Foundations of Data Science & Analytics: Artificial Neural Networks

Ezgi Siir Kibris

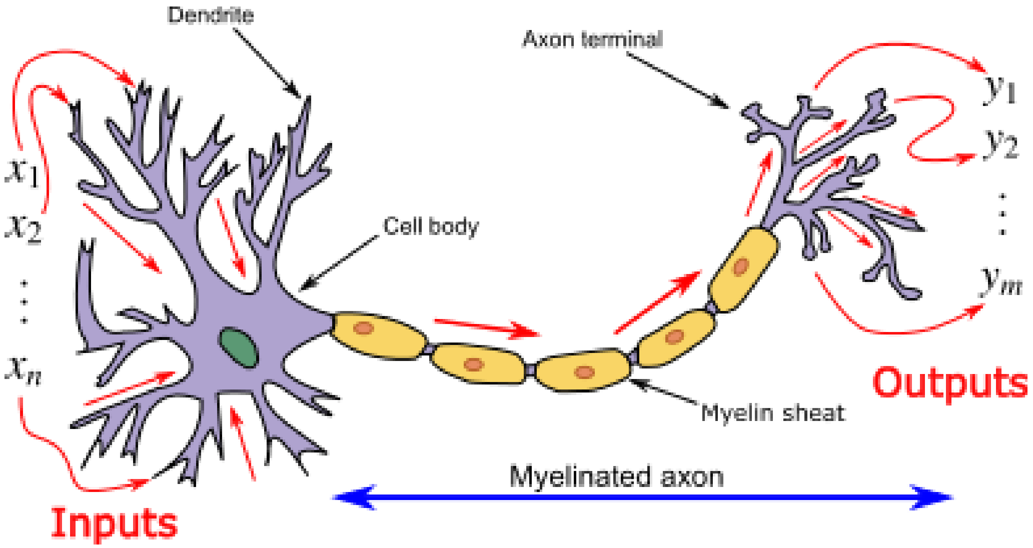
Introduction to Data Mining, 2nd Edition
by
Tan, Steinbach, Karpatne, Kumar

Classification Techniques

- **Base Classifiers**
  - Decision Tree based Methods
  - Rule-based Methods
  - Instance-based Methods (Nearest-neighbor)
  - Naïve Bayes
  - Support Vector Machines
  - **Neural Networks and Deep Learning**

- **Ensemble Classifiers**
  - Boosting, Bagging, Random Forests

Inspiration

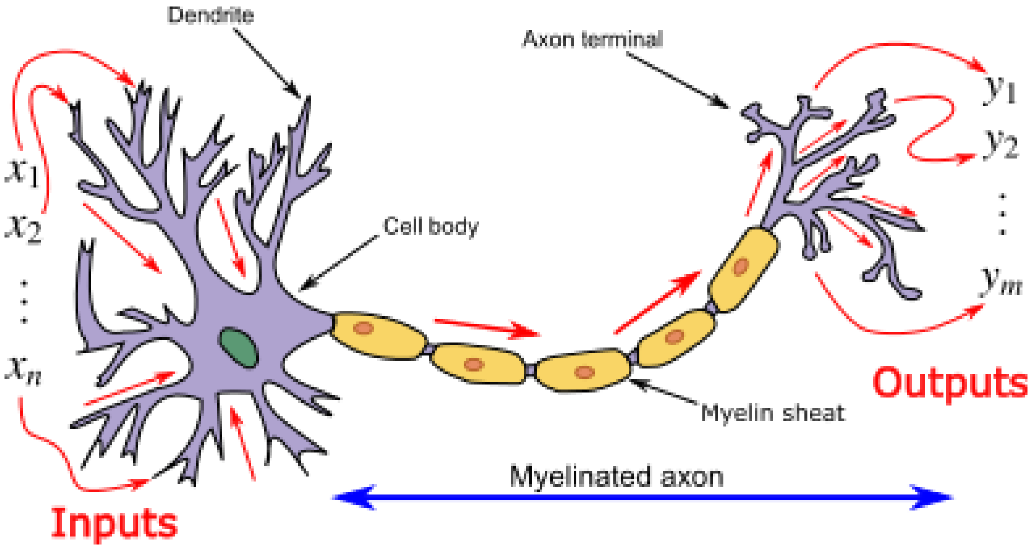


- **Dendrites:** Entry points in each neuron which take input from other neurons in the network in form of electrical impulses

- **Cell Body:** It generates inferences from the dendrite inputs and decides what action to take

- **Axon terminals:** They transmit outputs in form of electrical impulses to next neuron

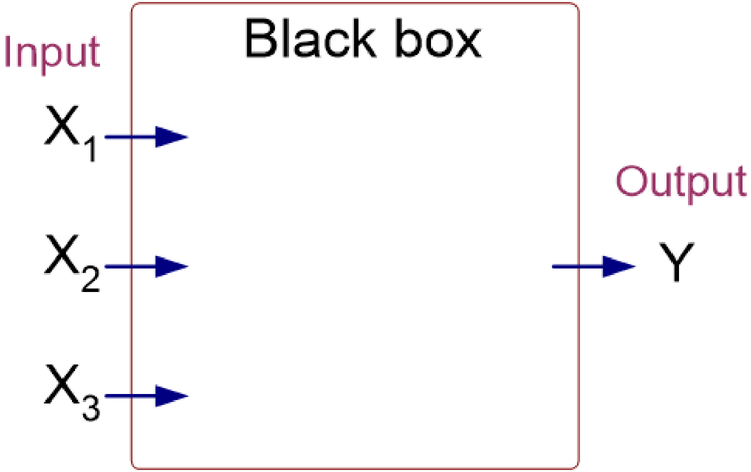
Inspiration



- Simple as a single neuron
- But extremely powerful when millions-billions of neurons are connected

ANN

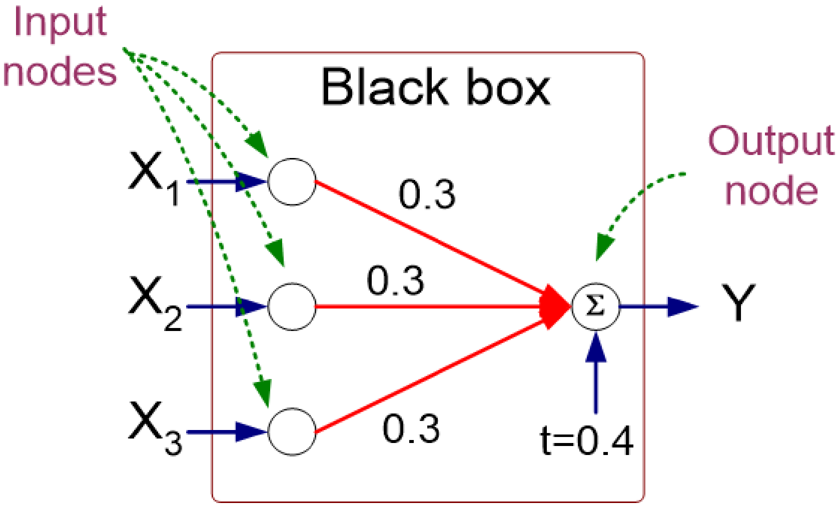
X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1



Output Y is 1 if at least two of the three inputs are equal to 1.

ANN

X_1	X_2	X_3	Y
1	0	0	-1
1	0	1	1
1	1	0	1
1	1	1	1
0	0	1	-1
0	1	0	-1
0	1	1	1
0	0	0	-1

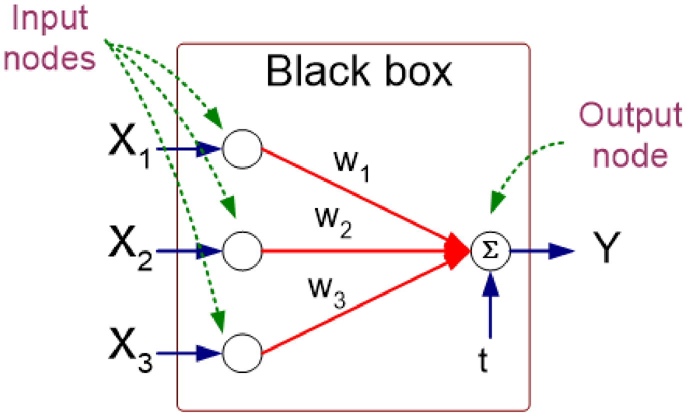


$$Y = sign(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4)$$

$$\text{where } sign(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

ANN

- Model is an assembly of inter-connected nodes and weighted links
- Output node sums up each of its input value according to the weights of its links
- Compare output node against some threshold t

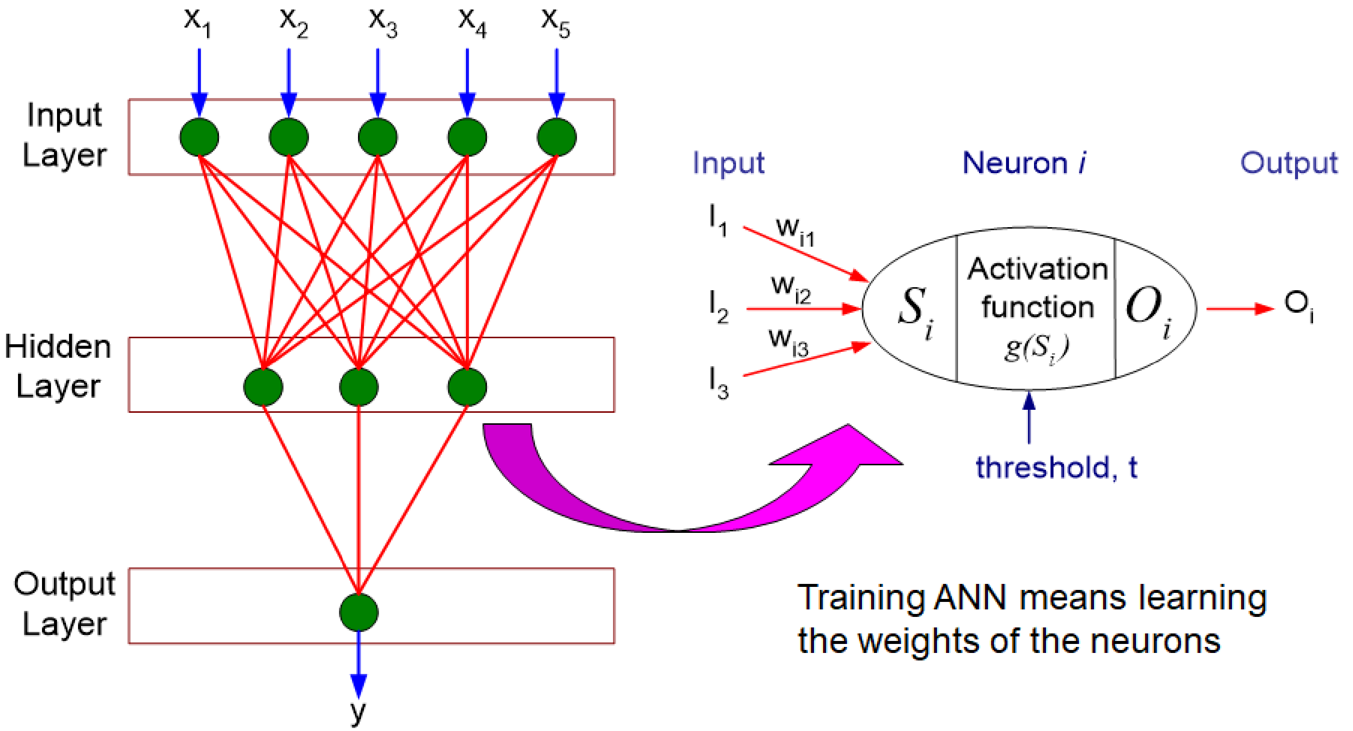


Perceptron Model

$$Y = sign(\sum_{i=1}^{d} w_i X_i - t)$$

$$= sign(\sum_{i=0}^{d} w_i X_i)$$

General Structure

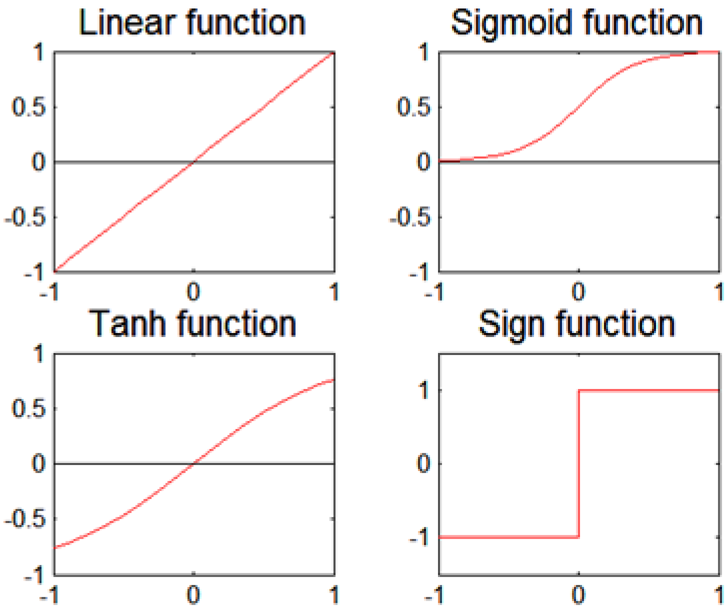


Training ANN means learning the weights of the neurons

Activation Functions

- Various types of activation functions f

$$Y = f(\sum_i w_i X_i)$$



Perceptron

- **Single layer network**
  - Contains only input and output nodes
- **Activation function: f = sign(w·x)**
- **Applying model is straightforward:**

$$Y = sign(0.3X_1 + 0.3X_2 + 0.3X_3 - 0.4)$$

$$\text{where } sign(x) = \begin{cases} 1 & \text{if } x \geq 0 \\ -1 & \text{if } x < 0 \end{cases}$$

Perceptron

- **Initialize the weights (w0, w1, …, wd)**
- **Repeat**
  - For each training example (xi, yi)
    - Compute f(w, xi)
    - Update the weights:

$$w^{(k+1)} = w^{(k)} + \lambda\left[y_i - f(w^{(k)}, x_i)\right]x_i$$

- **Until stopping condition is met**

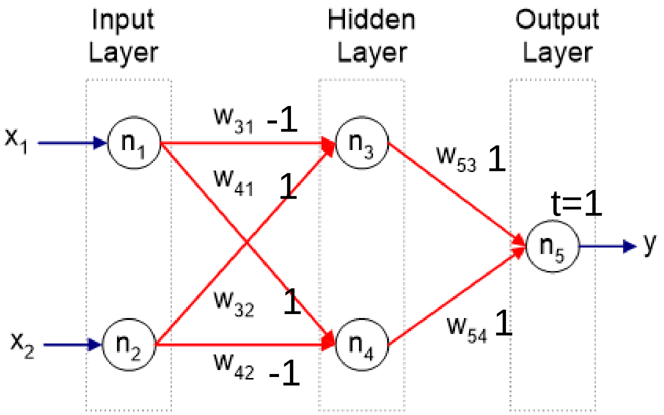
Solution for nonlinearly separable

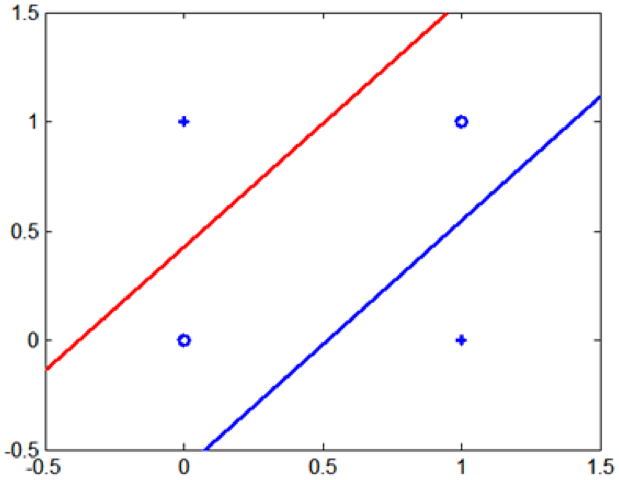
- **Hidden layers**
  - intermediary layers between input & output layers

- **More general activation functions (sigmoid, tanh, etc)**

Multilayer networks

Multi-layer neural network can solve any type of classification task involving nonlinear decision surfaces





Design Issue

- **Number of nodes in input layer**
  - One input node per binary/continuous feature
  - k or log k nodes for each categorical feature with k values
- **Number of nodes in output layer**
  - One output for binary class problem
  - k or log k nodes for k-class problem
- **Number of nodes in hidden layer**
- **Initial weights and biases**

Problems

- Multilayer ANN are universal approximators but could suffer from overfitting if the network is too large
- Model building can be very time consuming, but testing can be very fast
- Can handle redundant features because weights are automatically learned
- Sensitive to noise in training data
- Difficult to handle missing attributes