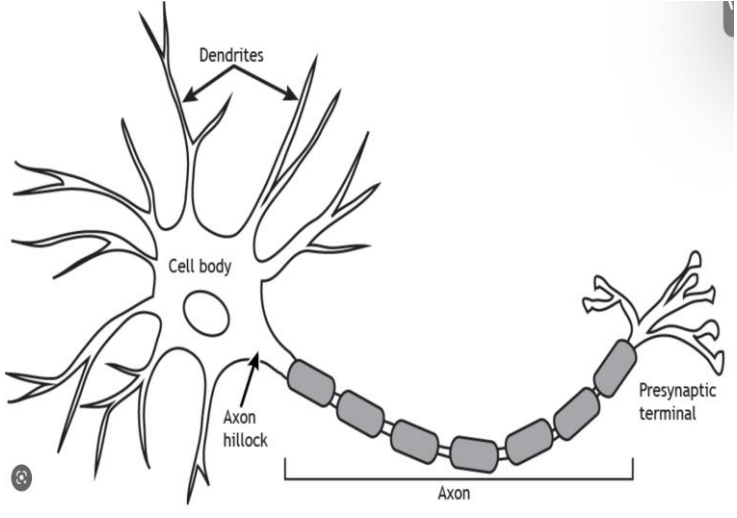
Neural Networks

Perceptron

Intuition

- A perceptron is a type of artificial neural network used for supervised learning tasks.
- It is based on the concept of a biological neuron, which receives inputs from other neurons and outputs a signal if the inputs exceed a certain threshold.
- Similarly, a perceptron receives inputs from the environment and outputs a prediction based on the inputs and its internal parameters.

Intuition


Dendrites
Cell body
Axon hillock
Presynaptic terminal
Axon

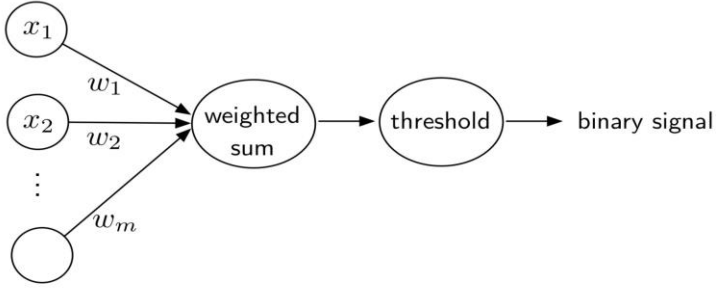
Introduction

- Perceptron is a type of linear binary classifier.
- It was introduced in 1957 by Frank Rosenblatt.
- The perceptron algorithm takes a set of input features and weights, computes a weighted sum of the inputs, and then applies an activation function to make a binary classification decision.
- It is trained using a supervised learning algorithm known as the Perceptron Learning Algorithm (PLA), which adjusts the weights and bias term in the direction of the misclassified samples.
- The perceptron algorithm is often used as a building block for more complex neural networks.
- It has many applications in areas such as computer vision, speech recognition, and natural language processing.

Introduction

A LOGICAL CALCULUS OF THE IDEAS IMMANENT IN NERVOUS ACTIVITY

WARREN S. MCCULLOCH and WALTER H. PITTS 1943



Definition

A perceptron is a binary classifier that takes a set of n input features $x_1, x_2, ..., x_n$ and corresponding weights $w_1, w_2, ..., w_n$, computes a weighted sum of the inputs $z = w_1x_1 + w_2x_2 + ... + w_nx_n$, adds a bias term b, and applies an activation function $f(z)$ to make a binary classification decision. The decision function of a perceptron can be written as:

$$y = f(\boldsymbol{w}^T \boldsymbol{x} + b)$$

where $\boldsymbol{w} = [w_1, w_2, ..., w_n]$ is the weight vector, $\boldsymbol{x} = [x_1, x_2, ..., x_n]$ is the input vector, b is the bias term, y is the output (0 or 1) of the perceptron, and f is the activation function.

Components

1. Input Layer: The input layer of the perceptron receives input values (features) that are fed into the perceptron for classification. The input values are typically represented as a vector $\boldsymbol{x} = [x_1, x_2, x_3, ..., x_n]$, where n is the number of input features.

2. Weight Vector: Each input feature x_i is multiplied by a corresponding weight w_i , and the resulting products are summed up. The weight vector is a vector $\boldsymbol{w} = [w_1, w_2, w_3, ..., w_n]$ that contains the weights for each input feature. The weights determine the importance of each feature in the classification process.

Components

3. Bias Term: The bias term, denoted by b, is added to the weighted sum of the inputs to shift the decision boundary of the perceptron. The bias term is essentially a constant that can be used to adjust the threshold at which the perceptron fires.

4. Activation Function: The output of the perceptron is obtained by applying an activation function to the weighted sum of the inputs and the bias term. The activation function determines whether the perceptron should output 0 or 1. The most commonly used activation function is the step function, which outputs 1 if the weighted sum of inputs plus the bias is greater than or equal to 0, and 0 otherwise. The step function can be represented as:

$$f(x) = \begin{cases} 1, & \text{if } x \geq 0 \\ 0, & \text{otherwise} \end{cases}$$

Components

5. Decision Function: The decision function of the perceptron can be written as:

$$y = f(\boldsymbol{w}^T \boldsymbol{x} + b)$$

where y is the output (0 or 1) of the perceptron, f is the activation function, $x_1, x_2, ..., x_n$ are the input features, $w_1, w_2, ..., w_n$ are the corresponding weights, and b is the bias term.

6. Learning Algorithm: The learning algorithm for the perceptron is the Perceptron Learning Algorithm (PLA), which is a supervised learning algorithm that adjusts the weights and bias term of the perceptron in order to minimize the classification error. The PLA updates the weights and bias term using the following update rule:

$$w_i = w_i + \alpha(y - \hat{y})x_i$$

$$b = b + \alpha(y - \hat{y})$$

where α is the learning rate, y is the true output (0 or 1), $\hat{y}$ is the predicted output (0 or 1), and x_i is the ith input feature. The update rule adjusts the weights and bias term in the direction of the misclassified sample. The learning rate controls the size of the weight and bias updates and is a hyperparameter that needs to be tuned. The PLA continues to iterate through the training data until the classification error is minimized.

For brevity

We can introduce a dummy input feature x_0 that always takes the value 1, and consider it as an input feature along with the other input features $x_1, x_2, \ldots, x_n$. Similarly, we can treat the bias term b as a weight w_0 associated with the dummy input feature x_0.
With this notation, we can write the update rules for the weights and bias as follows:

$$w_i = w_i + \alpha(y - \hat{y})x_i, \qquad i = 0, 1, 2, \ldots, n$$

where n is the number of input features.

Objective Function

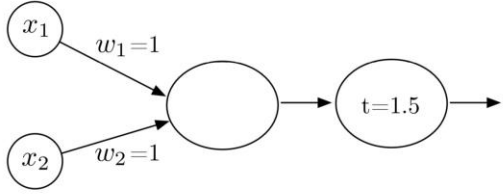
- The objective of the perceptron algorithm is to find a weight vector $\mathrm{w} \in \mathrm{R}^{n+1}$ that can correctly classify all the training examples.
- The objective function of perceptron is defined as the negative of the sum of the products of the true class labels $y^{(i)}$ and the predicted class labels $\mathrm{w}^T \mathrm{x}^{(i)}$, for all the misclassified examples.
- In other words, the objective function is minimized when all the training examples are correctly classified by the weight vector w.
- The objective function can be rewritten as

$$J(\mathrm{w}) = \sum_{i=1}^{m} \max(0, -y^{(i)}(\mathrm{w}^T \mathrm{x}^{(i)}))$$, where the 'max' function ensures that we only consider the misclassified examples.

Logical AND Gate

Bias=0

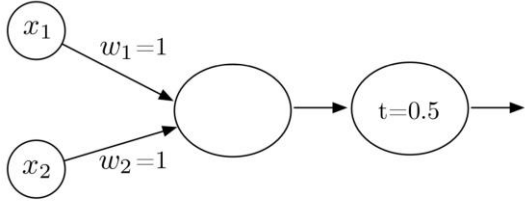
x_1	x_2	Out
0	0	0
0	1	0
1	0	0
1	1	1



Logical OR Gate

Bias=0

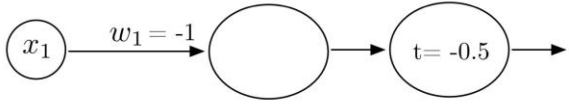
x_1	x_2	Out
0	0	0
0	1	1
1	0	1
1	1	1



Logical NOT Gate

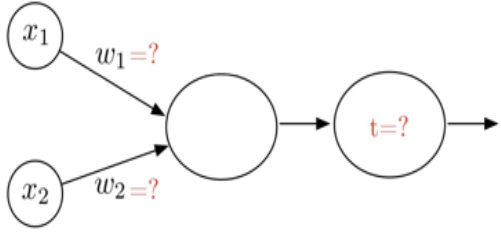
Bias=0

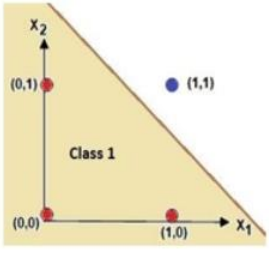
x_1	Out
0	1
1	0

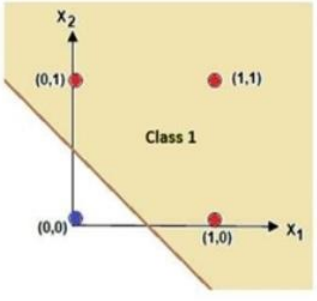


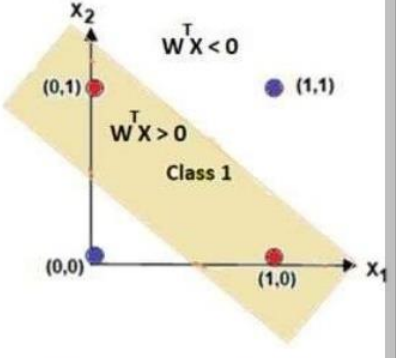
Logical XOR Gate

x_1	x_2	Out
0	0	0
0	1	1
1	0	1
1	1	0







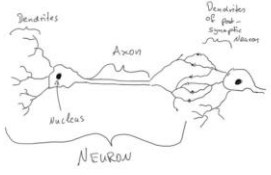


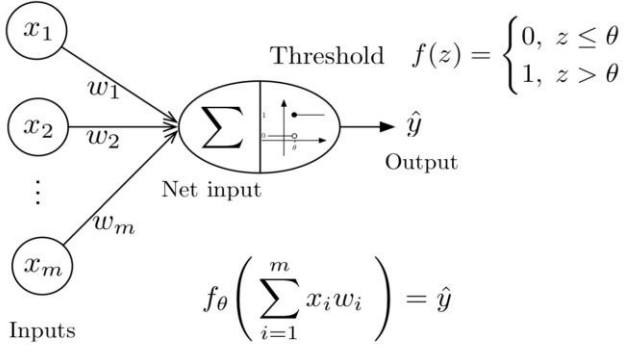
Truth Table - NAND		
X1	X2	Output Class
0	0	1
0	1	1
1	0	1
1	1	0

Truth Table - OR		
X1	X2	Output Class
0	0	0
0	1	1
1	0	1
1	1	1

Truth Table - XOR		
X1	X2	Output Class
0	0	0
0	1	1
1	0	1
1	1	0

Computational model for a Biological Neuron

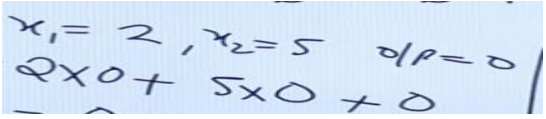




Perceptron Learning Algorithm

- If the prediction output is equal to the target, do nothing.
- If the prediction output is incorrect, there are two scenarios:
  - Scenario A: If the output is 0 and the target is 1, add the input vector to the weight vector.
  - Scenario B: If the output is 1 and the target is 0, subtract the input vector from the weight vector.

x1	x2	output
2	5	0
3	6	0
5	8	1
6	7	1

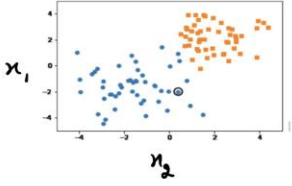


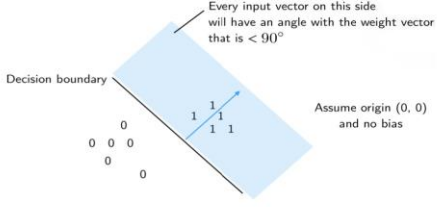
PLA

$$w_i = w_i + \alpha(y - \hat{y})x_i$$

$$b = b + \alpha(y - \hat{y})$$

Geometric Intuition





Limitations

- Perceptron is limited to linear classification boundaries, and cannot capture non-linear decision boundaries.
- It is a binary classifier, and cannot be used to solve problems that require multi-class classification or that have non-binary output.
- It may not converge if the classes are not linearly separable, making it ineffective for certain types of data.
- While there may be many "optimal" solutions in terms of 0/1 loss on the training data, most of these solutions will not perform well in terms of generalization, limiting the effectiveness of perceptron in real-world scenarios.