APPLICATION OF SUPERVISED LEARNING (CLASSIFICATION) IN DATA ANALYTICS

BACS3013

DATA SCIENCE

PRELUDE

- In predictive analytics, the aim is to build an analytical model predicting a target measure of interest. The target is then typically used to steer the learning process during an optimization procedure.
- Two types of predictive analytics can be distinguished: regression and classification. In regression, the target variable is continuous. Popular examples are predicting stock prices, loss given default (LGD), and customer lifetime value (CLV). In classification, the target is categorical.
- It can be binary (e.g., fraud, churn, credit risk) or multiclass (e.g., predicting credit ratings). Different types of predictive analytics techniques have been suggested in the literature. In what follows, we will discuss a selection of techniques with a particular focus on the practitioner's perspective.

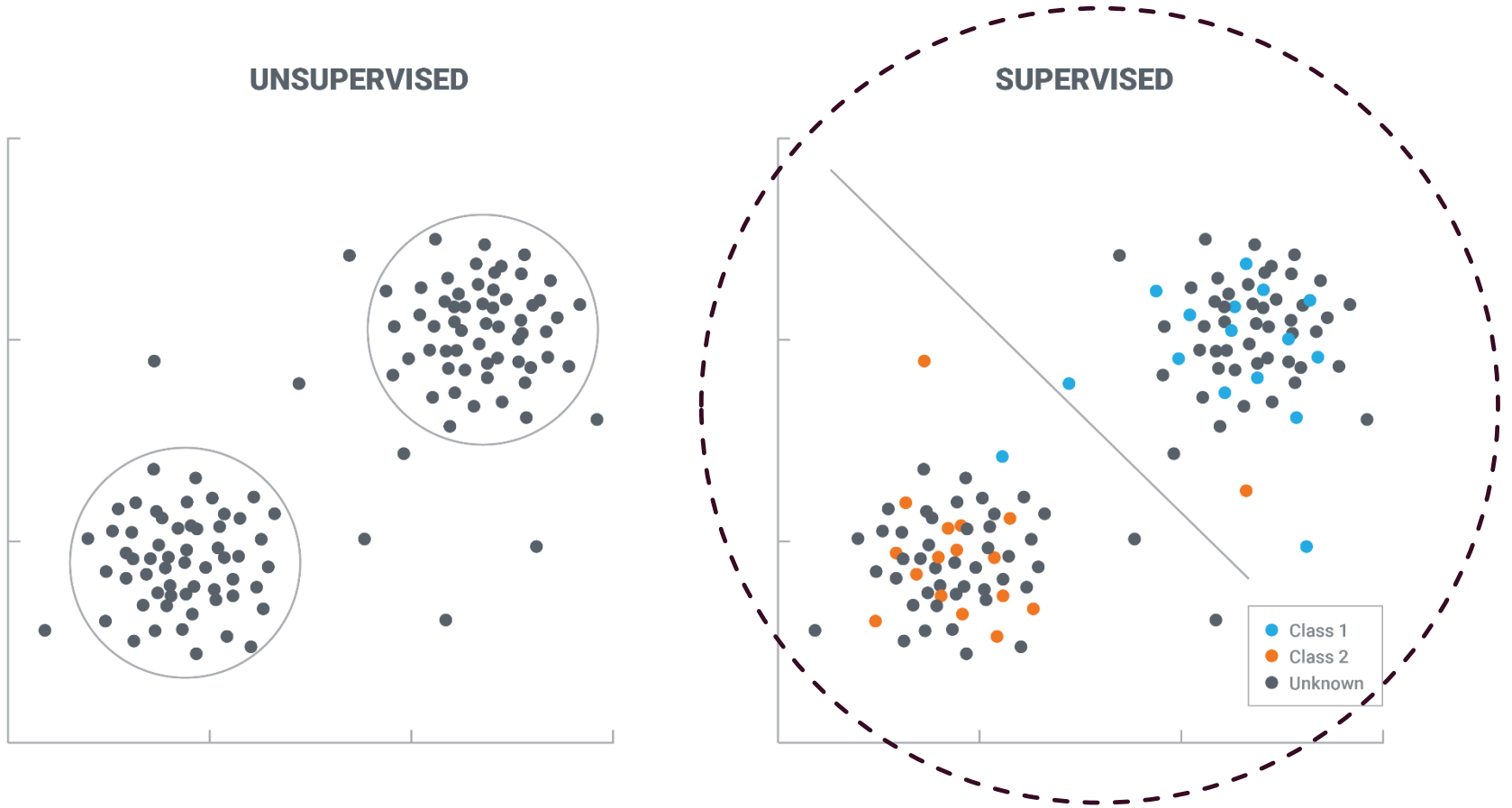
EXAMPLE

- **David:** If you go back in time, say, 10 years ago, one of the oft-repeated mantras was "the best practice is to review all your logs every day." You were expected to go through everything and say "hey, this log is kind of suspicious," but that was never really a scalable approach. It's certainly not realistic now that we're generating terabytes of data every day; you just can't do it.
- On the other hand, humans really are very good at finding patterns and noticing odd things. Computers are really good at doing repetitive work and working on a large scale, so one of the big roles for machine learning is to complement what an analyst can do. This allows you to apply your human smarts and pattern-recognition abilities on a big data scale.

INTRODUCTION

- Supervised learning is the machine learning task of inferring a function from labeled training data. The training data consist of a set of training examples. In supervised learning, each example is a pair consisting of an input object (typically a vector) and a desired output value (also called the supervisory signal). A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way

UNSUPERVISED VS SUPERVISED LEARNING



In supervised machine learning (on the right) known data is labeled into two classes (orange and blue) and unknown data (gray) is run through the model and the label is predicted. In unsupervised machine learning (on the left) unlabeled data (gray) is grouped into classes that the model determines.

ALGORITHMS THAT FACILITATE SUPERVISED LEARNING

- Linear Regression
- *K*-Nearest Neighbour
- Decision Tree
- Bayes Classification
- Artificial Neural Network
- Support Vector Machine

LINEAR REGRESSION

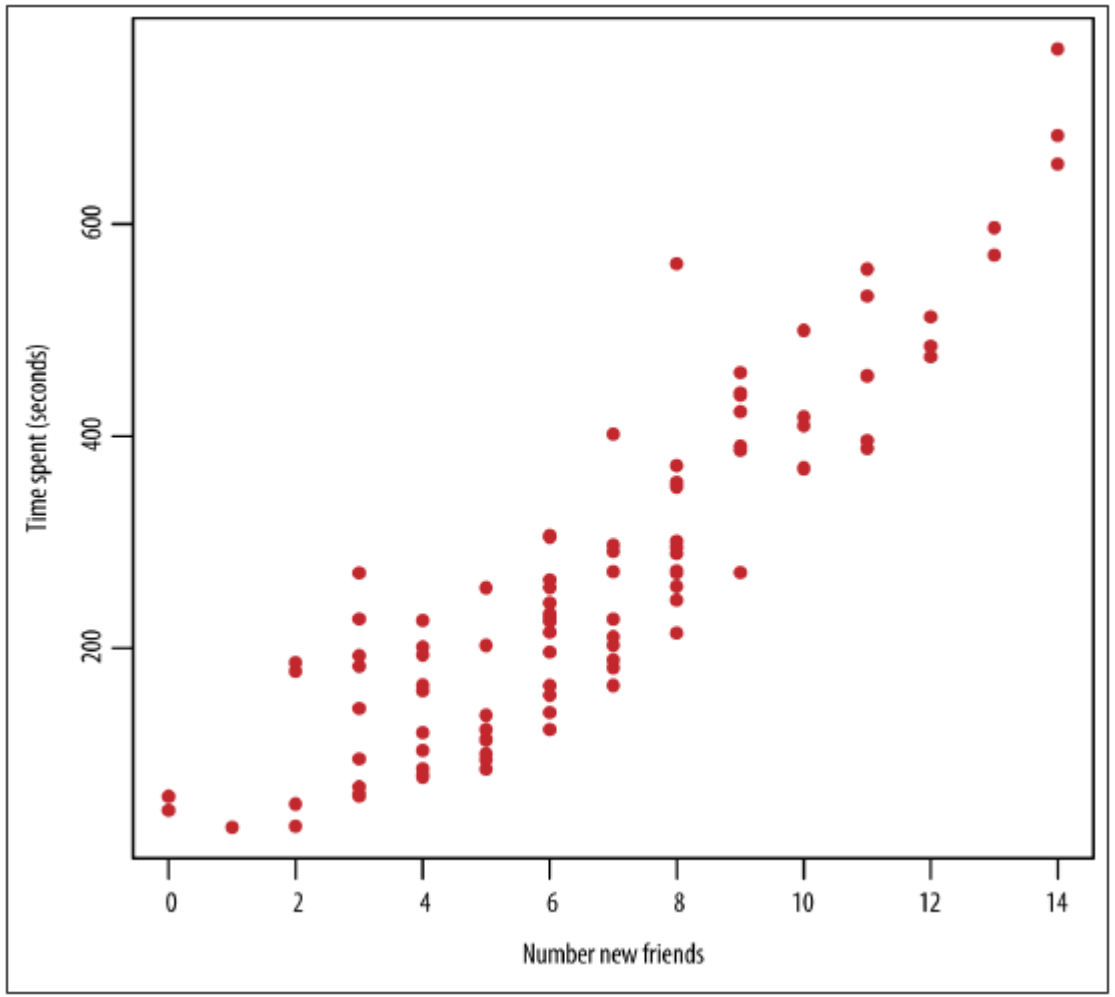
- One of the most common statistical method
- Used to express (linear) mathematical relationship between two variables.
- Example: Curious case of Social Media 

"Suppose you run a social media site, and you randomly sampled 100 users to study their behavior."

User #	No of friends	Time Spent
1	7	276
2	3	43
3	4	82
4	6	136
5	10	417
6	9	269
..	..	..

Ugh! Let's plot them!

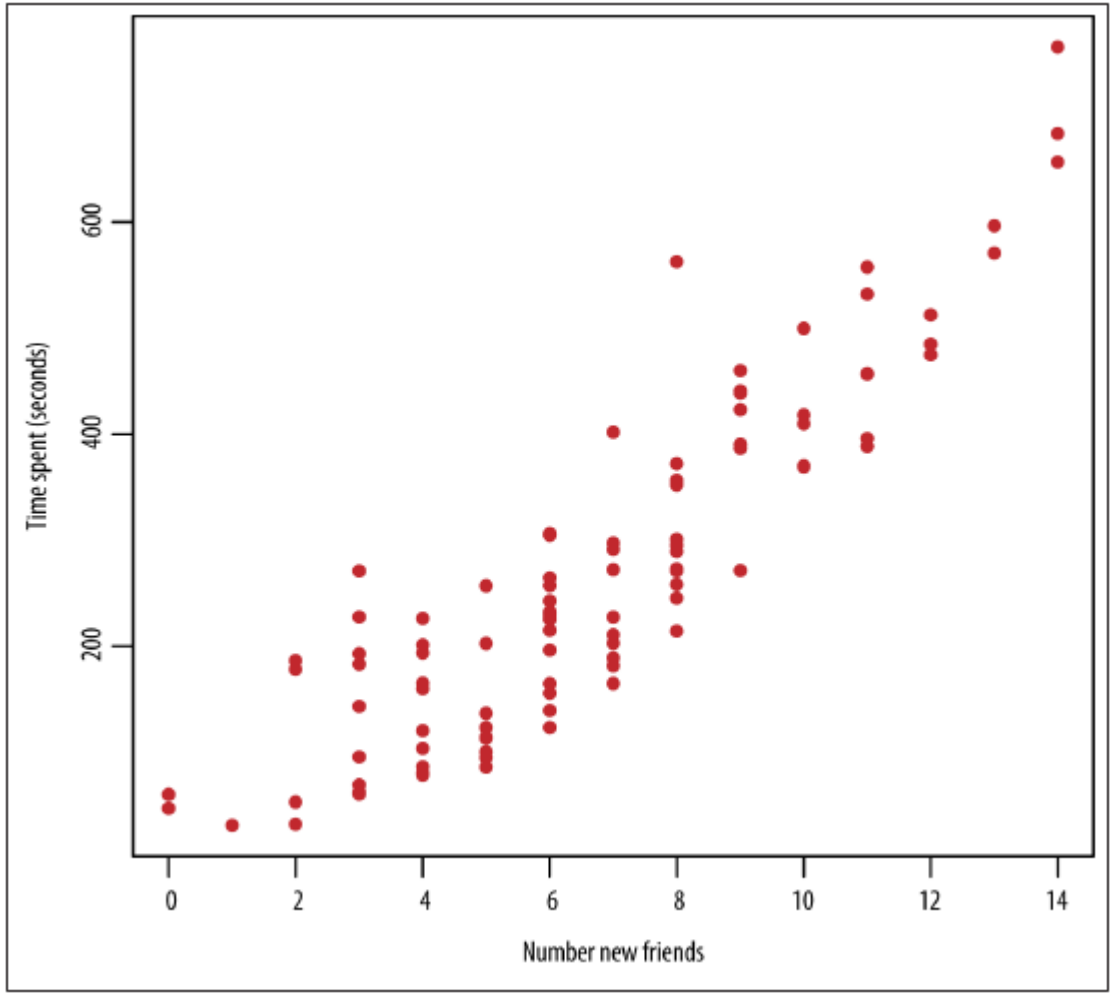
LINEAR REGRESSION

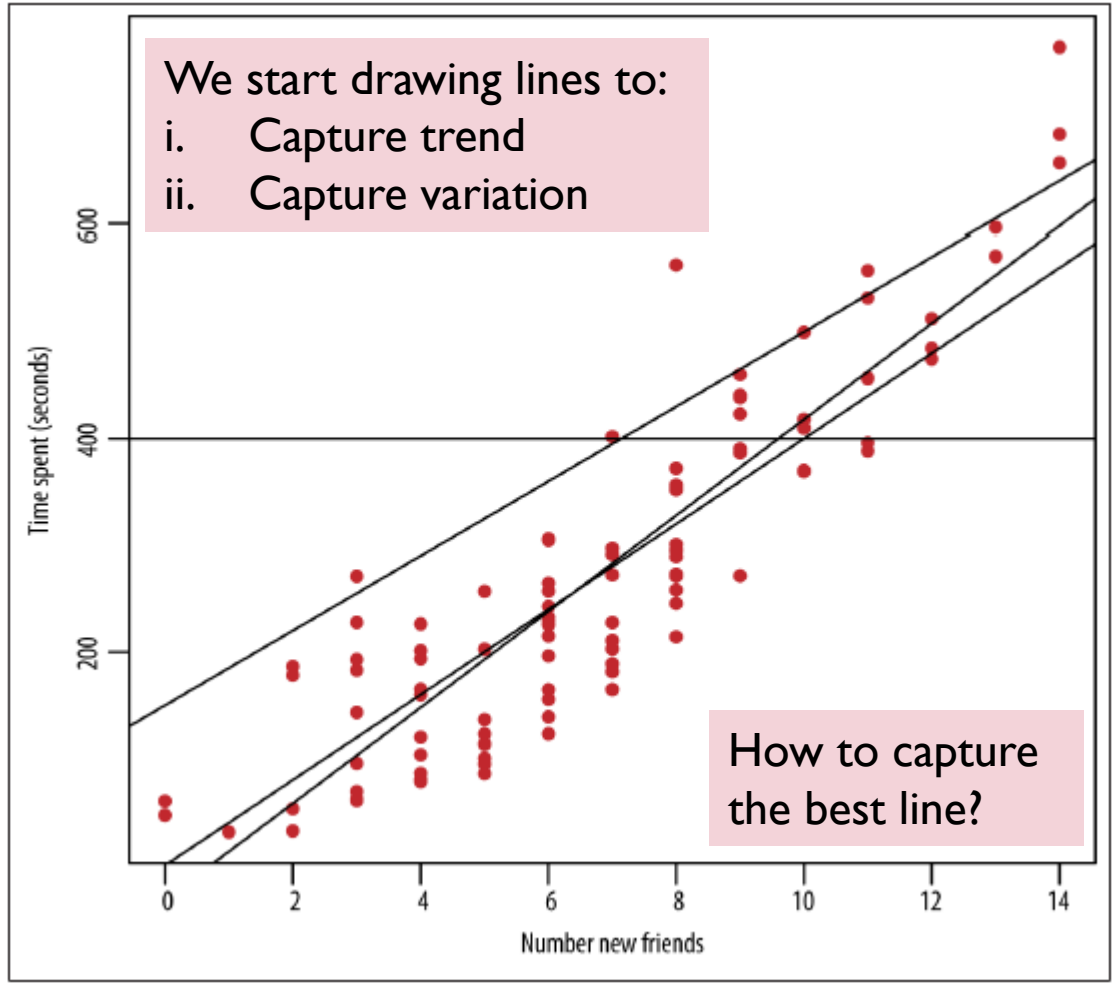


Looks Linear (The more friends a user has, the longer time he/she spends)

But how do we describe the relationship between the 2 variables?

LINEAR REGRESSION



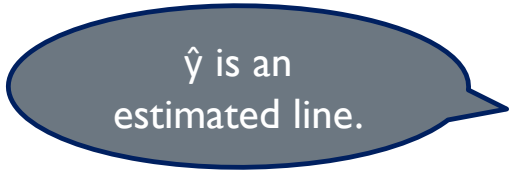


LINEAR REGRESSION

- Since we assumed that a linear relation exists between our variables, our model assumes the functional form:

$$y = \beta_1 x + \beta_0 \ (mx + c)$$

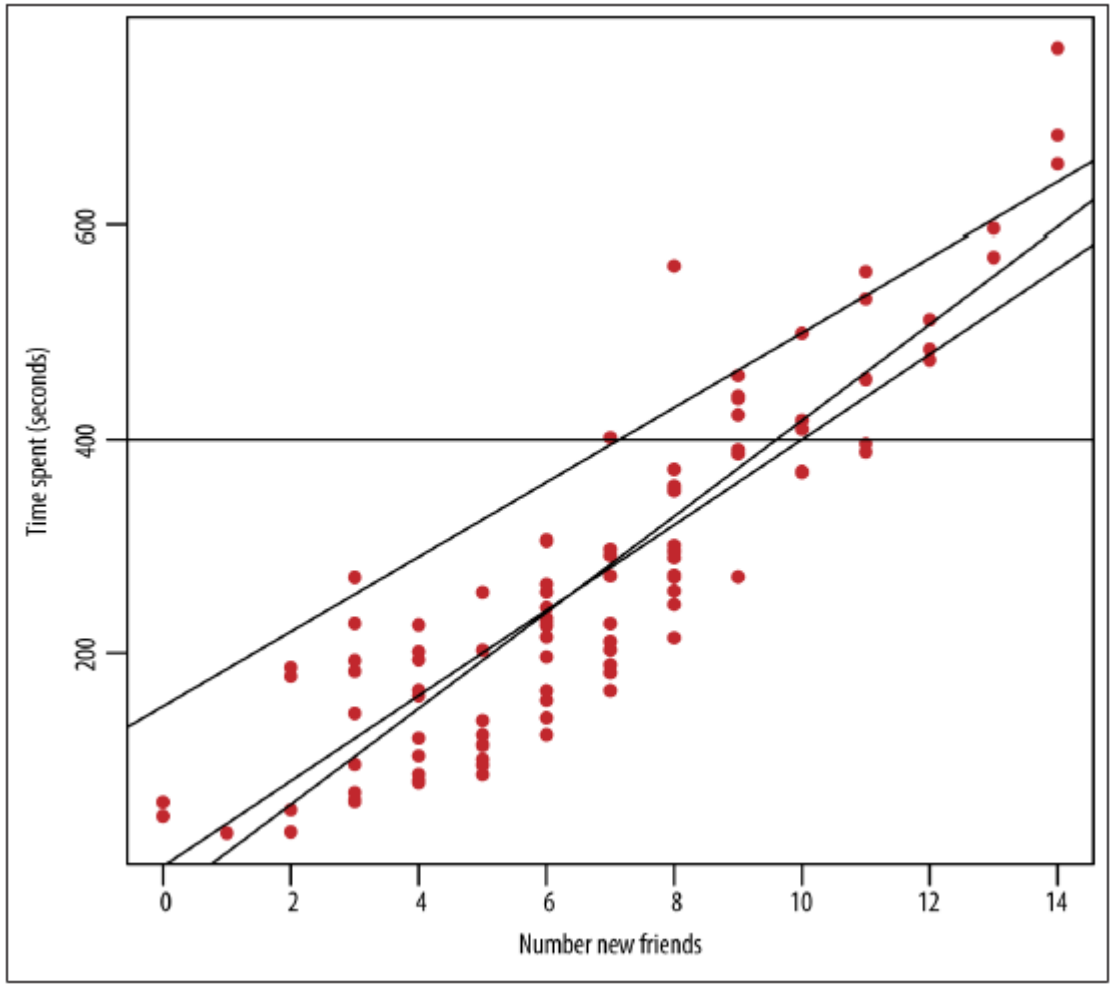
- Now we just have to find the best β_0 and β_1 using our observed data to estimate them. (x_1, y_1), (x_2, y_2) etc.
- To calculate β, we need to find the best line that minimizes the distance between all points and the line. (Least Square Estimation method)

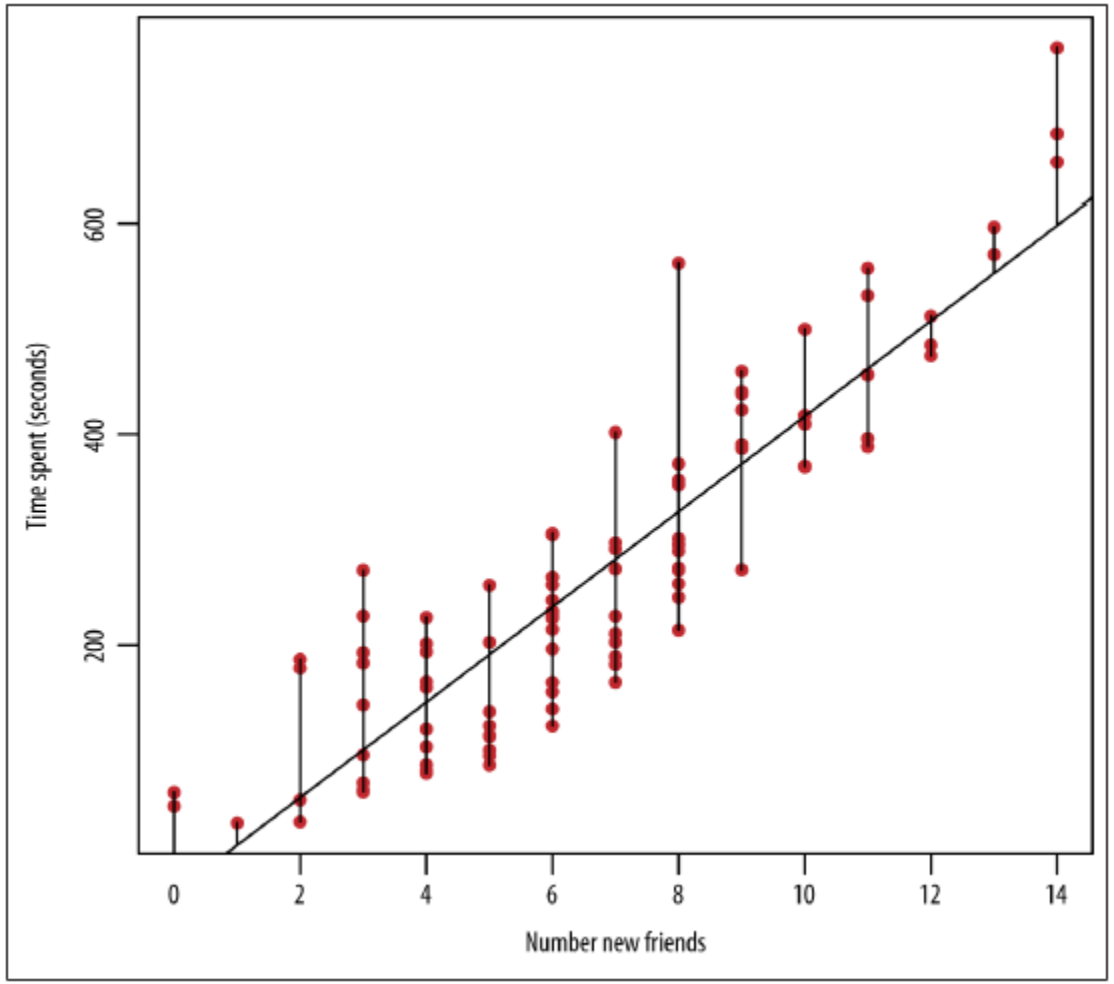


$$\hat{y} = 46x - 32$$

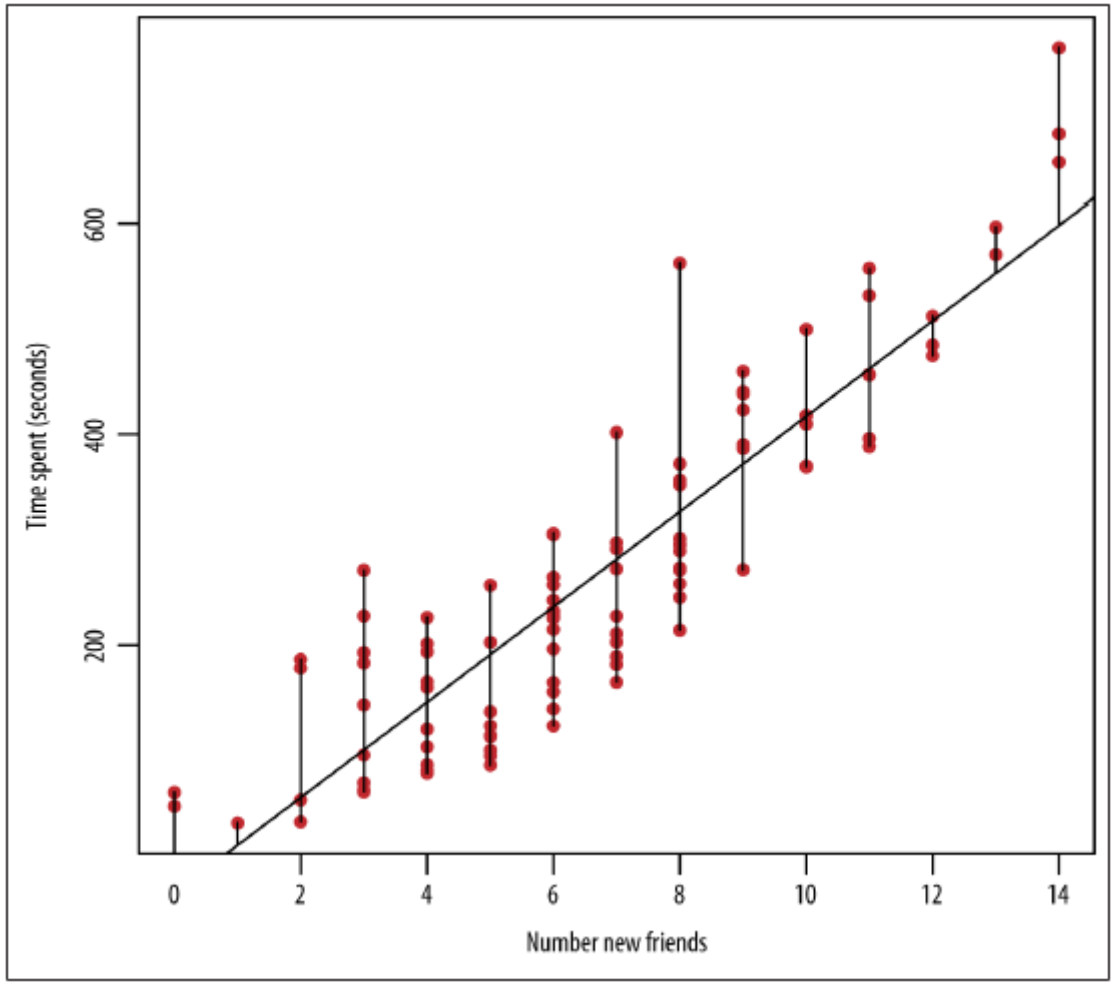
- Now that we have our model. Let's fit our model.

LINEAR REGRESSION





CONFIDENCE



Now that we have established this relationship,

$$\hat{y} = 46x - 32$$

If a new x-value of 5 (user has 5 new friends) came in, how confident are we in the output value of: 46(5) – 32 = 198s?

To increase our confidence, we need to:

i. Add in more predictors
ii. Transform the predictors
iii. Add in modeling assumption about the errors.

K-NEAREST NEIGHBOUR

- K-NN is an algorithm that can be used when you have a bunch of objects that have been classified, and other similar objects that haven't gotten classified, and you want a way to automatically classify them.
- The concept behind k-NN is to consider other items based on their attributes, look at their labels, and give the unassigned item the majority vote. (If there's a tie, you randomly select among the labels that have tied for first)
- 2 decisions are required:
  - How to define closeness/similarity?
  - How many neighbours should we look consider? (This value is *k*)

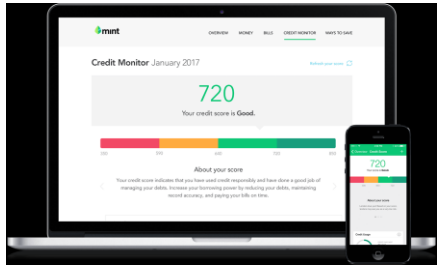
K-NEAREST NEIGHBOUR

- Example with credit score

Age	Income ('000)	Credit
69	3	Low
66	57	Low
49	79	Low
49	17	Low
58	26	High
44	71	High

We can represent these 3-dimensional information on a 2-dimensional plane

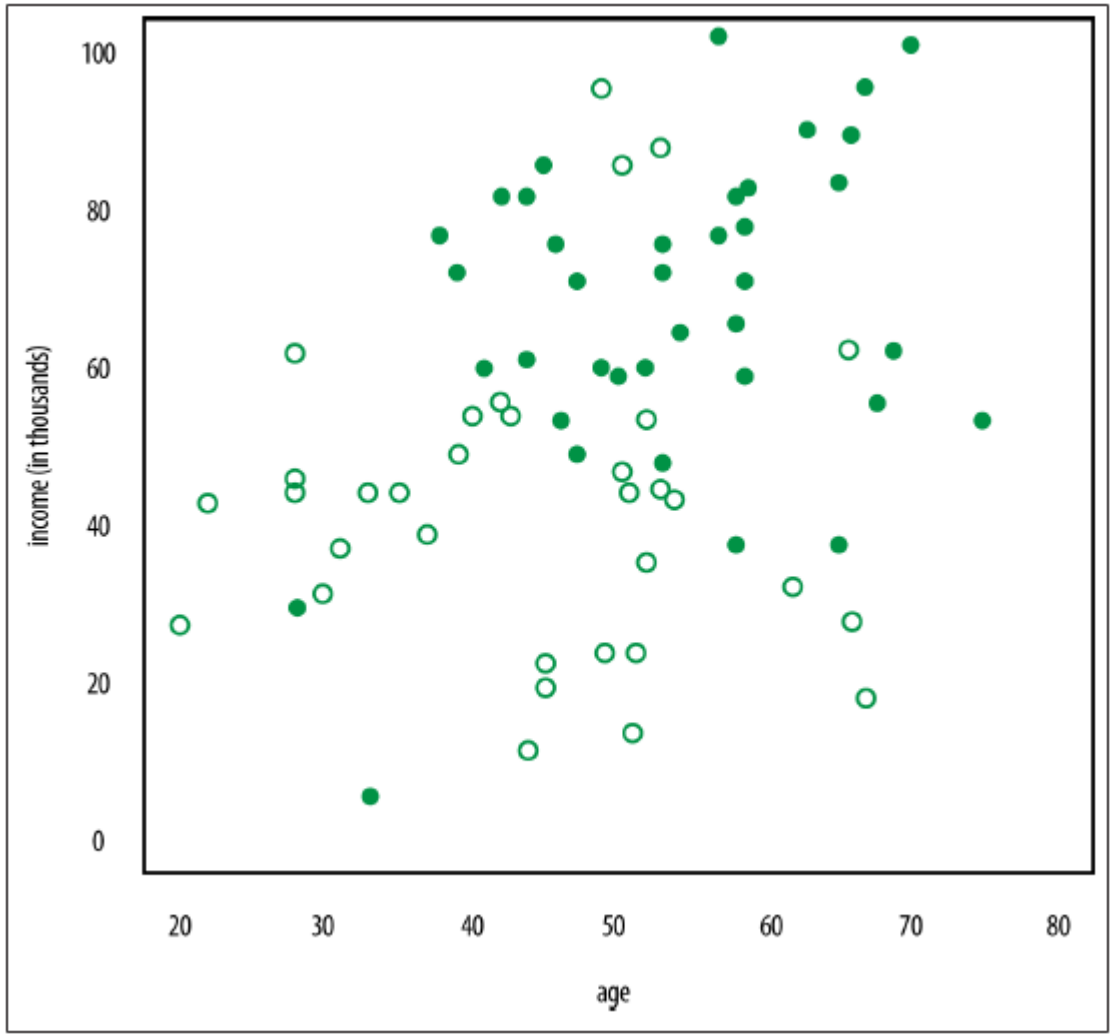
Hint: We can plot people as points on the plane and label them with an empty circle if they have low credit ratings.



K-NEAREST NEIGHBOUR

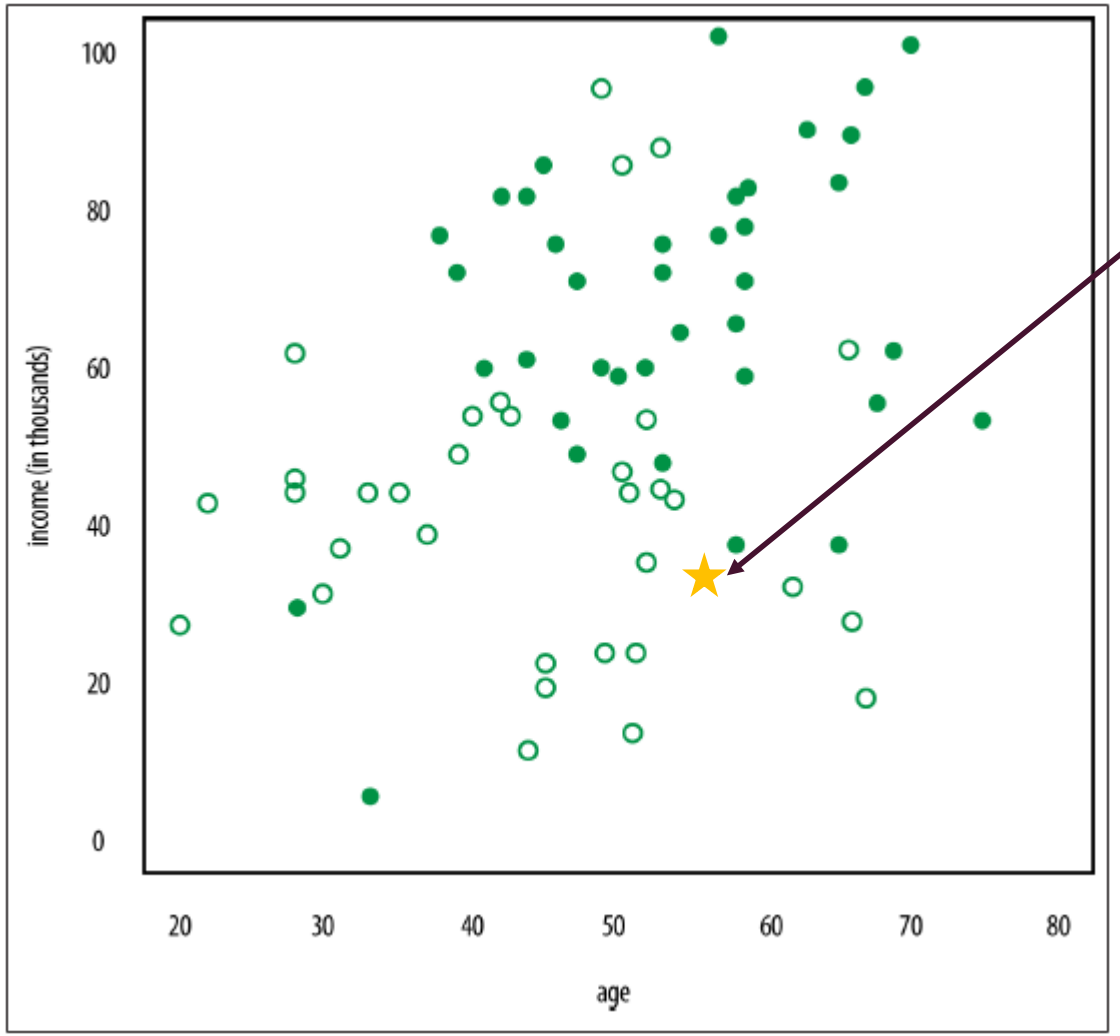
- Example with credit score

Age	Income ('000)	Credit
69	3	Low
66	57	Low
49	79	Low
49	17	Low
58	26	High
44	71	High



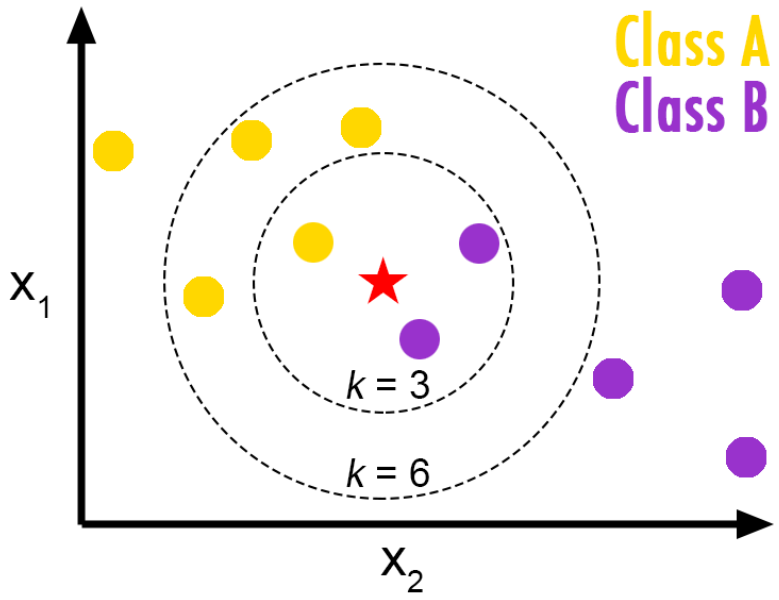
Gorgeous, ain't it?

K-NEAREST NEIGHBOUR

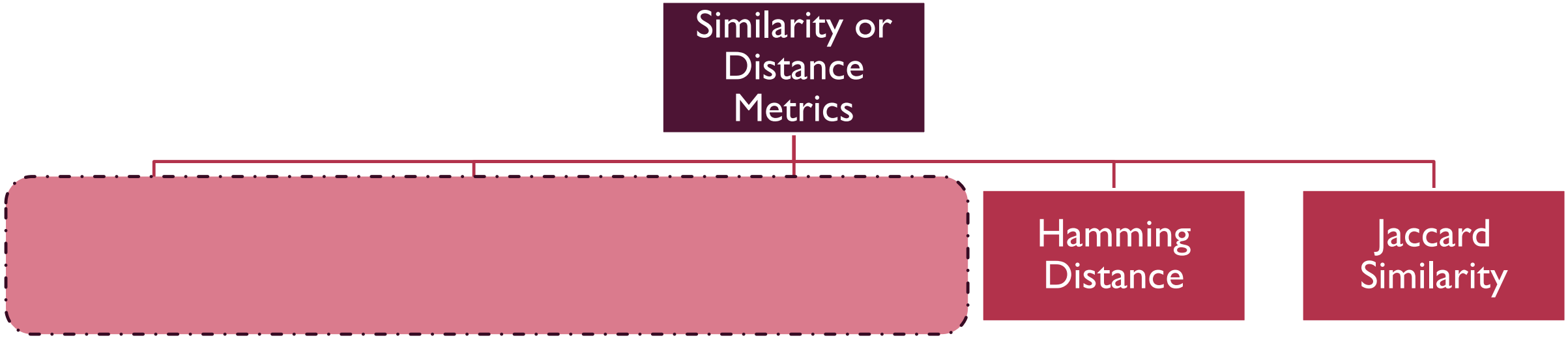


Question: What if there is a new guy who is 57 years old and makes RM37,000? What would his credit rating label most likely be?

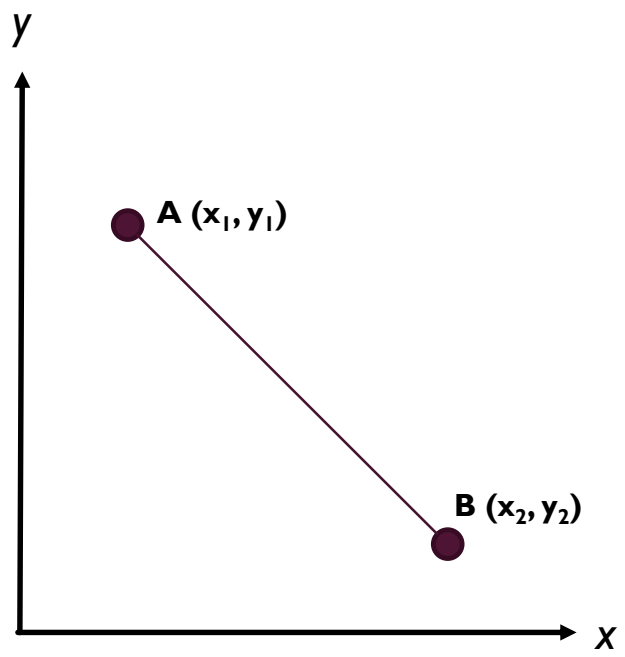
Answer: Measure the **distance** between ***k*-neighbours.** Eg.



SIMILARITY OR DISTANCE METRICS



EUCLIDEAN DISTANCE

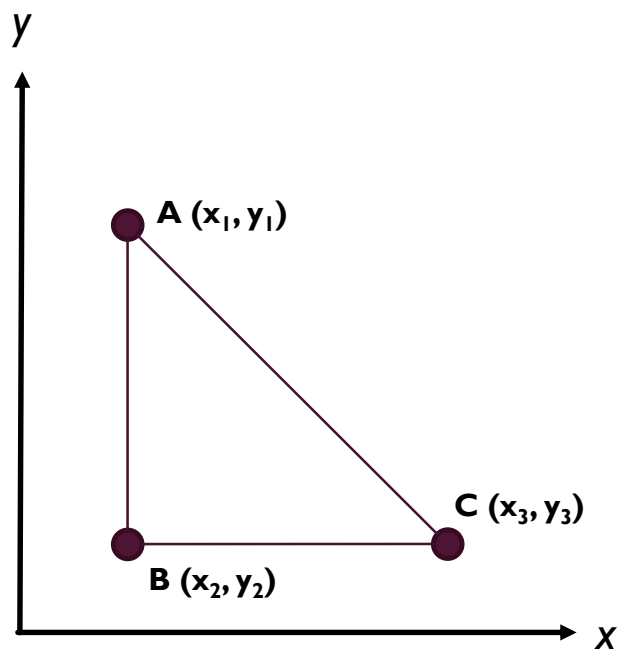


Euclidean distance is the most common use of distance. The Euclidean distance between two points is the length of the path connecting them. The Pythagorean theorem gives this distance between two points.

To calculate the distance between 2 points, we use:

$$d\,(A,B) = \sqrt{\sum_{i=1}^{n}(y_i - x_i)^2}$$

MANHATTAN DISTANCE

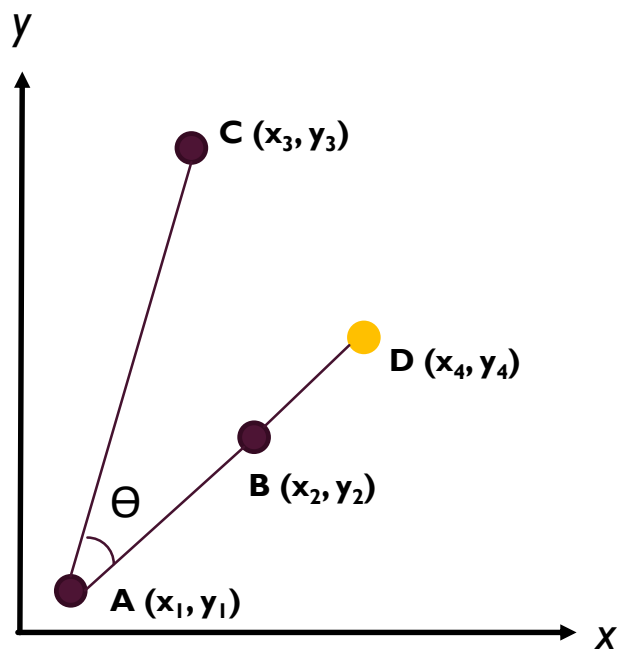


Manhattan distance is a metric in which the distance between two points is the sum of the absolute differences of their Cartesian coordinates. (In a simple way of saying it is the total sum of the difference between the x-coordinates and y-coordinates)

To calculate the distance between 2 points, we use:

$$d\,(A, C) = \sum_{i=1}^{n} |x_i - y_i|$$

COSINE SIMILARITY



Cosine similarity metric searches for the normalized dot product of two attributes. It is thus a judgement of orientation and not magnitude: Two vectors with the same orientation have a cosine similarity of 1, two vectors at 90° have a similarity of 0.

To measure the similarity between 2 points, we use:

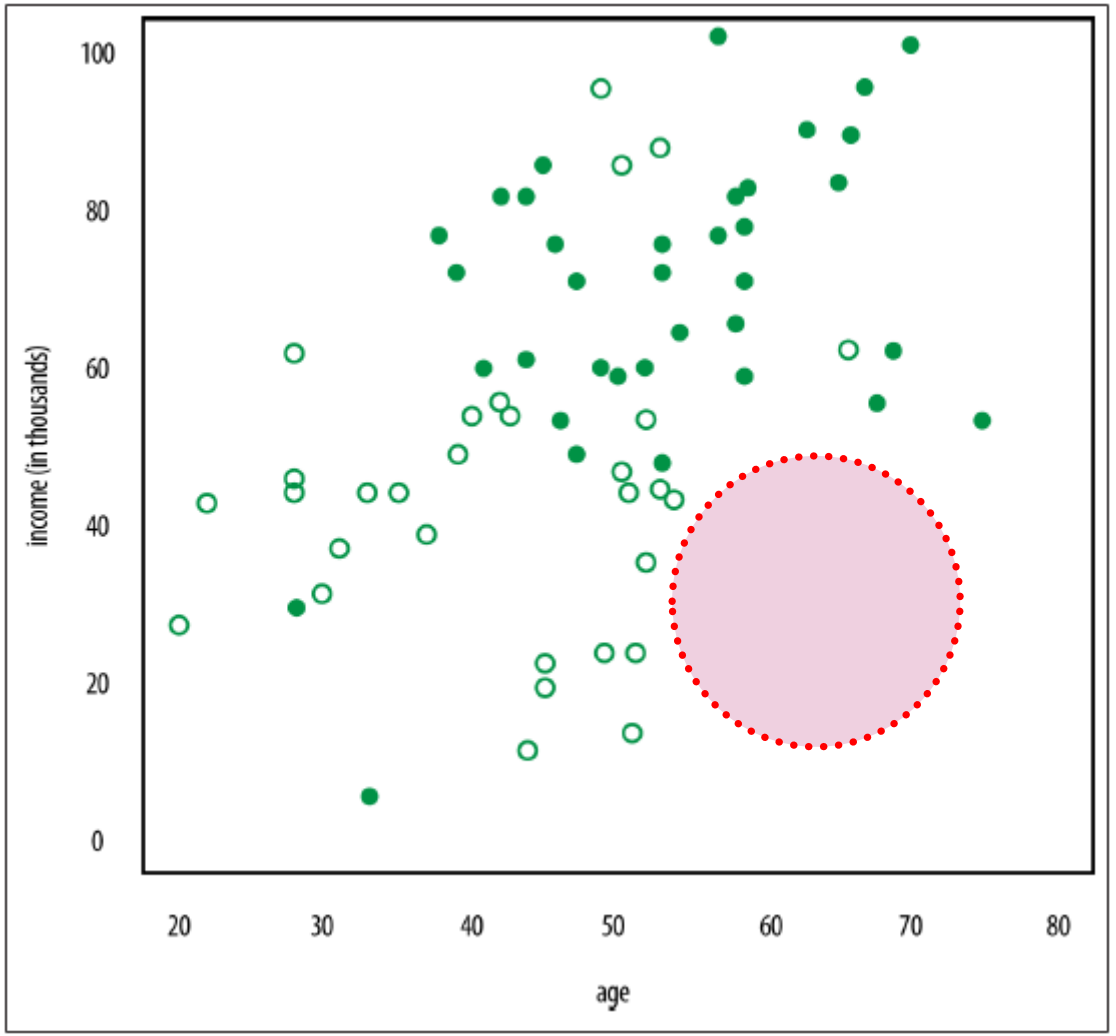
$$sim\ (A, B) = \cos(\Theta) = \frac{A.B}{\|A\|\|B\|}$$

COMING BACK TO K-NEAREST NEIGHBOUR

- Since we don't know which k-value to choose, we carry out multiple experiments with different k-values each time and say we get the following output in the form (k, misclassification rate)

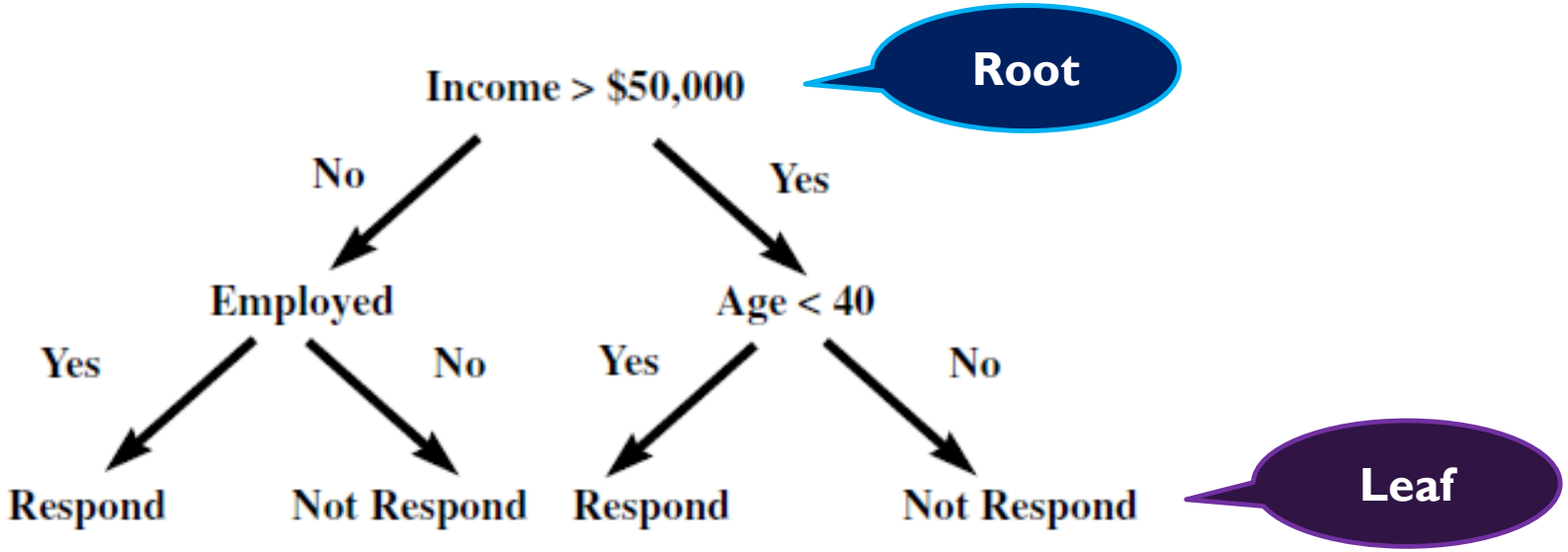
```
k, misclassification rate
1, 0.28
2, 0.315
3, 0.26
4, 0.255
5, 0.24
6, 0.26
7, 0.25
8, 0.25
9, 0.235
10, 0.24
```

In this dataset, it seems that the best k-value is 5. Note that the value for k may be different across various datasets.



DECISION TREE

- Decision trees are recursive partitioning algorithms (RPAs) that come up with a tree-like structure representing patterns in an underlying data set.
- The **top node** is the **root node** specifying a testing condition of which the outcome corresponds to a branch leading up to an internal node. The **terminal nodes** of the tree assign the classifications and are also referred to as the **leaf nodes**.

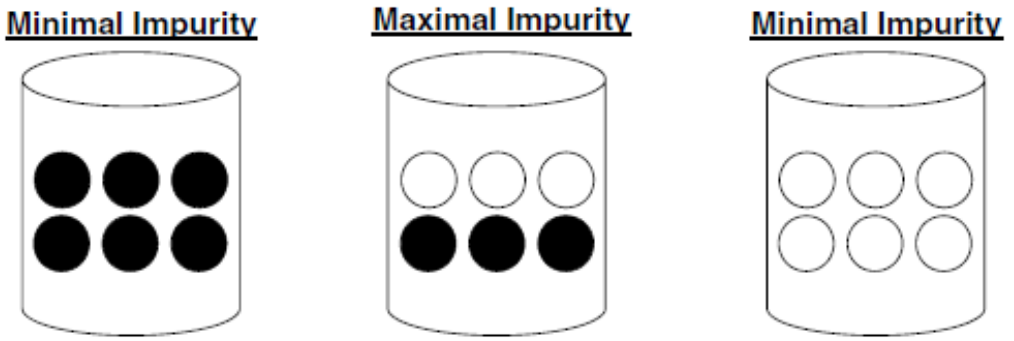


DECISION TREE – KEY DECISIONS TO BUILD A TREE

- Key decisions to consider when building a tree:
  i. Splitting Decision: Which variable to split at what value (e.g., age < 30 or not, income < 1,000 or not; marital status = married or not)
  ii. Stopping Decision: When to stop growing a tree?
  iii. Assignment Decision: What class (e.g., good or bad customer) to assign to a leaf node?

SPLITTING DECISION - ENTROPY

- Concept of Impurity vs Purity. Consider the following:



- Each contains white and black circles. Minimal impurity occurs when circles are either black or white. Maximal impurity occurs when the number of black and white circles are equivalent.
- Decision tree aims to minimize the impurity in the data (to get the purest leaf)
- To achieve that, we have the following measures:
  a. Entropy Gain: $E(S) = -p_G \log_2(p_G) - p_B \log_2(p_B)$
  b. Gini: $Gini(S) = 2p_G p_B$
  c. Chi-Squared Analysis

where p_G (p_B) being the proportions of good and bad respectively

EXAMPLE – CALCULATING ENTROPY

Entropy: A common way to measure impurity

Entropy: $E(S) = -p_G \log_2(p_G) - p_B \log_2(p_B)$

where

p_g, p_b is the probability of class g and b respectively

16/30 are **Green** circles; 14/30 are **Pink** crosses

Therefore:

$\log_2(p_g) = \log_2(16/30) = -.9$

$\log_2(p_b) = \log_2(14/30) = -1.1$

Entropy = -(16/30)(-.9) – (14/30)(-1.1) = .99

***Note**: Entropy comes from information theory. The higher the entropy the more the information content. (Good Training Set)*

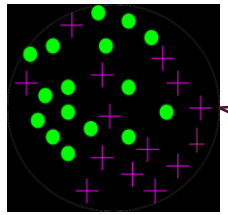
INFORMATION GAIN

- We want to determine which attribute is most useful for discriminating between the classes to be learned.
- Information Gain tells us exactly that! We will use it to decide the ordering attributes in the nodes of a decision tree.
- We will use it to determine the ordering of attributes in the nodes of a decision tree

INFORMATION GAIN

Information Gain = entropy(parent) – [average entropy(children)]

Entire Population: 30



Parent Entropy

$$= -\left(\frac{14}{30} log_2 \frac{14}{30}\right) - \left(\frac{16}{30} log_2 \frac{16}{30}\right) = 0.996$$

(Weighted) Average Entropy of Children = $\left(\frac{17}{30} \times 0.787\right) - \left(\frac{13}{30} \times 0.391\right) = 0.615$

Information Gain = 0.996 - 0.615 = 0.38 for this split

EXAMPLE

Training Set: 3 features and 2 classes

X	Y	Z	Class
I	I	I	A
I	I	0	A
0	0	I	B
I	0	0	B

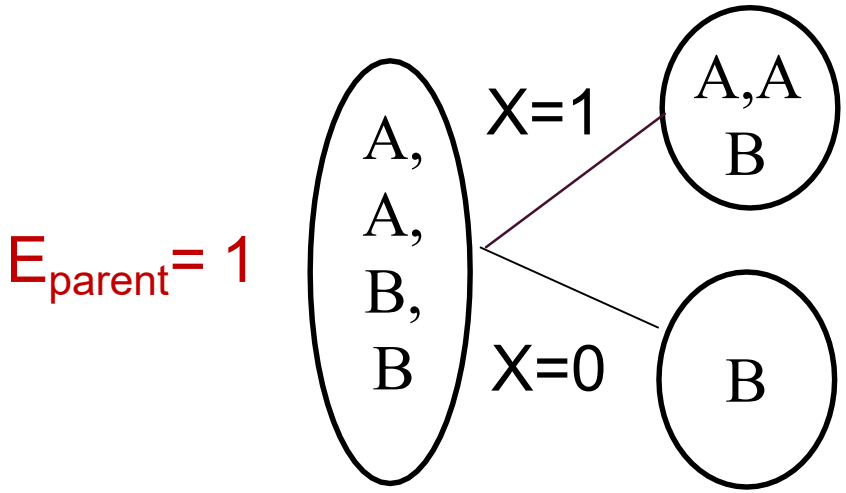
How do we distinguish class A from class B?

EXAMPLE

X	Y	Z	Class
I	I	I	A
I	I	0	A
0	0	I	B
I	0	0	B

Split on attribute X

If X is the best attribute, this node will be split further



$E_{child1} = -(1/3)\log_2(1/3) - (2/3)\log_2(2/3)$
$= .5284 + .39$
$= .9184$

$E_{child2} = 0$

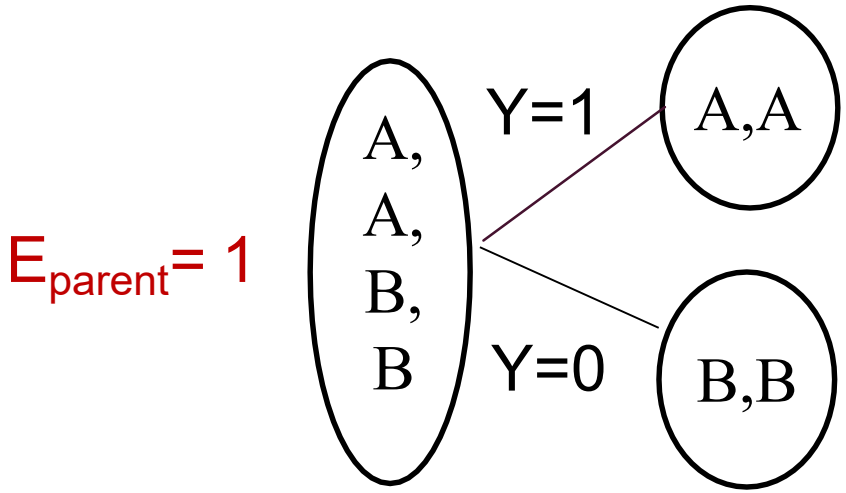
Therefore,
Gain = 1 – (3/4)(.9184) – (1/4)(0) = .3112

*If X is the best attribute, this node will be split further

EXAMPLE

X	Y	Z	Class
I	I	I	A
I	I	0	A
0	0	I	B
I	0	0	B

Split on attribute Y



$E_{parent} = 1$

$E_{child1} = 0$

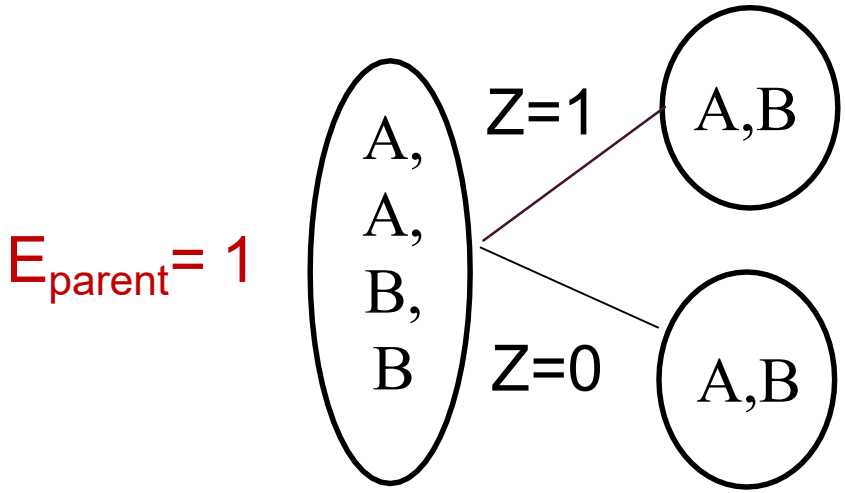
$E_{child2} = 0$

Therefore,
Gain = 1 –(1/2) 0 – (1/2)0 = 1; BEST!

EXAMPLE

X	Y	Z	Class
I	I	I	A
I	I	0	A
0	0	I	B
I	0	0	B

Split on attribute Z



$E_{parent} = 1$

$E_{child1} = 1$

$E_{child2} = 1$

Therefore,
Gain = 1 – (1/2)(1) – (1/2)(1) = 0; (No Gain, Worst)

(NAÏVE) BAYES CLASSIFICATION

- Classification techniques based on Bayes' Theorem with an assumption of independence among predictors.
- Assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. For example

If a fruit is **red**, **round**, and measures about 7cm in diameter, it's an Apple.

Am I an Apple then? Isn't Bayes a bit Naive?

(NAÏVE) BAYES CLASSIFICATION

- Naive Bayes model is easy to build and useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Likelihood Class prior probability

$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

Posterior Probability Predictor Prior Probability

- P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).
- P(c) is the prior probability of class.
- P(x|c) is the likelihood which is the probability of predictor given class.
- P(x) is the prior probability of predictor.

$$P(c \mid x) = P(x_1 \mid c) \times P(x_2 \mid c) \times \ldots \times P(x_n \mid c) \times P(c)$$

(NAÏVE) BAYES CLASSIFICATION - EXAMPLE

Weather	Play
Sunny	No
Overcast	Yes
Rainy	Yes
Sunny	Yes
Sunny	Yes
Overcast	Yes
Rainy	No
Rainy	No
Sunny	Yes
Rainy	Yes
Sunny	No
Overcast	Yes
Overcast	Yes
Rainy	No

Frequency Table		
Weather	No	Yes
Overcast		4
Rainy	3	2
Sunny	2	3
Grand Total	5	9

Likelihood table				
Weather	No	Yes		
Overcast		4	=4/14	0.29
Rainy	3	2	=5/14	0.36
Sunny	2	3	=5/14	0.36
All	5	9		
	=5/14	=9/14		
	0.36	0.64		

Problem: Players will play if weather is sunny. Is this statement is correct?

Step 1: Convert the data set into a **frequency table**

Step 2: Create **Likelihood table** by finding the probabilities like Overcast probability = 0.29 and probability of playing is 0.64.

Step 3: Now, use Naive Bayesian equation to calculate the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

(NAÏVE) BAYES CLASSIFICATION - SOLUTION

Problem: If the weather is sunny, is the game on?

P (Yes | Sunny) = P(Sunny | Yes) * P(Yes) / P (Sunny)

P (Sunny | Yes) = 3/9 = 0.33,
P (Sunny) = 5/14 = 0.36,
P (Yes) = 9/14 = 0.64

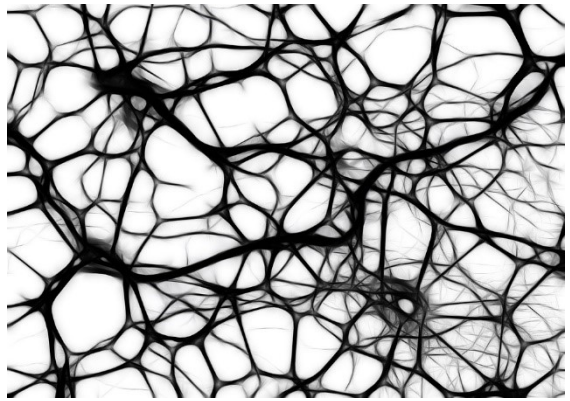
P (Yes | Sunny) = 0.33 * 0.64 / 0.36 = 0.60, which has higher probability.

Note: Naive Bayes uses a similar method to predict the probability of different class based on various attributes. This algorithm is mostly used in text classification and with problems having multiple classes.

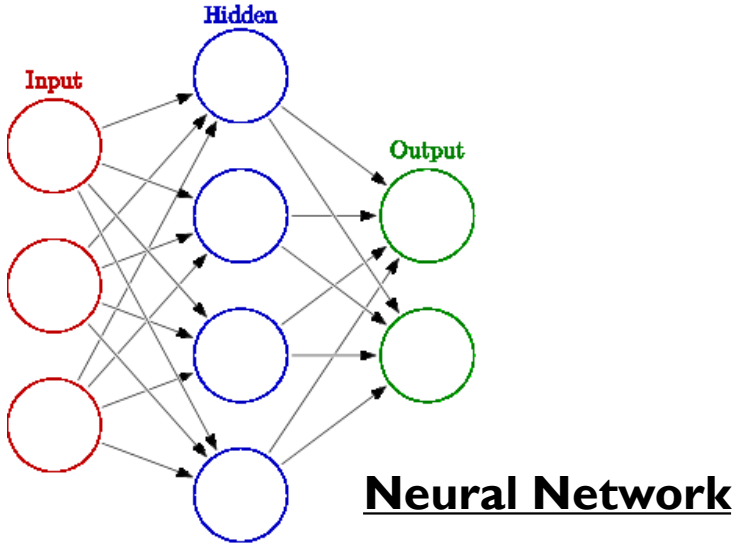
ARTIFICIAL NEURAL NETWORK (ANN)

- An Artificial Neural Network (ANN) is a predictive model motivated by the way the brain operates.
- Think of the brain as a collection of neurons wired together.
- Each neuron looks at the outputs of the other neurons that feed into it, does a calculation, and then either fires (if the calculation exceeds some threshold) or doesn't (if it doesn't).

Brain

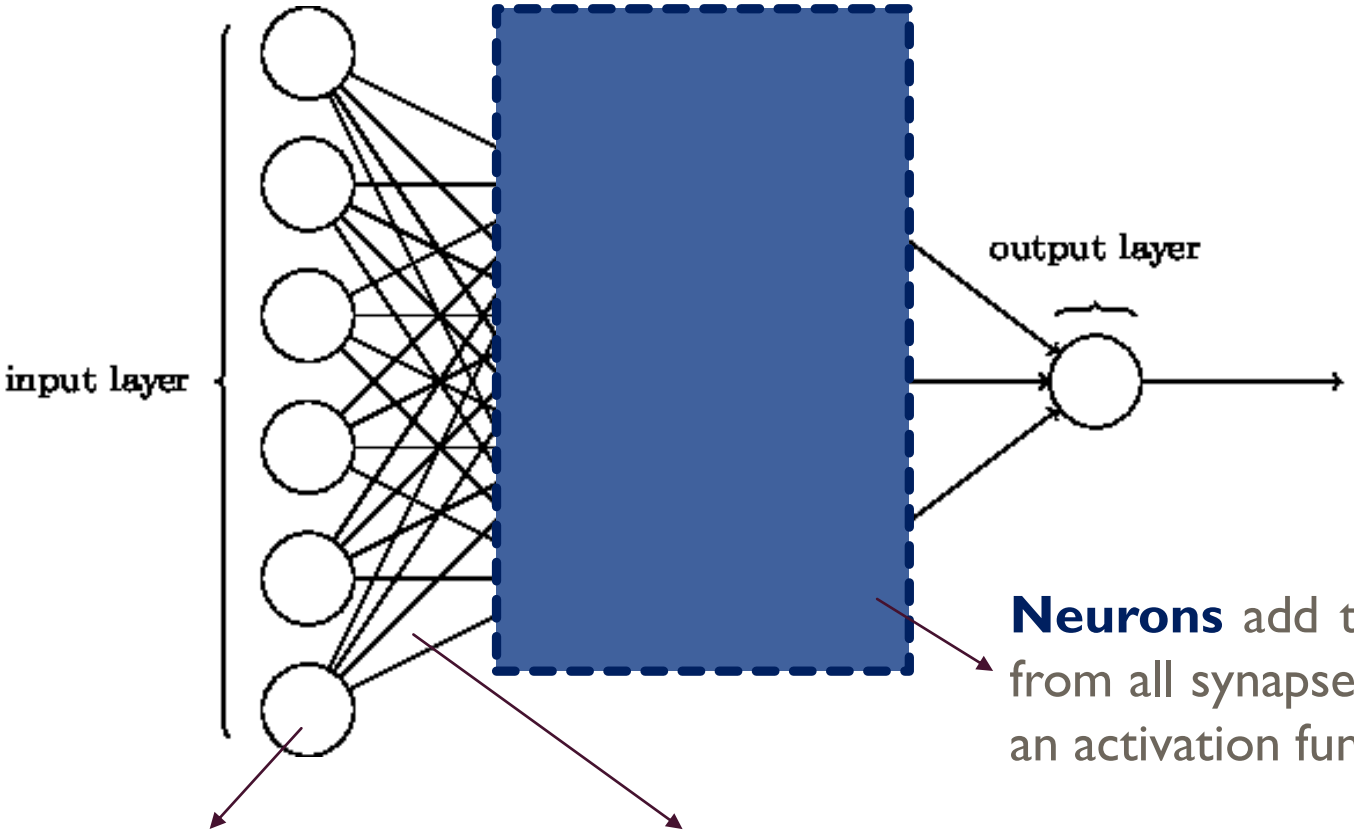


Neurons



Neural Network

ARTIFICIAL NEURAL NETWORK (ANN)



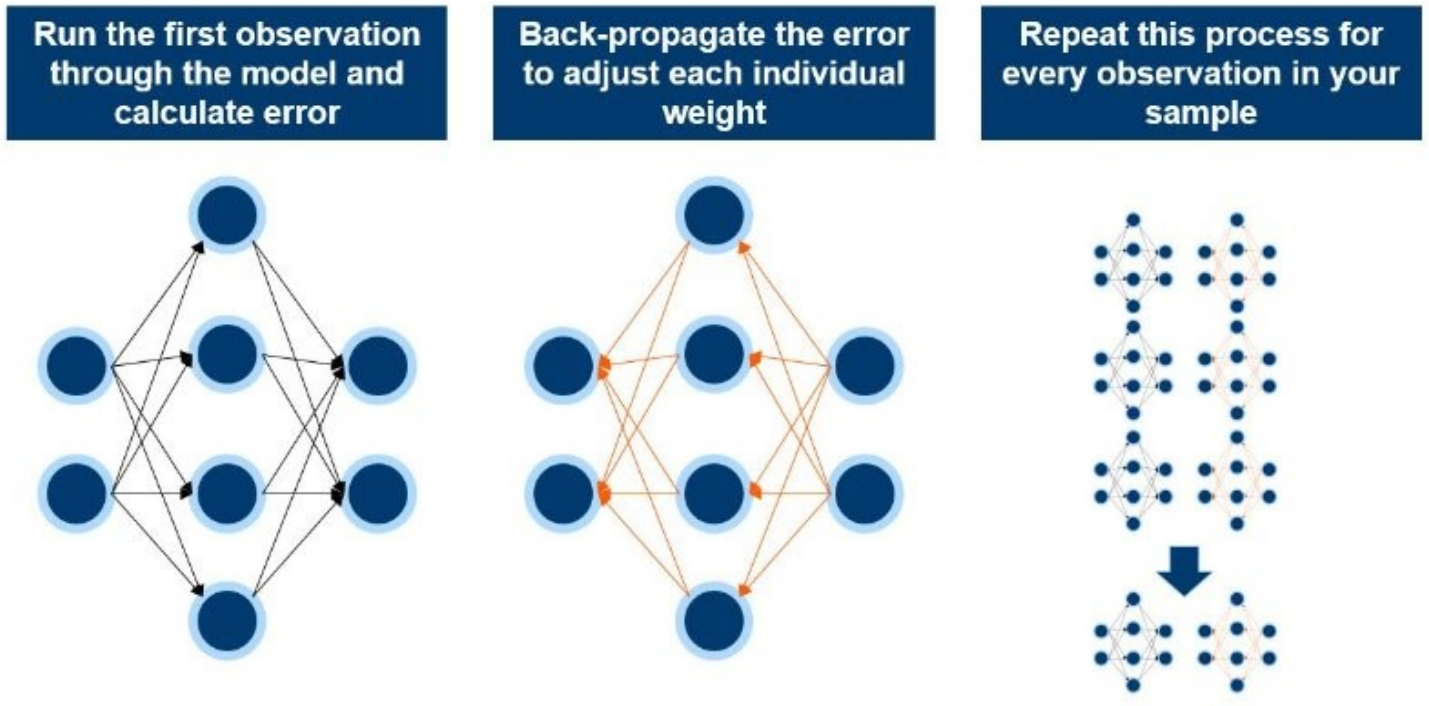
- Hidden layers: ANN uses it to make sense of something complicated.
- The term "deep" learning came from having many hidden layers.
- These layers are known as "hidden", since they are not visible as a network output.

Neurons add the outputs from all synapses and apply an activation function.

The **Circles** represent neurons

The **Lines** represent synapses. Synapses take the input and multiply it by a "weight"

ARTIFICIAL NEURAL NETWORK (ANN)



- Training a neural network :

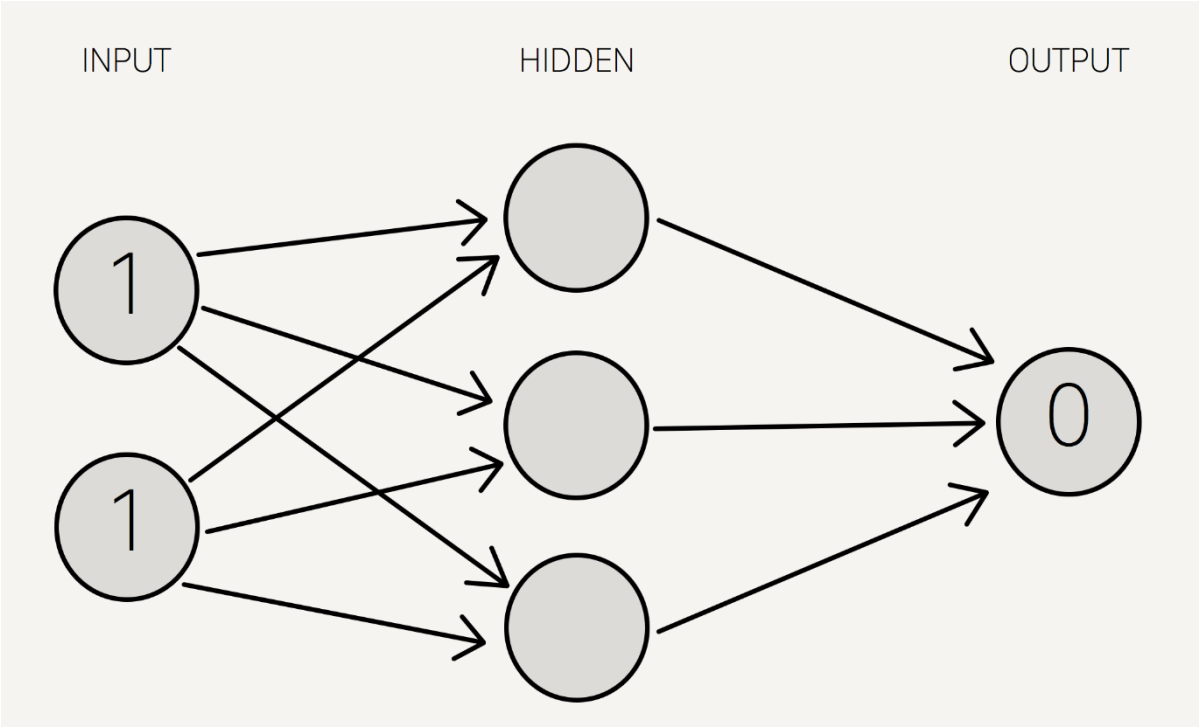
i. Forward Propagation - Apply a set of weights to the input data and calculate an output. For the first forward propagation, the set of weights is selected randomly.

ii. Backward Propagation - Measure the margin of error of the output and adjust the weights accordingly to decrease the error.

- ANN repeats both forward and back propagation until the weights are calibrated to accurately predict an output.

ANN - CALCULATION

We use (1,1) => 0 to demonstrate forward propagation

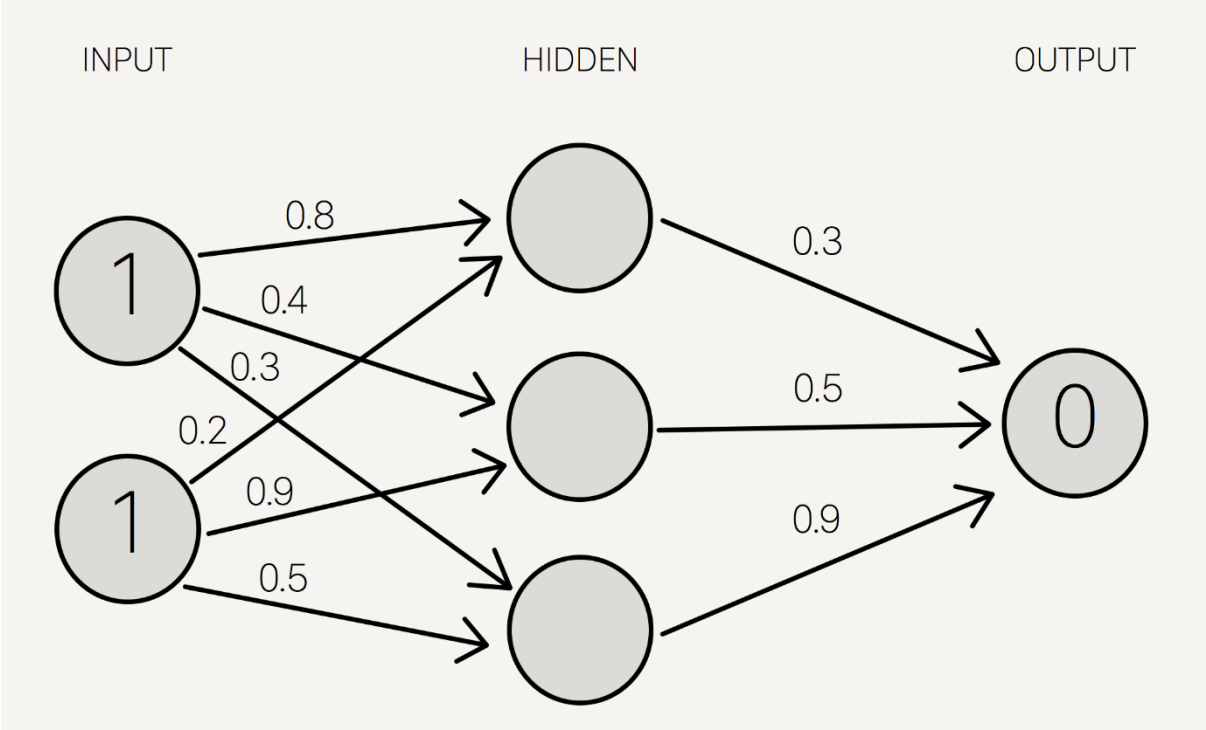


Simple Dataset

Input	Output
0, 0	0
0, 1	1
1, 0	1

ANN - CALCULATION

1. Assign weights to all synapses



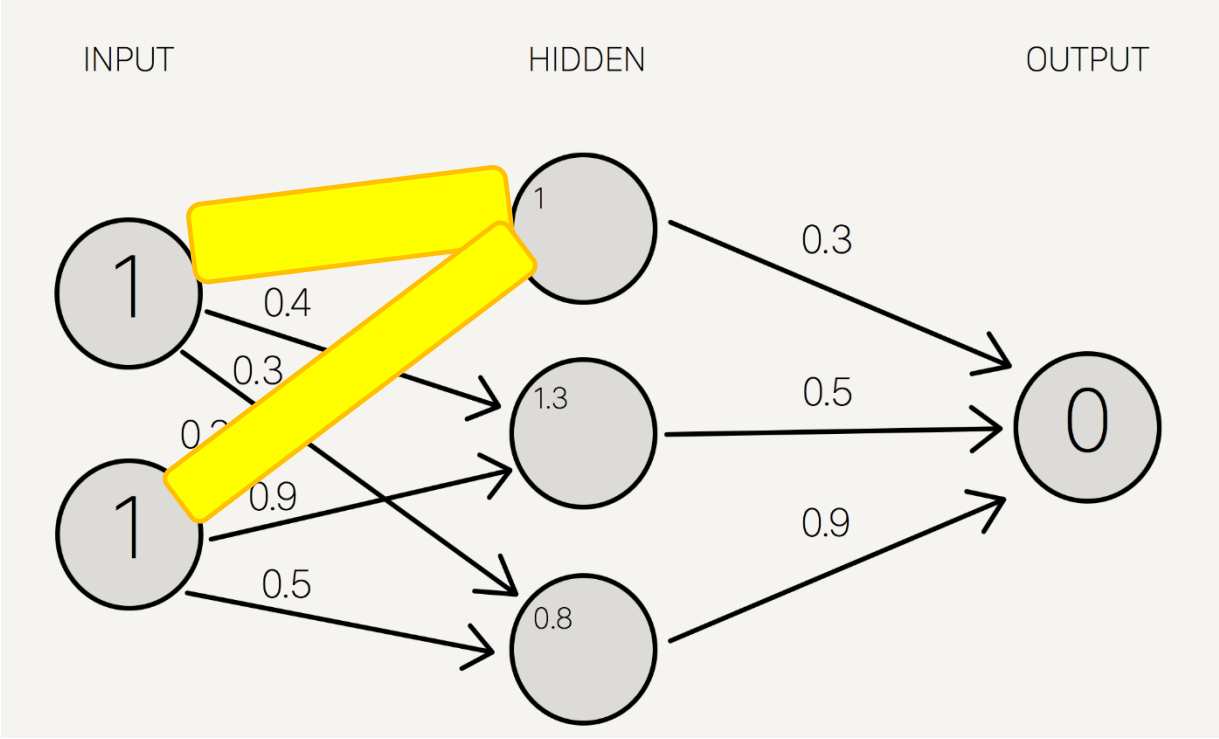
Note: Weights are selected randomly (based on Gaussian distribution).

Since it is the first iteration, the initial weights will be between 0 and 1.

However, final weights don't need to be.

ANN - CALCULATION

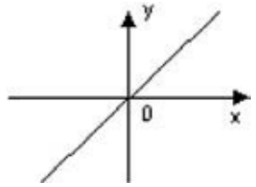
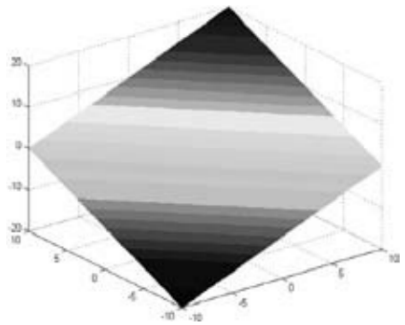
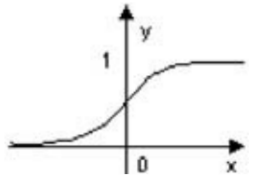
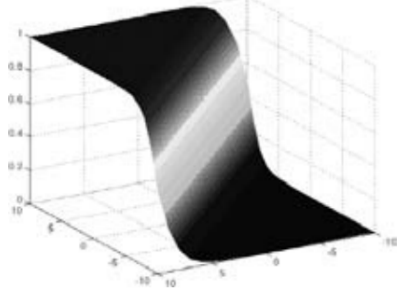
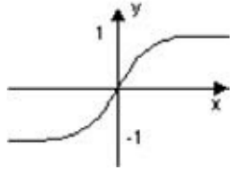
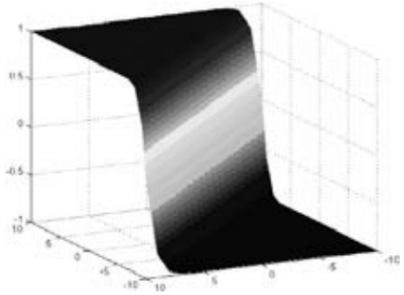
2. Sum products of inputs with respective weight.



(1*0.8) + (1*0.2) = 1
(1*0.4) + (1*0.9) = 1.3
(1*0.3) + (1*0.5) = 0.8

To get the final value, apply an activation function to the hidden layer sums

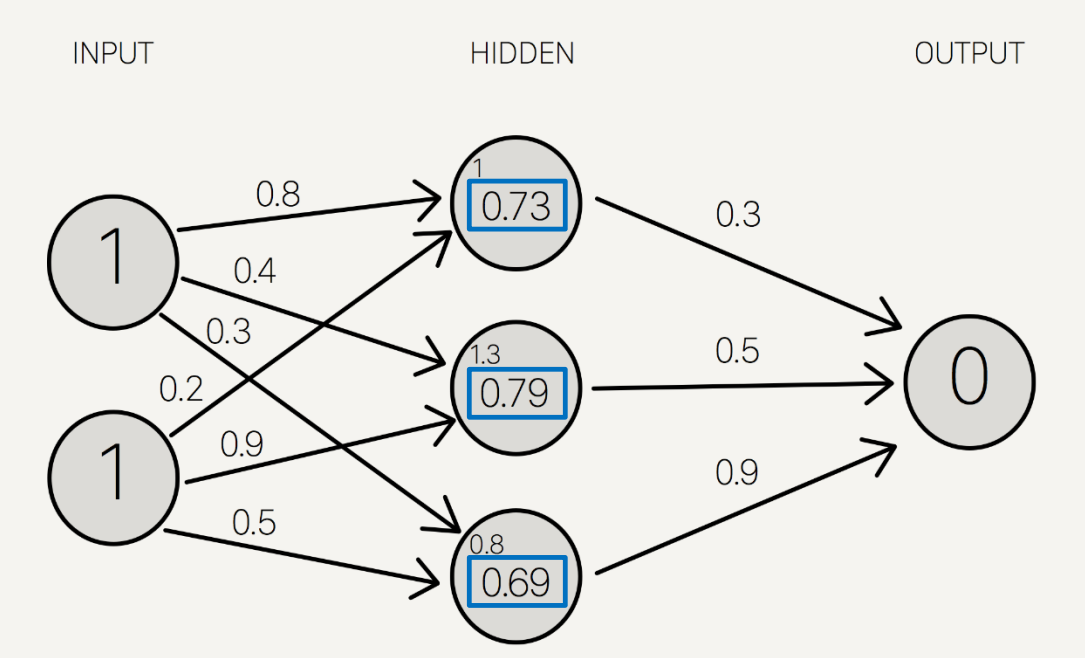
TYPES OF ACTIVATION FUNCTION

Activation Function	Mathematical Equation	2D Graphical Representation	3D Graphical Representation
Linear	$y = x$		
Sigmoid (logistic)	$y = \frac{1}{1 + e^{-x}}$		
Hyperbolic tangent	$y = \frac{1 - e^{-2x}}{1 + e^{2x}}$		

Let's choose sigmoid

ANN - CALCULATION

3. Apply S(x) to the three hidden layer sum.

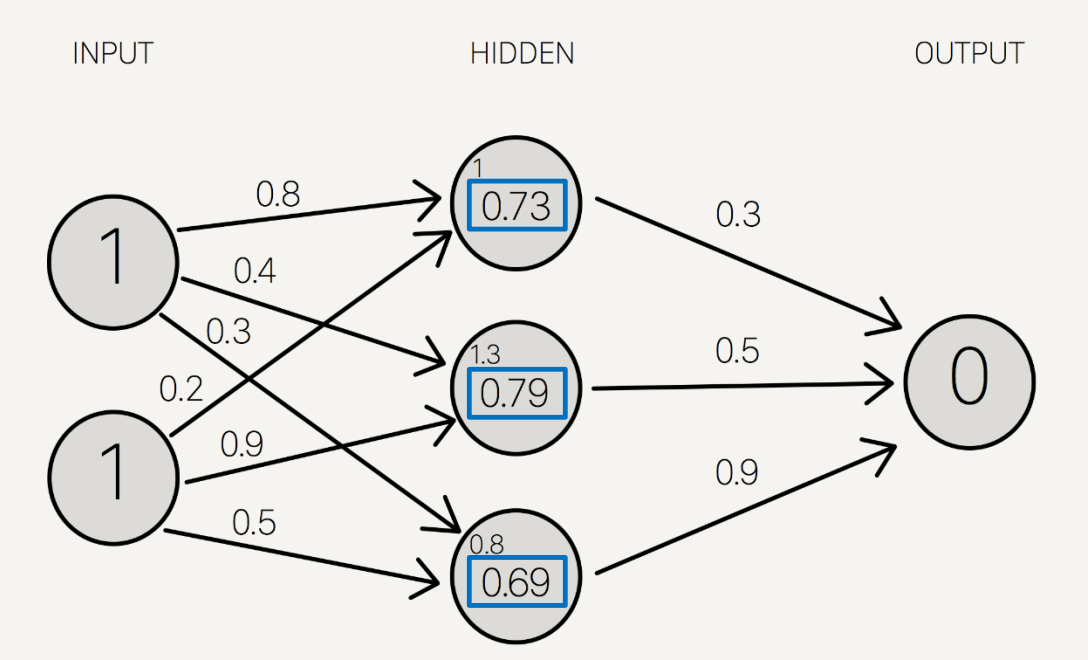


S(1.0) = 0.7311
S(1.3) = 0.7858
S(0.8) = 0.6900

Then, sum the product of the hidden layer results with the second set of weights (also determined at random the first time around) to determine the output sum.

ANN - CALCULATION

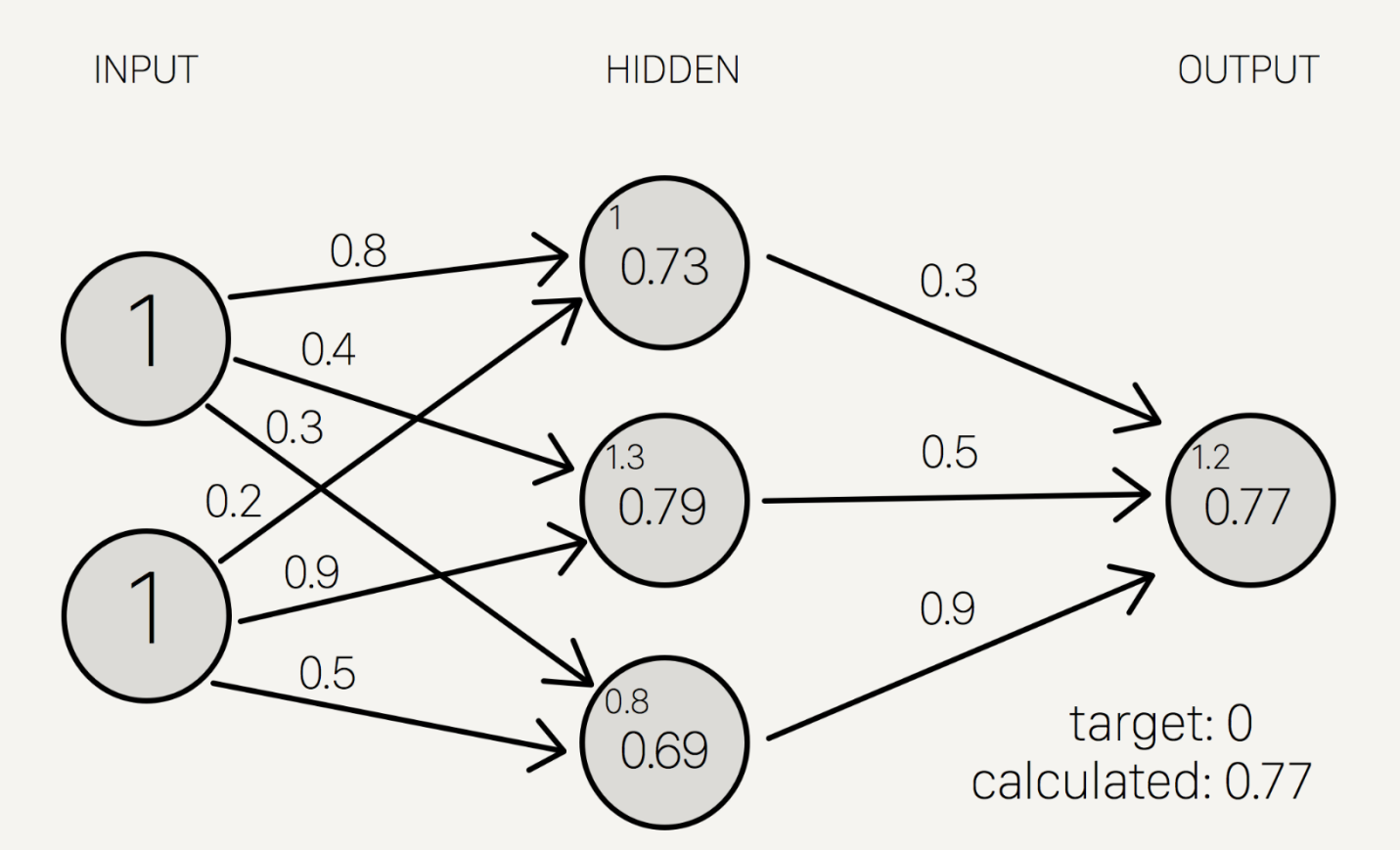
4. Calculate sum of product of hidden layers and apply activation function



Output sum
= (0.73 * 0.3) + (0.79 * 0.5) + (0.69 * 0.9)
= 1.235

S(1.235) = 0.7747

COMPLETE DIAGRAM (FOR NOW)



Question 1: Why is the calculated output off the target mark?

Answer: Because our weights were initialized randomly.

Action: We need to adjust the weight via back propagation.

BACK PROPAGATION – ADJUST THE WEIGHTS

Calculating the incremental change to these weights happens in two steps:

1) we find the margin of error of the output result to back out the necessary change in the output sum (delta output sum)

2) Extract the change in weights by multiplying delta output sum by the hidden layer results.

Output sum margin of error = target – calculated

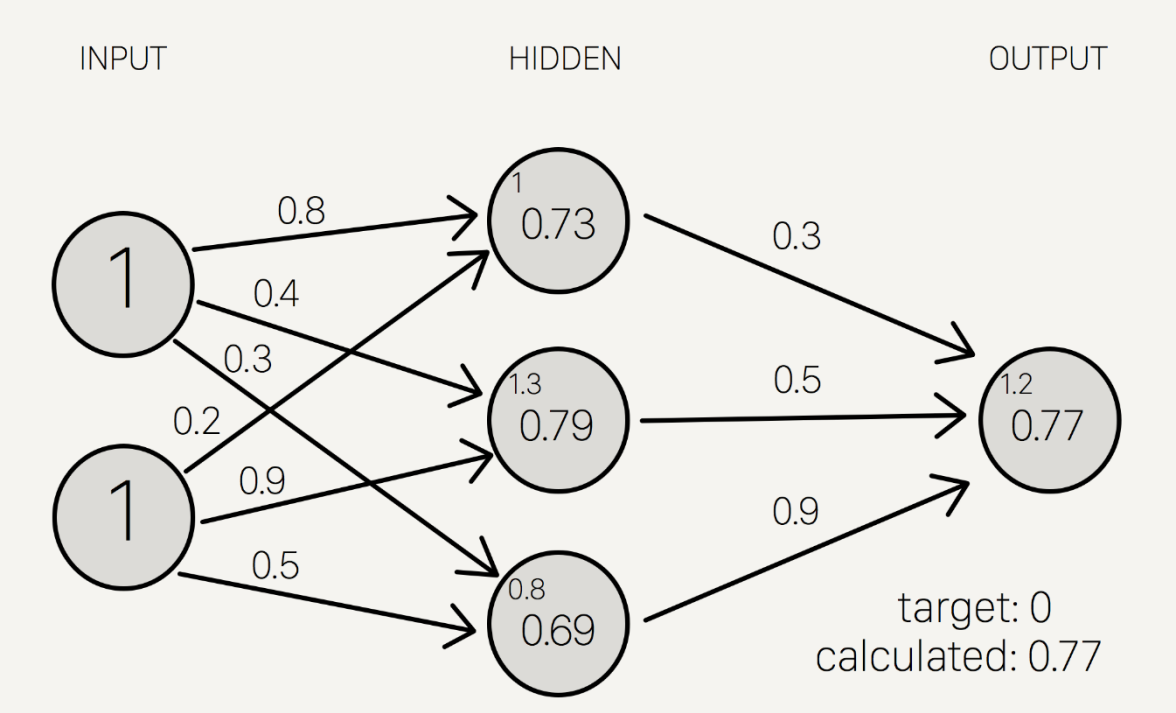
BACK PROPAGATION – ADJUST THE WEIGHTS

Output Sum of Margin Error = Target – Calculated
= 0 – 0.77
= -0.77

Calculate: Delta Output Sum = Derivative of the activation (sigmoid) function and apply to the output sum

Delta output sum = S'(output sum) * (output sum margin error)
Delta output sum = S'(1.235) * (-0.77)
Delta output sum = -0.1344 (proposed change)

BACK PROPAGATION



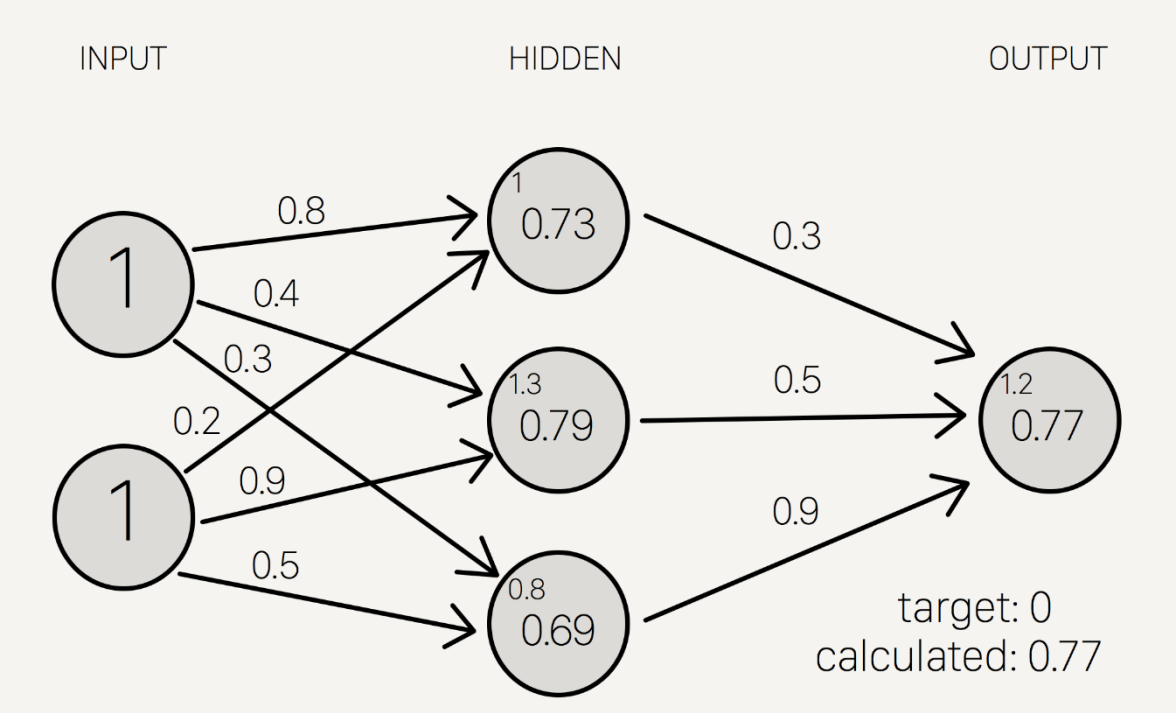
hidden result 1 = 0.73
hidden result 2 = 0.79
hidden result 3 = 0.69

Delta weights = delta output sum * hidden layer results
Delta weights = -0.1344 * [0.73, 0.79, 0.69]
Delta weights = [-0.0983, -0.1056, -0.0941]

Hence

old w7 = 0.3	->	new w7 = 0.202
old w8 = 0.5	->	new w8 = 0.394
old w9 = 0.9	->	new w9 = 0.806

BACK PROPAGATION



To determine the change in weight between input and hidden layer, we perform similar calculations.

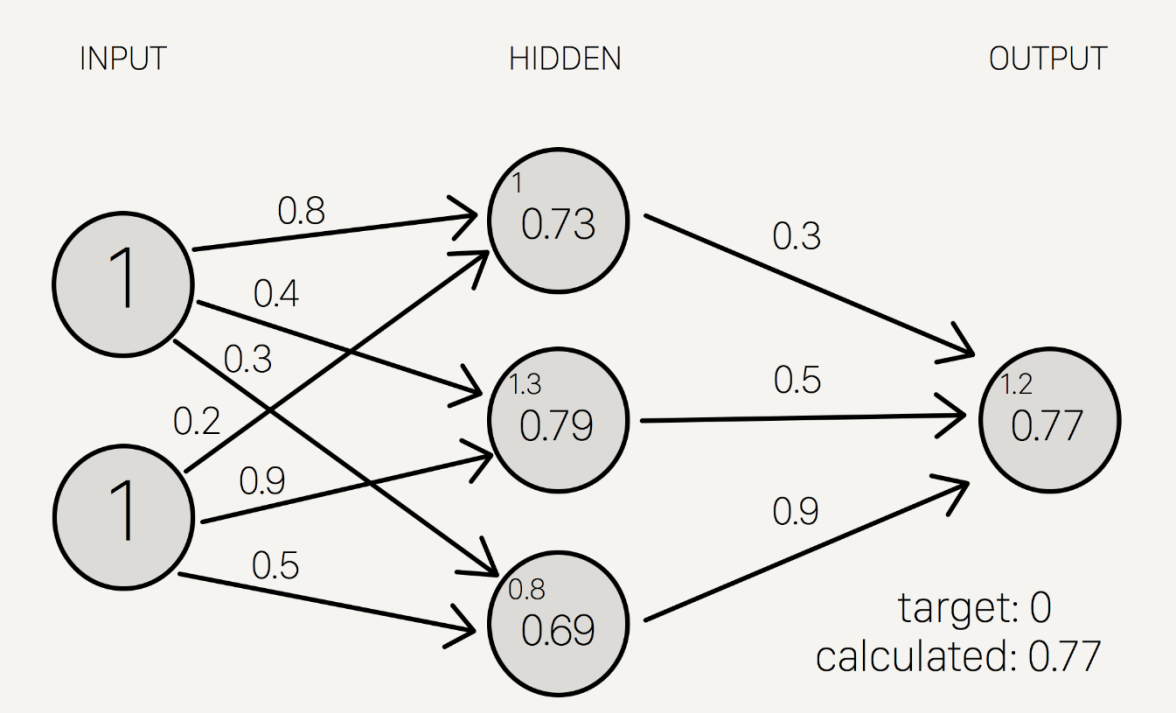
Delta hidden sum = delta output sum * hidden-to-outer weights * S'(hidden sum)

Delta hidden sum = -0.1344 * [0.3, 0.5, 0.9] * S'([1, 1.3, 0.8])

Delta hidden sum = [-0.0403, -0.0672, -0.1209] * [0.1966, 0.1683, 0.2139]

Delta hidden sum = [-0.0079, -0.0113, -0.0259]

BACK PROPAGATION



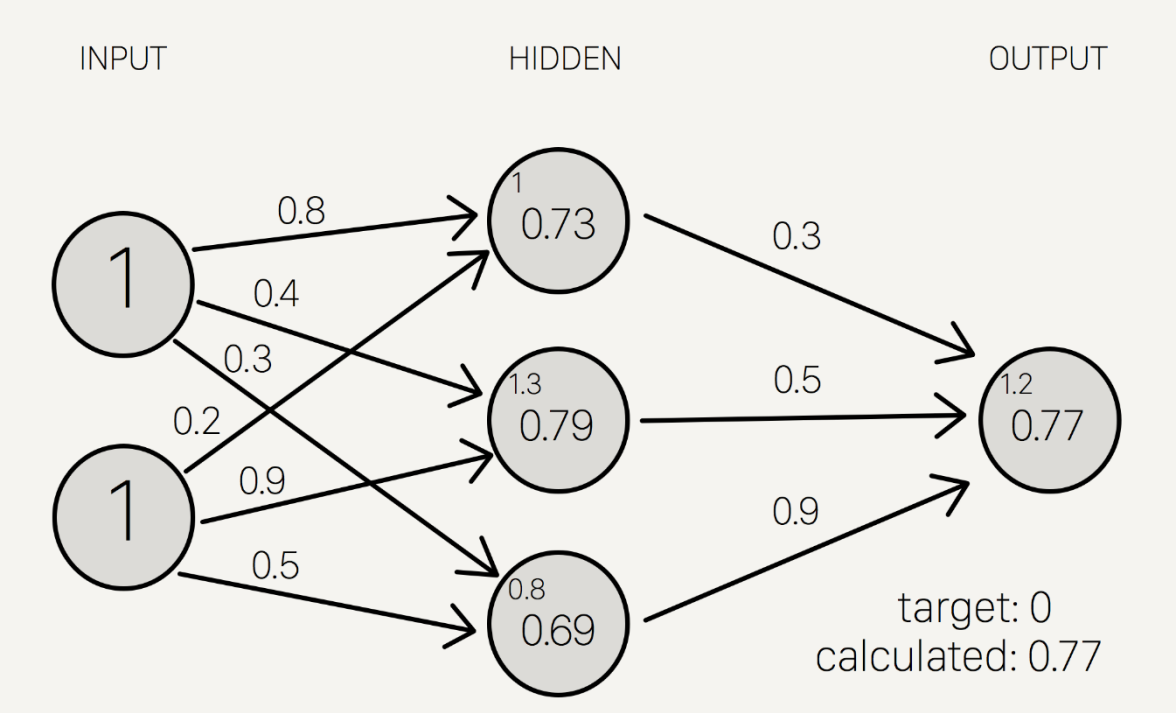
Then multiply the results with the input data.

input 1 = 1
input 2 = 1

Delta weights = delta hidden sum * input
Delta weights = [-0.0079, -0.0113, -0.0259] * [1, 1]
Delta weights = [-0.0079, -0.0113, -0.0259, -0.0079, -0.0113, -0.0259]

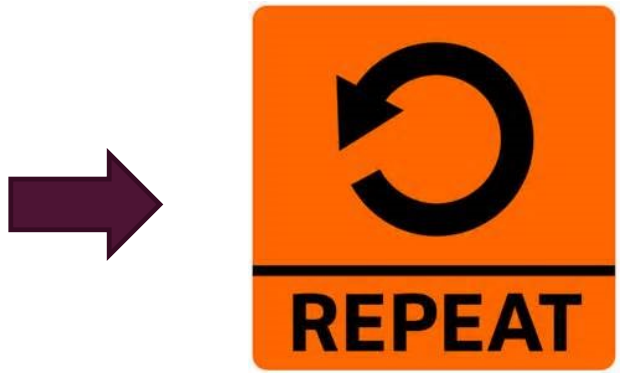
old w1 = 0.8	>	new w1 = 0.7921
old w2 = 0.4	>	new w2 = 0.3887
old w3 = 0.3	>	new w3 = 0.2741
old w4 = 0.2	>	new w4 = 0.1921
old w5 = 0.9	>	new w5 = 0.8887
old w6 = 0.5	>	new w6 = 0.4741

BACK PROPAGATION



New weights.

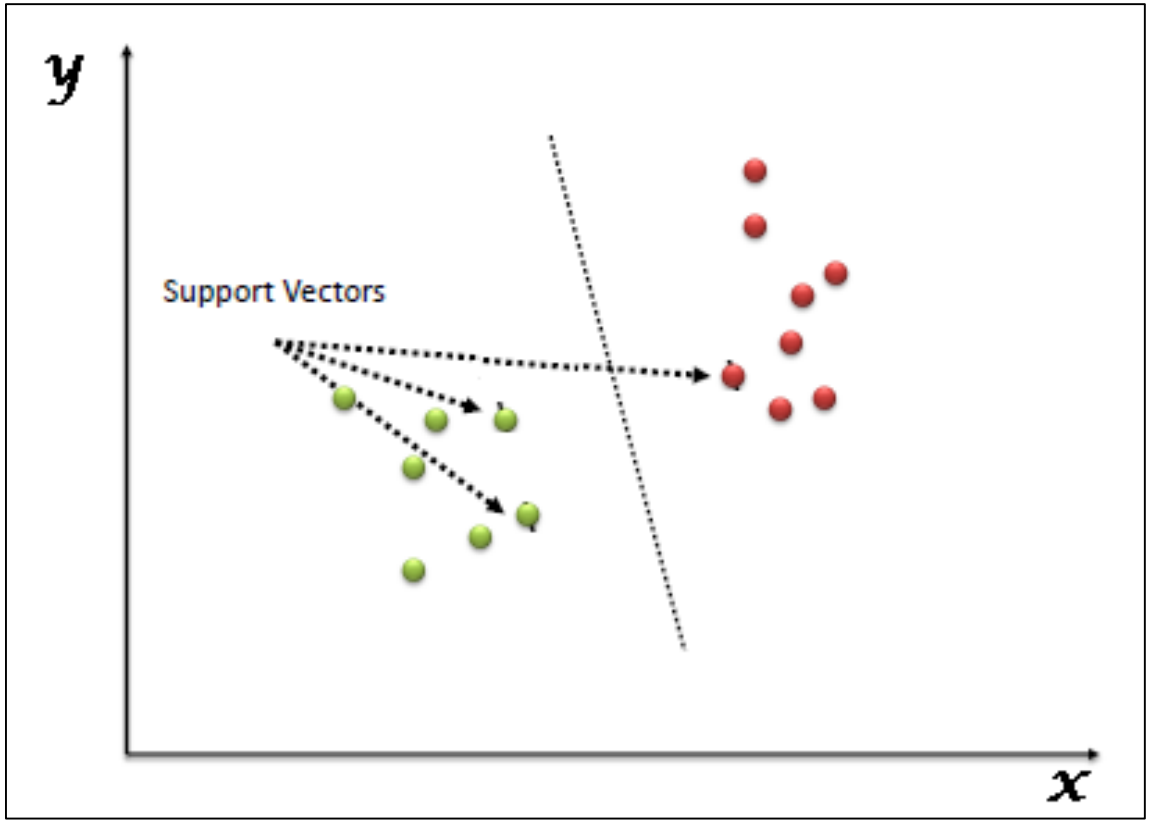
old	new
w1: 0.8	w1: 0.7921
w2: 0.4	w2: 0.3887
w3: 0.3	w3: 0.2741
w4: 0.2	w4: 0.1921
w5: 0.9	w5: 0.8887
w6: 0.5	w6: 0.4741
w7: 0.3	w7: 0.2020
w8: 0.5	w8: 0.3940
w9: 0.9	w9: 0.8060



SUPPORT VECTOR MACHINE (SVM)

- A supervised machine learning algorithm which can be used for classification or regression challenges.
- Mostly used in classification problems.
- Each data item is represented as a point in an *n*-dimensional space (where n is number of features you have) with the value of each feature being the coordinate value
- Then, we find the hyper-plane that differentiate the two classes very well

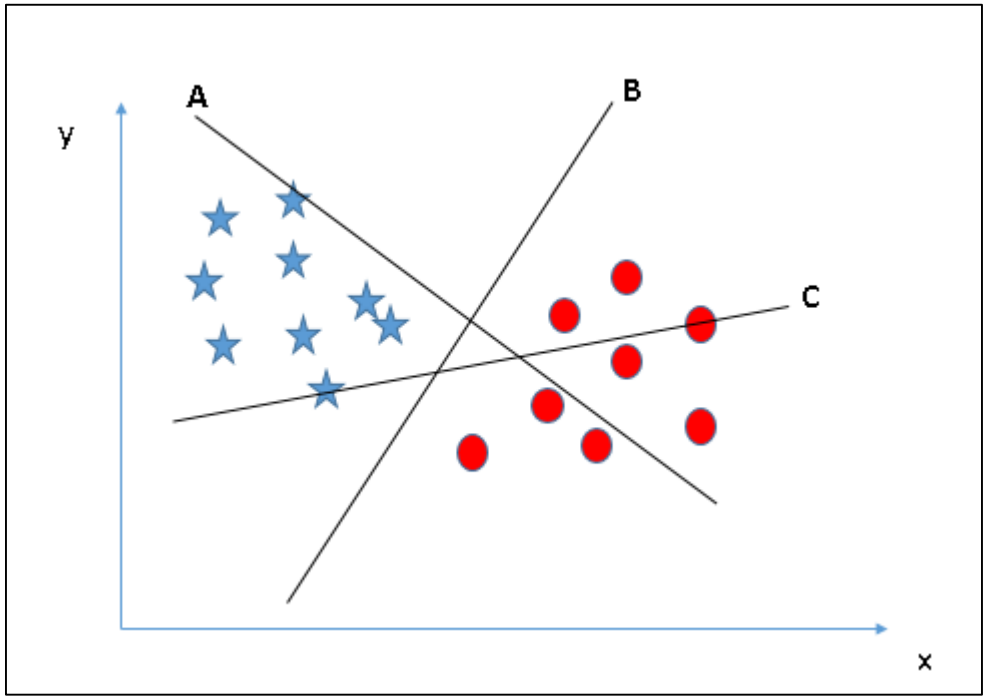
SUPPORT VECTOR MACHINE (SVM)



- A hyperplane in a 2-dimensional plane is a straight line (2-1)
- A hyperplane in a n-dimensional plane is a n-1 hyperplane.
- Support vectors are the elements of the training set that would change the position of the dividing hyperplane if removed.

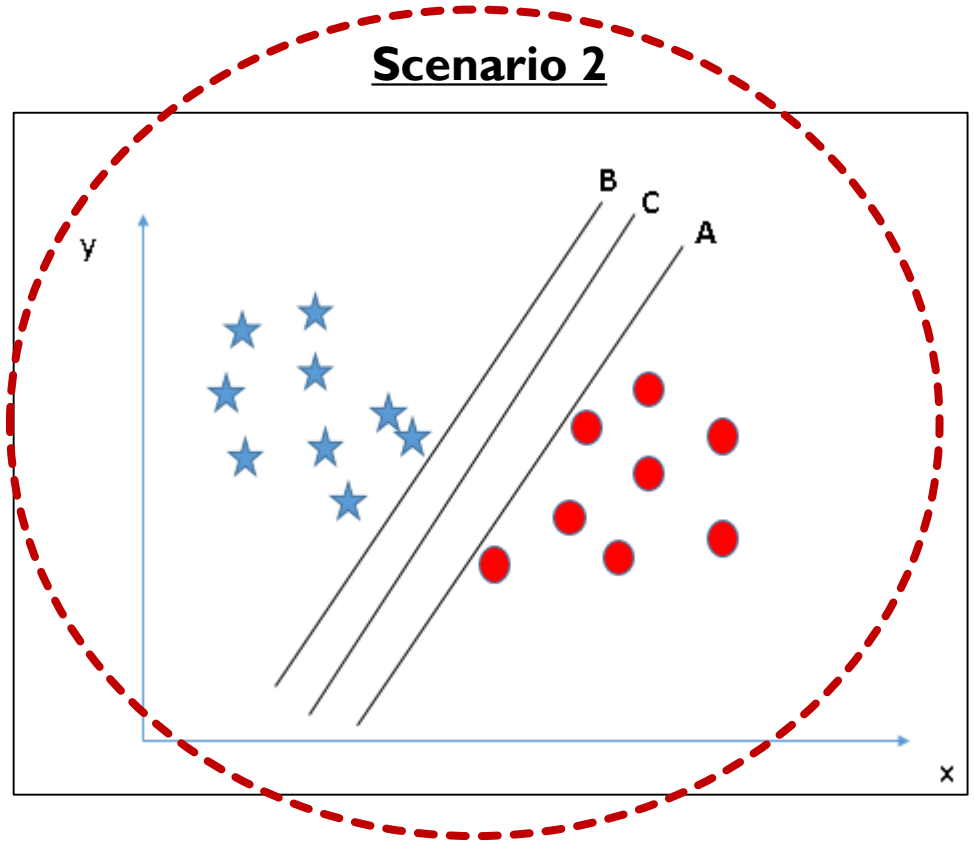
SUPPORT VECTOR MACHINE (SVM)

Scenario 1



Identify the hyper plane that classifies stars and circles

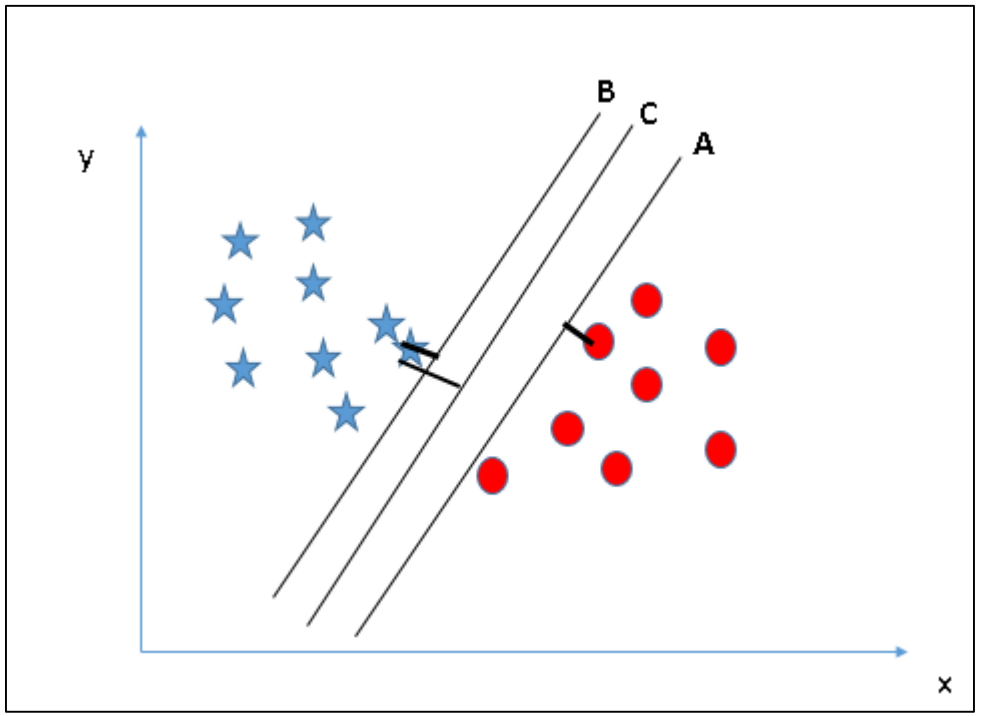
Scenario 2



Identify the best hyper plane that classifies stars and circles

SUPPORT VECTOR MACHINE (SVM)

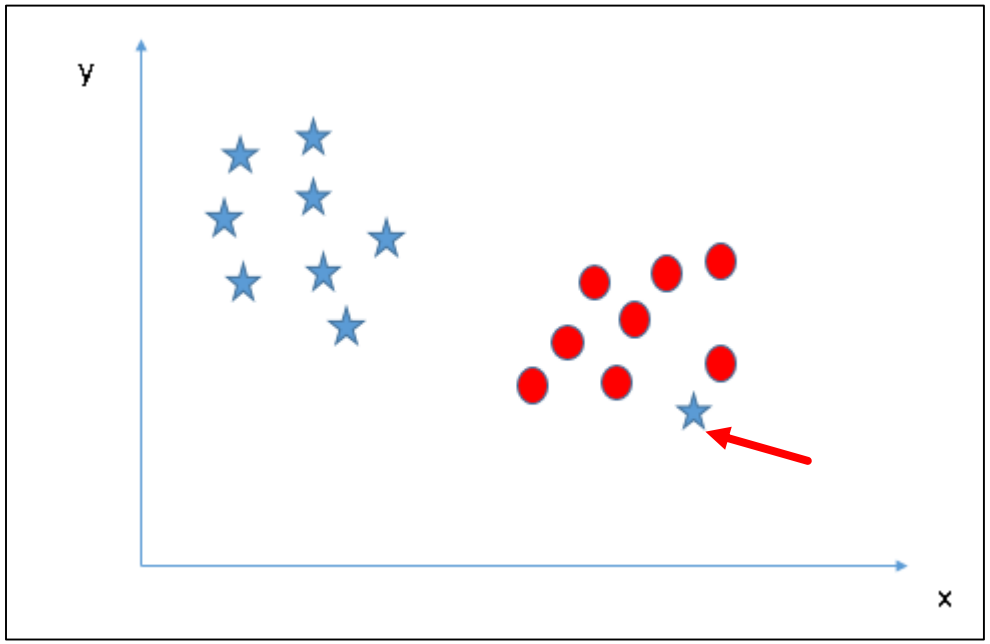
Scenario 2



- Maximizing the distances between nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called **Margin**.
- Margin for hyper-plane C is high as compared to both A and B. Hence, the right hyper-plane is C.
- Another reason for selecting the hyper-plane with higher margin is robustness. A hyper-plane having low margin has a high chance of miss-classification.

SUPPORT VECTOR MACHINE (SVM)

Scenario 3

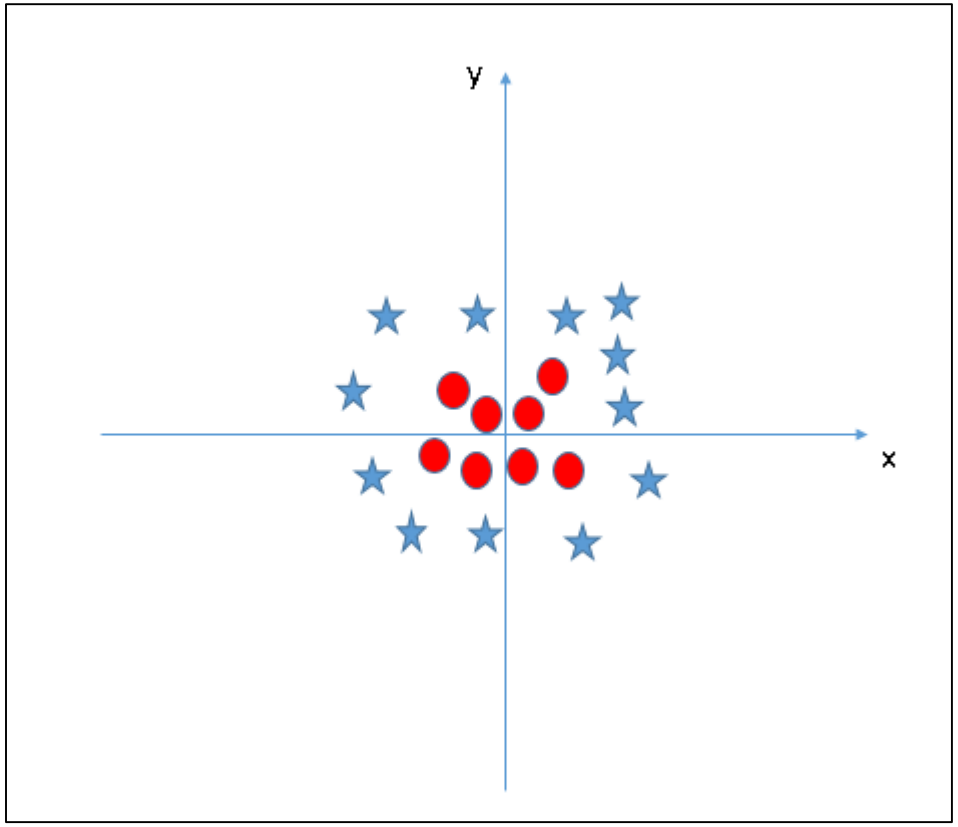


Identify the hyper plane that classifies stars and circles

- SVM is robust to outliers.
- SVM has a feature to ignore outliers and find the hyper-plane that has maximum margin.

SUPPORT VECTOR MACHINE (SVM)

Scenario 3



We can't have linear hyper plane to classify them!

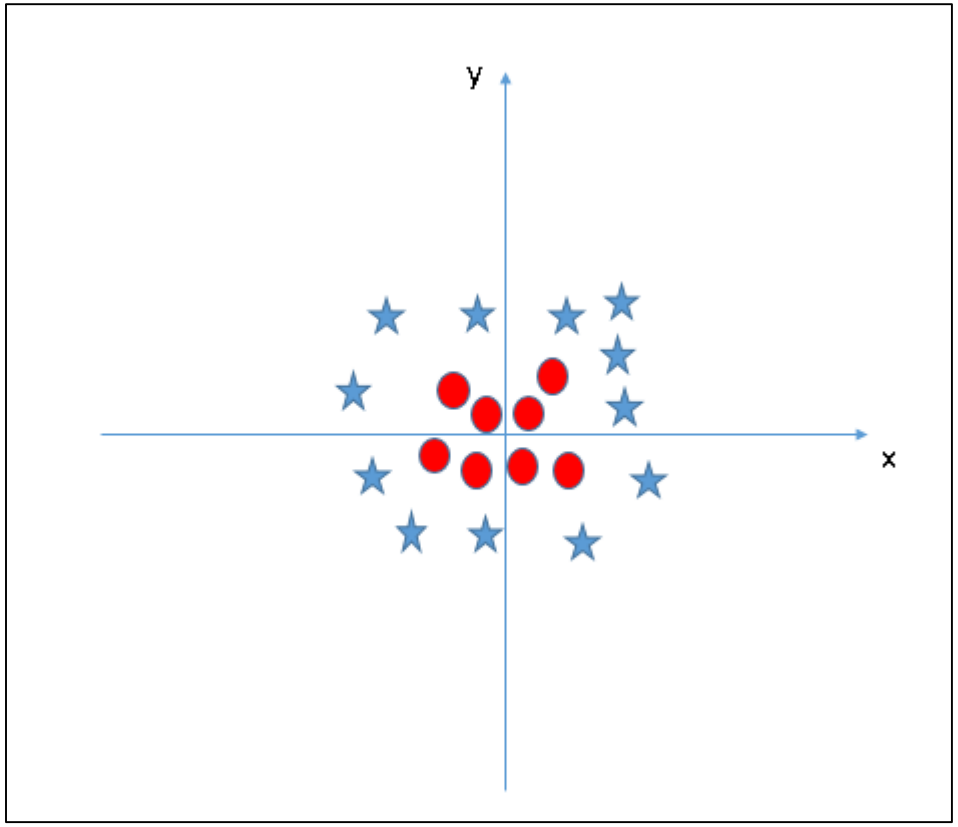
- What happens when we have non-linear data?
- SVM solves this problem by introducing another feature, z.

$$z=x^2+y^2$$

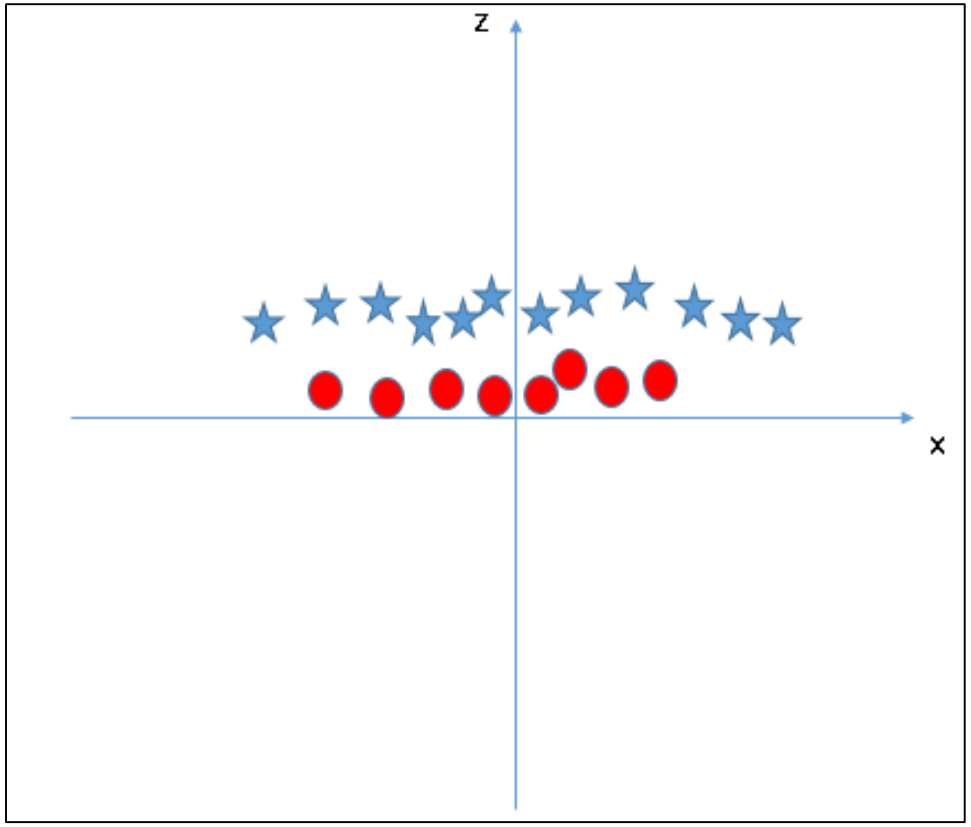
- You will notice how these data are converted to become a "linear" form.

SUPPORT VECTOR MACHINE (SVM)

Before Conversion

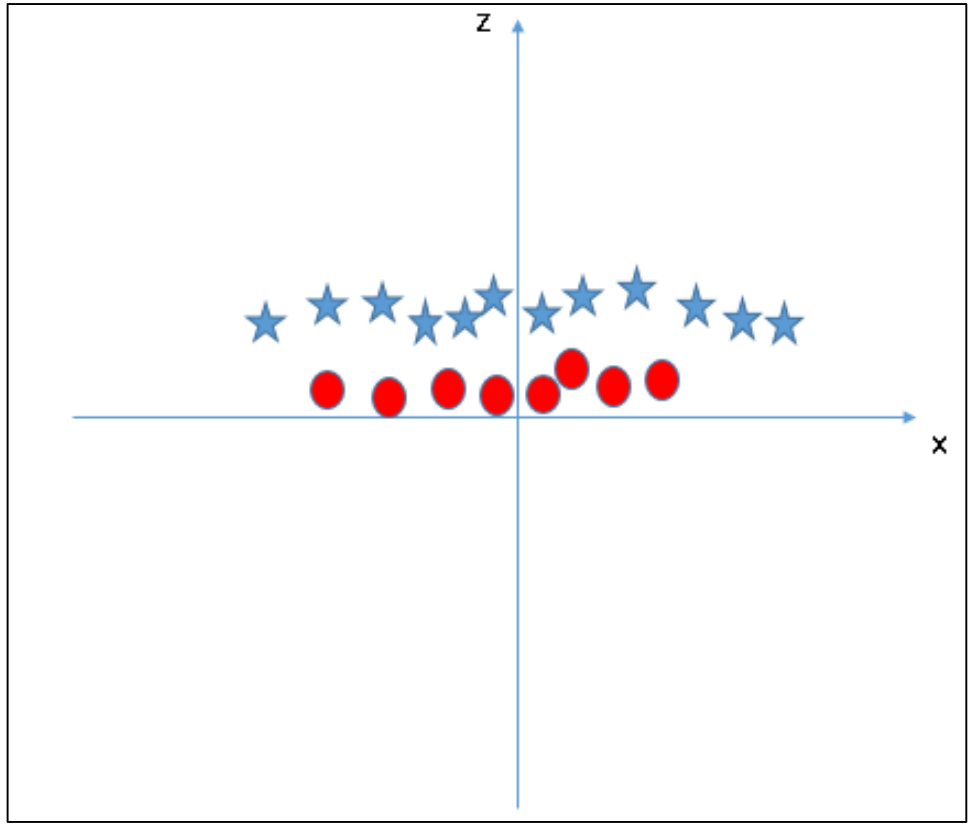


After Conversion



SUPPORT VECTOR MACHINE (SVM)

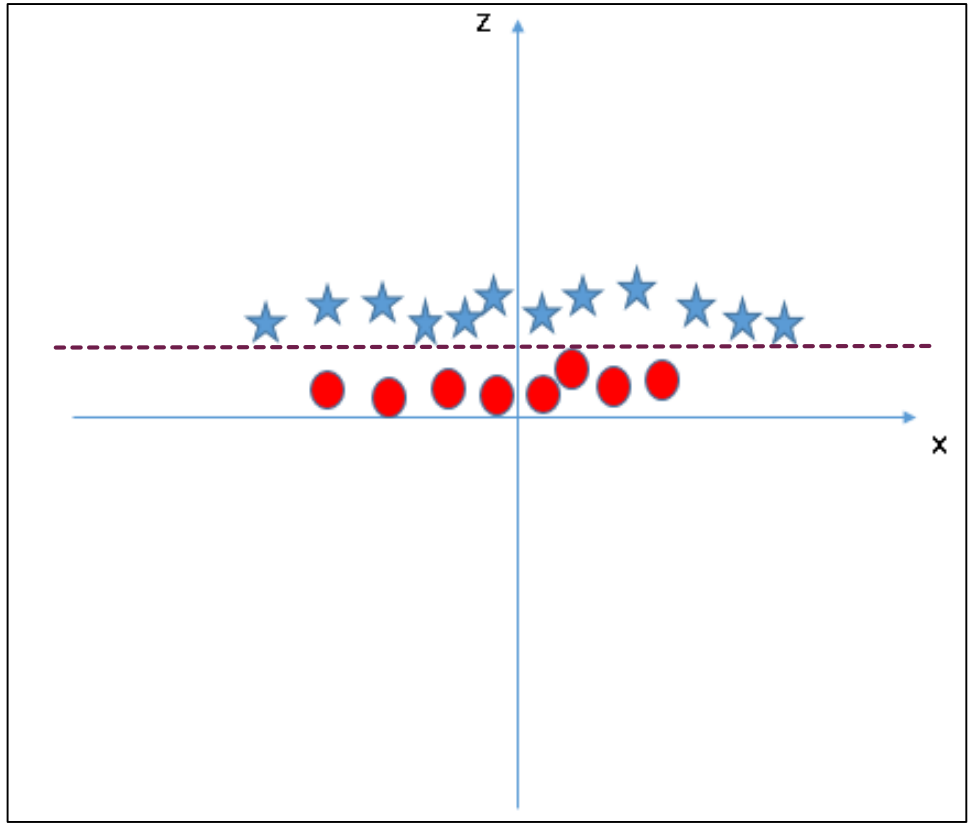
Scenario 3



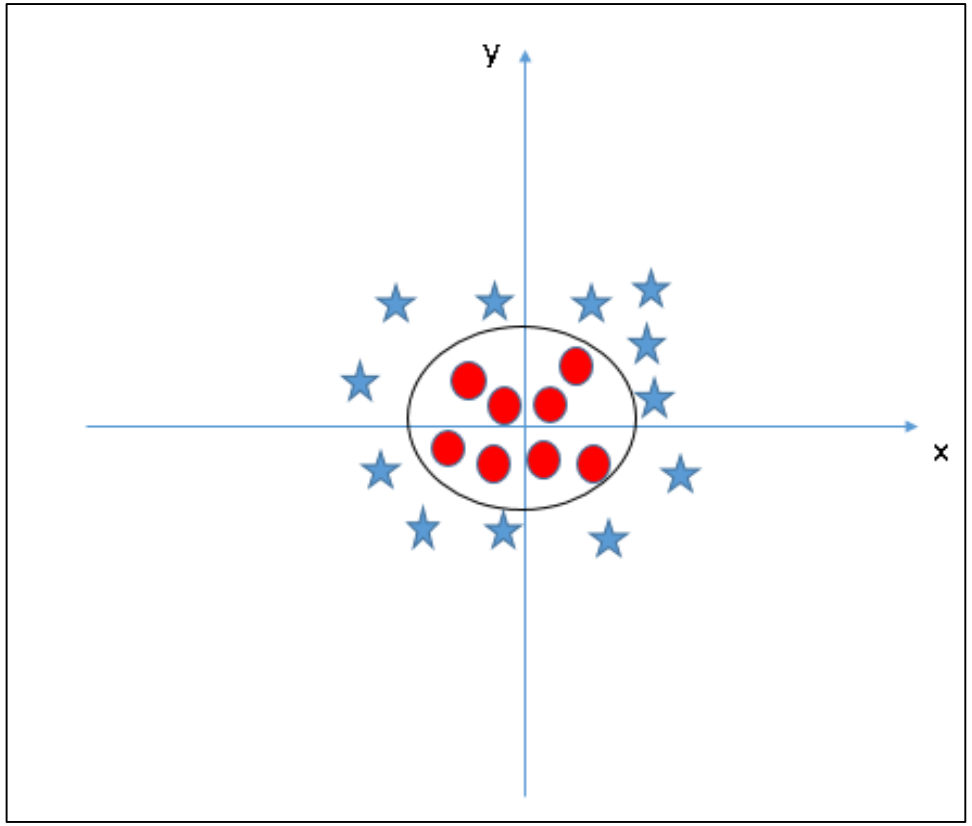
- In SVM, this feature is handled with a technique known as "Kernel Trick"
- Kernel methods take low dimensional input space and transform it to a higher dimensional space (Converts not separable problem to separable problem)

SUPPORT VECTOR MACHINE (SVM)

(Kernel) plot of X-Z



Original plot of X-Y

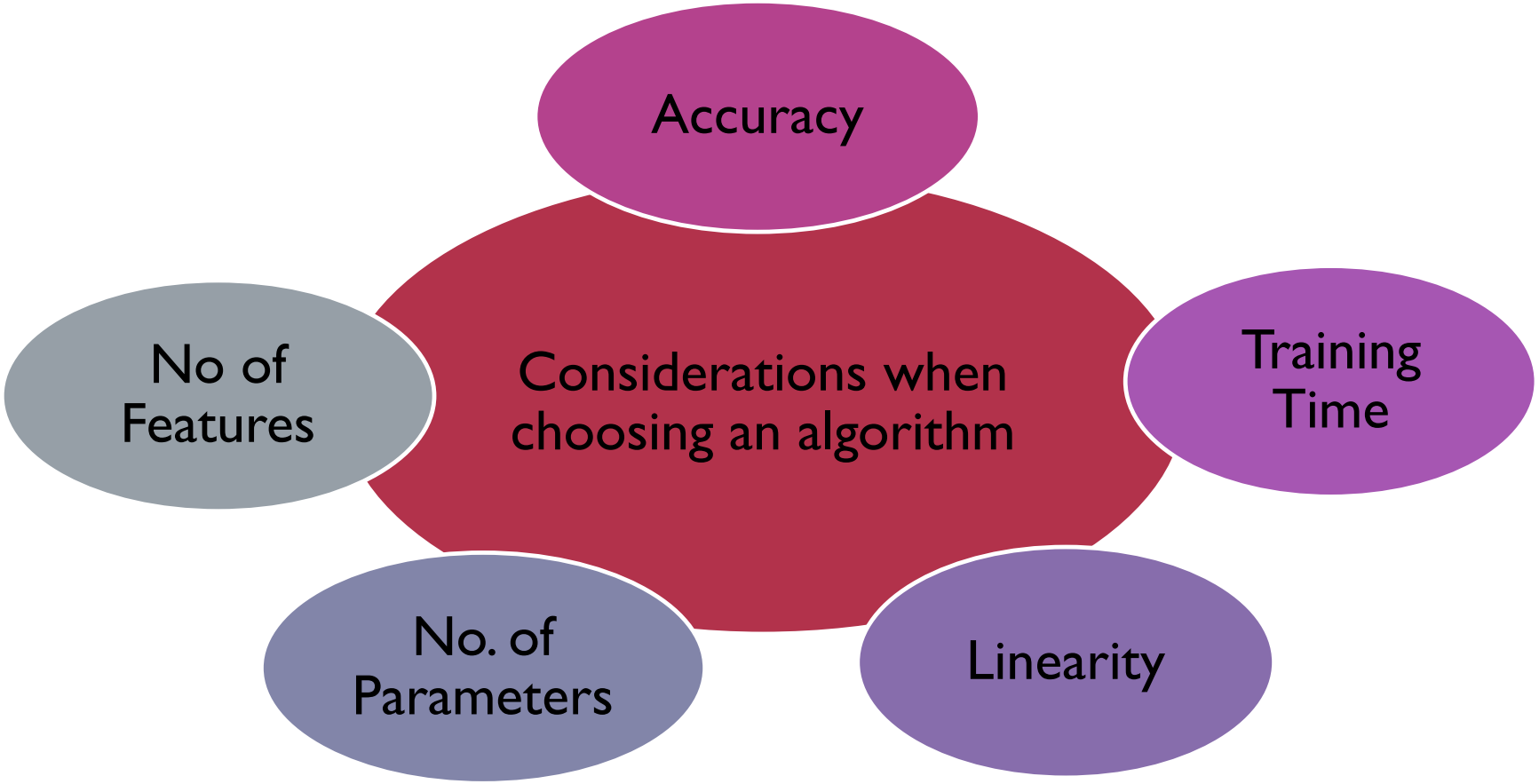


MODEL EVALUATION AND SELECTION

Considerations when choosing a machine learning algorithm:

1. Accuracy
2. Training Time
3. Linearity
4. Number of Parameters
5. Number of features

MODEL EVALUATION AND SELECTION



MODEL EVALUATION AND SELECTION

- Accuracy
  - It is not always necessary to get accurate results
  - Approximation is sometimes sufficient
  - It cuts processing time significantly
  - Tends to avoid overfitting
- Training Time
  - Time to train a model varies greatly across algorithm
  - It is highly correlated to accuracy
  - Dependent on the size of the training data set as well

MODEL EVALUATION AND SELECTION

- Linearity
  - Linear classification algorithm assumes classes can be separated linearly
  - However, if the data are not linearly separable, it may result in low accuracy
- No of parameters
  - No of parameter denotes the flexibility of an algorithm
  - When the right combination of parameters is yield, it will bring about high accuracy.
  - However, requires a lot of trial and error work.
- No of features
  - In some dataset, no of features can be very large compared to the number of data points
  - This characteristic may bog down some machine learning algorithms (Resulting in substantial training time)
  - SVM usually handles these kind of data well.

MODEL EVALUATION AND SELECTION

Algorithms	Advantages (Pros)	Disadvantages (Cons)	Use cases
Linear regression	• Very fast (runs in constant time) • Easy to understand the model • Less prone to overfitting	• Unable to model complex relationships • Unable to capture nonlinear relationships without first transforming the inputs	• The first look at a dataset • Numerical data with lots of features
Decision trees	• Fast • Robust to noise and missing values • Accurate	• Complex trees are hard to interpret • Duplication within the same sub-tree is possible	• Star classification • Medical diagnosis • Credit risk analysis
Neural networks	• Extremely powerful • Can model even very complex relationships • No need to understand the underlying data • Almost works by "magic"	• Prone to overfitting • Long training time • Requires significant computing power for large datasets • Model is essentially unreadable	• Images • Video • "Human-intelligence" type tasks like driving or flying • Robotics

MODEL EVALUATION AND SELECTION

Algorithms	Advantages (Pros)	Disadvantages (Cons)	Use cases
Support Vector Machines	• Can model complex, nonlinear relationships • Robust to noise (because they maximize margins)	• Need to select a good kernel function • Model parameters are difficult to interpret • Sometimes numerical stability problems • Requires significant memory and processing power	• Classifying proteins • Text classification • Image classification • Handwriting recognition
K-Nearest Neighbors	• Simple • Powerful • No training involved ("lazy") • Naturally handles multiclass classification and regression	• Expensive and slow to predict new instances • Must define a meaningful distance function • Performs poorly on high-dimensionality datasets	• Low-dimensional datasets • Computer security: intrusion detection • Fault detection in semiconductor manufacturing • Video content retrieval • Gene expression • Protein-protein interaction