Single Layer Perceptron (SLP)

05 October 2025 09:55

Single Layer Perceptron

what is a perceptron?

Perception: derived from the word 'perceive'

- w.r.t. psychology / human context

→ Perception is the process by which we interpret and make sense of sensory information.

what we see, hear, touch, smell etc.

(machines + softwares)

In AI (Deep learning)

Perception refers to how machines interpret raw data from the world — turning it into meaningful information

PERCEP + TRON

PERCEP: derived from the word perception

TRON: some sort of electronics devices → GPUs.

→ to mimicking the human intelligence using electronics.

① algorithms & frameworks

② Hardware such as computers with GPUs

(TensorFlow) ⊕ (electronics devices) → Perceptron based model.

SINGLE LAYER PERCEPTRON (SLP)

In machine learning, the perceptron is an algorithm for supervised learning of binary classifiers

History [edit]

See also: History of artificial intelligence § Perceptrons

[ANN was invented in 1943.]

The artificial neuron network was invented in 1943 by Warren McCulloch and Walter Pitts in *A logical calculus of the ideas immanent in nervous activity*.[5]

In 1957, Frank Rosenblatt was at the Cornell Aeronautical Laboratory. He simulated the perceptron on an IBM 704.[6][7] Later, he obtained funding by the Information Systems Branch of the United States Office of Naval Research and the Rome Air Development Center, to build a custom-made computer, the Mark I Perceptron. It was first publicly demonstrated on 23 June 1960.[8] The machine was "part of a previously secret four-year NPIC [the US' National Photographic Interpretation Center] effort from 1963 through 1966 to develop this algorithm into a useful tool for photo-interpreters".[9]

Rosenblatt described the details of the perceptron in a 1958 paper.[10] His organization of a perceptron is constructed of three kinds of cells ("units"): AI, AII, R, which stand for "projection", "association" and "response". He presented at the first international symposium on AI, *Mechanisation of Thought Processes*, which took place in 1958 November.[11]

https://en.wikipedia.org/wiki/Perceptron

SLP was developed by Frank Rosenblatt in 1958 to do a simple binary classification

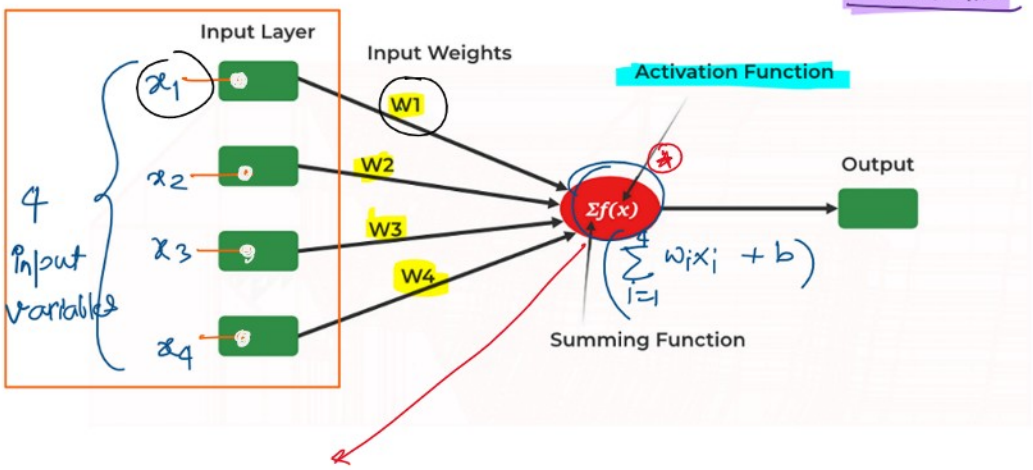
A SLP consists of only one layer of weights that directly connects the input features to the output

It is a feed-forward ANN with NO HIDDEN LAYER

* It is one of the simplest types of ANNs designed to mimic the way neurons work in the brain

Structure of a Single-Layer Perceptron (SLP)

Illustrative



Weighted sum of inputs

$$x|_z = w_1 x_1 + w_2 x_2 + w_3 x_3 + w_4 x_4 + b$$

weights

bias

$$x|_z = \sum_{i=1}^{4} w_i x_i + b$$

$x|z$ → Activation Function (AF) $f(z)$ → Threshold t

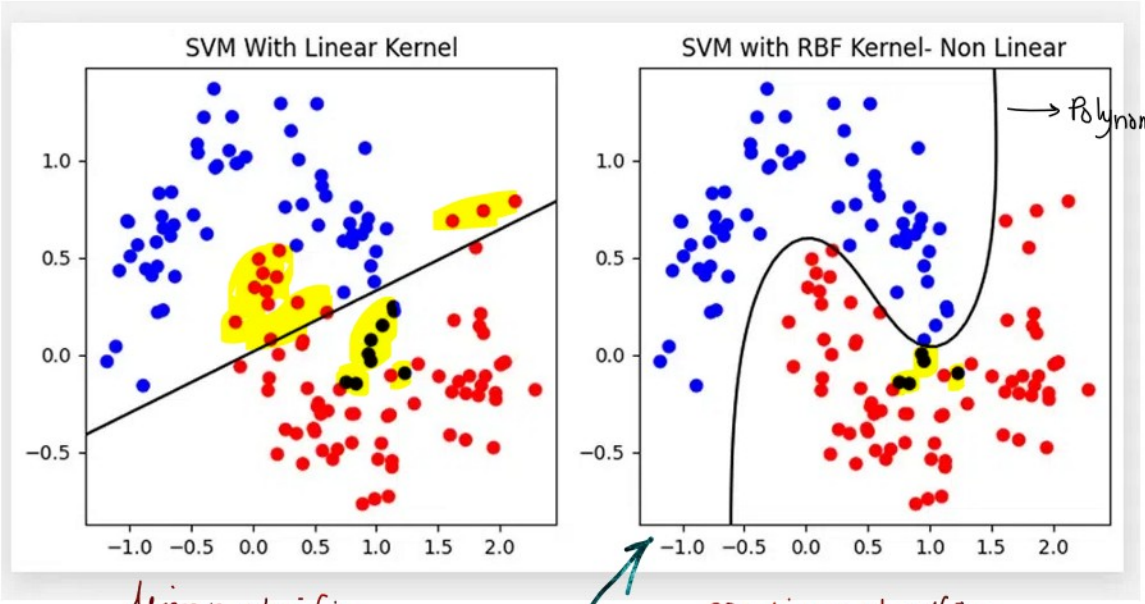
- $\geq t$ → class #1
- $< t$ → class #0

Output layer

(Separate section will be discussed later)

In logistic regression:
- ≥ 0.5 → class #1
- < 0.5 → class #0

Activation Function (AF)



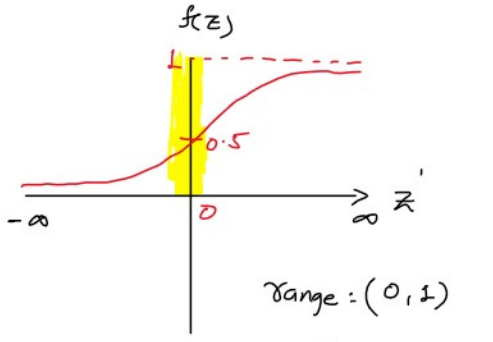
An activation function introduces non-linearity into a neural network model.

Without activation function, the neural network would just be a linear regression model, no matter how many layers are added.

lets the network learn complex, non-linear patterns like images, speech and text.

Sigmoid Function

$$f(z) = \left(\frac{1}{1+e^{-z}}\right)$$

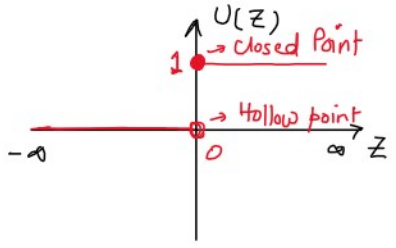


range : (0, 1)

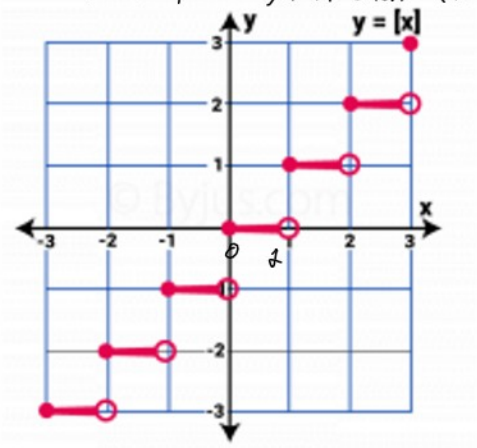
S-shaped curve

Unit step Activation Function

$$U(z) = \begin{cases} 1 & z \geq 0 \\ 0 & z < 0 \end{cases}$$



Greatest Integer Function (GIF)



$[x]$

$[0.3] = 0$

Building SLP using a tiny dataset (Binary classification)

SLP Demonstration using Excel

Let us take a tiny dataset having:

- two features say x_1 and x_2 - two input variables
- target or label → y values
- 4 training rows or training examples.

Feature 1 : X1	Feature 2 : X2	Label (Y)
2	1	1
1	-1	0
-1	-2	0
-2	1	1

→ We'll build SLP using Excel

→ and then move the SLP to Python using our own functions

PERCEPTRON LEARNING RULE

$$w_j := w_j - \alpha * (\hat{y}^{(i)} - y^{(i)}) * x_j^{(i)}$$

$$b := b - \alpha * (\hat{y}^{(i)} - y^{(i)})$$

Error

Given error = 0, both weight and bias need not be updated

$$w_1)_{new} = w_1)_{old} - \alpha * error * x_1$$

$$w_{new} = w_{old} - 0.1 * error * x_1$$