PARULINSTITUTEOF ENGINEERING &TECHNOLOGY
FACULTY OF ENGINEERING & TECHNOLOGY
PARULUNIVERSITY

Subject: Machine Learning

Unit 3 : Supervised Learning – II

Computer Science & Engineering

Dr. Sudhanshu Singh (Associate Prof. PIET-CSE)

Outline

- K-NN classifier
- Logistic regression
- Perceptrons
- Single layer Perceptrons
- Multi-layer Perceptrons
- Support Vector Machines (SVM)
- Linear SVM
- Non-linear SVM

K-Nearest Neighbor(KNN) Algorithm

K-Nearest Neighbor is one of the simplest Machine Learning algorithms based on Supervised Learning technique.

K-NN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories.

K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

K-NN algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.

K-Nearest Neighbor(KNN) Algorithm

History

It was first described in the early 1950s.

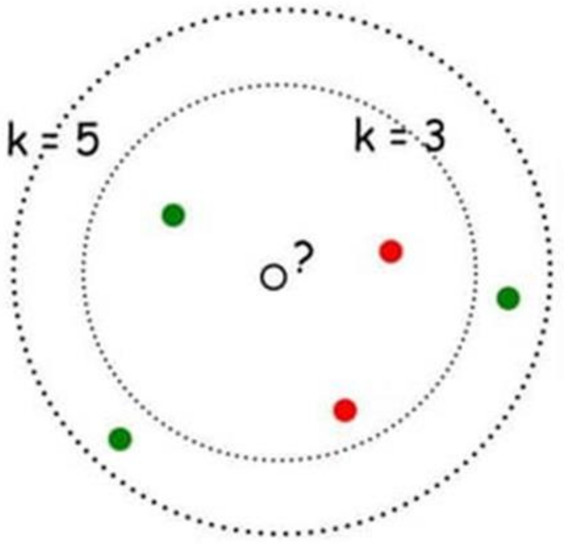
The method is labor intensive when given large training sets.

Gained popularity, when increased computing power became available.

Used widely in area of pattern recognition and statistical estimation.

K-Nearest Neighbor(KNN) Example

When k=3 or k=5??



with k = 3, 🔴
with k = 5, 🟢

Remarks

Similarity Function Based.

Choose an odd value of k for 2 class problem.

k must not be multiple of number of classes.

Closeness

• The Euclidean distance between two points or tuples, say,

$$dist(X_1, X_2) = \sqrt{\sum_{i=1}^{n} (x_{1i} - x_{2i})^2}.$$

Min-max normalization can be used to transform a value v of a numeric attribute A to v0 in the range [0,1] by computing

$$v' = \frac{v - min_A}{max_A - min_A},$$

What if attributes are categorical??

How can distance be computed for attribute such as colour?

Simple Method: Compare corresponding value of attributes

Other Method: Differential grading

What about missing values ??

If the value of a given attribute A is missing in tuple X1 and/or in tuple X2, we assume the maximum possible difference.

For categorical attributes, we take the difference value to be 1 if either one or both of the corresponding values of A are missing.

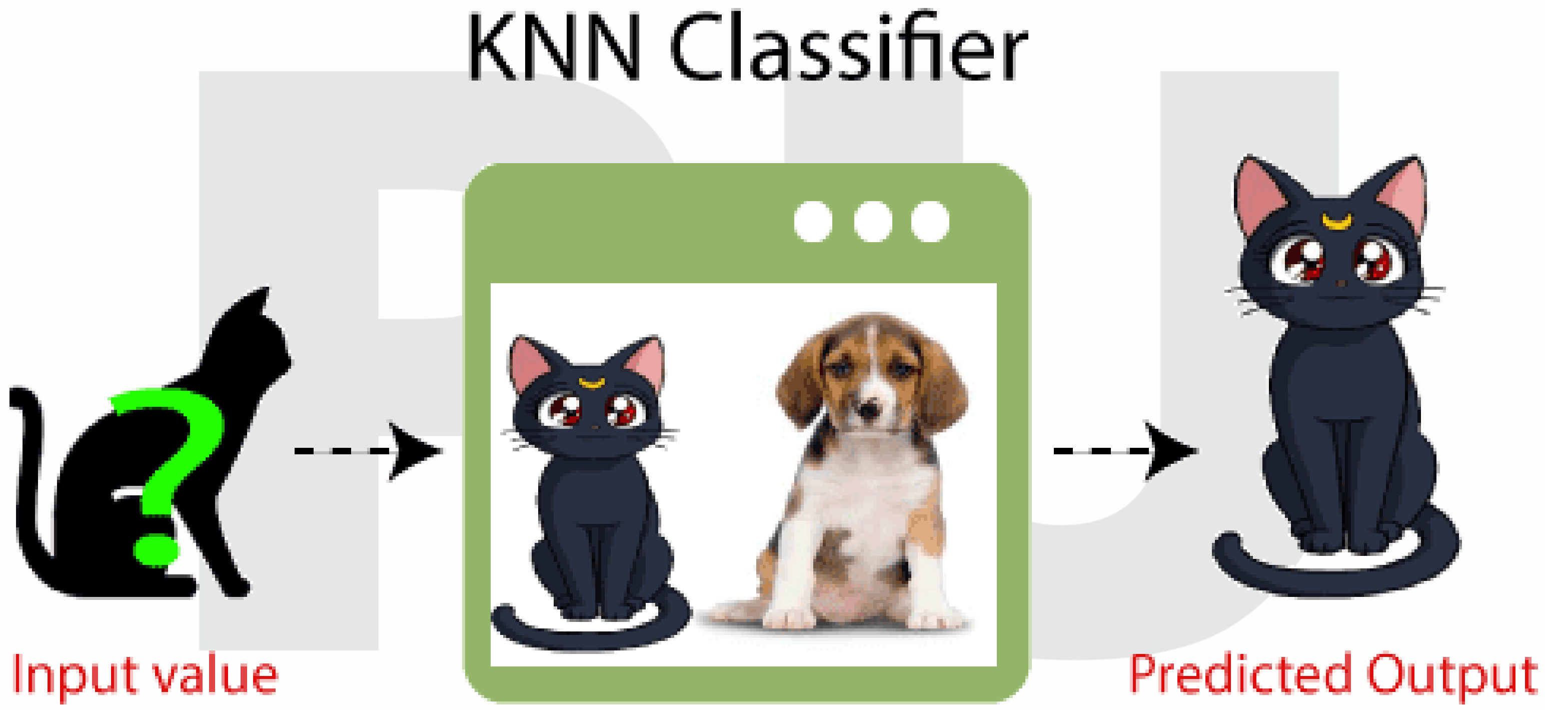
If A is numeric and missing from both tuples X1 and X2, then the difference is also taken to be 1.

K-Nearest Neighbor(KNN) Algorithm

- It is also called a **lazy learner algorithm** because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

- KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.
- **Example:** Suppose, we have an image of a creature that looks similar to cat and dog, but we want to know either it is a cat or dog. So for this identification, we can use the KNN algorithm, as it works on a similarity measure. Our KNN model will find the similar features of the new data set to the cats and dogs images and based on the most similar features it will put it in either cat or dog category.

K-Nearest Neighbor(KNN) Algorithm

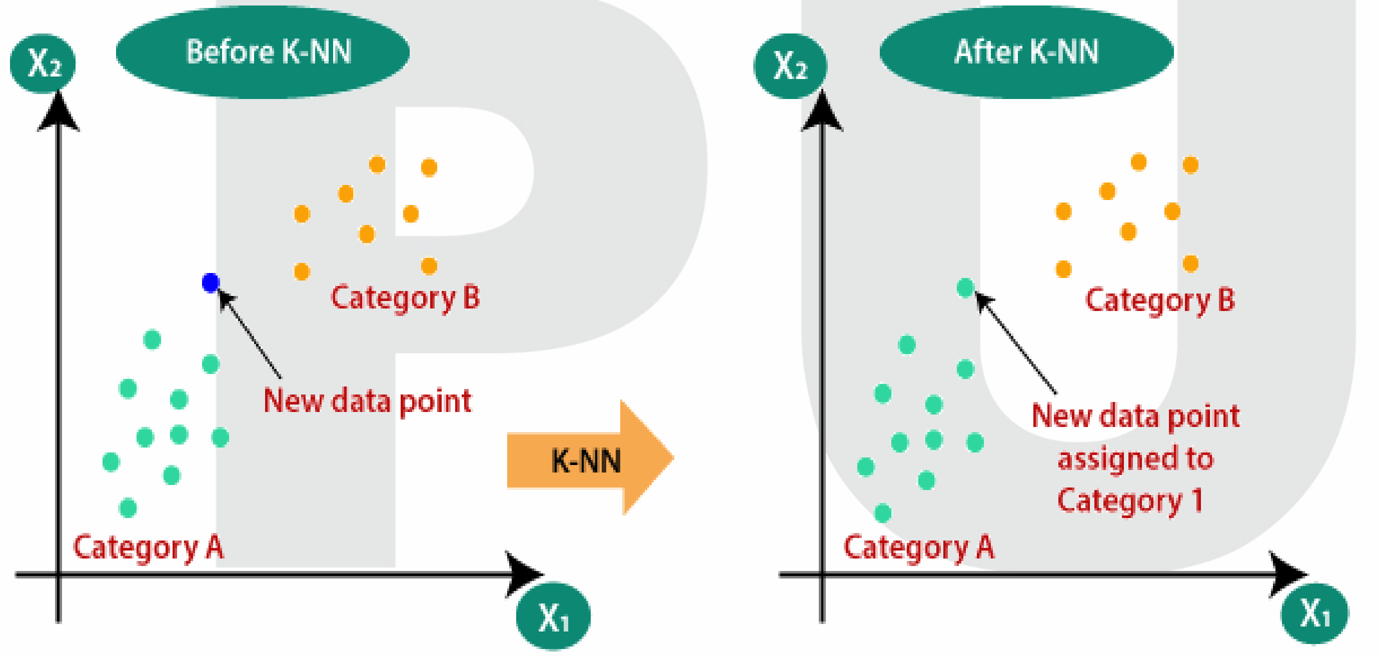


Why do we need a K-NN Algorithm?

- Suppose there are two categories, i.e. Category A and Category B, and we have a new data point x1, so this data point will lie in which of these categories.

- To solve this type of problem, we need a K-NN algorithm. With the help of K-NN, we can easily identify the category or class of a particular dataset.

- Consider the below diagram:.


Before K-NN
X₂
Category B
New data point
Category A
X₁
K-NN
After K-NN
X₂
Category B
New data point
assigned to
Category 1
Category A
X₁

How does K-NN work?

The K-NN working can be explained on the basis of the below algorithm:

Step-1: Select the number K of the neighbors

Step-2: Calculate the Euclidean distance of **K number of neighbors**

Step-3: Take the K nearest neighbors as per the calculated Euclidean distance.

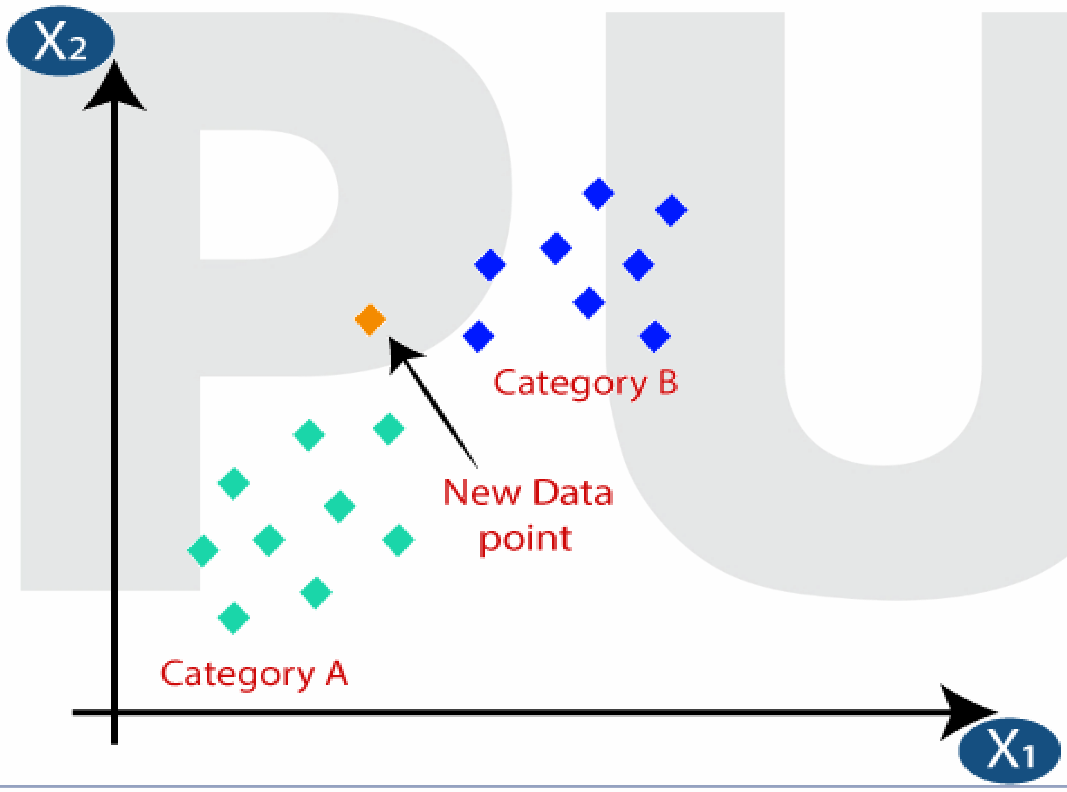
Step-4: Among these k neighbors, count the number of the data points in each category.

Step-5: Assign the new data points to that category for which the number of the neighbor is maximum.

Step-6: Our model is ready.

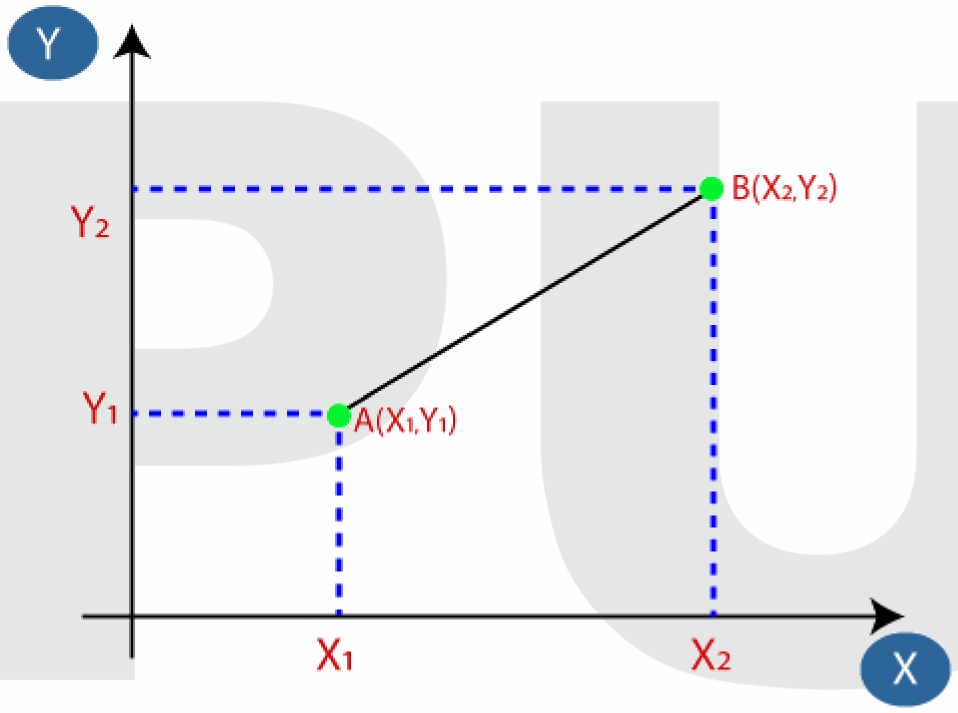
How does K-NN work?

Suppose we have a new data point and we need to put it in the required category. Consider the below image



How does K-NN work?

- Firstly, we will choose the number of neighbors, so we will choose the k=5.
- Next, we will calculate the Euclidean distance between the data points.
- The Euclidean distance is the distance between two points, which we have already studied in geometry. It can be calculated as:

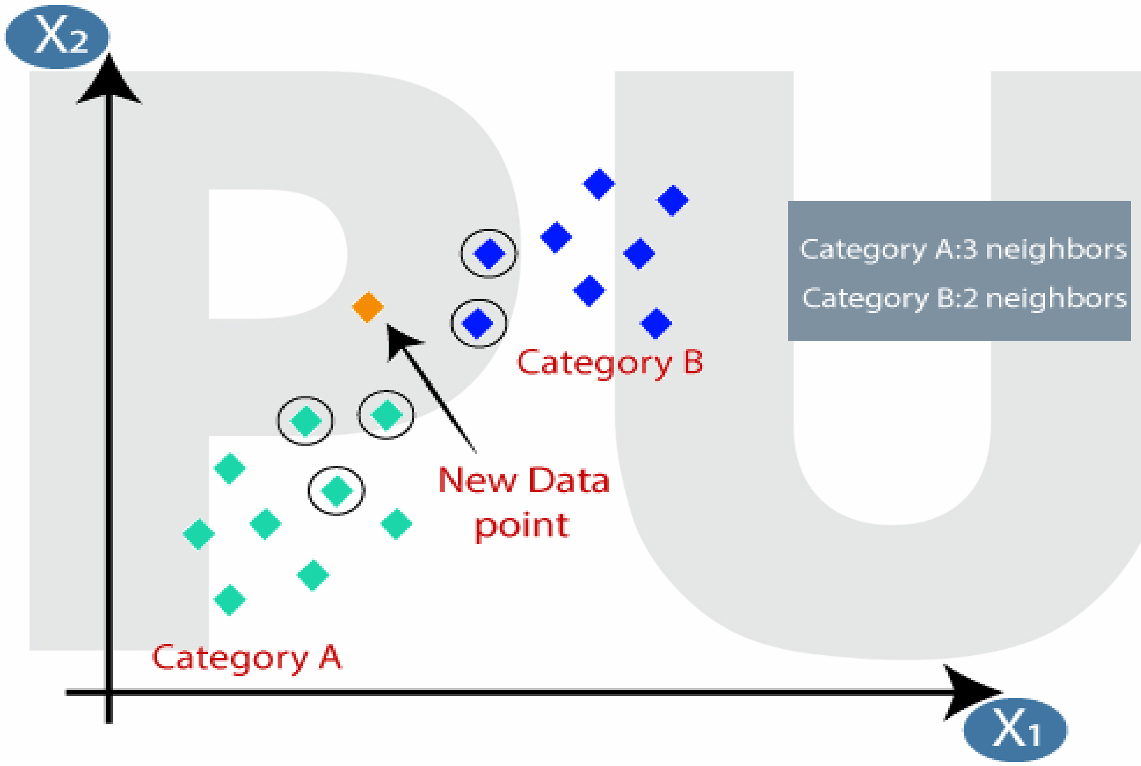


Euclidean Distance between A_1 and B_2 = $\sqrt{(X_2-X_1)^2+(Y_2-Y_1)^2}$

How does K-NN work?

- By calculating the Euclidean distance we got the nearest neighbors, as three nearest neighbors in category A and two nearest neighbors in category B. Consider the Given image:

- As we can see the 3 nearest neighbors are from category A, hence this new data point must belong to category A.


X2
X1
Category A
Category B
New Data point
Category A:3 neighbors
Category B:2 neighbors

How to select the value of K in the K-NN Algorithm?

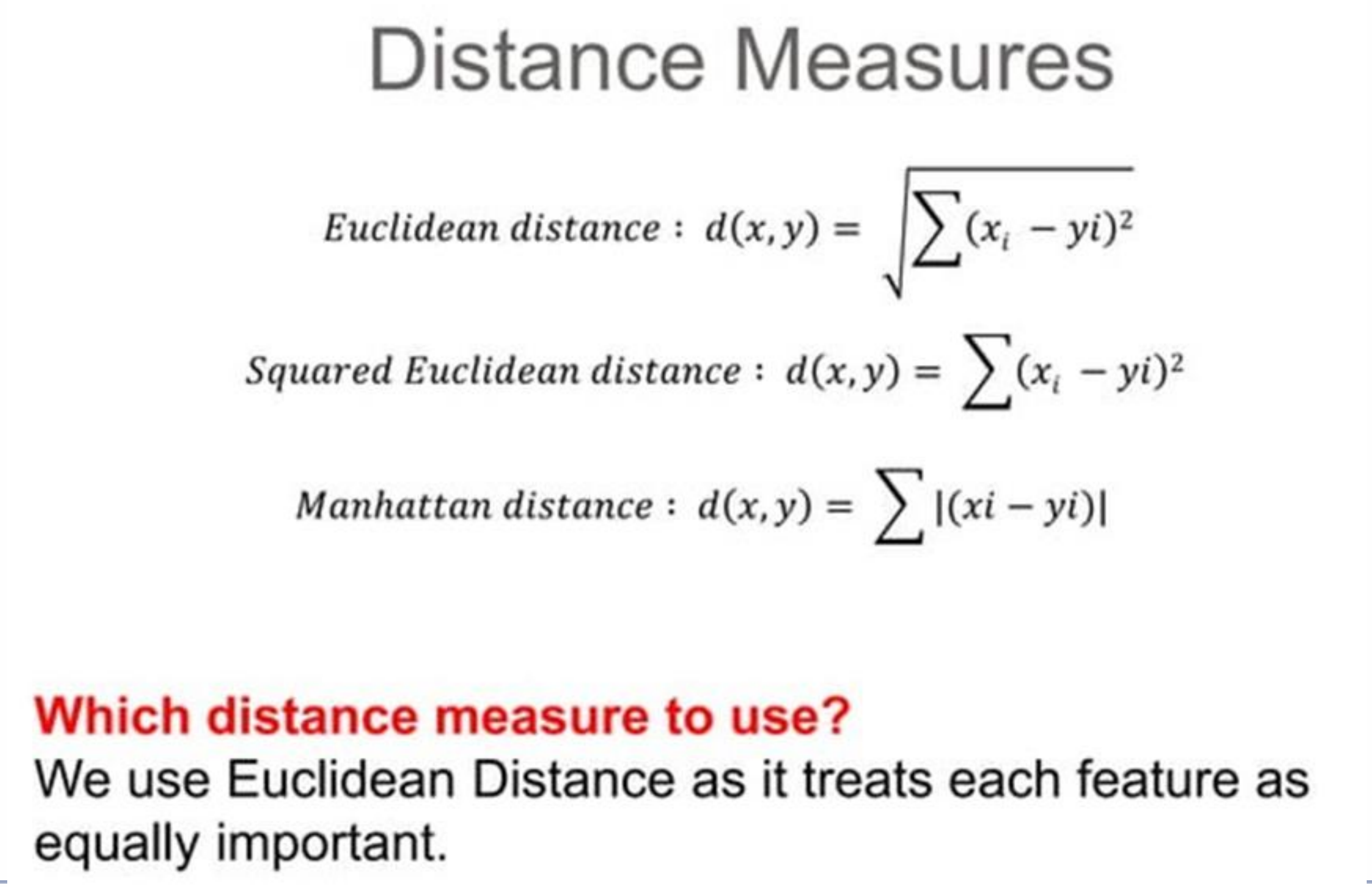
Below are some points to remember while selecting the value of K in the K-NN algorithm:

- There is no particular way to determine the best value for "K", so we need to try some values to find the best out of them. The most preferred value for K is 5.

- A very low value for K such as K=1 or K=2, can be noisy and lead to the effects of outliers in the model.

- Large values for K are good, but it may find some difficulties.

KNN Algorithm Example



Distance Measures

Euclidean distance : $d(x,y) = \sqrt{\sum (x_i - yi)^2}$

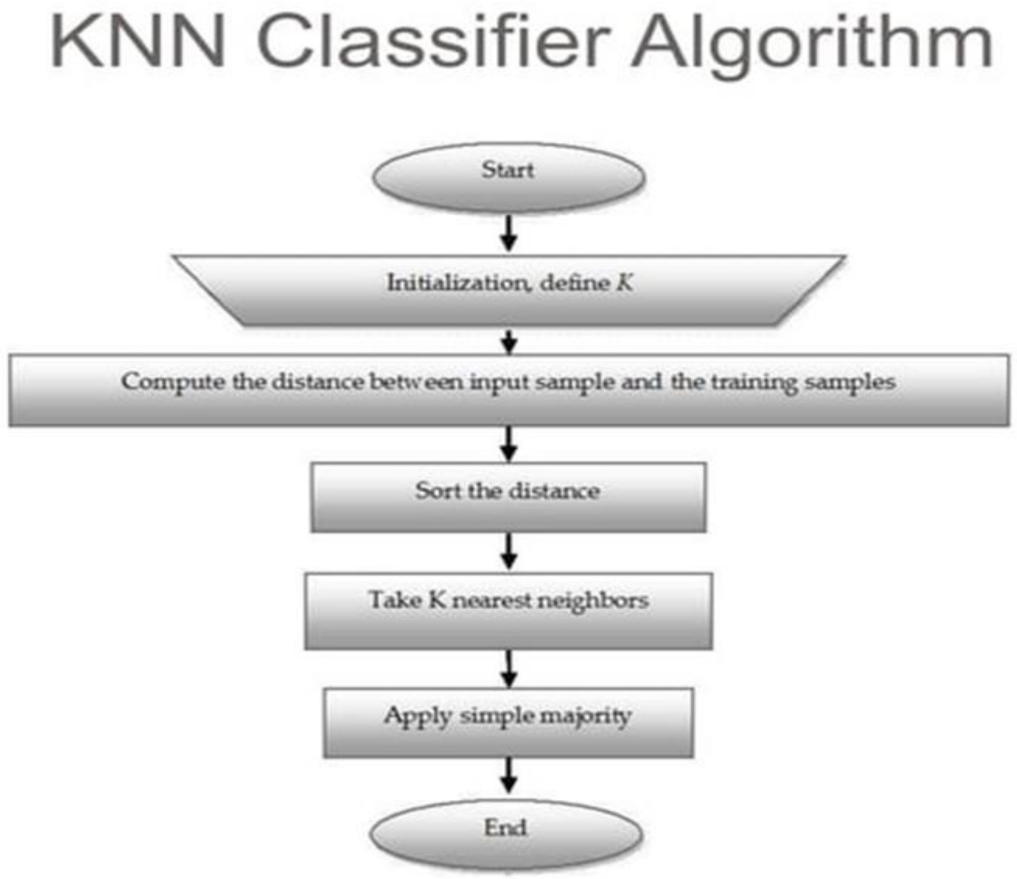
Squared Euclidean distance : $d(x,y) = \sum (x_i - yi)^2$

Manhattan distance : $d(x,y) = \sum |(xi - yi)|$

Which distance measure to use?

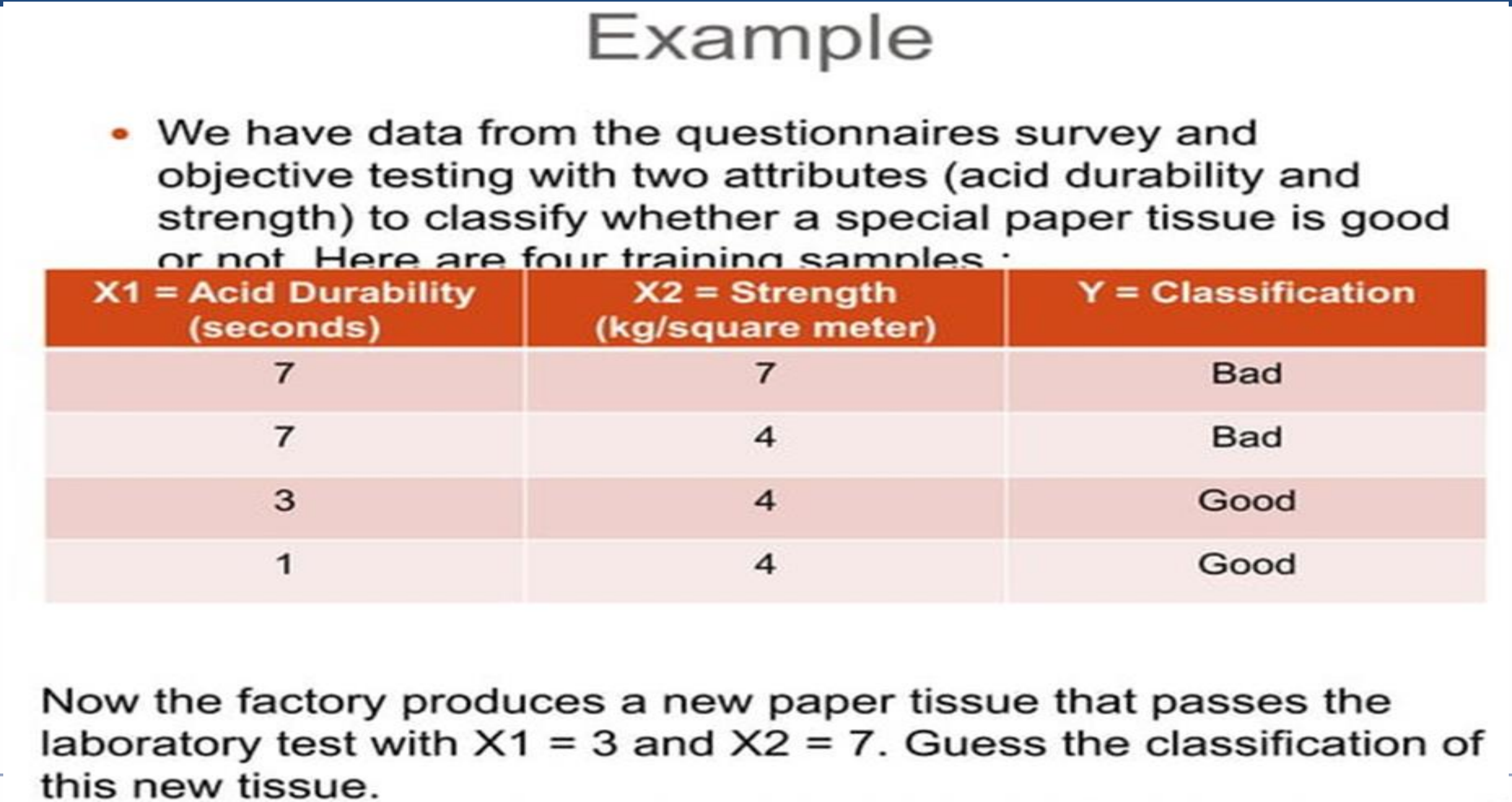
We use Euclidean Distance as it treats each feature as equally important.

KNN Algorithm



DIGITAL LEARNING CONTENT

KNN Algorithm Example

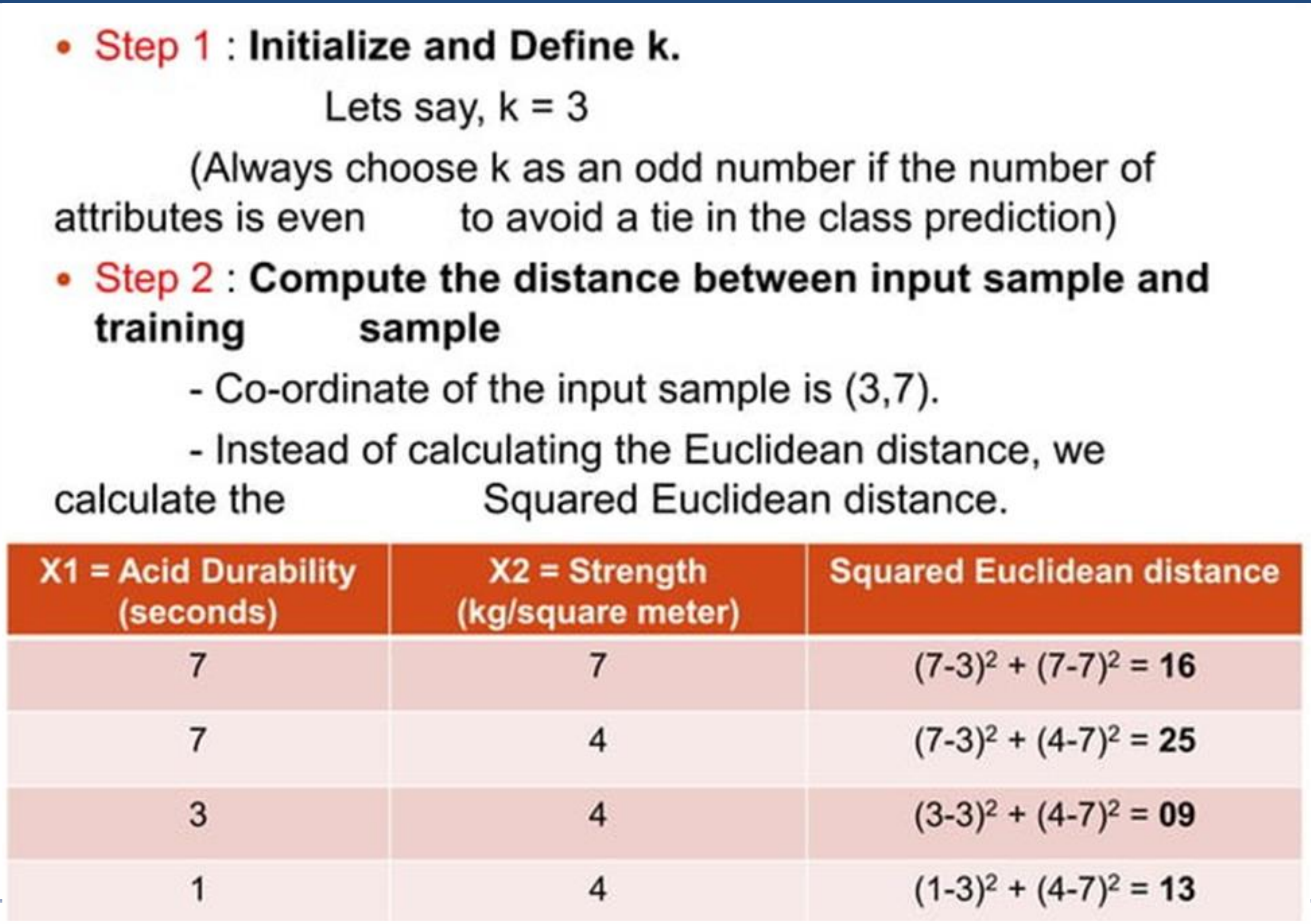


Example

- We have data from the questionnaires survey and objective testing with two attributes (acid durability and strength) to classify whether a special paper tissue is good or not. Here are four training samples :

X1 = Acid Durability (seconds)	X2 = Strength (kg/square meter)	Y = Classification
7	7	Bad
7	4	Bad
3	4	Good
1	4	Good

Now the factory produces a new paper tissue that passes the laboratory test with X1 = 3 and X2 = 7. Guess the classification of this new tissue.



- Step 1 : **Initialize and Define k.**

  Lets say, k = 3

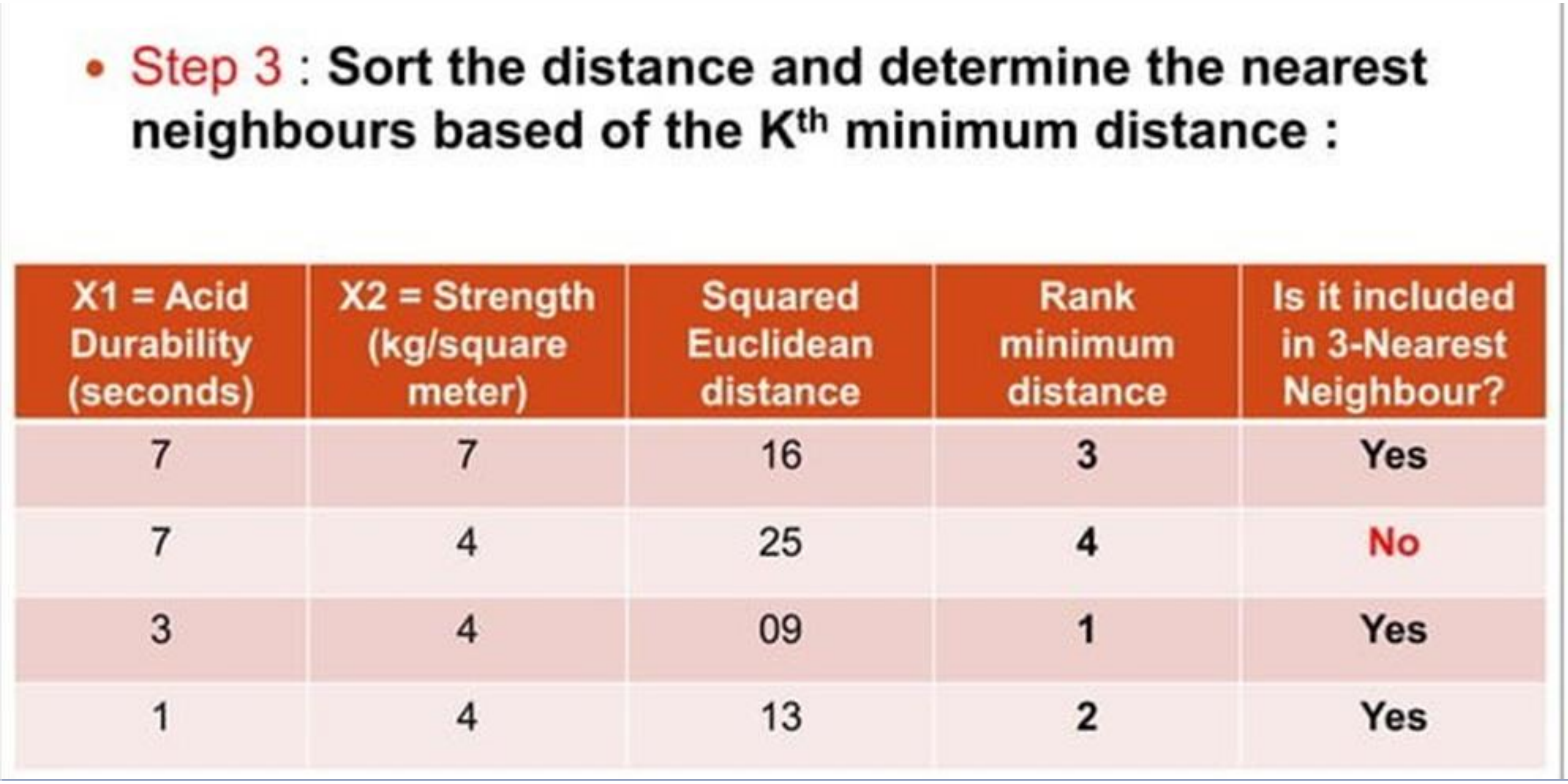
  (Always choose k as an odd number if the number of attributes is even to avoid a tie in the class prediction)

- Step 2 : **Compute the distance between input sample and training sample**

  - Co-ordinate of the input sample is (3,7).

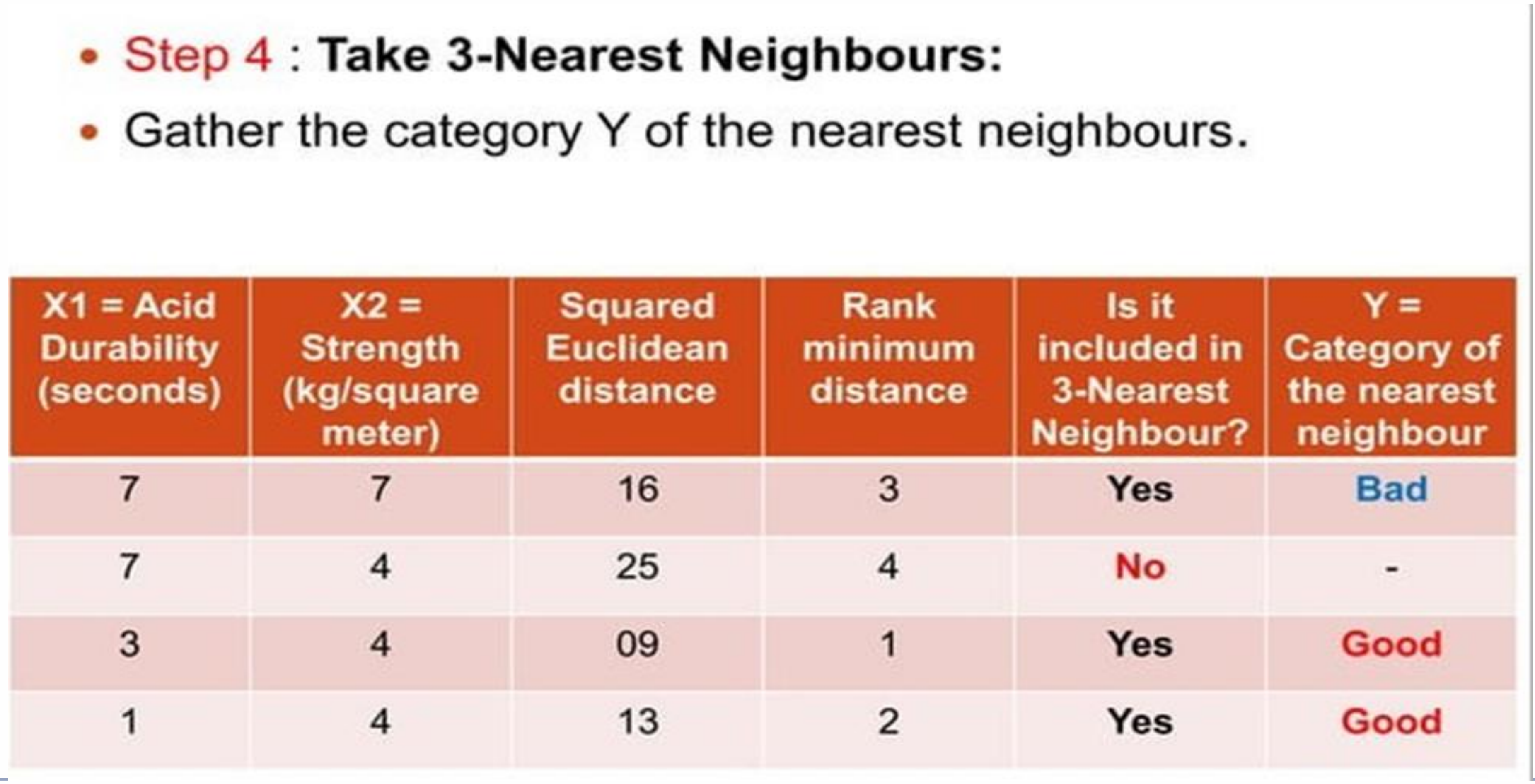
  - Instead of calculating the Euclidean distance, we calculate the Squared Euclidean distance.

X1 = Acid Durability (seconds)	X2 = Strength (kg/square meter)	Squared Euclidean distance
7	7	$(7-3)^2 + (7-7)^2 =$ **16**
7	4	$(7-3)^2 + (4-7)^2 =$ **25**
3	4	$(3-3)^2 + (4-7)^2 =$ **09**
1	4	$(1-3)^2 + (4-7)^2 =$ **13**



- Step 3 : **Sort the distance and determine the nearest neighbours based of the K^th^ minimum distance :**

X1 = Acid Durability (seconds)	X2 = Strength (kg/square meter)	Squared Euclidean distance	Rank minimum distance	Is it included in 3-Nearest Neighbour?
7	7	16	3	Yes
7	4	25	4	No
3	4	09	1	Yes
1	4	13	2	Yes



- Step 4 : **Take 3-Nearest Neighbours:**
- Gather the category Y of the nearest neighbours.

X1 = Acid Durability (seconds)	X2 = Strength (kg/square meter)	Squared Euclidean distance	Rank minimum distance	Is it included in 3-Nearest Neighbour?	Y = Category of the nearest neighbour
7	7	16	3	Yes	Bad
7	4	25	4	No	-
3	4	09	1	Yes	Good
1	4	13	2	Yes	Good

- Step 5 : **Apply simple majority**

- Use simple majority of the category of the nearest neighbours as the prediction value of the query instance.

- We have 2 “good” and 1 “bad”. Thus we conclude that the new paper tissue that passes the laboratory test with X1 = 3 and X2 = 7 is included in the “good” category.

Iris Dataset Example using Weka

- Iris dataset contains 150 sample instances belonging to 3 classes. 50 samples belong to each of these 3 classes.
- **Statistical observations** :
- Let's denote the true value of interest as $\theta\ (expected)$ and the value estimated using some algorithm as $\hat{\theta}.(observed)$
- Kappa Statistics : The kappa statistic measures the agreement of prediction with the true class -- 1.0

- Root Mean Square Error :

$$\mathrm{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\hat{\theta}_i - \theta_i\right)^2}$$

- Relative Absolute Error :

$$\mathrm{RAE} = \frac{\sum_{i=1}^{N} |\hat{\theta}_i - \theta_i|}{\sum_{i=1}^{N} |\bar{\theta} - \theta_i|}$$

Complexity

Basic KNN algorithm stores all examples
.
Suppose we have n examples each of dimension d

O(d) to compute distance to one examples • O(nd) to computed distances to all examples • Plus O(nk) time to find k closest examples

Total time: O(nk+nd)

Very expensive for a large number of samples • But we need a large number of samples for kNN to to work well!!

Advantages of KNN Algorithm:

It is simple to implement.

It is robust to the noisy training data

It can be more effective if the training data is large.

Disadvantages of KNN Algorithm:

Always needs to determine the value of K which may be complex some time.

The computation cost is high because of calculating the distance between the data points for all the training samples..

Applications of KNN Classifier

Used in classification

Used to get missing values

Used in pattern recognition

Used in gene expression

Used in protein-protein prediction

Used to get 3D structure of protein

Used to measure document similarity

Conclusion

KNN is what we call lazy learning (vs. eager learning)

Conceptually simple, easy to understand and explain

Very flexible decision boundaries

Not much learning at all!

It can be hard to find a good distance measure

Irrelevant features and noise can be very detrimental

Typically can not handle more than a few dozen attributes

Computational cost: requires a lot computation

KNN Algorithm Example

https://www.vtupulse.com/machine-learning/knn-solved-example-diabetic-patient/

https://www.freecodecamp.org/news/k-nearest-neighbors-algorithm-classifiers-and-model-example/

Logistic Regression

Logistic regression is one of the most popular Machine Learning algorithms, which comes under the Supervised Learning technique. It is used for predicting the categorical dependent variable using a given set of independent variables.

Logistic regression predicts the output of a categorical dependent variable. Therefore the outcome must be a categorical or discrete value. It can be either Yes or No, 0 or 1, true or False, etc. but instead of giving the exact value as 0 and 1, it gives the probabilistic values which lie between 0 and 1.

Logistic Regression is much similar to the Linear Regression except that how they are used. Linear Regression is used for solving Regression problems, whereas **Logistic regression is used for solving the classification problems.**

Logistic Regression

In Logistic regression, instead of fitting a regression line, we fit an "S" shaped logistic function, which predicts two maximum values (0 or 1).

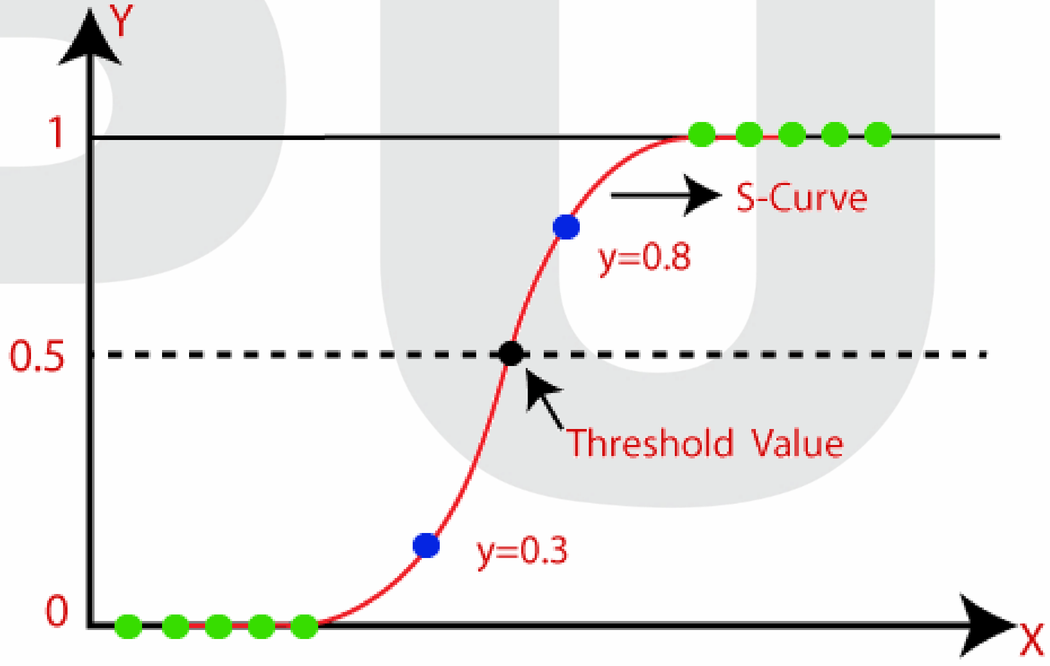
t is based on the concept of regression, but instead of predicting continuous values, it predicts the probability of an instance belonging to a particular class.

Logistic Regression is widely used in various domains, including healthcare, finance, marketing, and social sciences.

It is especially useful when the dependent variable (the variable being predicted) is categorical or binary

Logistic Regression

Logistic Regression can be used to classify the observations using different types of data and can easily determine the most effective variables used for the classification. The below image is showing the logistic function:



Logistic Function (Sigmoid Function)

The logistic function, also known as the sigmoid function, is a key component of logistic regression.

The sigmoid function maps any real-valued number to a value between 0 and 1.

The value of the logistic regression must be between 0 and 1, which cannot go beyond this limit, so it forms a curve like the "S" form. The S-form curve is called the Sigmoid function or the logistic function.

It has an S-shaped curve, which allows it to model the probability of an instance belonging to a specific class.

Logistic Function (Sigmoid Function)

The sigmoid function is defined as follows: **f(x) = 1 / (1 + e^(-x)).**

Here, x is the input value, and f(x) represents the output value, which lies between 0 and 1.

As x approaches positive infinity, f(x) approaches 1, indicating a higher probability of belonging to the positive class.

Similarly, as x approaches negative infinity, f(x) approaches 0, indicating a lower probability of belonging to the positive class.

The midpoint of the sigmoid function occurs at x = 0, where f(0) = 0.5, indicating an equal probability of belonging to either class.

Decision Boundary in Logistic Regression

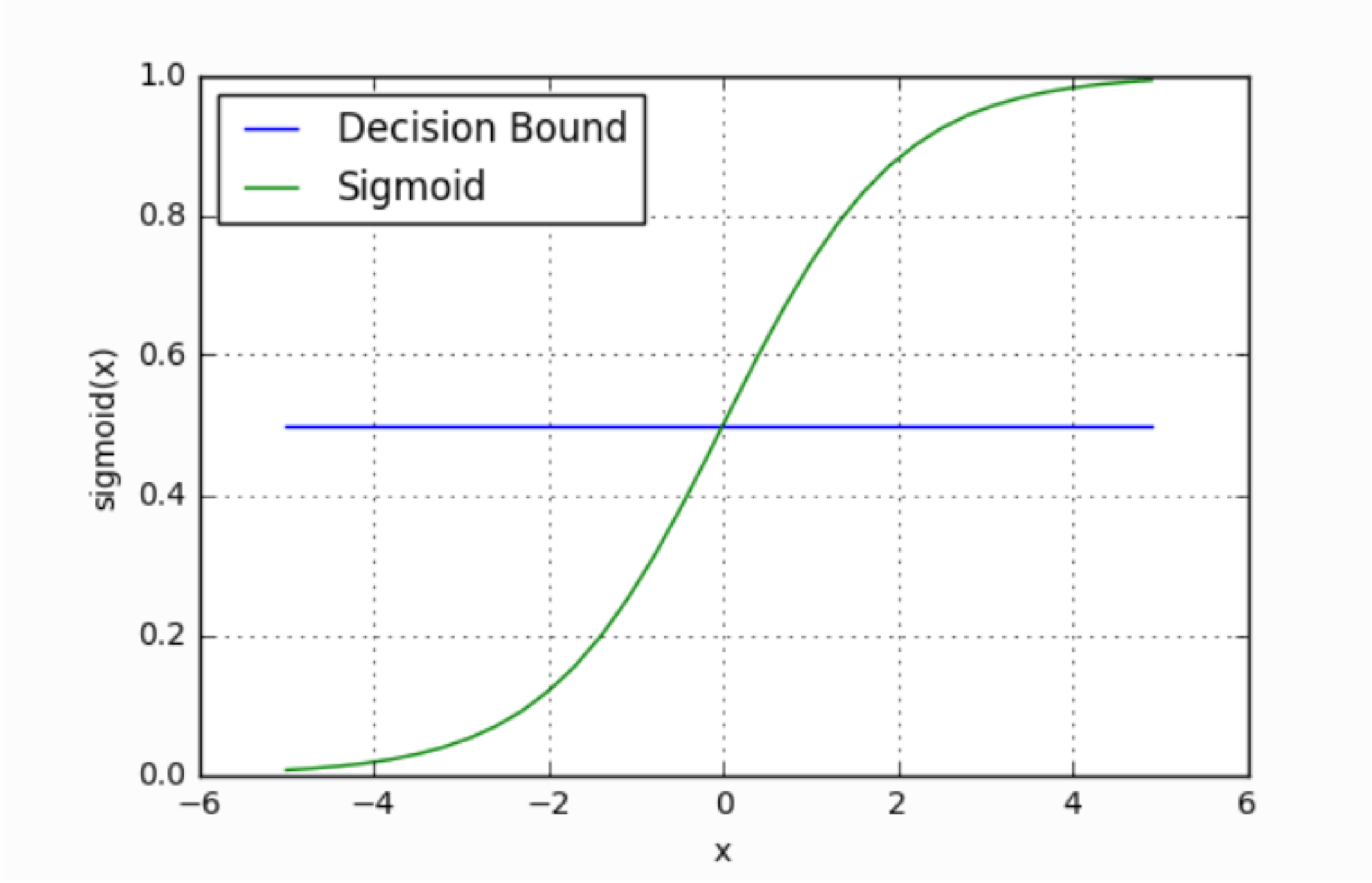
In a two-dimensional feature space, the decision boundary is represented by a line that separates the positive and negative instances.

The decision boundary is derived from the sigmoid function output, where the predicted probability is compared to the threshold value.

Instances with predicted probabilities above the threshold are classified as the positive class, and those below the threshold are classified as the negative class.

The decision boundary can be linear or non-linear, depending on the relationship between the input variables and the output probability.

Decision Boundary in Logistic Regression



Decision Boundary in Logistic Regression

Logistic Regression uses a decision boundary to classify instances into different classes based on their predicted probabilities.

The decision boundary is a threshold value that separates the instances.

If the predicted probability is above the threshold, the instance is classified as the positive class; otherwise, it is classified as the negative class.

The decision boundary can be adjusted to control the trade-off between false positives and false negatives

Advantages of Logistic Regression

Logistic Regression is computationally efficient and can handle large datasets.

It provides interpretable results by estimating the impact of each input variable on the output probability.

Logistic Regression can handle both categorical and continuous input variables.

It is robust to noise and outliers in the data.

Limitations of Logistic Regression

Logistic Regression assumes a linear relationship between the input variables and the log-odds of the output probability.

It may not perform well if the classes are not linearly separable.

It is sensitive to irrelevant or correlated features.

Logistic Regression is not suitable for multi-class classification without modifications..

Real-world Use Cases of Logistic Regression

Predicting customer churn in a telecommunications company based on customer demographics and usage patterns.

Medical diagnosis, such as predicting the likelihood of a patient having a certain disease based on symptoms and test results.

Credit scoring, determining the probability of default for loan applicants based on their financial information.

Spam email classification based on the content and metadata of emails.

Sentiment analysis, classifying the sentiment of customer reviews as positive or negativ..

Type of Logistic Regression:

On the basis of the categories, Logistic Regression can be classified into three types:

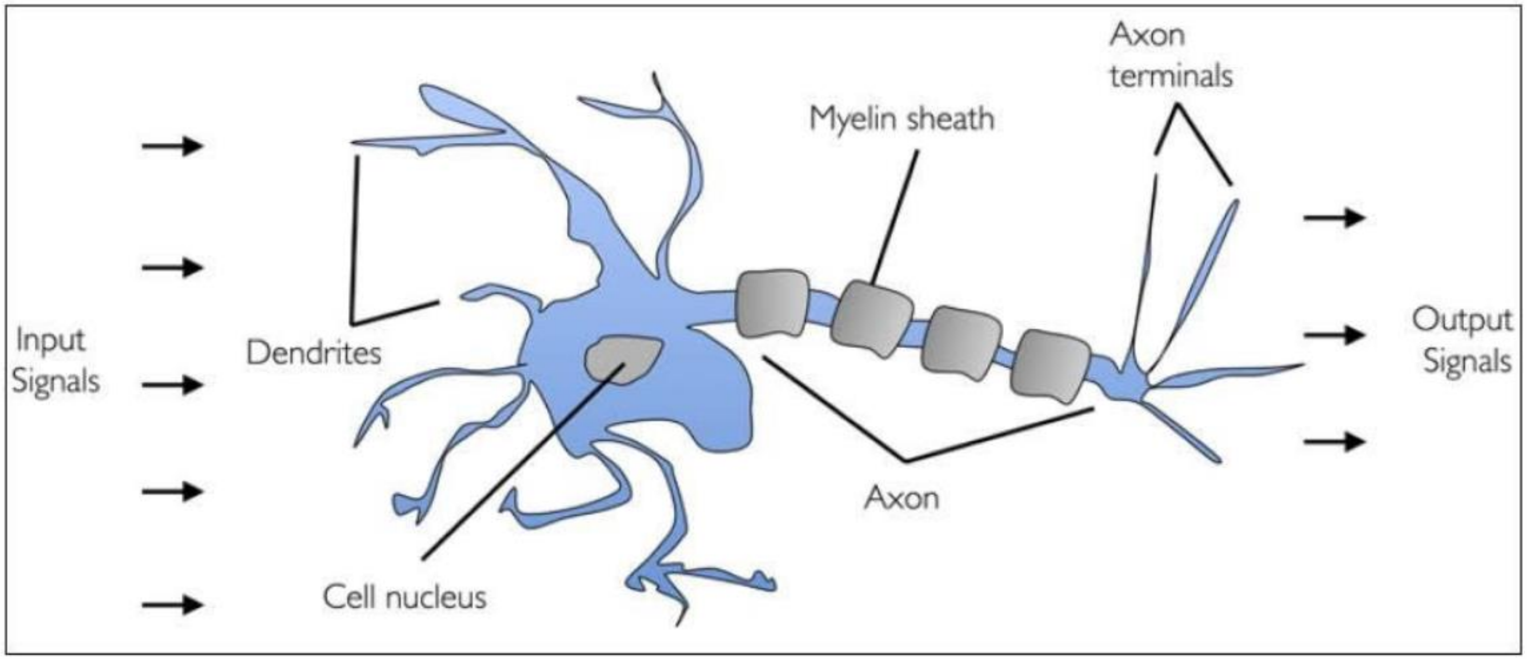
Binomial: In binomial Logistic regression, there can be only two possible types of the dependent variables, such as 0 or 1, Pass or Fail, etc.

Multinomial: In multinomial Logistic regression, there can be 3 or more possible unordered types of the dependent variable, such as "cat", "dogs", or "sheep“

Ordinal: In ordinal Logistic regression, there can be 3 or more possible ordered types of dependent variables, such as "low", "Medium", or "High".

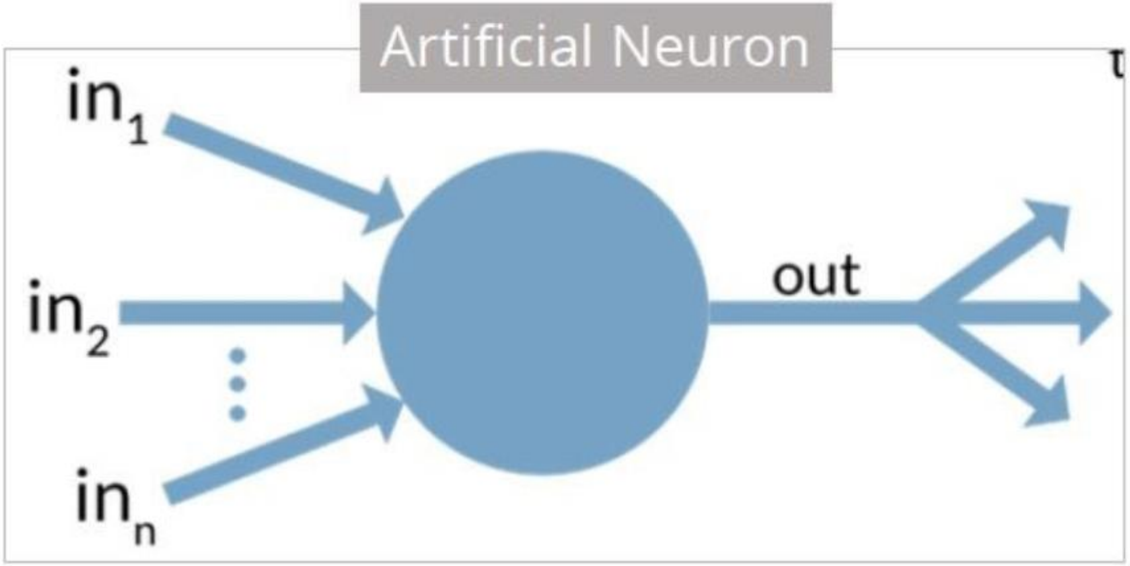
Perceptrons

Biological Neuron



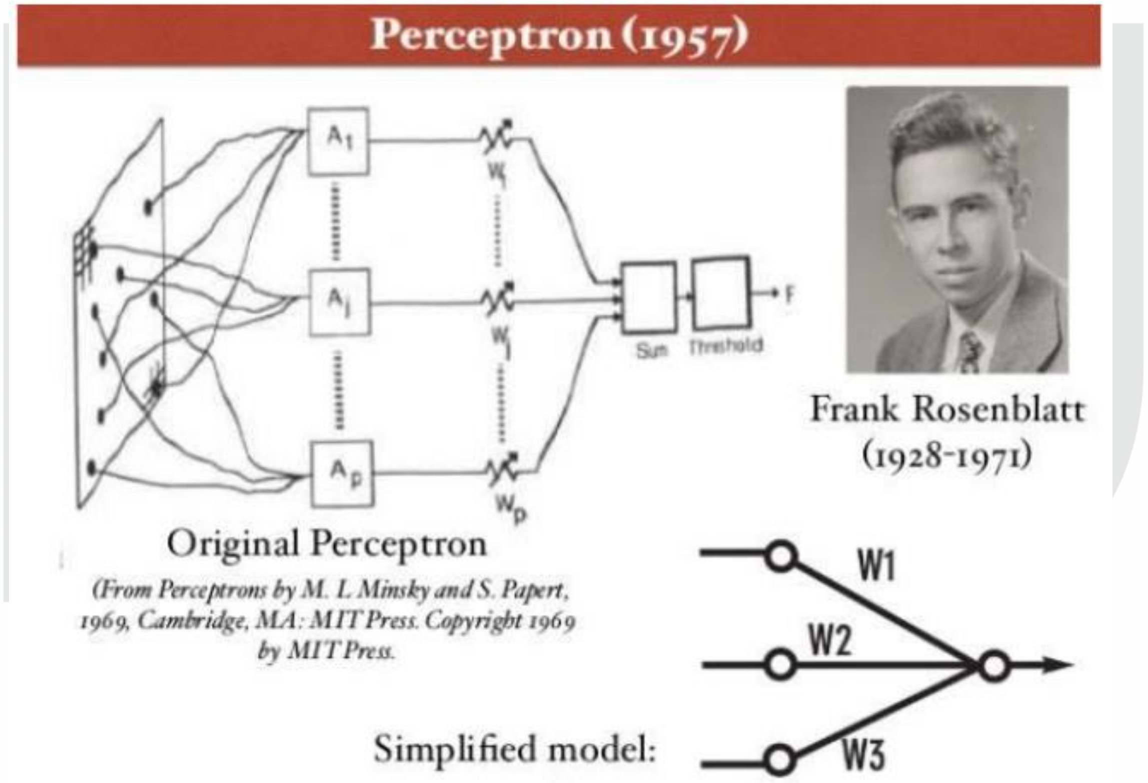
Artificial Neuron

Researchers Warren McCullock and Walter Pitts published their first concept of simplified brain cell in 1943.
This was called McCullock-Pitts (MCP) neuron.
They described such a nerve cell as a simple logic gate with binary outputs. •
Multiple signals arrive at the dendrites and are then integrated into the cell body, and, if the accumulated signal exceeds a certain threshold, an output signal is generated that will be passed on by the axon.



Perceptron

A perceptron is a neural network unit (an artificial neuron) that does certain computations to detect features or business intelligence in the input data.



Perceptron

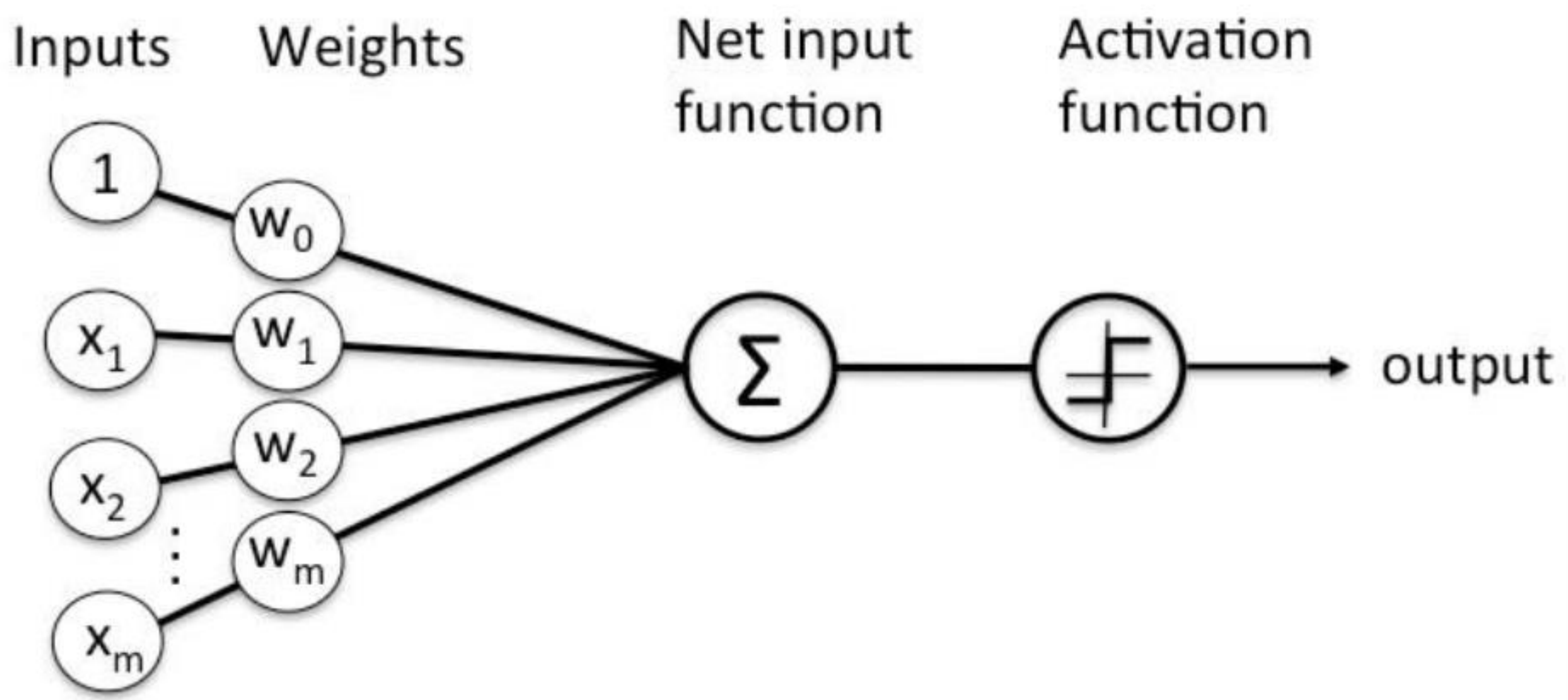
Perceptron was introduced by Frank Rosenblatt in 1957.

He proposed a Perceptron learning rule based on the original MCP neuron.

A Perceptron is an algorithm for supervised learning of binary classifiers.

This algorithm enables neurons to learn and processes elements in the training set one at a time.

Perceptron


Inputs
Weights
Net input function
Activation function
1
x_1
x_2
x_m
w_0
w_1
w_2
w_m
Σ
output

Perceptron

There are two types of Perceptrons

Single layer and Multilayer.

Single layer Perceptrons can learn only linearly separable patterns.

Multilayer Perceptrons or feed forward neural networks with two or more layers have the greater processing power.

The Perceptron algorithm learns the weights for the input signals in order to draw a linear decision boundary.

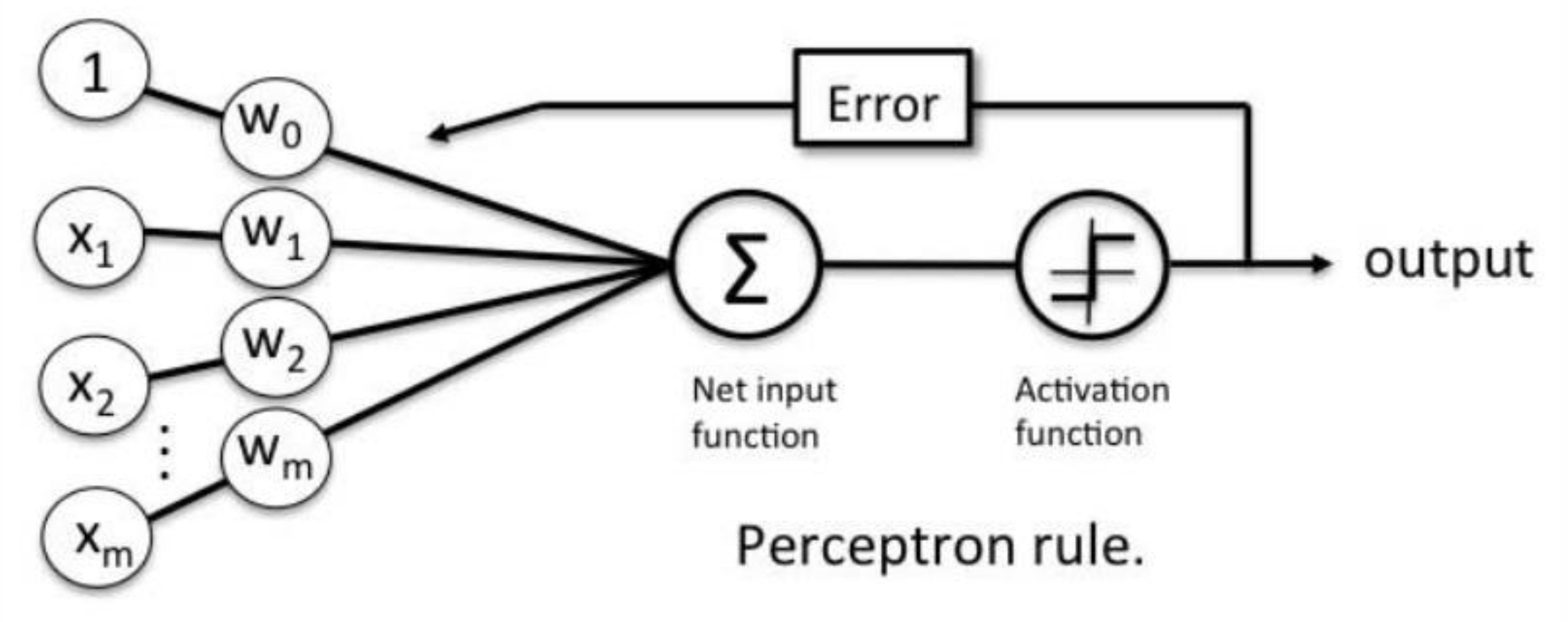
This enables you to distinguish between the two linearly separable classes +1 and -1. •

Note: Supervised Learning is a type of Machine Learning used to learn models from labeled training data. It enables output prediction for future or unseen data.

Perceptron Learning Rule

Perceptron Learning Rule states that the algorithm would automatically learn the optimal weight coefficients.

The input features are then multiplied with these weights to determine if a neuron fires or not.



Perceptron rule.

Perceptron function

Perceptron is a function that maps its input “x,” which is multiplied with the learned weight coefficient; an output value ”f(x)”is generated.

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

In the equation given above:
“w” = vector of real-valued weights
“b” = bias (an element that adjusts the boundary away from origin without any dependence on the input value)
“x” = vector of input x values

Perceptron function

“m” = number of inputs to the Perceptron

$$\sum_{i=1}^{m} w_i x_i$$

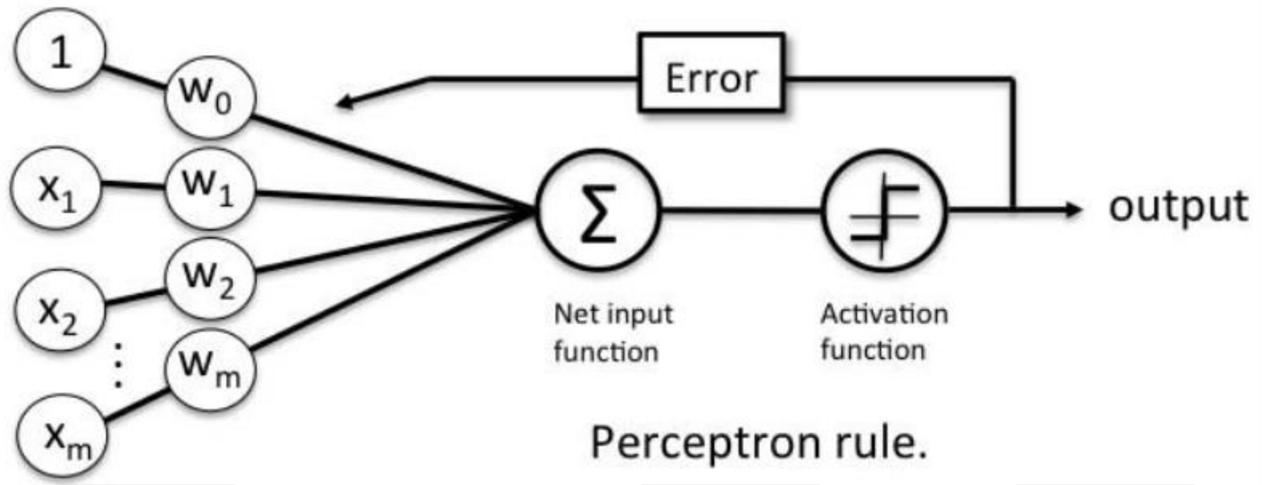
The output can be represented as “1” or “0.” It can also be represented as “1” or “-1” depending on which activation function is used.

Inputs of Perceptron

A Perceptron accepts inputs, moderates them with certain weight values, then applies the transformation function to output the final result.

The above below shows a Perceptron with a Boolean output.

Perceptron function



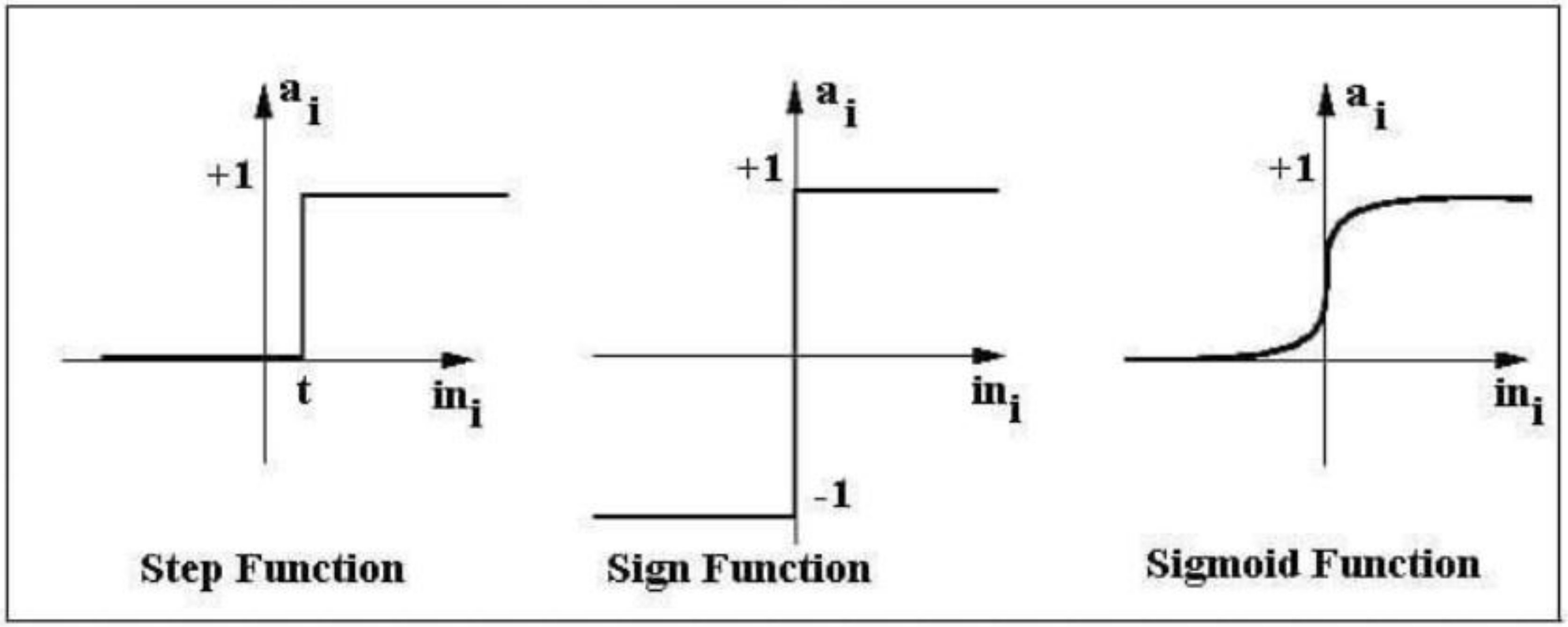
A Boolean output is based on inputs such as salaried, married, age, past credit profile, etc. It has only two values: Yes and No or True and False.

The summation function "∑" multiplies all inputs of "x" by weights "w" and then adds them up as follows:

$$w_0 + w_1 x_1 + w_2 x_2 + \cdots + w_n x_n$$

Activation function

The activation function applies a step rule (convert the numerical output into +1 or -1) to check if the output of the weighting function is greater than zero or not



Example

For example: If ∑ wixi> 0 => then final output “o” = 1 (issue bank loan)
Else, final output “o” = -1 (deny bank loan)

Step function gets triggered above a certain value of the neuron output; else it outputs zero. Sign Function outputs +1 or -1 depending on whether neuron output is greater than zero or not. Sigmoid is the S-curve and outputs a value between 0 and 1.

Output of Perceptron

Perceptron with a Boolean output:
Inputs: x1…xn
Output: o(x1….xn)

$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1x_1 + w_2x_2 + \cdots + w_nx_n > 0 \\ -1 & \text{otherwise} \end{cases}$$

Weights: wi=> contribution of input xi to the Perceptron output;
w0=> bias or threshold

Support Vector Machines (SVM)

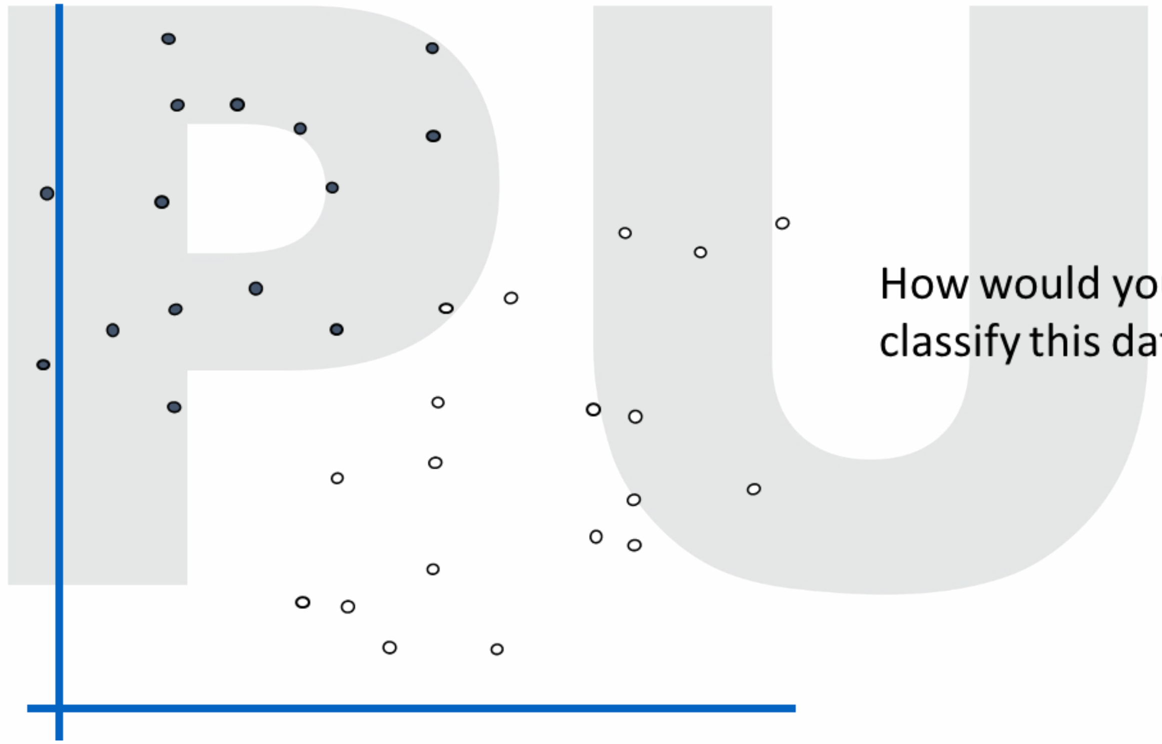
Linear Classifiers

$$\begin{tikzcd} & \alpha \arrow[d] & \\ \boldsymbol{x} \arrow[r] & \boxed{f} \arrow[r] & y^{est} \end{tikzcd}$$

$f(\mathbf{x}, \mathbf{w}, b) = sign(\mathbf{w} \cdot \mathbf{x} - b)$

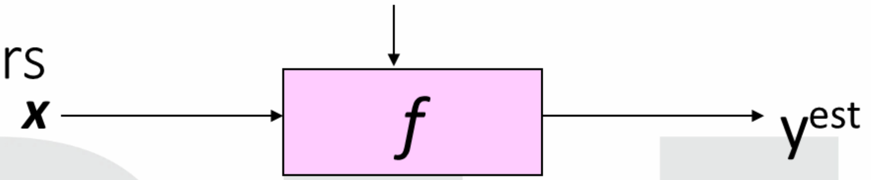
● denotes +1

○ denotes -1



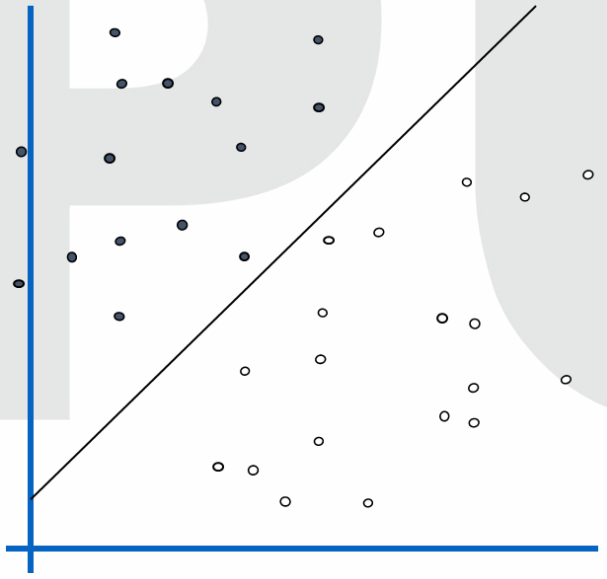
How would you classify this data?

Linear Classifiers



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\,\mathbf{x} - b)$

● denotes +1
○ denotes -1



How would you classify this data?

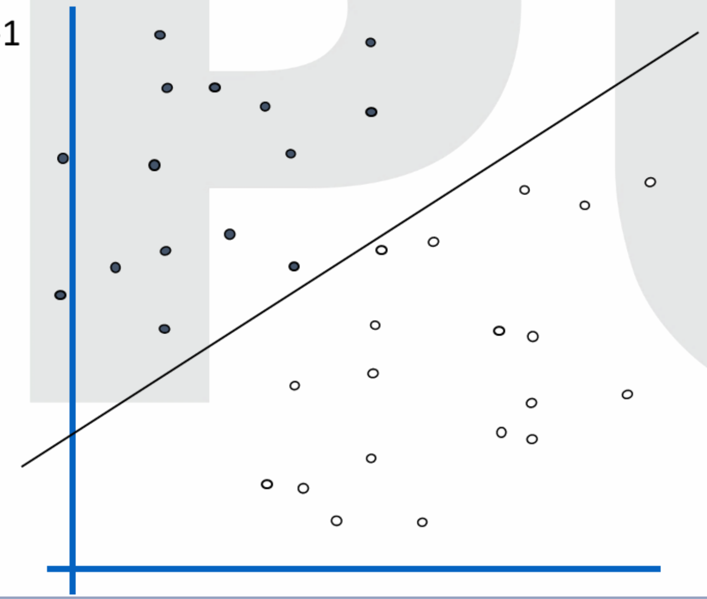
Linear Classifiers

$$\begin{tikzcd} x \arrow[r] & \boxed{f} \arrow[r] & y^{est} \end{tikzcd}$$

$f(x,w,b) = sign(w. x - b)$

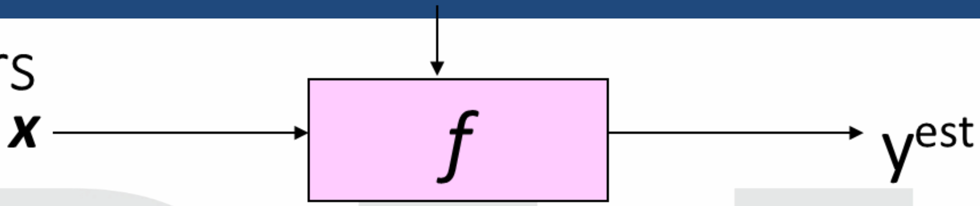
● denotes +1

○ denotes -1



How would you classify this data?

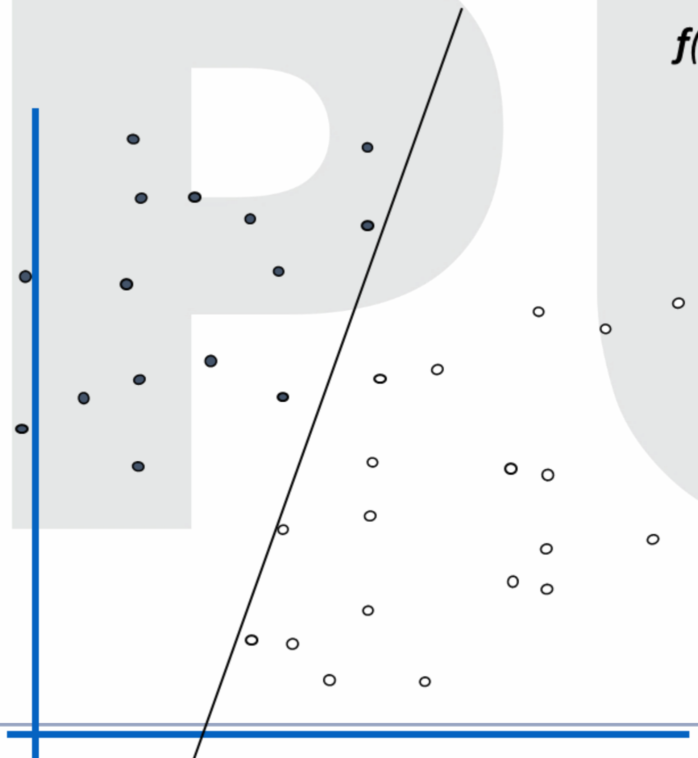
Linear Classifiers



$f(x,w,b) = sign(w. x - b)$

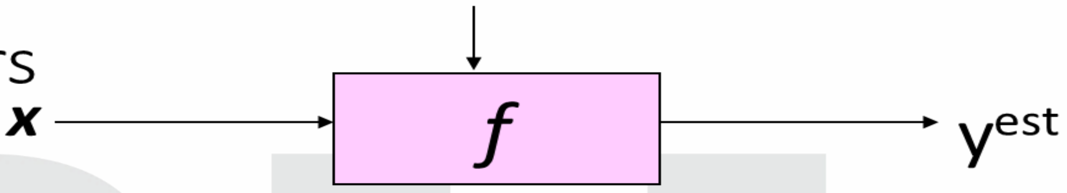
● denotes +1

○ denotes -1



How would you classify this data?

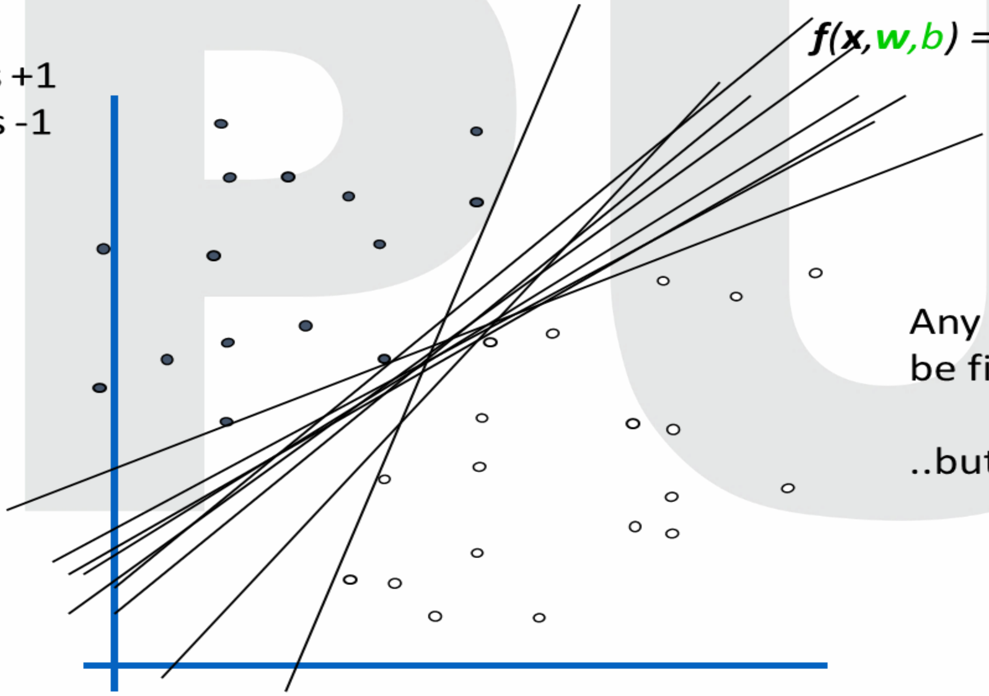
Linear Classifiers



$f(\mathbf{x},\mathbf{w},b) = sign(\mathbf{w}.\,\mathbf{x} - b)$

● denotes +1

○ denotes -1



Any of these would be fine..

..but which is best?

Classifier Margin

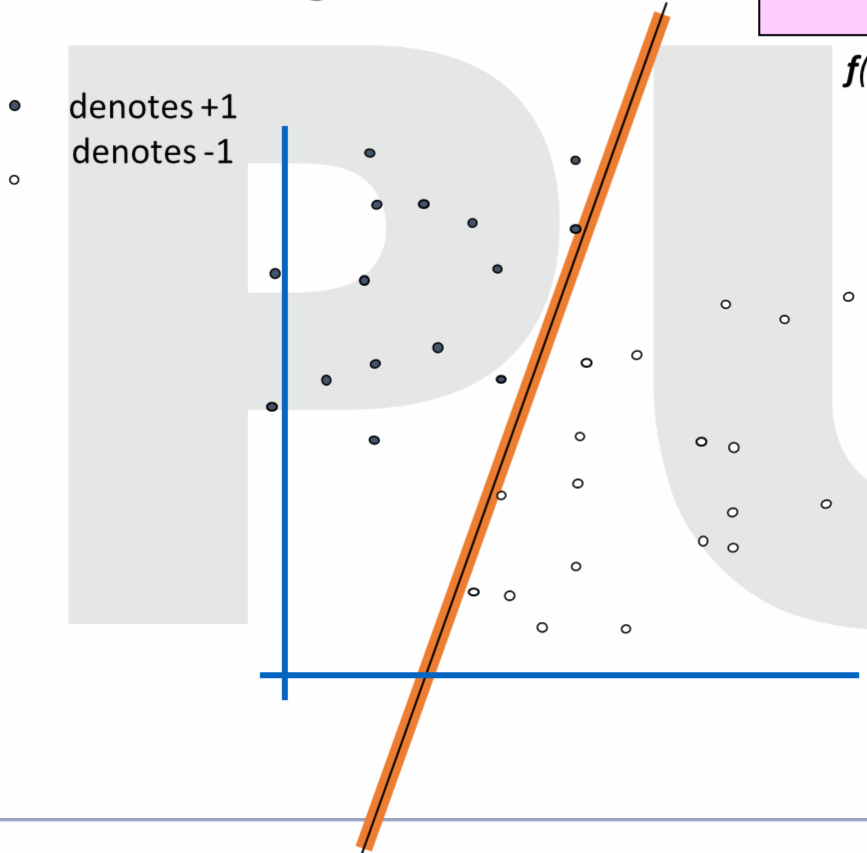
$$x \longrightarrow \boxed{f} \longrightarrow y^{est}$$

α

$f(x,w,b) = sign(w. x - b)$

● denotes +1

○ denotes -1



Define the margin of a linear classifier as the width that the boundary could be increased by before hitting a datapoint.

In machine learning, **support vector machines** (**SVMs**, also **support vector networks**) are supervised max-margin models with associated learning algorithms that analyze data for classification and regression analysis.

Developed at AT&T Bell Laboratories by Vladimir Vapnik with colleagues (Boser et al., 1992, Guyon et al., 1993, Cortes and Vapnik, 1995,Vapnik et al., 1997) SVMs are one of the most studied models, being based on statistical learning frameworks of VC theory proposed by Vapnik (1982, 1995) and Chervonenkis (1974).

In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, which represent the data only through a set of pairwise similarity comparisons between the original data observations and representing the data by these transformed coordinates in the higher dimensional feature space.

Thus, SVMs use the kernel trick to implicitly mapping their inputs into high-dimensional feature spaces. Being max-margin models, SVMs are resilient to noisy data (for example, mis-classified examples). SVMs can also be used for regression tasks, where the objective becomes ϵ-sensitive.

SVMs are based on the idea of finding a hyperplane that best divides a data set into two classes as shown here. As we're in a two-dimensional space, you can think of the hyperplane as a line that linearly separates the blue points from the red points.

ADVANTAGES

- Accurate in high dimension place
- Memory efficient

DISADVANTAGES

- Small datasets
- Prone to overfitting

APPLICATIONS

- Image Recognition
- Spam detection

SVMs can be used to solve various real-world problems:
SVMs are helpful in text and hypertext categorization, as their application can significantly reduce the need for labeled training instances in both the standard inductive and transductive settings.

Some methods for shallow semantic parsing are based on support vector machines. Classification of images can also be performed using SVMs.
Experimental results show that SVMs achieve significantly higher search accuracy than traditional query refinement schemes after just three to four rounds of relevance feedback.

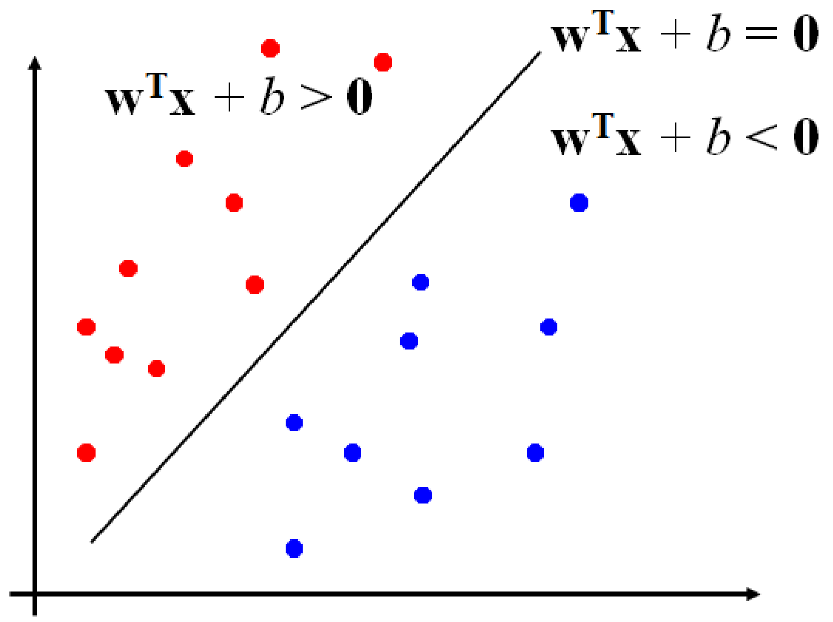
This is also true for image segmentation systems, including those using a modified version SVM that uses the privileged approach as suggested by Vapnik.

Classification of satellite data like SAR data using supervised SVM.
Hand-written characters can be recognized using SVM.

Linear SVM

Perceptron Revisited: Linear Separators

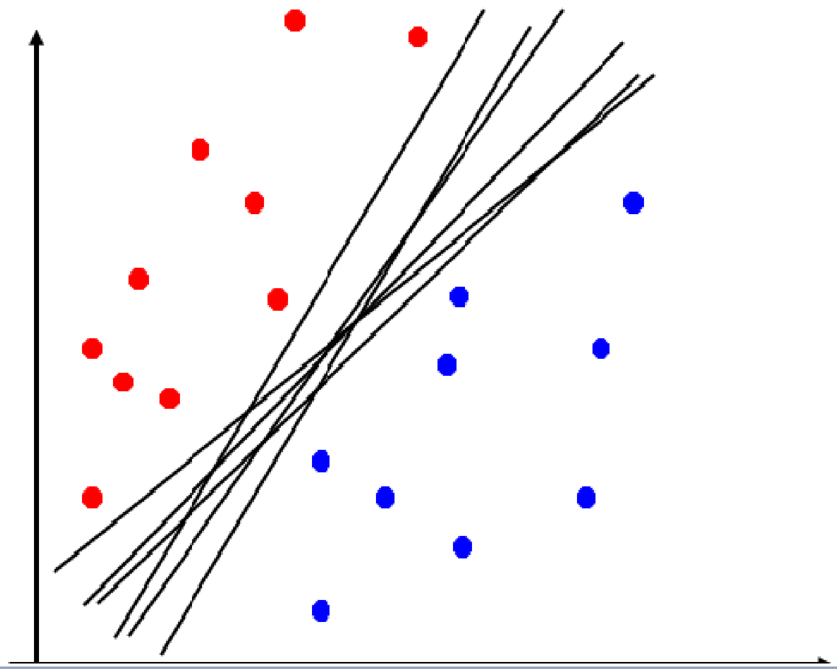
- Binary classification can be viewed as the task of separating classes in feature space:


$\mathbf{w}^T\mathbf{x} + b = 0$
$\mathbf{w}^T\mathbf{x} + b > 0$
$\mathbf{w}^T\mathbf{x} + b < 0$

$$f(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x} + b)$$

Linear Separators

Which of the linear separators is optimal?

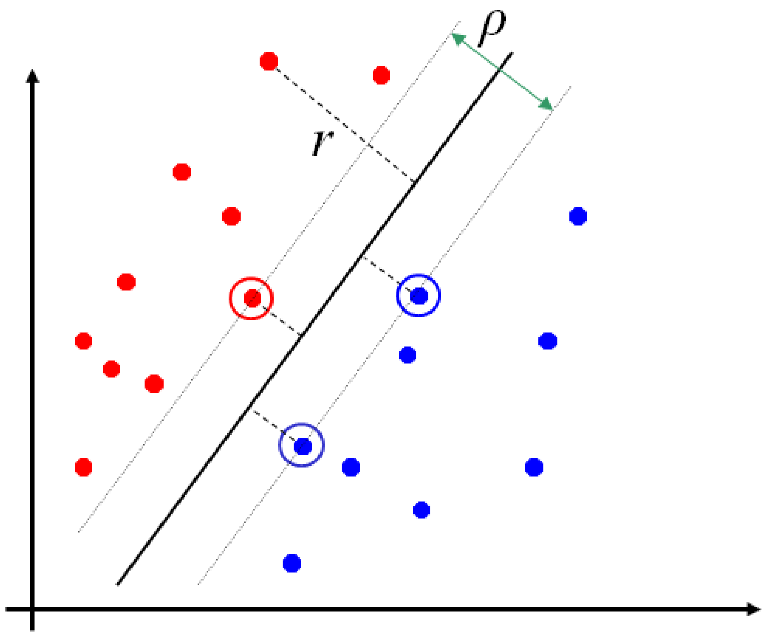


Classification Margin

Distance from example $\mathbf{x}_i$ to the separator is $r = \frac{\mathbf{w}^T\mathbf{x}_i + b}{\|\mathbf{w}\|}$

Examples closest to the hyperplane are ***support vectors***.

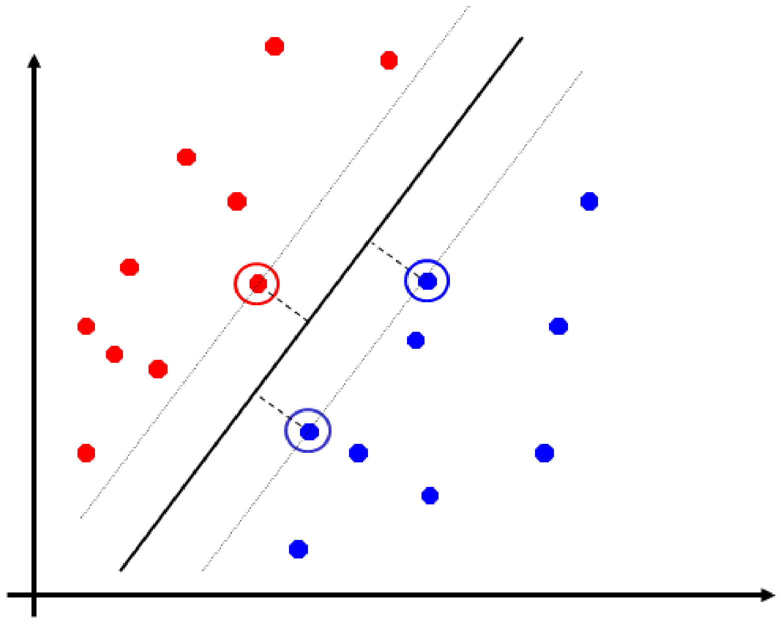
Margin ρ of the separator is the distance between support vectors.



Maximum Margin Classification

Maximizing the margin is good according to intuition and PAC theory.

Implies that only support vectors matter; other training examples are ignorable.



Linear SVM Mathematically

Let training set $\{(\mathbf{x}_i, y_i)\}_{i=1..n}$, $\mathbf{x}_i \in \mathbf{R}^d$, $y_i \in \{-1, 1\}$ be separated by a hyperplane with margin ρ. Then for each training example $(\mathbf{x}_i, y_i)$:

$$\begin{array}{ll} \mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b \leq -\rho/2 & \text{if } y_i = -1 \\ \mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b \geq \rho/2 & \text{if } y_i = 1 \end{array} \quad \Leftrightarrow \quad y_i(\mathbf{w}^{\mathbf{T}}\mathbf{x}_i + b) \geq \rho/2$$

For every support vector $\mathbf{x}_s$ the above inequality is an equality. After rescaling $\mathbf{w}$ and b by $\rho/2$ in the equality, we obtain that distance between each $\mathbf{x}_s$ and the hyperplane is $r = \frac{y_s(\mathbf{w}^T\mathbf{x}_s + b)}{\|\mathbf{w}\|} = \frac{1}{\|\mathbf{w}\|}$

Then the margin can be expressed through (rescaled) $\mathbf{w}$ and b as:

$$\rho = 2r = \frac{2}{\|\mathbf{w}\|}$$

Linear SVMs Mathematically (cont.)

- Then we can formulate the *quadratic optimization problem:*

Find $\mathbf{w}$ and b such that

$$\rho = \frac{2}{\|\mathbf{w}\|} \quad \text{is maximized}$$

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $\quad y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Which can be reformulated as:

Find $\mathbf{w}$ and b such that

$\mathbf{\Phi}(\mathbf{w}) = ||\mathbf{w}||^2 = \mathbf{w}^T\mathbf{w}$ is minimized

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $\quad y_i\,(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Solving the Optimization Problem

Find **w** and b such that

$\mathbf{\Phi(w)} = \mathbf{w}^T\mathbf{w}$ is minimized

and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i (\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Need to optimize a *quadratic* function subject to *linear* constraints.

Quadratic optimization problems are a well-known class of mathematical programming problems for which several (non-trivial) algorithms exist.

The solution involves constructing a *dual problem* where a *Lagrange multiplier* α_i is associated with every inequality constraint in the primal (original) problem:

Find $\alpha_1 \ldots \alpha_n$ such that

Q(**α**) $= \Sigma\alpha_i - \frac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized and

(1) $\Sigma\alpha_i y_i = 0$

(2) $\alpha_i \geq 0$ for all α_i

The Optimization Problem Solution

Given a solution $\alpha_1 \ldots \alpha_n$ to the dual problem, solution to the primal is:

$$\boxed{\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x}_i \qquad b = y_k - \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x}_k \quad \text{for any } \alpha_k > 0}$$

Each non-zero α_i indicates that corresponding $\mathbf{x}_i$ is a support vector.

Then the classifying function is (note that we don't need **w** explicitly):

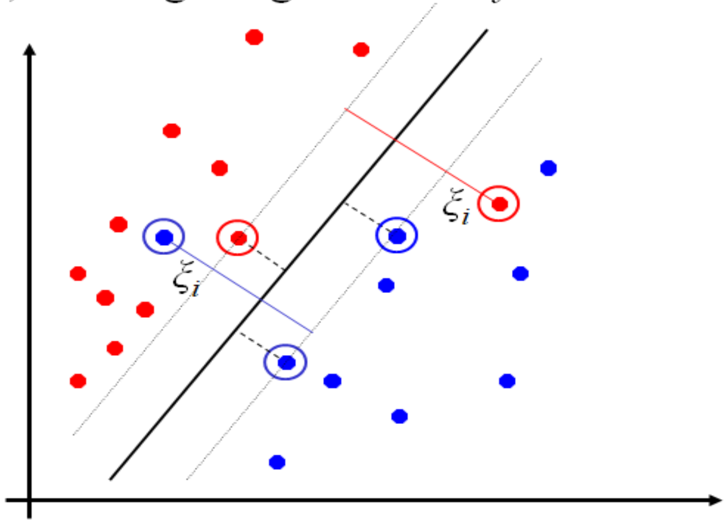
$$\boxed{f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b}$$

Notice that it relies on an *inner product* between the test point **x** and the support vectors **x**$_i$ – we will return to this later.
Also keep in mind that solving the optimization problem involved computing the inner products $\mathbf{x}_i^T\mathbf{x}_j$ between all training points.

Soft Margin Classification

What if the training set is not linearly separable?

Slack variables ξ_i can be added to allow misclassification of difficult or noisy examples, resulting margin called *soft*.



Soft Margin Classification Mathematically

The old formulation:

Find **w** and b such that
$\Phi(\mathbf{w}) = \mathbf{w}^T\mathbf{w}$ is minimized
and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1$

Modified formulation incorporates slack variables:

Find **w** and b such that
$\Phi(\mathbf{w}) = \mathbf{w}^T\mathbf{w} + C\Sigma\xi_i$ is minimized
and for all $(\mathbf{x}_i, y_i)$, $i=1..n$: $y_i(\mathbf{w}^T\mathbf{x}_i + b) \geq 1 - \xi_i$, $\xi_i \geq 0$

Parameter C can be viewed as a way to control overfitting: it "trades off" the relative importance of maximizing the margin and fitting the training data.

Soft Margin Classification – Solution

- Dual problem is identical to separable case (would *not* be identical if the 2-norm penalty for slack variables $C\Sigma\xi_i^2$ was used in primal objective, we would need additional Lagrange multipliers for slack variables):

Find $\alpha_1 \ldots \alpha_N$ such that
$\mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - ½\Sigma\Sigma\alpha_i\alpha_j y_i y_j \mathbf{x}_i^T\mathbf{x}_j$ is maximized and
(1) $\Sigma\alpha_i y_i = 0$
(2) $0 \leq \alpha_i \leq C$ for all α_i

- Again, $\mathbf{x}_i$ with non-zero α_i will be support vectors.
- Solution to the dual problem is:

$\mathbf{w} = \Sigma\alpha_i y_i \mathbf{x}_i$
$b = y_k(1 - \xi_k) - \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x}_k$ for any k s.t. $\alpha_k > 0$

Again, we don't need to compute $\mathbf{w}$ explicitly for classification:

$f(\mathbf{x}) = \Sigma\alpha_i y_i \mathbf{x}_i^T\mathbf{x} + b$

Theoretical Justification for Maximum Margins

Vapnik has proved the following:

The class of optimal linear separators has VC dimension h bounded from above as

$$h \leq \min\left\{\left\lceil \frac{D^2}{\rho^2} \right\rceil, m_0\right\} + 1$$

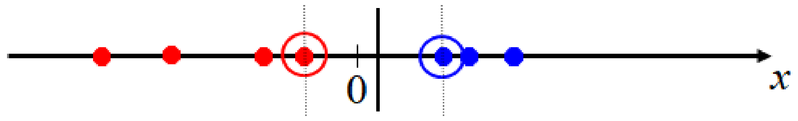
where ρ is the margin, D is the diameter of the smallest sphere that can enclose all of the training examples, and m_0 is the dimensionality.

Intuitively, this implies that regardless of dimensionality m_0 we can minimize the VC dimension by maximizing the margin ρ.

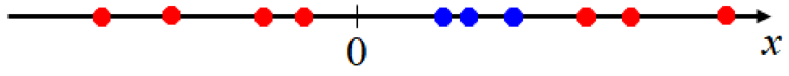
Thus, complexity of the classifier is kept small regardless of dimensionality.

Non-Linear SVM

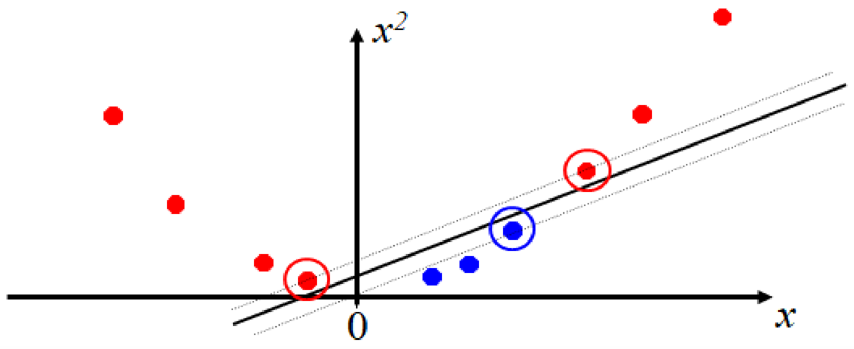
Datasets that are linearly separable with some noise work out great:



But what are we going to do if the dataset is just too hard?

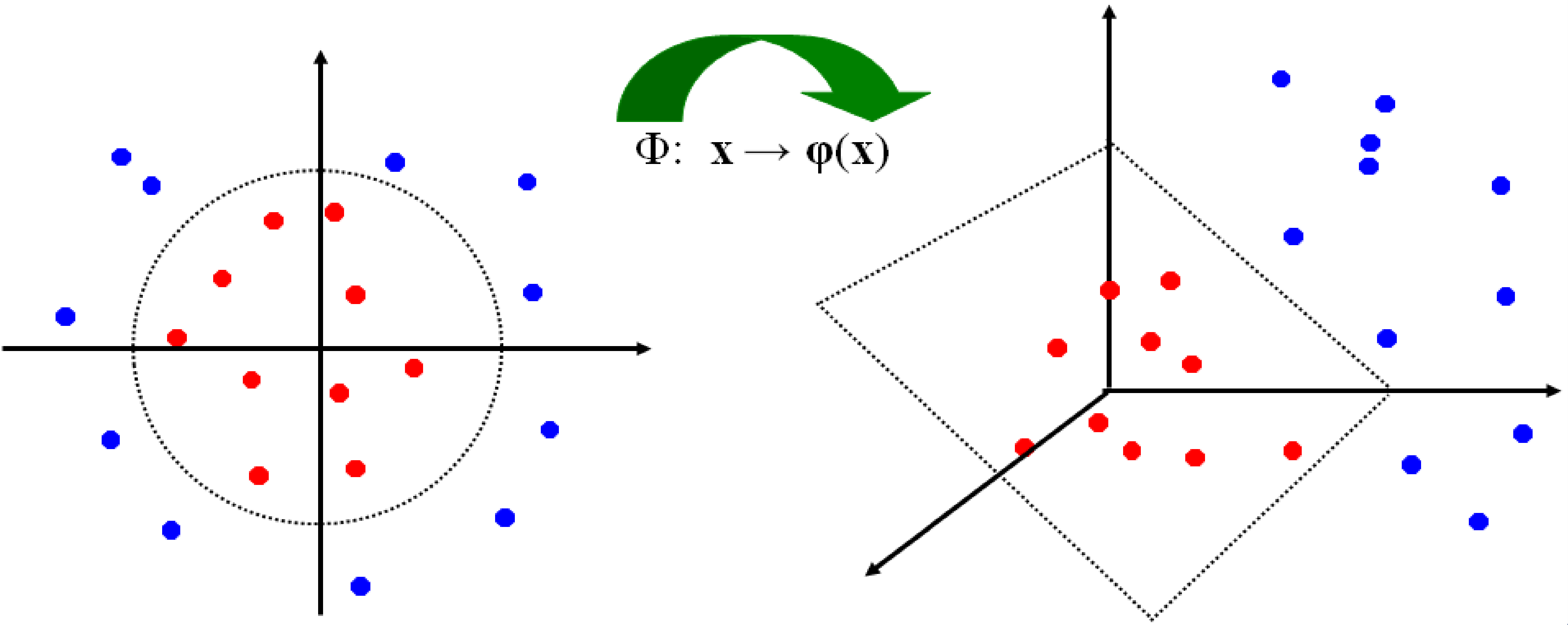


How about… mapping data to a higher-dimensional space:



Non-linear SVMs: Feature spaces

- General idea: the original feature space can always be mapped to some higher-dimensional feature space where the training set is separable:



The "Kernel Trick"

The linear classifier relies on inner product between vectors $K(\mathbf{x}_i,\mathbf{x}_j)=\mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j$

If every datapoint is mapped into high-dimensional space via some transformation Φ: $\mathbf{x} \longrightarrow \varphi(\mathbf{x})$, the inner product becomes:

$$K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^{\mathbf{T}}\varphi(\mathbf{x}_j)$$

A *kernel function* is a function that is eqiuvalent to an inner product in some feature space.

Example:

2-dimensional vectors $\mathbf{x}=[x_1\ \ x_2]$; let $K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j)^2$,

Need to show that $K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^{\mathbf{T}}\varphi(\mathbf{x}_j)$:

$$K(\mathbf{x}_i,\mathbf{x}_j)=(1 + \mathbf{x}_i^{\mathbf{T}}\mathbf{x}_j)^2_{,}= 1+ x_{i1}^2x_{j1}^2 + 2\, x_{i1}x_{j1}\, x_{i2}x_{j2}+ x_{i2}^2x_{j2}^2 + 2x_{i1}x_{j1} + 2x_{i2}x_{j2}=$$

$$= [1\ \ x_{i1}^2\ \ \sqrt{2}\, x_{i1}x_{i2}\ \ x_{i2}^2\ \ \sqrt{2}x_{i1}\ \ \sqrt{2}x_{i2}]^{\mathbf{T}}\, [1\ \ x_{j1}^2\ \ \sqrt{2}\, x_{j1}x_{j2}\ \ x_{j2}^2\ \ \sqrt{2}x_{j1}\ \ \sqrt{2}x_{j2}] =$$

$$= \varphi(\mathbf{x}_i)^{\mathbf{T}}\varphi(\mathbf{x}_j), \quad \text{where } \varphi(\mathbf{x}) = [1\ \ x_1^2\ \ \sqrt{2}\, x_1x_2\ \ x_2^2\ \ \sqrt{2}x_1\ \ \sqrt{2}x_2]$$

Thus, a kernel function *implicitly* maps data to a high-dimensional space (without the need to compute each $\varphi(\mathbf{x})$ explicitly).

What Functions are Kernels?

For some functions $K(\mathbf{x}_i,\mathbf{x}_j)$ checking that $K(\mathbf{x}_i,\mathbf{x}_j)= \varphi(\mathbf{x}_i)^{\mathbf{T}}\varphi(\mathbf{x}_j)$ can be cumbersome.

Mercer's theorem:

Every semi-positive definite symmetric function is a kernel

Semi-positive definite symmetric functions correspond to a semi-positive definite symmetric Gram matrix:

K=

$K(\mathbf{x}_1,\mathbf{x}_1)$	$K(\mathbf{x}_1,\mathbf{x}_2)$	$K(\mathbf{x}_1,\mathbf{x}_3)$	…	$K(\mathbf{x}_1,\mathbf{x}_n)$
$K(\mathbf{x}_2,\mathbf{x}_1)$	$K(\mathbf{x}_2,\mathbf{x}_2)$	$K(\mathbf{x}_2,\mathbf{x}_3)$		$K(\mathbf{x}_2,\mathbf{x}_n)$
…	…	…	…	…
$K(\mathbf{x}_n,\mathbf{x}_1)$	$K(\mathbf{x}_n,\mathbf{x}_2)$	$K(\mathbf{x}_n,\mathbf{x}_3)$	…	$K(\mathbf{x}_n,\mathbf{x}_n)$

Examples of Kernel Functions

Linear: $K(\mathbf{x}_i,\mathbf{x}_j) = \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j$

– Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is $\mathbf{x}$ itself

Polynomial of power p: $K(\mathbf{x}_i,\mathbf{x}_j) = (1 + \mathbf{x}_i^{\mathrm{T}}\mathbf{x}_j)^p$

– Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ has $\binom{d+p}{p}$ dimensions

Gaussian (radial-basis function): $K(\mathbf{x}_i,\mathbf{x}_j) = e^{-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}}$

– Mapping Φ: $\mathbf{x} \rightarrow \boldsymbol{\varphi}(\mathbf{x})$, where $\boldsymbol{\varphi}(\mathbf{x})$ is *infinite-dimensional*: every point is mapped to *a function* (a Gaussian); combination of functions for support vectors is the separator.

Higher-dimensional space still has *intrinsic* dimensionality d (the mapping is not *onto*), but linear separators in it correspond to *non-linear* separators in original space.

Non-linear SVMs Mathematically

Dual problem formulation:

$$\boxed{\begin{array}{l}\text{Find } \alpha_1 \ldots \alpha_n \text{ such that} \\ \mathbf{Q}(\boldsymbol{\alpha}) = \Sigma\alpha_i - \tfrac{1}{2}\Sigma\Sigma\alpha_i\alpha_j y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \text{ is maximized and} \\ (1)\ \ \Sigma\alpha_i y_i = 0 \\ (2)\ \alpha_i \geq 0 \text{ for all } \alpha_i\end{array}}$$

The solution is:

$$\boxed{f(\mathbf{x}) = \Sigma\alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b}$$

Optimization techniques for finding α_i's remain the same!

SVM applications

SVMs were originally proposed by Boser, Guyon and Vapnik in 1992 and gained increasing popularity in late 1990s.

SVMs are currently among the best performers for a number of classification tasks ranging from text to genomic data.

SVMs can be applied to complex data types beyond feature vectors (e.g. graphs, sequences, relational data) by designing kernel functions for such data.

SVM techniques have been extended to a number of tasks such as regression [Vapnik *et al.* ’97], principal component analysis [Schölkopf *et al.* ’99], etc.

Most popular optimization algorithms for SVMs use *decomposition* to hill-climb over a subset of α_i’s at a time, e.g. SMO [Platt ’99] and [Joachims ’99]
Tuning SVMs remains a black art: selecting a specific kernel and parameters is usually done in a try-and-see manner.

Thank You!!!

DIGITAL LEARNING CONTENT

Parul® University

www.paruluniversi