IITU

Neural Networks

Compiled by
G. Pachshenko

Pachshenko Galina Nikolaevna

Associate Professor of Information System Department,

Candidate of Technical Science

Week 3
Lecture 3

Topics

- Perceptron
- The perceptron learning algorithm
- Major components of a perceptron
- AND operator
- OR operator
- Neural Network Learning Rules
- Hebbian Learning Rule

Machine Learning Classics: The Perceptron

Perceptron (Frank Rosenblatt, 1957)

First learning algorithm for neural networks;

Originally introduced for character classification, where each character is represented as an image;

In machine learning, the **perceptron** is an algorithm for supervised learning of *binary classifiers* (functions that can decide whether an input, represented by a vector of numbers, *belongs to some specific class or not*.

The *binary classifier* defines that there should be only two categories for classification.

Classification is an example of **supervised learning**.

The perceptron learning algorithm (PLA)

The learning algorithm for the perceptron *is online*, meaning that instead of considering the entire data set at the same time, *it only looks at one example at a time*, processes it and goes on to the next one.

Following are the major components of a perceptron:

Input: All the features become the input for a perceptron. We denote the input of a perceptron by [x1, x2, x3, ..,xn], where x represents the feature value and n represents the total number of features. We also have special kind of input called the *bias*. In the image, we have described the value of the BIAS as w0.

Weights: The values that are computed over the time of training the model. Initially, we start the value of weights with some initial value and these values get updated for each training error. We represent the weights for perceptron by [w1,w2,w3,.. wn].

Weighted summation: Weighted summation is the sum of the values that we get after the multiplication of each weight [wn] associated with the each feature value [xn].

We represent the weighted summation by $\boldsymbol{\Sigma w_i x_i}$ for all i -> [1 to n].

Bias: A bias neuron allows a classifier to shift the decision boundary left or right. In algebraic terms, the bias neuron allows a classifier to translate its decision boundary. It aims to "move every point a constant distance in a specified direction." Bias helps to train the model faster and with better quality.

Step/activation function: The role of activation functions is to make neural networks nonlinear. For
linear classification, for example, it becomes necessary to make the perceptron as linear as possible.

Output: The *weighted summation* is passed to the *step/activation function* and whatever value we get after computation is our predicted output.

Inputs:

1 or 0

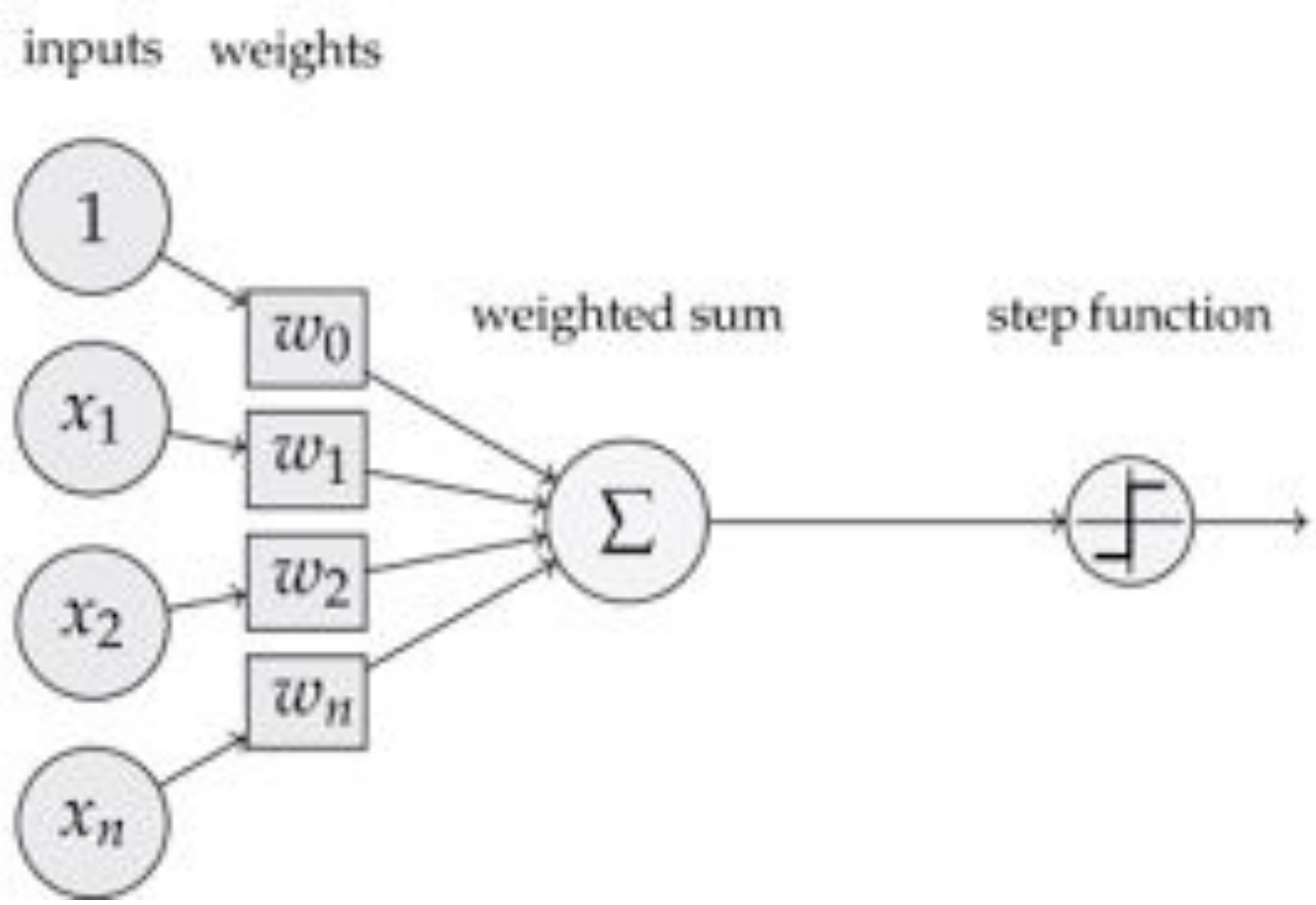
Outputs:

1 or 0

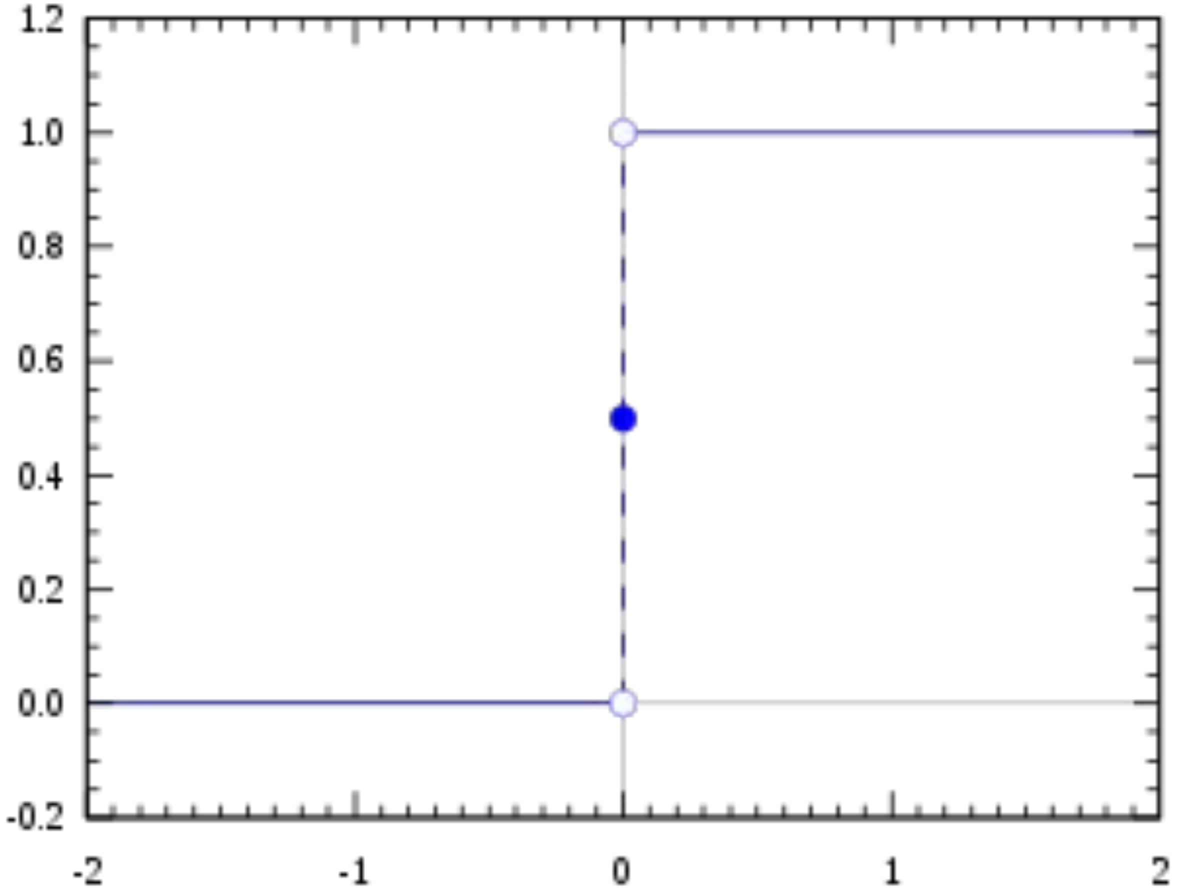
Description:

- Firstly, the features for an example are given as input to the perceptron.
- These input features get multiplied by corresponding weights (starting with initial value).
- The summation is computed for the value we get after multiplication of each feature with the corresponding weight.
- The value of the summation is added to the bias.
- The step/activation function is applied to the new value.

Perceptron


inputs
weights
weighted sum
step function
1
x_1
x_2
x_n
w_0
w_1
w_2
w_n
Σ

Step function



Perceptron: Learning Algorithm

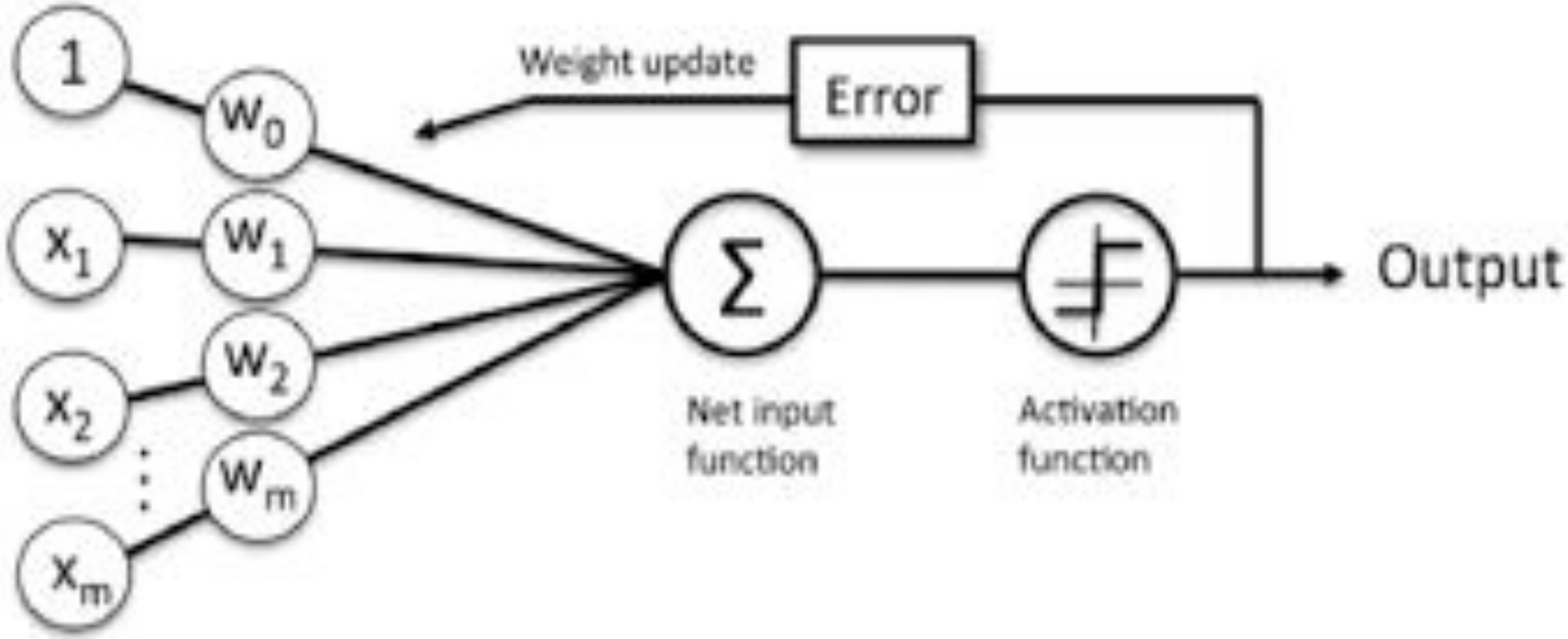
The algorithm proceeds as follows:

Initial random setting of weights;

The input is a random sequence.

For each element of class C1, if output = 1 (correct) do nothing, otherwise update weights;

For each element of class C2, if output = 0 (correct) do nothing, otherwise update weights.


1
w_0
x_1
w_1
x_2
w_2
x_m
w_m
Weight update
Error
Σ
Net input function
Activation function
Output

Perceptron Learning Algorithm

We want to train the perceptron to classify inputs correctly

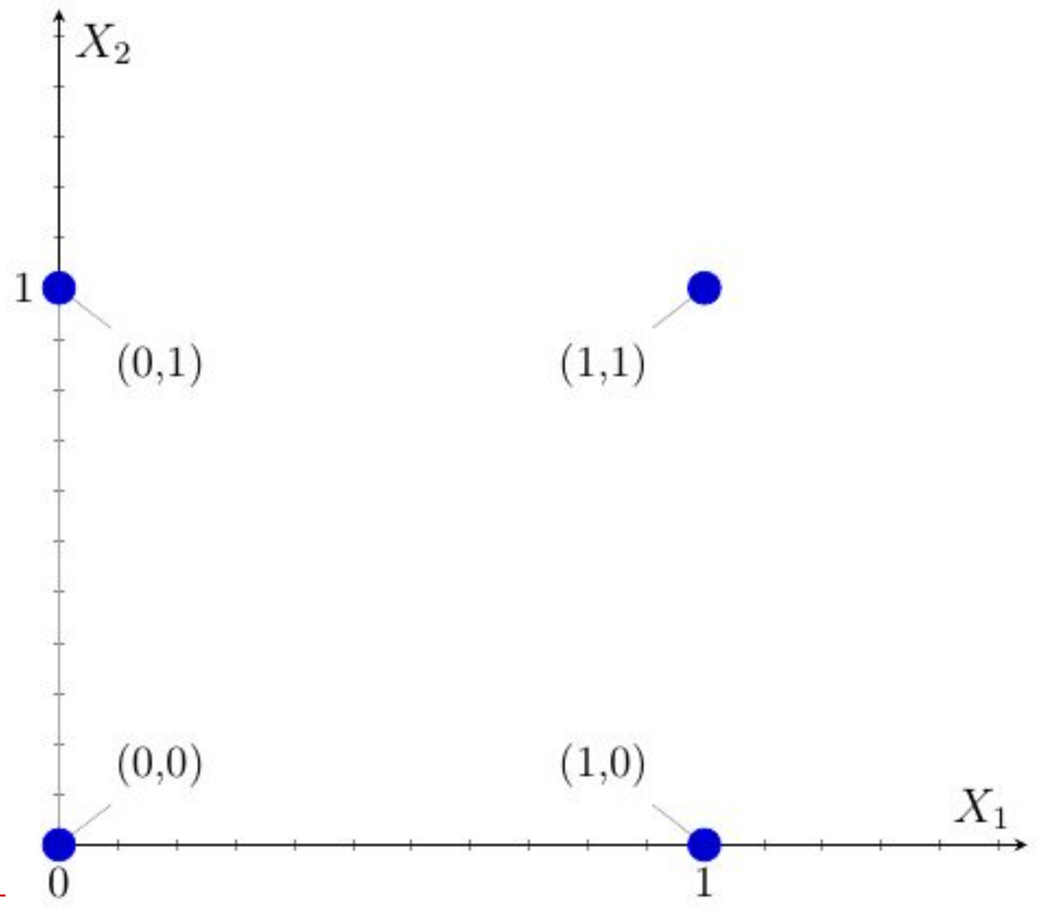
Accomplished by adjusting the connecting weights and the bias

Can only properly handle linearly separable sets

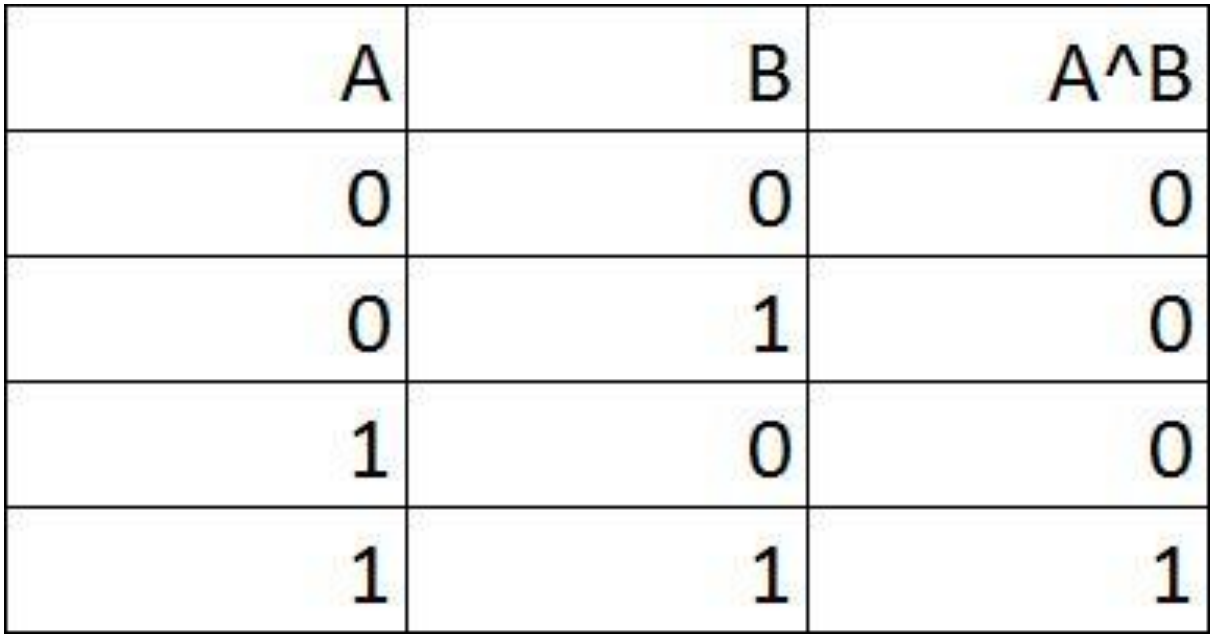
The **perceptron** is a machine learning algorithm used to determine whether an input belongs to one class or another.

For example, the perceptron algorithm can determine the **AND operator** - given binary inputs and , is (AND) equal to 0 or 1

AND operator

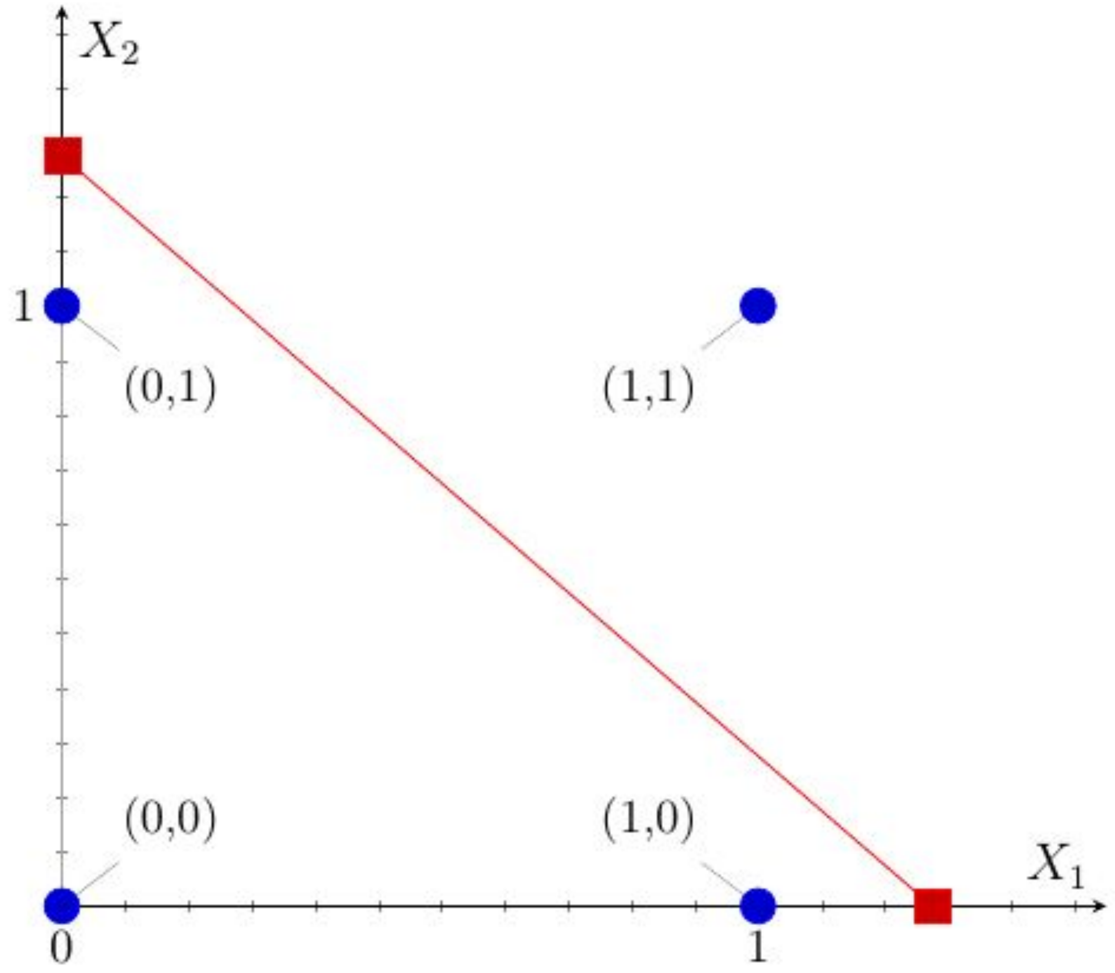


AND operator

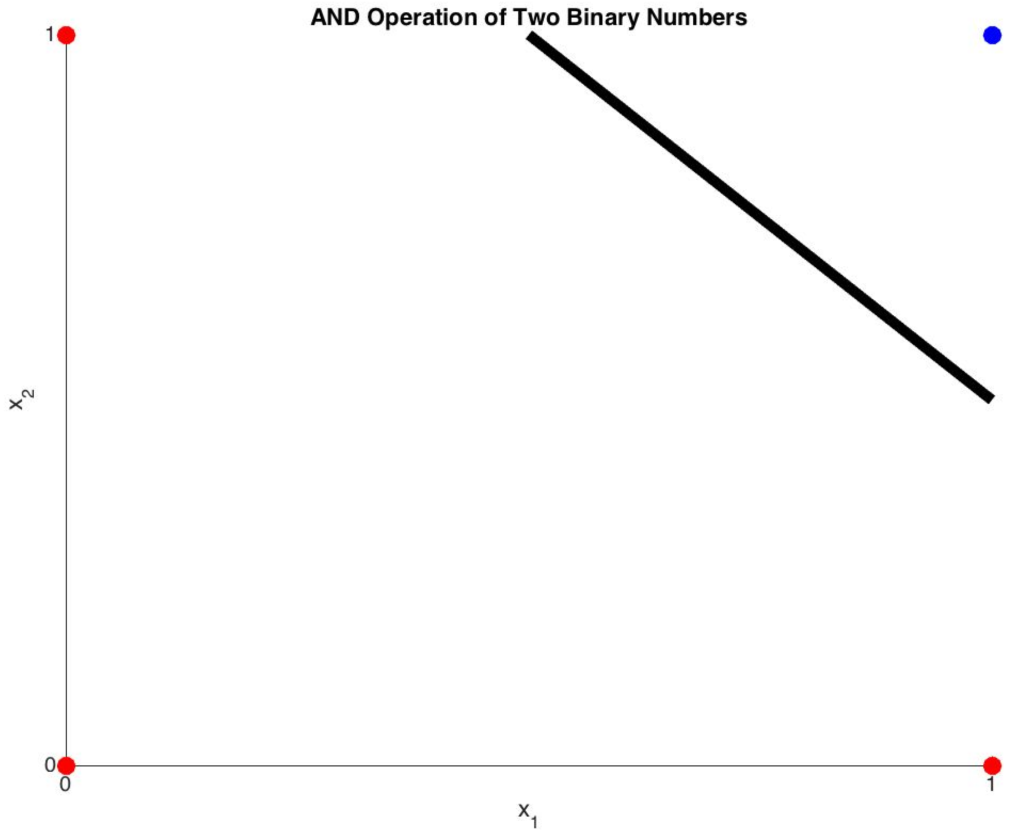


A	B	A^B
0	0	0
0	1	0
1	0	0
1	1	1

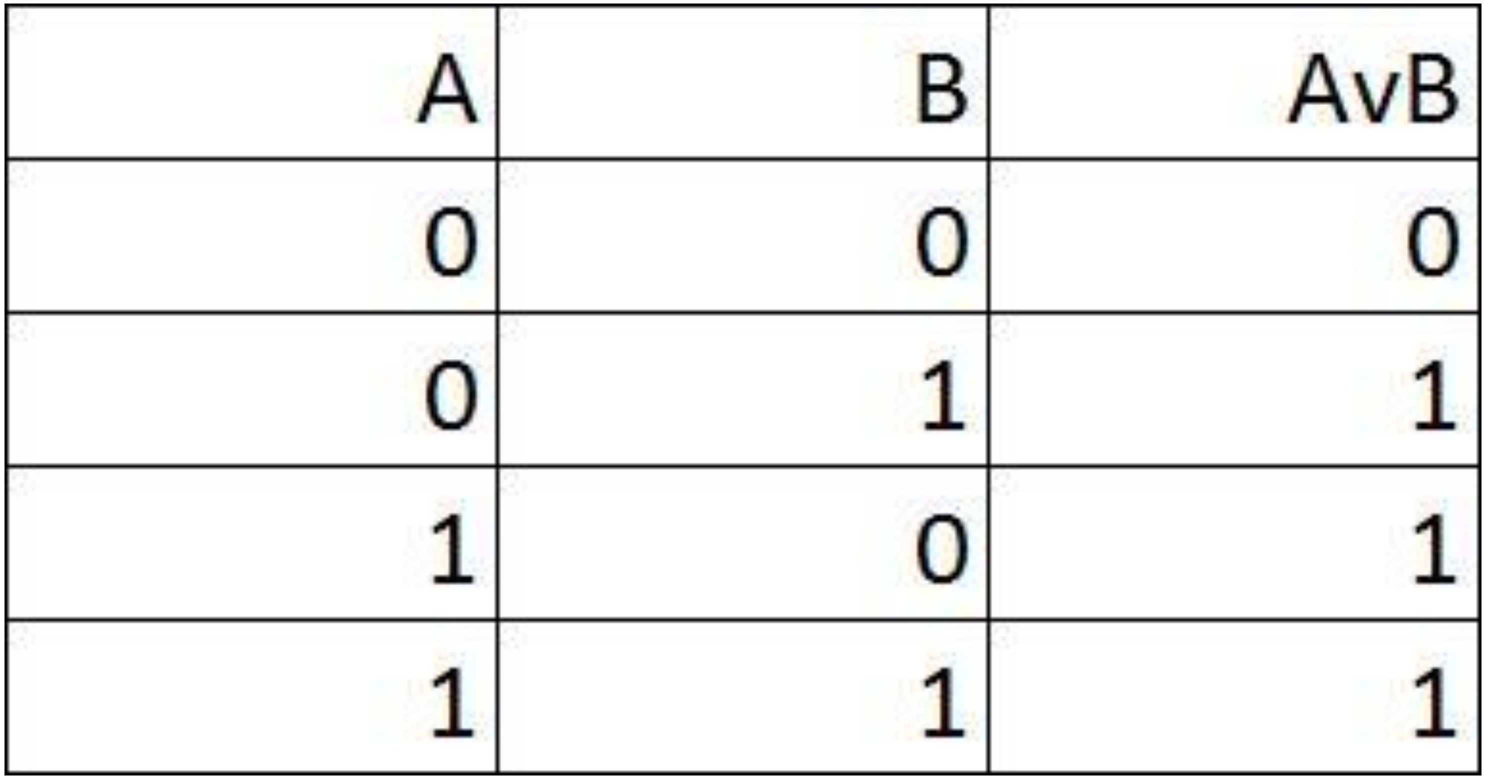
AND operator



The AND operation between two numbers. A red dot represents one class (AND) and a blue dot represents the other class (AND). The line is the result of the perceptron algorithm, which separates all data points of one class from those of the other.

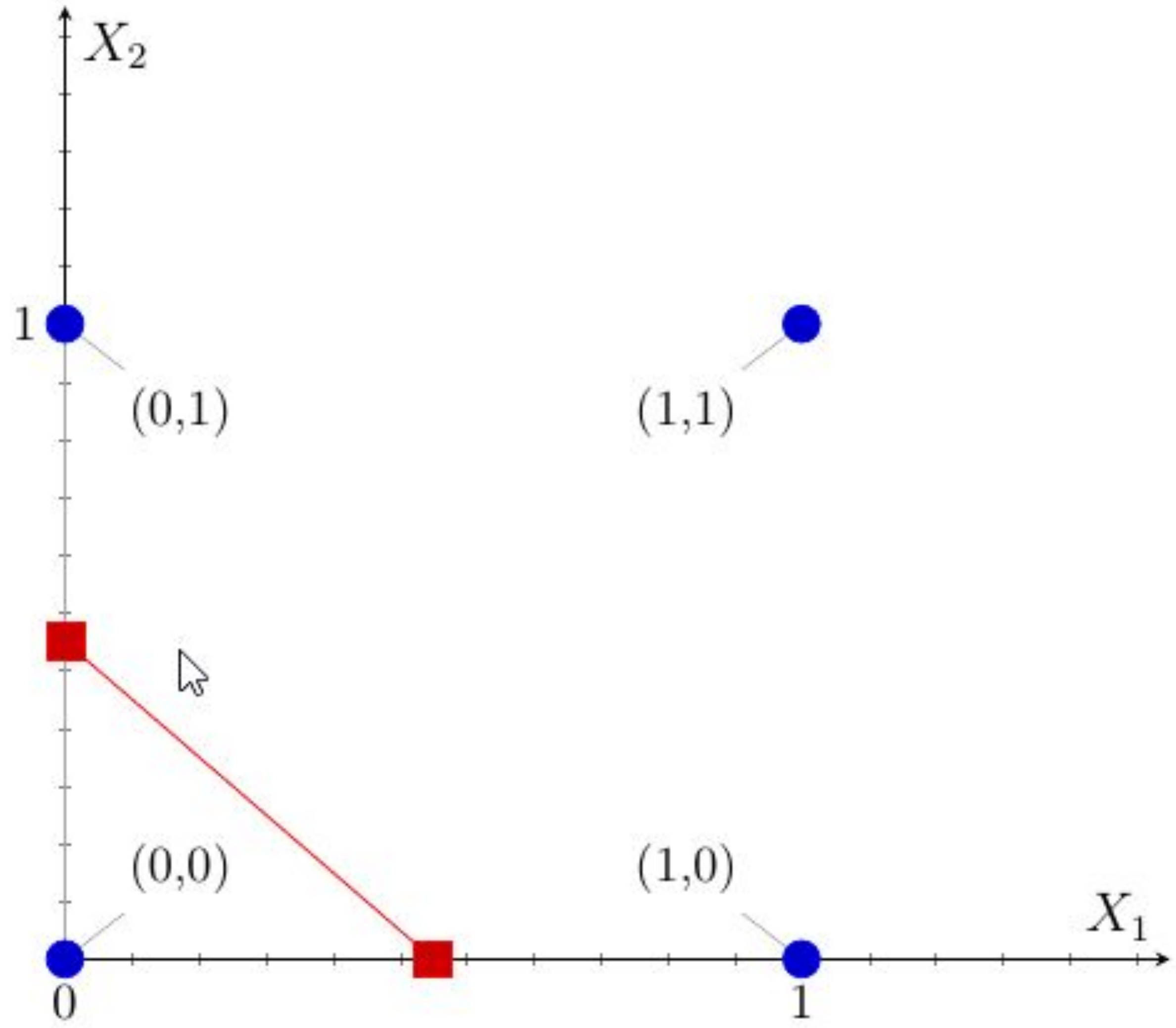


OR operator

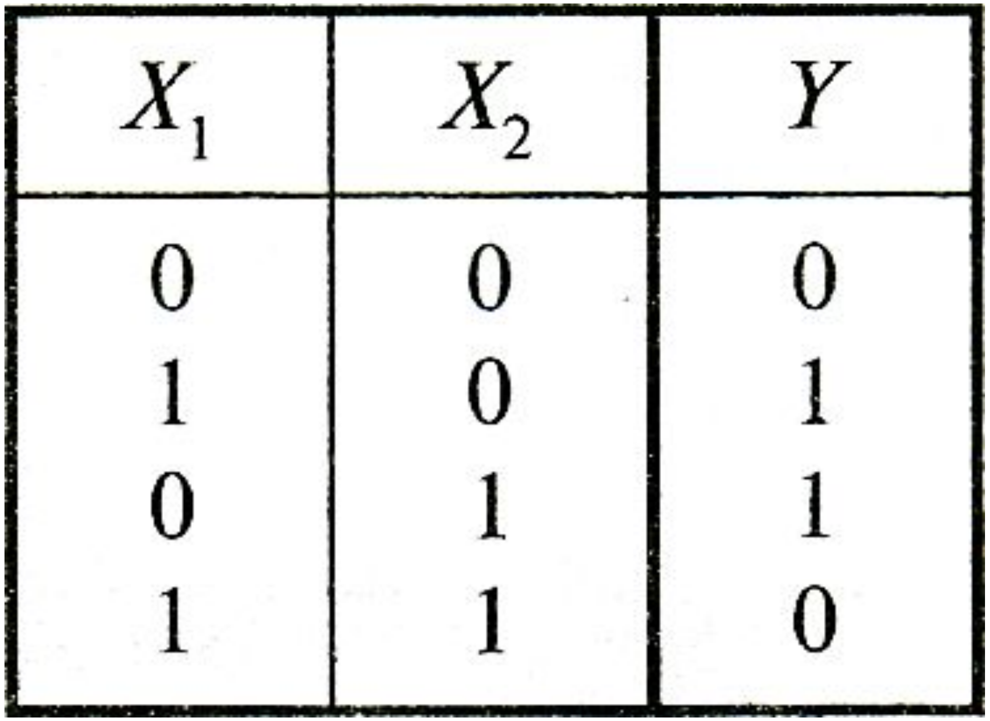


A	B	AvB
0	0	0
0	1	1
1	0	1
1	1	1

OR operator

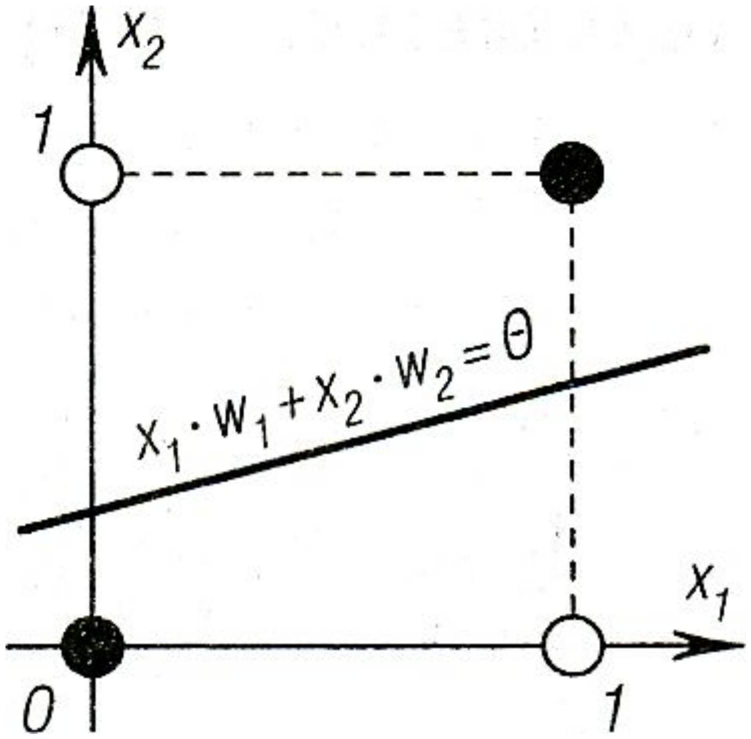


XOR
Not linearly separable sets

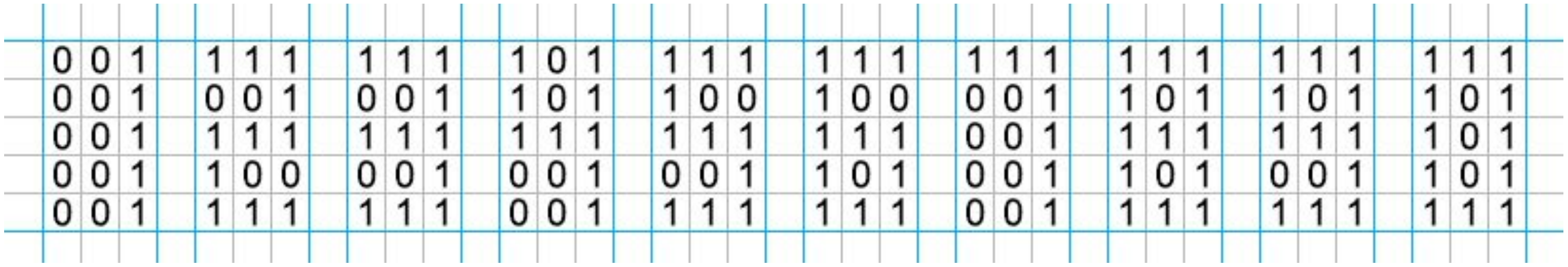


X_1	X_2	Y
0	0	0
1	0	1
0	1	1
1	1	0

XOR
Not linearly separable sets



Character classification



0 0 1	1 1 1	1 1 1	1 0 1	1 1 1	1 1 1	1 1 1	1 1 1	1 1 1	1 1 1
0 0 1	0 0 1	0 0 1	1 0 1	1 0 0	1 0 0	0 0 1	1 0 1	1 0 1	1 0 1
0 0 1	1 1 1	1 1 1	1 1 1	1 1 1	1 1 1	0 0 1	1 1 1	1 1 1	1 0 1
0 0 1	1 0 0	0 0 1	0 0 1	0 0 1	1 0 1	0 0 1	1 0 1	0 0 1	1 0 1
0 0 1	1 1 1	1 1 1	0 0 1	1 1 1	1 1 1	0 0 1	1 1 1	1 1 1	1 1 1

Character classification

1 – 001001001001001

……………………………….

9 – 111101111001111

0 – 111101101101111

Neural Network Learning Rules

We know that, during ANN learning, to change the input/output behavior, we need to adjust the weights. Hence, a method is required with the help of which the weights can be modified. These methods are called Learning rules, which are simply algorithms or equations.

Hebbian Learning Rule

This rule, one of the oldest and simplest, was introduced by Donald Hebb in his book *The Organization of Behavior* in 1949.

It is a kind of feed-forward, unsupervised learning.

The Hebbian Learning Rule is a learning rule that specifies how much the weight of the connection between two units should be increased or decreased in proportion to the product of their activation.

Rosenblatt’s initial perceptron rule

Rosenblatt’s initial perceptron rule is fairly simple and can be summarized by the following steps:

Initialize the weights to 0 or small random numbers.

For each training sample:

- Calculate the *output* value.
- Update the weights.

Perceptron learning rule

The weight adjustment in the perceptron learning rule is performed by

Wi+1 := wi + η(y − o)xi

where η > 0 is the learning rate, y is he *desired output*,

o ∈ {0, 1} is the *computed output*, x is the *actual input* to the neuron.

Step 1 $\eta > 0$ is chosen, range [0,5; 0,7].

where $\eta > 0$ is the learning rate

Step 2 Weigts are initialized at small random values,

The running error E is set to 0

Step 3 Training starts here.

For each element of class C1, if output = 1 (correct) do nothing, otherwise update weights;

For each element of class C2, if output = 0 (correct) do nothing, otherwise update weights.

Step 4

Weights are updated

$$w_N(i+1) = w_N(i) + \eta \delta x_N$$

$$\delta = T - OUT$$

Step 5 Cumulative cycle error is computed by adding the present error to initial error.

Step 6

If $i < N$ then $i := i + 1$ and we continue the training by going back to Step 3, otherwise we go to **Step 7**

Step 7 The training cycle is completed. For errow $E = 0$ terminate the training session. If $E > 0$ then E is set to 0, $N := 1$ and we initiate a new training cycle by going to **Step 3**

The output value is the class label predicted by the unit step function that we defined earlier.

The value for updating the weights at each increment is calculated by the learning rule

- **Hebbian learning rule –** It identifies, how to modify the weights of nodes of a network.
- **Perceptron learning rule –** Network starts its learning by assigning a random value to each weight.

Thank you
for your attention!