MACHINE LEARNING

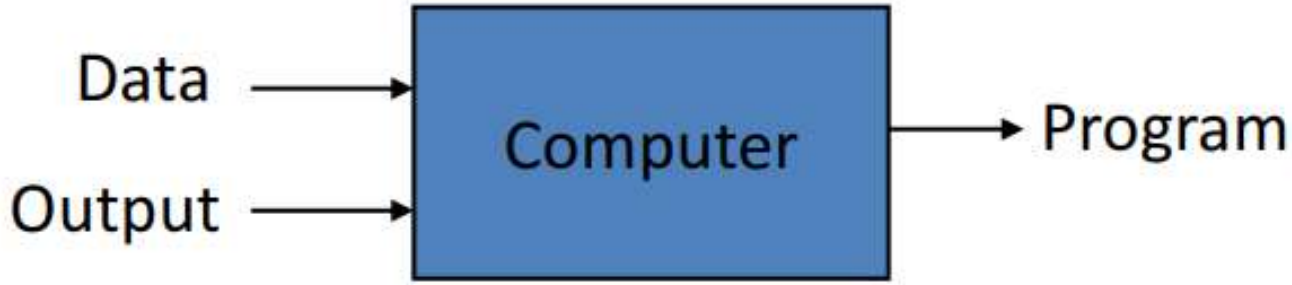
What is Machine Learning

- Machine learning is an application of artificial intelligence that involves algorithms and data that automatically analyse and make decision by itself without human intervention.
-
- It describes how computer perform tasks on their own by previous experiences.

- Therefore we can say in machine language artificial intelligence is generated on the basis of experience.

Traditional Programming

Machine Learning

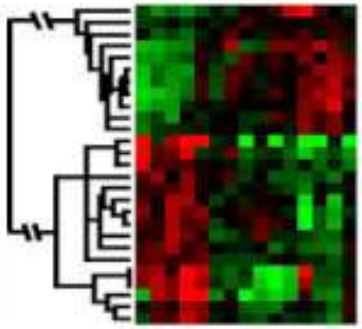


When Do We Use Machine Learning?

ML is used when:

- Human expertise does not exist (navigating on Mars)
- Humans can't explain their expertise (speech recognition)
- Models must be customized (personalized medicine)
- Models are based on huge amounts of data (genomics)



Learning isn't always useful:

- There is no need to "learn" to calculate payroll

Types of Machine Learning

There are three types of machine learning

- Supervised learning
  - Given: training data + desired outputs (labels)
- Unsupervised learning
  - Given: training data (without desired outputs)
- Semi-supervised learning
  - Given: training data + a few desired outputs
- Reinforcement learning
  - Rewards from sequence of actions

Machine Learning Uses

- Traffic prediction
- Virtual Personal Assistant
- Speech recognition
- Email spam and malware filtering
- Bioinformatics
- Natural language processing

Difference Between Machine Learning And Artificial Intelligence

- Artificial Intelligence is a concept of creating intelligent machines that stimulates human behaviour

whereas

- Machine learning is a subset of Artificial intelligence that allows machine to learn from data without being programmed.

Advantages of Machine Learning

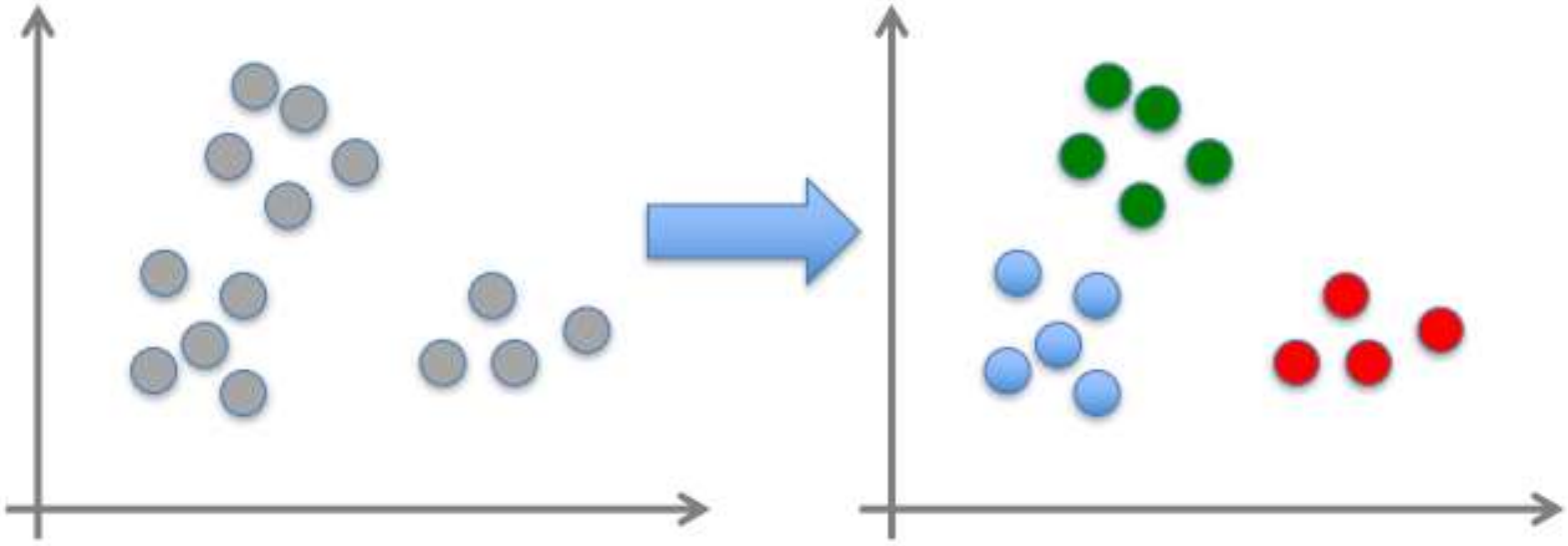
- Fast, Accurate, Efficient.
- Automation of most applications.
- Wide range of real life applications.
- Enhanced cyber security and spam detection.
- No human Intervention is needed.
- Handling multi dimensional data.

Disadvantages of Machine Learning

- It is very difficult to identify and rectify the errors.
- Data Acquisition.
- Interpretation of results Requires more time and space.

Unsupervised Learning

- Given $x_1, x_2, \ldots, x_n$ (without labels)
- Output hidden structure behind the x's
  - E.g., clustering



Steps of machine learning

- Import the dataset
- Explore the data to figure out what they look like
- Pre-process the data
- Split the data into attributes and labels
- Divide the data into training and testing sets
- Train the SVM algorithm
- Make some predictions
- Evaluate the results of the algorithm

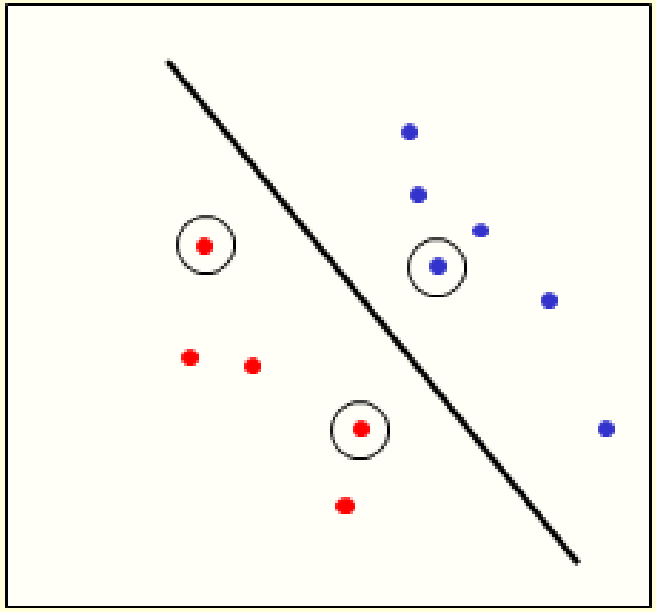
SVM

- **Support Vector Machine**(SVM) is a supervised machine learning algorithm used for both classification and regression.

- The objective of SVM algorithm is to find a hyper plane in an N-dimensional space that distinctly classifies the data points. The dimension of the hyper plane depends upon the number of features

Support Vector Machines

- The line that maximizes the minimum margin is a good bet.
  - The model class of "hyper-planes with a margin of m" has a low VC dimension if m is big.
- This maximum-margin separator is determined by a subset of the datapoints.
  - Datapoints in this subset are called "support vectors".
  - It will be useful computationally if only a small fraction of the datapoints are support vectors, because we use the support vectors to decide which side of the separator a test case is on.



The support vectors are indicated by the circles around them.

Definitions

Define the hyperplane H such that:

$\mathbf{x}_i \cdot \mathbf{w} + b \geq +1$ when $y_i = +1$

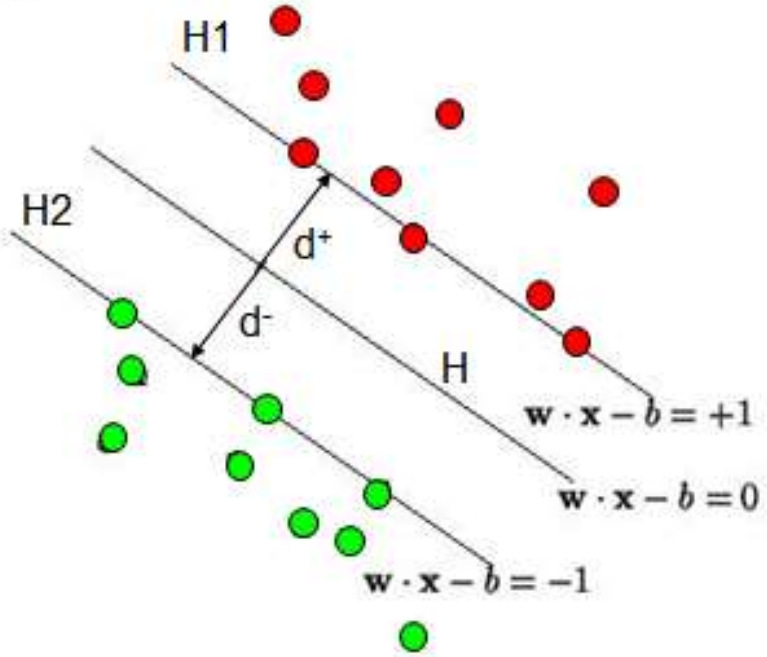
$\mathbf{x}_i \cdot \mathbf{w} + b \leq -1$ when $y_i = -1$

H1 and H2 are the planes:

H1: $\mathbf{x}_i \cdot \mathbf{w} + b = +1$

H2: $\mathbf{x}_i \cdot \mathbf{w} + b = -1$

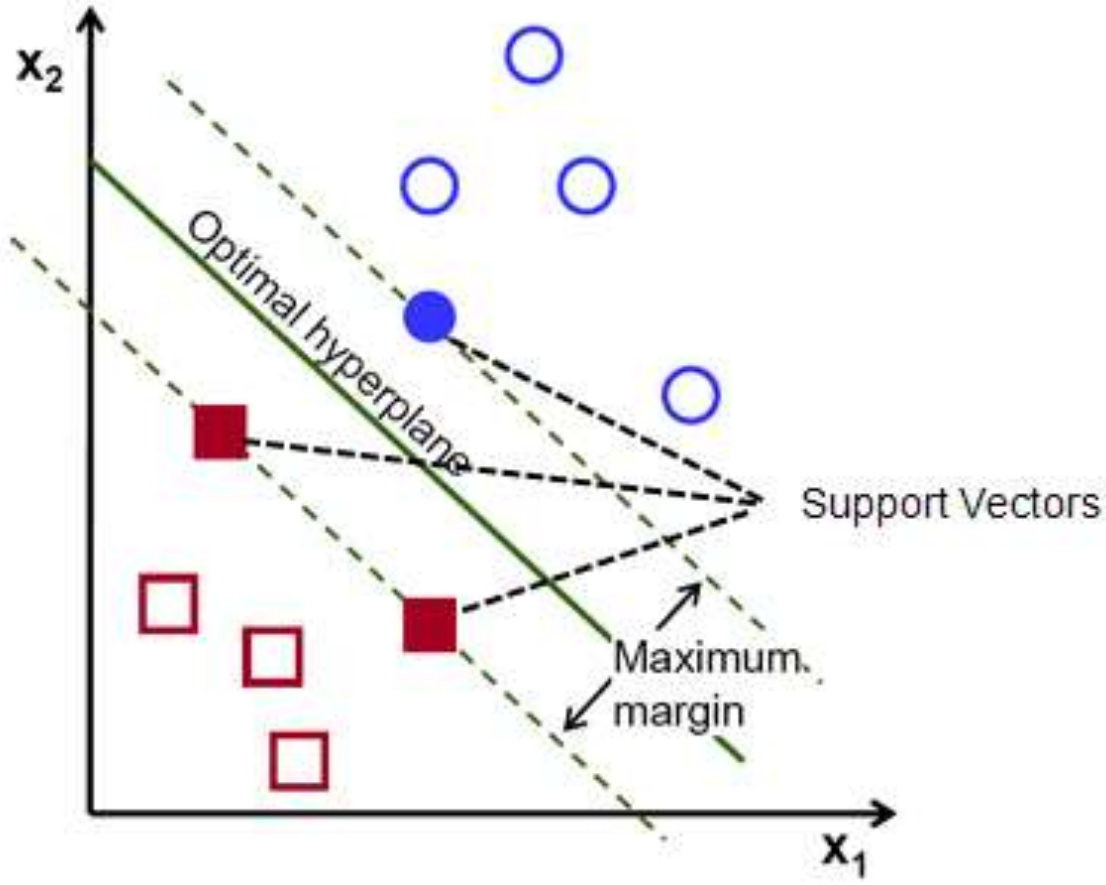
The points on the planes H1 and H2 are the Support Vectors



d+ = the shortest distance to the closest positive point

d- = the shortest distance to the closest negative point

The margin of a separating hyperplane is $d^+ + d^-$.


x_2
Optimal hyperplane
Support Vectors
Maximum
margin
x_1

Linear Kernel Function

- These are commonly recommended for text classification because most of these types of classification problems are linearly separable.
- The linear kernel works really well when there are a lot of features, and text classification problems have a lot of features. Linear kernel functions are faster than most of the others and you have fewer parameters to optimize.
- Here's the function that defines the linear kernel:

```
f(X) = w^T * X + b
```

In this equation, **w** is the weight vector that you want to minimize, **X** is the data that you're trying to classify, and **b** is the linear coefficient estimated from the training data. This equation defines the decision boundary that the SVM returns.

Gaussian Radial Basis function

- One of the most powerful and commonly used kernels in SVMs. Usually the choice for non-linear data.
- Here's the equation for an RBF kernel:

```
f(X1, X2) = exp(-gamma * ||X1 - X2||^2)
```

In this equation, **gamma** specifies how much a single training point has on the other data points around it. **||X1 - X2||** is the dot product between your features.

Polynomial Kernel Function

- The polynomial kernel isn't used in practice very often because it isn't as computationally efficient as other kernels and its predictions aren't as accurate.
- Here's the function for a polynomial kernel:

```
f(X1, X2) = (a + X1^T * X2) ^ b
```

This is one of the more simple polynomial kernel equations you can use. **f(X1, X2)** represents the polynomial decision boundary that will separate your data. **X1** and **X2** represent your data.

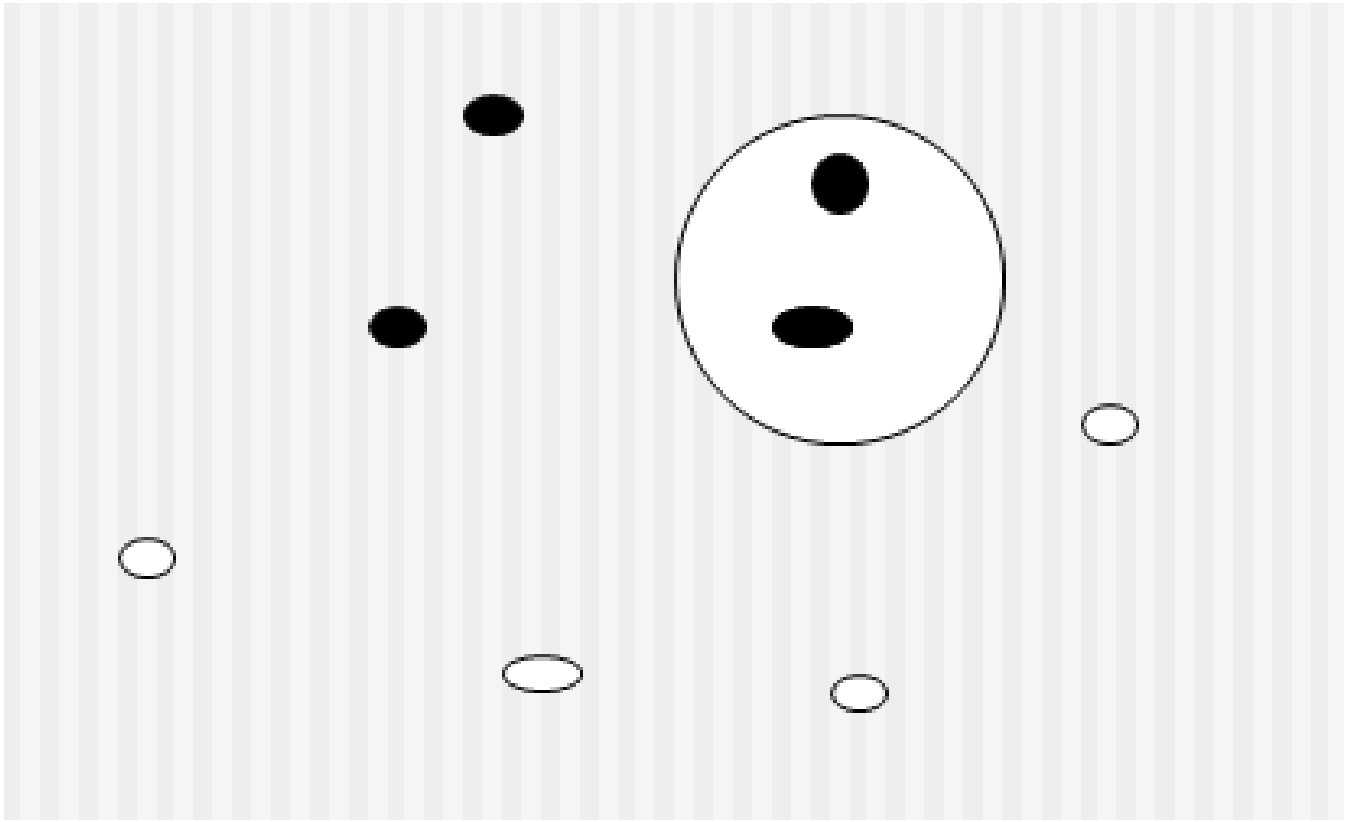
Performance

- Support Vector Machines work very well in practice.
  - The user must choose the kernel function and its parameters, but the rest is automatic.
  - The test performance is very good.
- They can be expensive in time and space for big datasets
  - The computation of the maximum-margin hyper-plane depends on the square of the number of training cases.
  - We need to store all the support vectors.
- SVM's are very good if you have no idea about what structure to impose on the task.
- The kernel trick can also be used to do PCA in a much higher-dimensional space, thus giving a non-linear version of PCA in the original space.

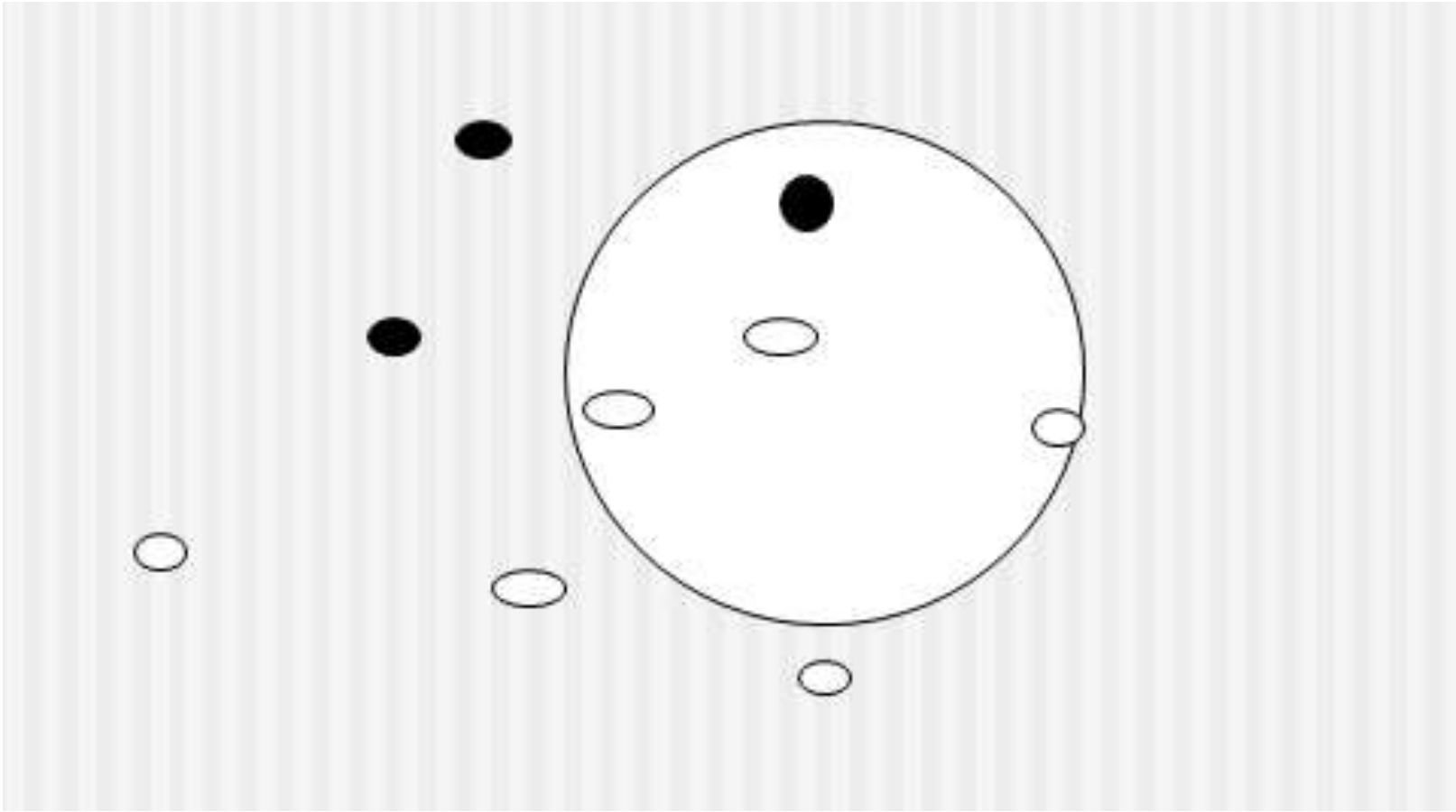
K-Nearest Neighbor Learning

- Features
  - All instances correspond to points in an n-dimensional Euclidean space
  - Classification is delayed till a new instance arrives
  - Classification done by comparing feature vectors of the different points
  - Target function may be discrete or real-valued

1-Nearest Neighbor



3-Nearest Neighbor



K-Nearest Neighbor

- An arbitrary instance is represented by

$$(a_1(x), a_2(x), a_3(x),.., a_n(x))$$

$a_i(x)$ denotes features

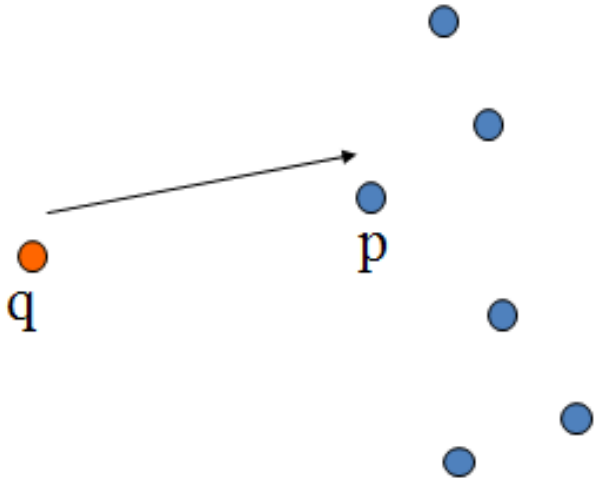
- Euclidean distance between two instances

$d(x_i, x_j)=\text{sqrt}\ (\text{sum for r=1 to n}\ (a_r(x_i) - a_r(x_j))^2)$

- Continuous valued target function
  - mean value of the k nearest training examples

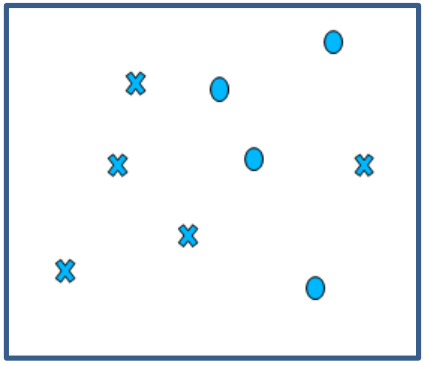
Nearest Neighbor Search

- Given: a set P of n points in R^d
- Goal: a data structure, which given a query point q, finds the *nearest neighbor* p of q in P

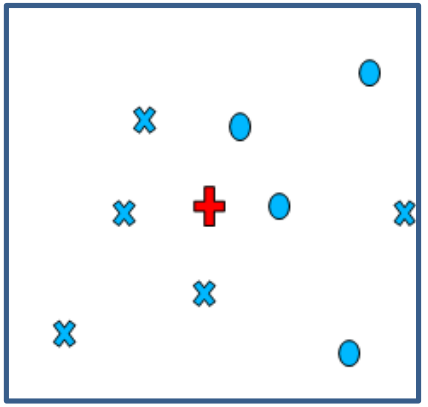


K-NN

- (K-l)-NN: Reduce complexity by having a threshold on the majority.

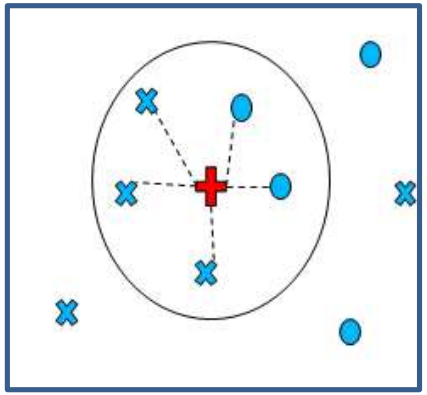


- K=5

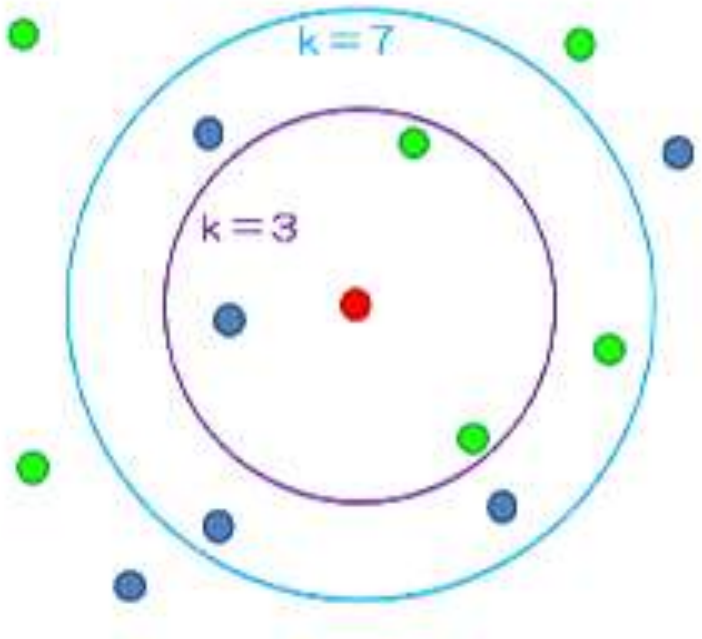


- Select 5 Nearest Neighbors as Value of K=5 by taking their Euclidean Distances

- Decide if majority of Instances over a given value of K Here, K=5.



Different Values of K



- True positive (TP): Number of correct matches.
- False negative (FN): Matches that are not correctly detected.
- False positive (FP): Matches that are incorrect.
- True negative (TN): Non-matches that are correctly rejected.
- True positive rate (TPR): $\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$
- False positive rate (FPR): $\mathrm{FPR} = \frac{\mathrm{FP}}{\mathrm{FP+TN}}$
- Accuracy (ACC): $\mathrm{ACC} = \frac{\mathrm{TP+TN}}{\mathrm{TP+FN+FP+TN}}$
- Precision: $\mathrm{P} = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$
- Recall: $\mathrm{R} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$
- Specificity: $\mathrm{S} = \frac{\mathrm{TN}}{\mathrm{TN+FP}}$

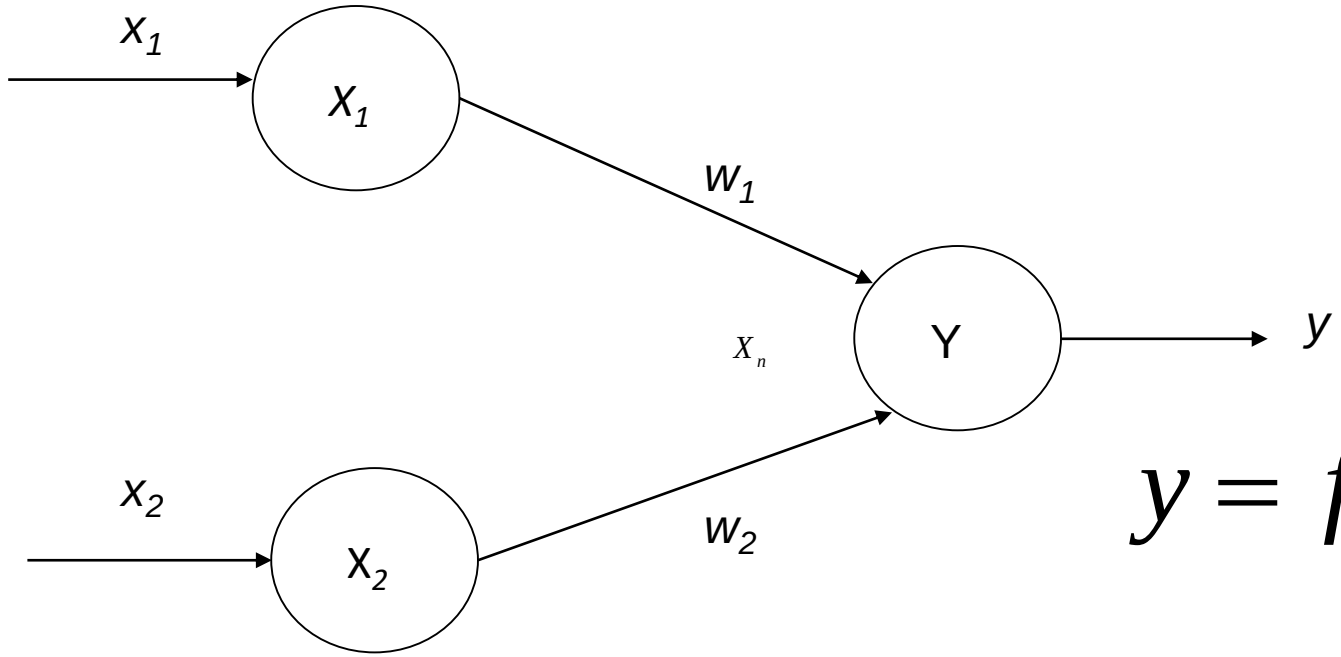
Artificial Neural Networks

- NN are constructed and implemented to model the human brain.
- Performs various tasks such as pattern-matching, classification, optimization function, approximation, vector quantization and data clustering.
- These tasks are difficult for traditional computers

.

- ANN posess a large number of processing elements called nodes/neurons which operate in parallel.
- Neurons are connected with others by connection link.
- Each link is associated with weights which contain information about the input signal.
- Each neuron has an internal state of its own which is a function of the inputs that neuron receives- Activation level

Artificial Neural Networks



$$y = f(y_{in})$$

$$y_{in} = x_1 w_1 + x_2 w_2$$

Neuron Modeling for ANN

$$o = f(\mathbf{w}^t\mathbf{x}), \text{ or}$$

$$o = f\left(\sum_{i=1}^{n} w_i x_i\right)$$

Is referred to activation function. Domain is set of activation values *net.*

$$net \triangleq \mathbf{w}^t\mathbf{x}$$

Scalar product of weight and input vector

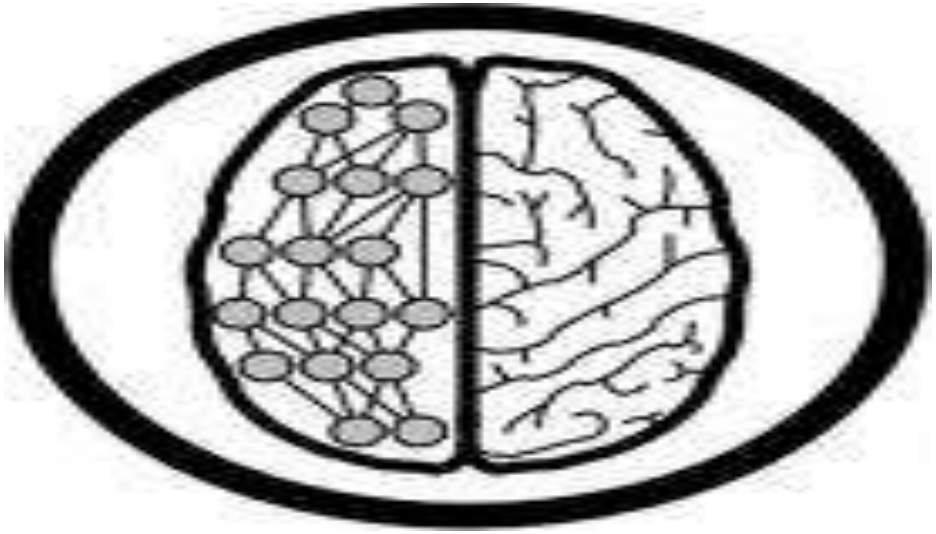
Neuron as a processing node performs the operation of summation of its weighted input.

Artificial Neural Networks

- The “building blocks” of neural networks are the **neurons**.
  - In technical systems, we also refer to them as **units** or **nodes**.
- Basically, each neuron
  - receives **input** from many other neurons.
  - changes its internal state (**activation**) based on the current input.
  - sends **one output signal** to many other neurons, possibly including its input neurons (recurrent network).

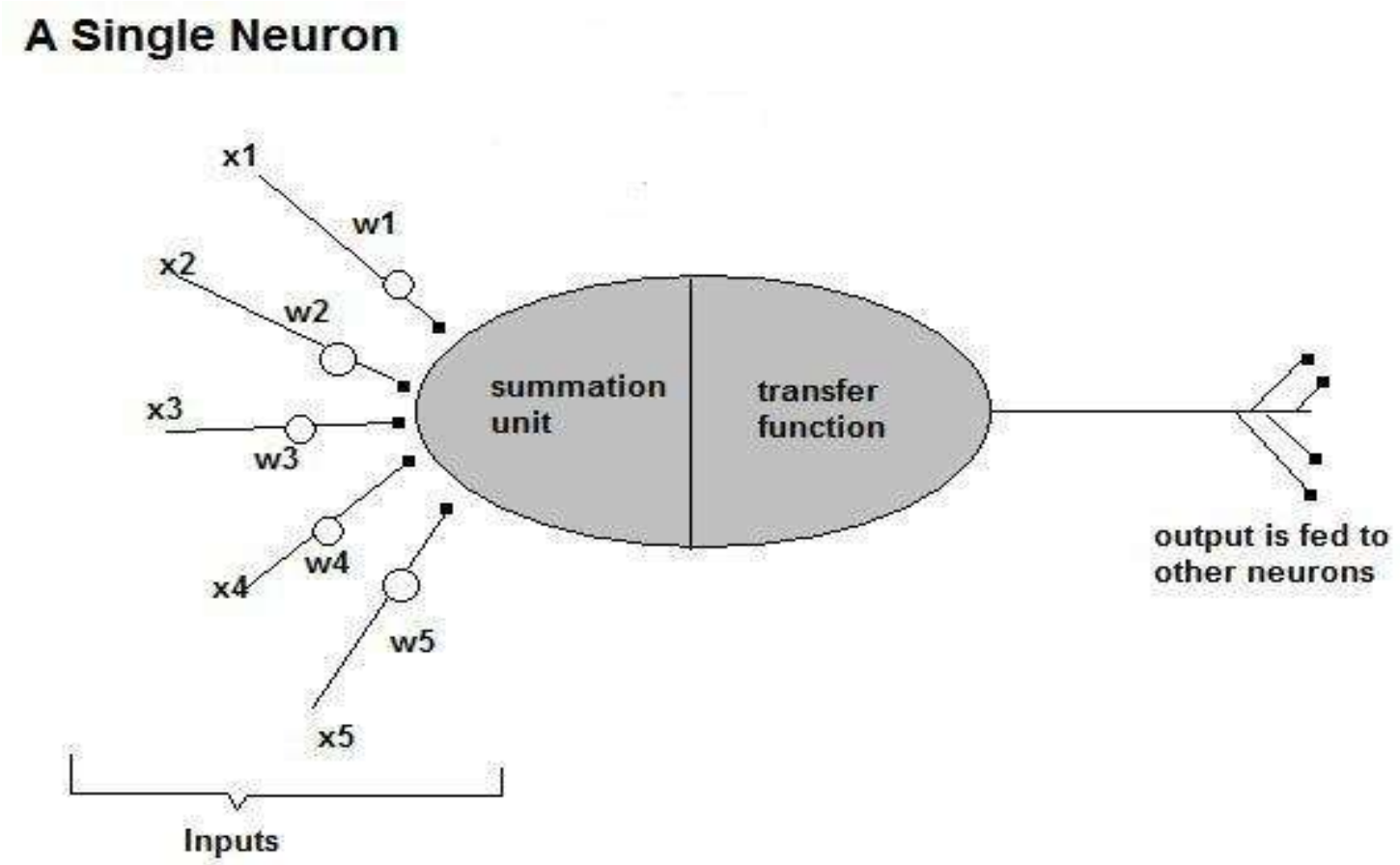
How do ANNs work?

- An artificial neural network (ANN) is either a hardware implementation or a computer program which strives to simulate the information processing capabilities of its biological exemplar. ANNs are typically composed of a great number of interconnected artificial neurons. The artificial neurons are simplified models of their biological counterparts.
- ANN is a technique for solving problems by constructing software that works like our brains.

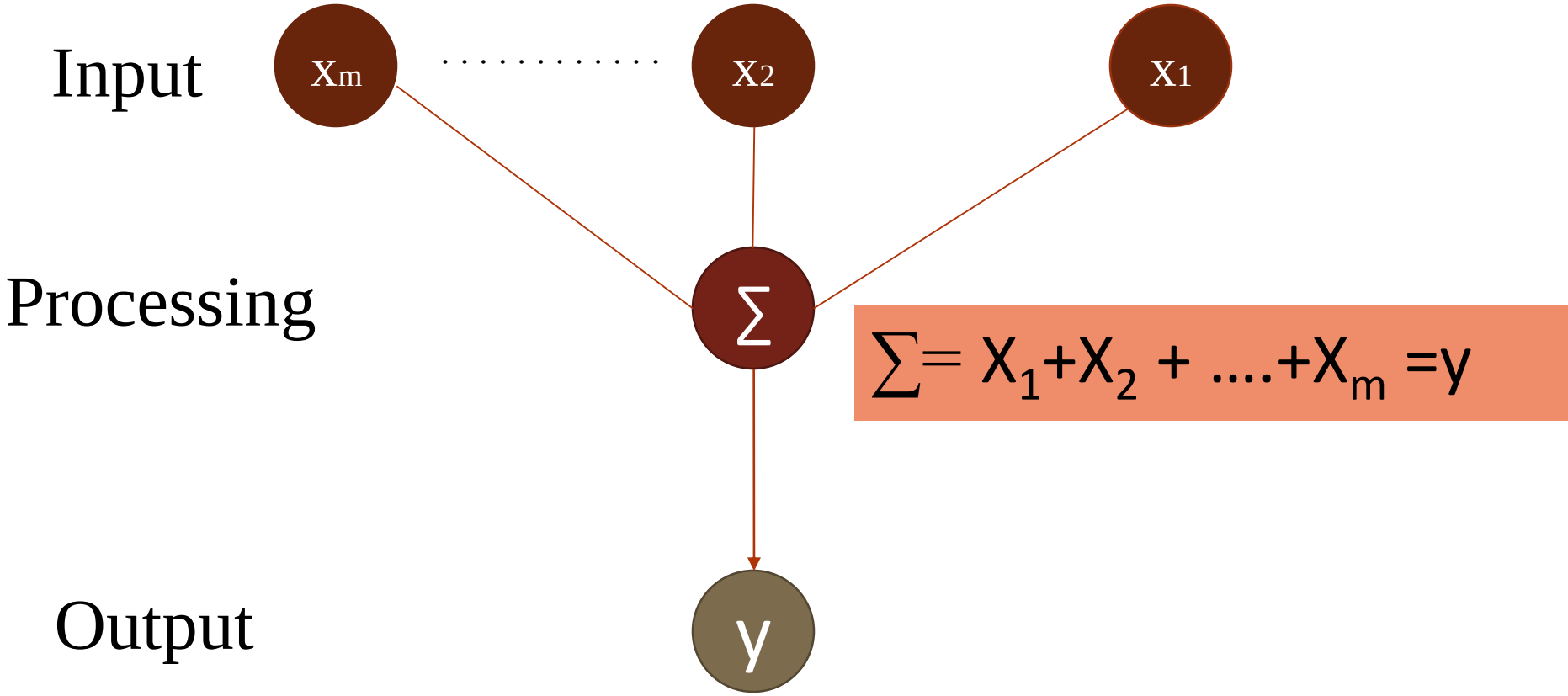


How do ANNs work?

- Now, let us have a look at the model of an artificial neuron.



How do ANNs work?



How do ANNs work?

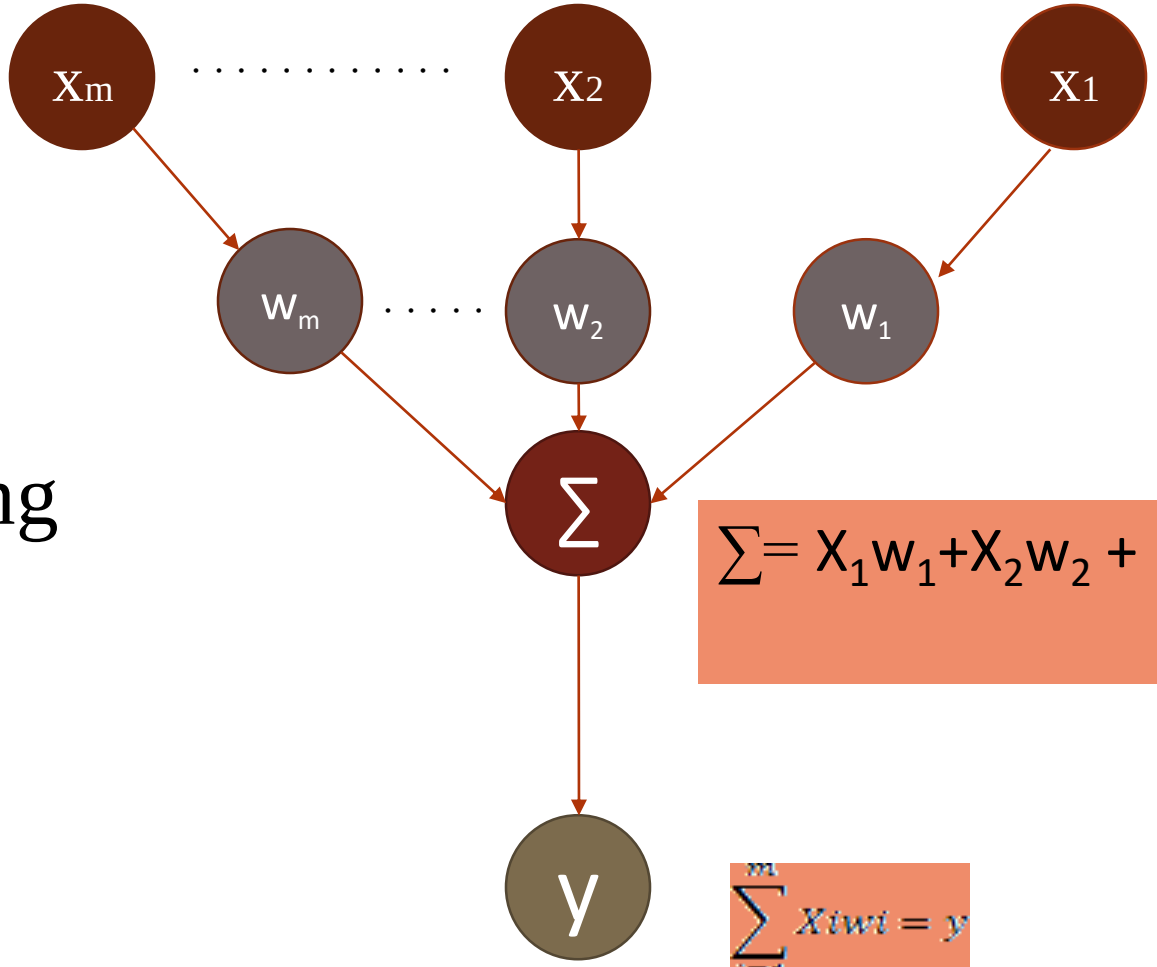
Not all inputs are equal

Input

weights

Processing

Output

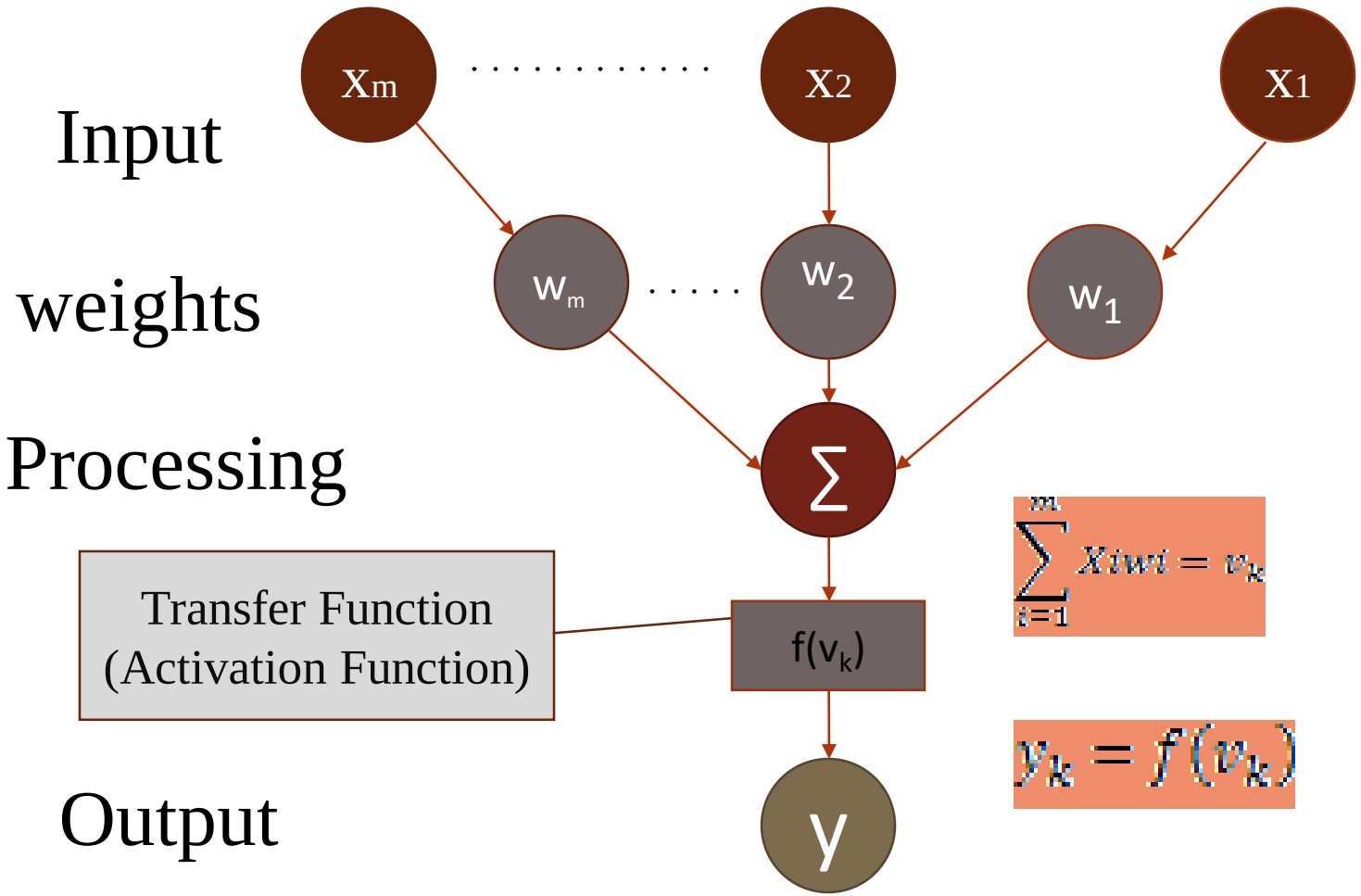


$$\sum = X_1w_1 + X_2w_2 + \ldots + X_mw_m = y$$

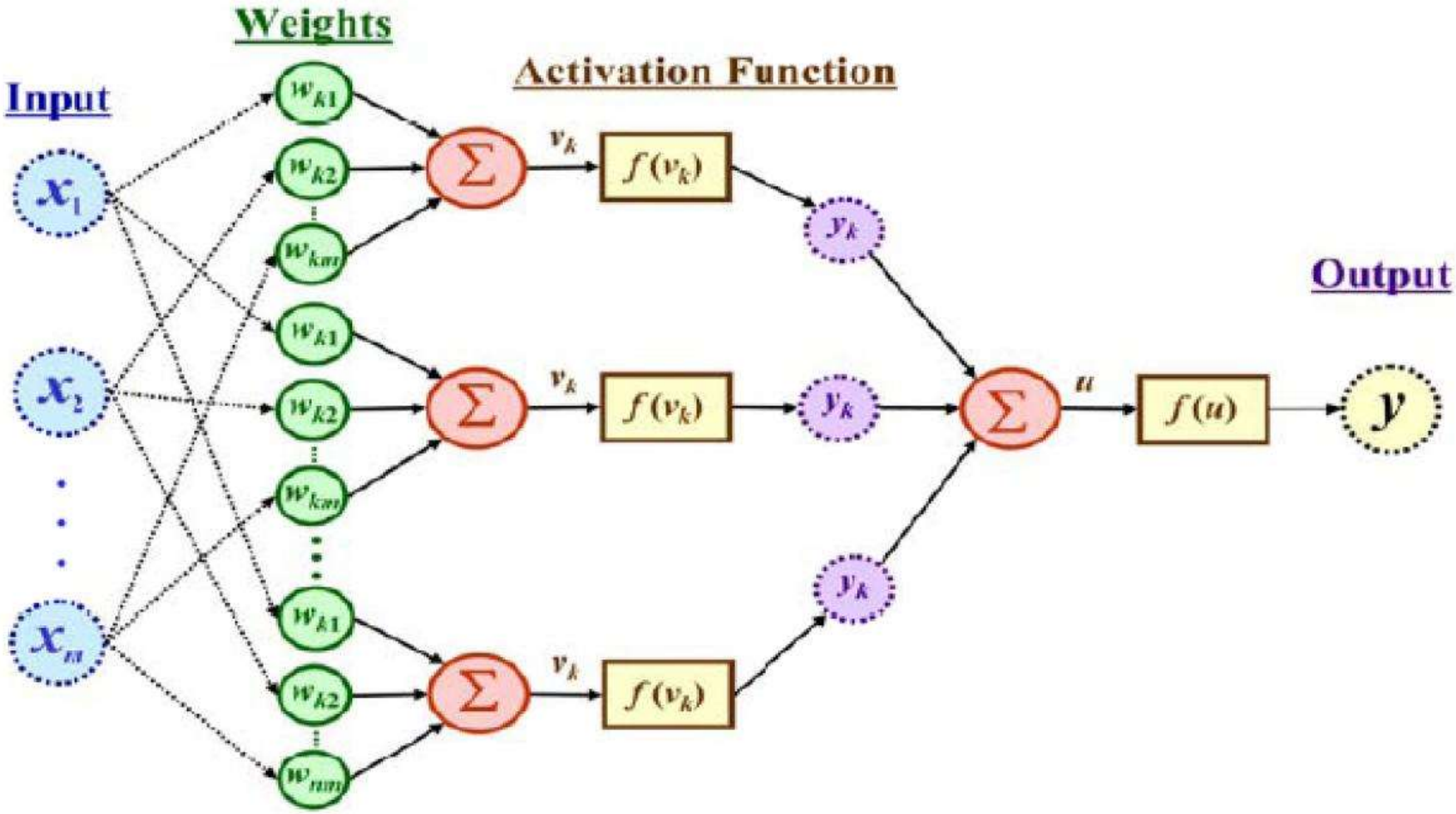
$$\sum_{i=1}^{m} Xiwi = y$$

How do ANNs work?

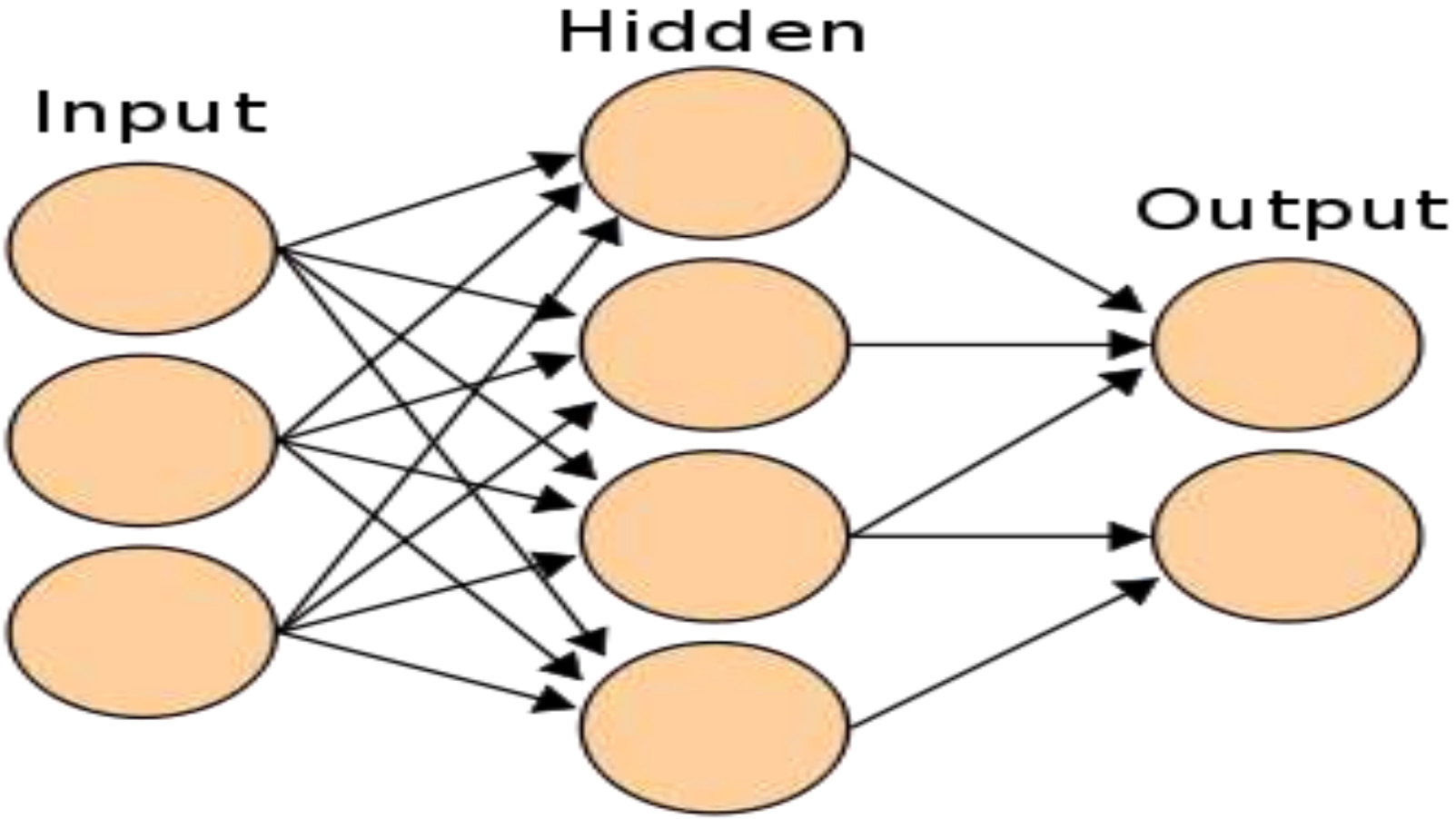
The signal is not passed down to the next neuron verbatim



The output is a function of the input, that is affected by the weights, and the transfer functions



Three types of layers: Input, Hidden, and Output



Artificial Neural Networks

- An ANN can:
  1. compute *any computable* function, by the appropriate selection of the network topology and weights values.
  2. learn from experience!
     - Specifically, by trial-and-error