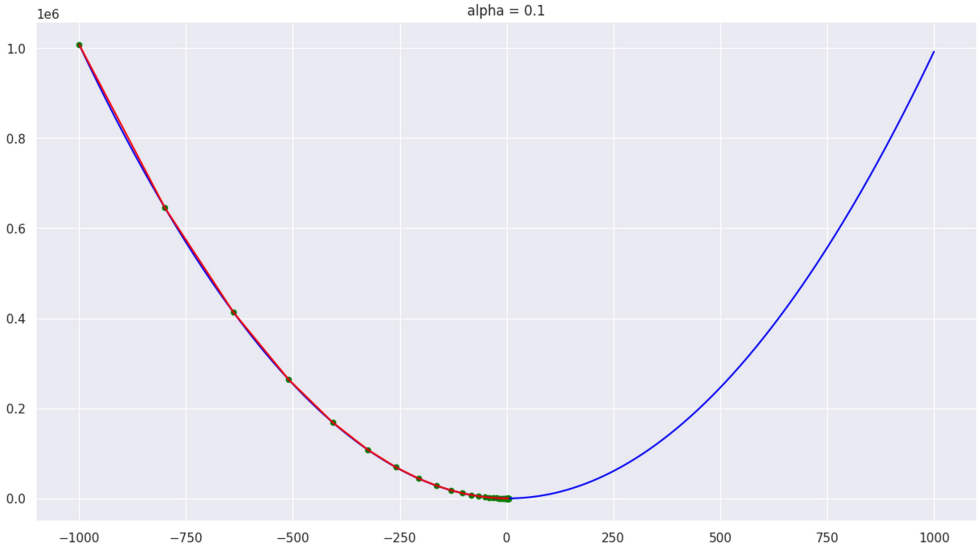
$\theta_0 = \theta_0 - \alpha \frac{\partial}{\partial \theta_0} J(\theta_0, \theta_1)$

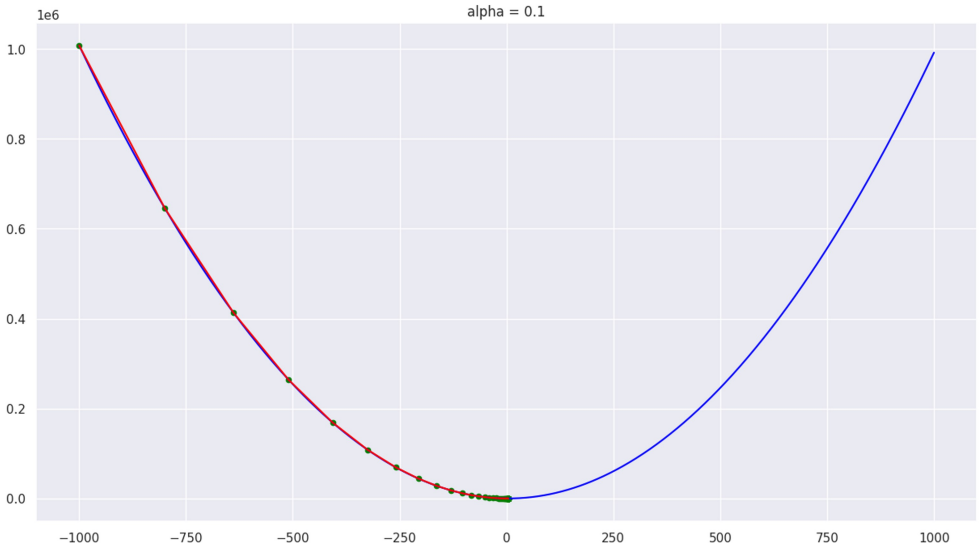
$\theta_1 = \theta_1 - \alpha \frac{\partial}{\partial \theta_1} J(\theta_0, \theta_1)$

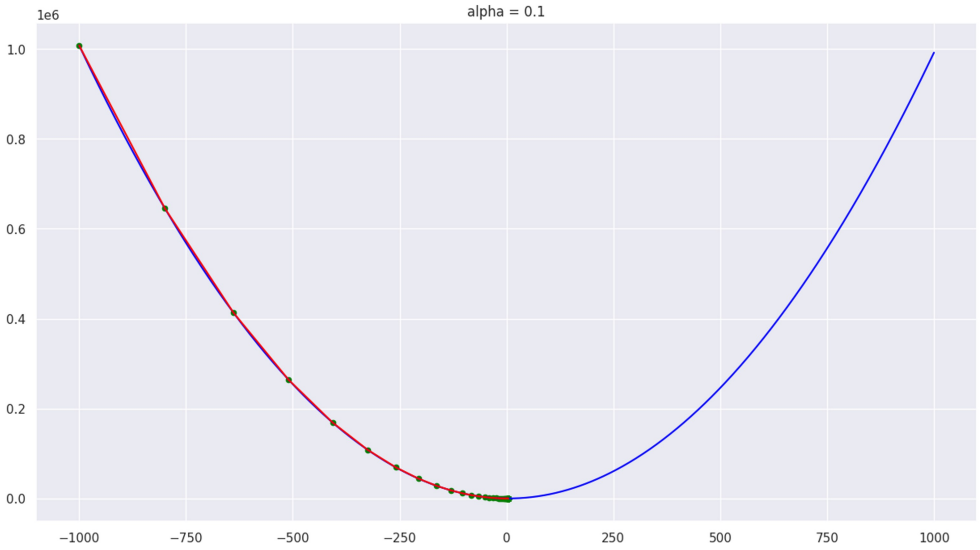
α is known as the learning rate that actually decides How slow, fast we are reaching towards minima

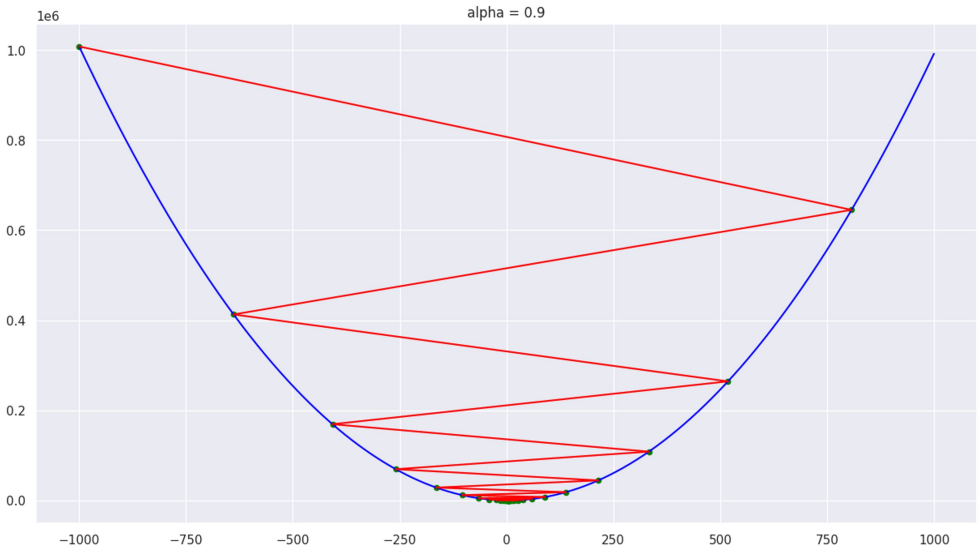
Given below figures show a graph of function $f(x) = (x-3)(x-5)$ and Gradient descendent with different values of alpha.











Multivariate linear regression:

Multivariate linear regression:

$$h_\theta(x) = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \cdots \theta_n x_n$$

$$= \theta^T x$$

where, $\theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \vdots \\ \theta_n \end{bmatrix}$, $x = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_n \end{bmatrix}$

Cost function :

$$J(\theta) = \frac{1}{2m} \sum \varepsilon_i^2$$

where ε_i = error in i^{th} instance of prediction

$$\varepsilon_i^2 = (h_\theta(x_i) - y_i)^2$$

$$h_\theta(x_i) = \theta_0 + \theta_1 x_{i1} + \theta_2 x_{i2} + \cdots \theta_n x_{in}$$

differentiating ε_i^2 wrt θ_j

$$\frac{\partial \varepsilon_i^2}{\partial \theta_j} = 2(h_\theta(x_i) - y_i)^2 \cdot x_{ij}$$

So, $$\frac{\partial J(\theta)}{\partial \theta_j} = \frac{1}{m} \sum_{i=1}^{m} (h_\theta(x_i) - y_i) \cdot x_{ij}$$

Feature Scaling:

- Make sure that features are on the similar scale.

$x_1 \rightarrow$ Number of bedrooms : 1-5

$x_2 \rightarrow$ Size of room : (0 - 2000)

$$x_1 = \frac{x_1}{5} \quad \& \quad x_2 = \frac{x_2}{2000}$$

$$0 \le x_1 \le 1 \qquad 0 \le x_2 \le 1$$

Logistic regression: classification:

Linear regression $\Rightarrow$ $h_\theta(x) = \theta^T x$

Logistic regression $\Rightarrow$ $h_\theta(x) = g_\theta(\theta^T x)$

where, $g(x) = \frac{1}{1+e^{-x}}$ } Sigmoid or logistic fn.

Due to this $h_\theta(x) \in [0,1]$

if $h_\theta(x) < 0.5 \rightarrow$ predict 0

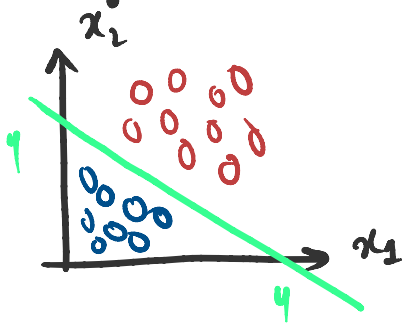
else predict 1.

Here, $h_\theta(x)$ represent the probability of any feature being salient.

$$h_\theta(x) = P(y=1 \mid x; \theta)$$

probability that $y=1$ given x & parametrised by θ.

Decision boundary



Let's take $\theta = \begin{bmatrix} -3 \\ 1 \\ 1 \end{bmatrix} = \theta_{mis}$

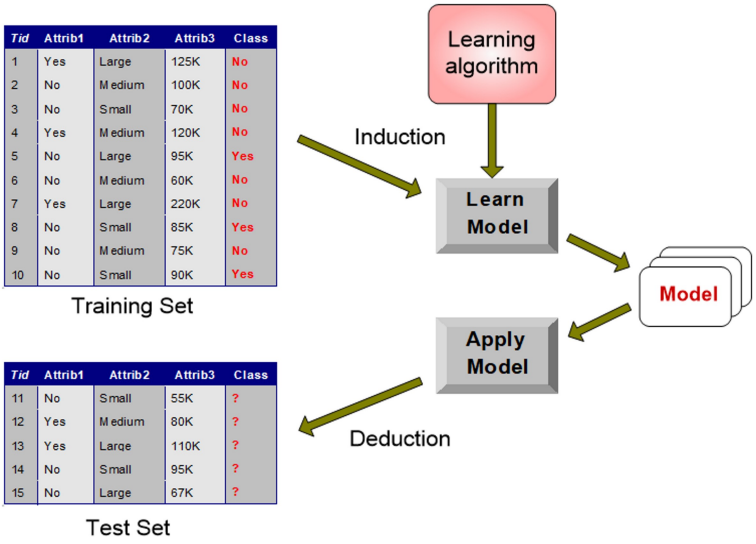
then

$$\theta^T x = -3 + x_1 + x_2$$

So, $g(\theta^T x) = 0.5$ if $\theta^T x = 0$

i.e $\boxed{x_1 + x_2 = 3}$ → This is our decision boundary.

Decision boundary is a property of hypothesis function.



Loss function :

- Determine how good or bad our hypothesis fn is

a) Zero - One loss function { generally used in classification problems }

$$L_{0/1}(h) = \frac{1}{m} \sum_{i=1}^{m} \delta_{h(x_i) \neq y}$$

where, $\delta_{h(x_i) \neq y} = \begin{cases} 1 & \text{if } h(x_i) \neq y \\ 0 & \text{otherwise} \end{cases}$

b) Squared loss function { generally used in regression settings }

$$L_{\text{Squared}}(h) = \frac{1}{m} \sum (h(x_i) - y_i)^2$$

1) If prediction is far away from actual then loss suffered will grow quadratically and that leads to better prediction.

2) If prediction is so close to actual then its square will be more less and then less attention will be given to that example.

Absolute loss

$$L_{abs}(h) = \frac{1}{m} \sum |h(x_i) - y_i|$$

To optimise we should reach

$$h(x) = \text{Median}_{P(y|x)}[y]$$

optimal hypothesis fn:

$$h = \text{argmin}_{h \in H} L(h)$$

h will be the hypothesis for which the loss is minimum.

cost function for logistic regression:

$$\text{Cost}(h_\theta(x), y) = \begin{cases} -\log(h_\theta(x)) & \text{if } y = 1 \\ -\log(1 - h_\theta(x)) & \text{if } y = 0 \end{cases}$$

$$\text{Cost}(h_\theta(x), y) = -y \log(h_\theta(x)) - (1 - y) \log(1 - h_\theta(x))$$

KNN Algorithm

Friday, February 17, 2023 11:01 AM

we have a point 'x' that has to classified into some class Let $S_x \subseteq D$ and $|S_x| = K$ be a set of K Nearest neighbour of point X.

then our hypothesis function is -

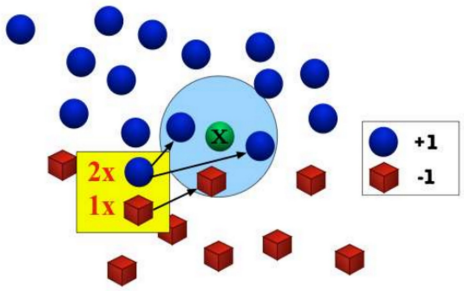
$$h(x) = \text{mode}(\{ y' : (x', y') \in S_x \})$$

i.e select the class of high frequency.

Assumption taken for this problem is -

Similar inputs have Similar Output.

Example: $K = 3$



Distance function:

$$\text{dist}(x_1, x_2) = \left\{ \sum_{i=1}^{d} |x_1 - x_2|^p \right\}^{\frac{1}{p}}$$

This is Known as "MinKowski distance"

This is Known as "MinKowski distance"

$p = 1 \Rightarrow$ Manhattan Distance

$p = 2 \Rightarrow$ Euclidean Distance

Curse of dimensionality:

- As the dimension of feature space increases the closeness of points decreases.

Let's illustrate the above statement with an example:

Lets take a cubic of edge length = 1 as our dataset i.e it encloses all the points in our dataset.

Then, to enclose K Neighbours we take the edge length of min hypercube = l

Now, $l^d \approx \frac{K}{N}$ Assuming distribution is uniform over volume.

$$l \approx \left(\frac{K}{N}\right)^{\frac{1}{d}}$$

So, as dimension increases value of ℓ will increases.

d	ℓ
2	0.1
10	0.63
100	0.955
1000	0.9954

k-NN summary

- k-NN is a simple and effective classifier if distances reliably reflect a semantically meaningful notion of the dissimilarity. (It becomes truly competitive through metric learning)
- As $n \to \infty$, k-NN becomes provably very accurate, but also very slow.
- As $d \gg 0$, points drawn from a probability distribution stop being similar to each other, and the kNN assumption breaks down.

Decision Tree

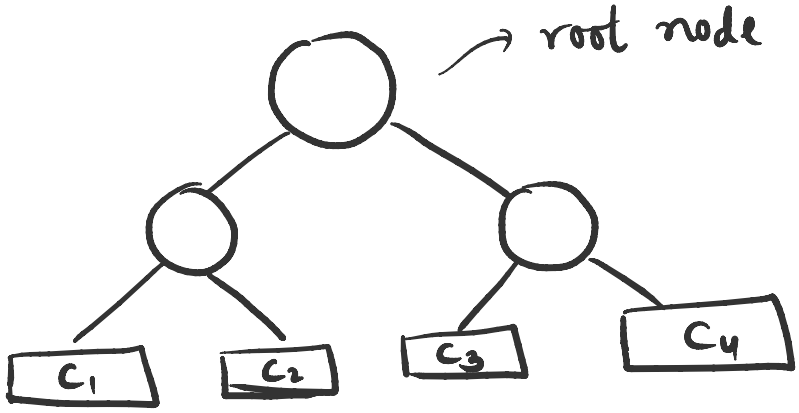
Friday, February 17, 2023 11:39 AM

Given a database $D = \{t_1, t_2 \dots t_n\}$ where t_i is the tuple in database, which is define by set of attribute $A = \{A_1, A_2, A_3 \dots A_n\}$. Also given a set of classes $C = \{c_1, c_2 \dots c_n\}$

Decision tree :

○ → internal nodes

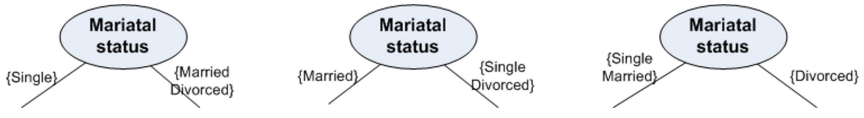
▭ → leaf nodes



Node splitting in building binary tree:

a) Binary attribute

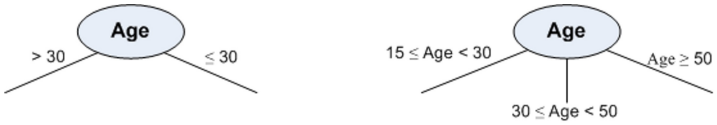
b) Nominal attribute



c) Ordinal attribute

- Order should be maintained properly:

d) Numerical attribute



Concept of Entropy:

- This is needed to chose the order of attribute selection
- First time it was used to measure information content in a message.

Entropy Calculation:

The minimum number of Yes/No question asked to reach to the answer.

Let suppose we have 'm' values

then its entropy is $n = \log_2 m$

Alternate definition of 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.

Example:

00 : North
01 : East
10 : West

→ contains 4 values

→ Number of bits

01 : East
10 : West
11 : South

→ Number of bits needed = 2

If distribution of values is not uniform

Example: Let take four values A, B, C and D and with probability P_1, P_2, P_3 & P_4 respectively.

Then, for this we use huffman coding not simple coding

$A \rightarrow b_1$: bits to represent A

$B \rightarrow b_2$

$C \rightarrow b_3$

$D \rightarrow b_4$

So, on an average number of bits required are

$$b_{avg} = \sum P_i b_i$$

So, if an object occurs with possibility P then Number of bits required to represent it is

$$= \log_2 \left(\frac{1}{P}\right) = -\log_2 P$$

So, $$b_{avg} = -\sum P_i \log P_i = E$$

↳ entropy

Log entropy

Entropy of a dataset

$$E = -\sum p_i \log(p_i)$$

E is always non-negative

$$0 \leq E \leq \frac{1}{K} \log_2 K$$

→ when all data is distributed uniformly over K class

$m = 2^K$ in case of uniform distribution

→ when all data is distributed in one class only

- Different algorithms have been proposed to take a good control over
  1. Choosing the best attribute to be splitted, and
  2. Splitting criteria

ID_3 Algorithm:

→ Several terms

$$E(D) = -\sum p_i \log(p_i)$$

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

Number of sub dataset 'j' which is partition on Attribute 'A'

Then information gain $\alpha(A, D)$ will be

Then information gain $\alpha(A,D)$ will be

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

Frequency table : A shortcut method for calculating α

Data set *Table A*

X	Y	Class
1	1	A
1	2	B
2	1	A
2	2	B
3	2	A
3	1	B
4	2	A
4	1	B

X *Table X*

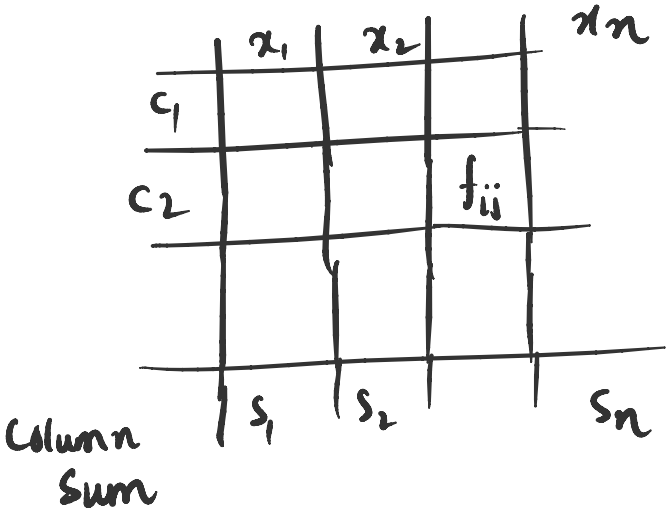
	1	2	3	4
A	1	1	1	1
B	1	1	1	1
C.Sum	2	2	2	2

Frequency table of X

Y *Table Y*

	1	2
A	2	2
B	2	2
C.Sum	4	4

Frequency table of Y



$V = f_{ij} \log f_{ij}$ for all i & j

$S = S_i \log S_i$ for all i

then, $E_x(D) = (-V+S)/N$

$$\alpha(X, D) = E(D) - E_X(D)$$

CART Algorithm:

Inspite of using entropy of dataset it uses $G(D)$ i.e the measure of impurity of dataset

$$G(D) = \sum 1 - p_i^2$$

$$0 \leq G(D) \leq 1 - \frac{1}{K}$$

- 0: when there is only single class
- $1 - \frac{1}{K}$: when data is distributed uniformly

Now, similar to ID_3 we divide D into D_1 and D_2 on attribute A then.

$$\underbrace{\gamma(A, D)}_{\text{Gini Index / Impurity reduction}} = G(D) - G_A(D)$$

Calculation using frequency table

$$G(A, D) = 1 - \frac{1}{|D|} \sum_{j=1}^{m} \underbrace{\frac{1}{|D_j|}} \sum_{i=1}^{K} f_{ij}^2$$

$$G(A,D) = 1 - \frac{1}{|D|} \sum_{j=1} \underbrace{|D_j|}_{\text{Column Sum}} \sum_{i=1} \cdots$$

Algorithm C4.5 –

$$\text{Split information} = -\sum_{j=1}^{m} \frac{|D_j|}{|D|} \log \frac{|D_j|}{|D|} = E_A^*(D)$$

↳ Here 'm' shows number of values of Attribute 'A'

then our golden ratio is,

$$\beta = \frac{\alpha(A,D)}{E_A^*(D)}$$

$$0 \leq \underbrace{E_A^*(D)}_{\text{Single attribute value.}} \leq \log_2 m$$

↳ m shows the number of distinct values of A.

calculation of $\underbrace{E^*(A,D)}_{\text{Spit info.}}$ using frequency table

$\sum_{i=1}^{m} S_i \log S_j$

$$= - \sum_{j=1}^{m} \frac{s_j}{|D|} \log_2 \frac{s_j}{|D|}$$