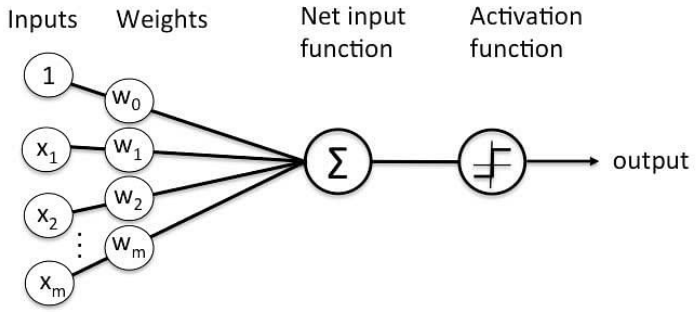
Report about Neural Networks

Zhivotovsky Dmitry
5130203/20102

Saint-Petersburg
2024

Perceptron

Perceptron is an algorithm for supervised learning of binary classifiers. This algorithm enables neurons to learn and processes elements in the training set one at a time. The following figure is a graphical representation of a perceptron.



Vector Representation of Data:

Inputs: x = [x_1, … , x_n], where each *xi* is a feature in the input vector.
Output: $y \in \{0,1\}$ (binary classification output).

Mathematical Formulation:

The combination function takes the input vector (x) to produce a combined value, or net input, (c). The combination is computed as bias plus a linear combination of the synaptic weights and the inputs in the perceptron,

$$c = w_0 + \sum_{i=1}^{n} w_i \cdot x_i$$

Linear Combination:

The Perceptron computes a linear combination of the inputs:

$$z = w_0 + \sum_{i=1}^{n} w_i \cdot x_i = X^T \cdot W$$

Where:

$W = [w_0, w_1, w_2, \dots, w_n]$ are the weights;
$X = [x_1, x_2, \dots, x_n]$ are the inputs.

Activation Function:

$$\hat{y} = f(z) = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{if } z < 0 \end{cases}$$

Loss Function:

The loss function is the misclassification error:

$$\ell(\boldsymbol{x}, \boldsymbol{y}) = \left(\boldsymbol{w}_{\hat{y}}^T \boldsymbol{x} + b_{\hat{y}}\right) - \left(\boldsymbol{w}_y^T \boldsymbol{x} + b_y\right)$$

- $\ell(\boldsymbol{x}, \boldsymbol{y}) > 0$ if $\hat{y} \neq y$
- $\ell(\boldsymbol{x}, \boldsymbol{y}) = 0$ if $\hat{y} = y$

Prediction Calculation:

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

Gradient Descent Algorithm:

The Perceptron uses uses stochastic gradient descendent algorithm (SGD) to optimize the weights and bias by minimizing the log loss.

Gradients and Weight Updates:

The gradient of the loss function with respect to weights is:

$$\nabla_w L = (\hat{y} - y) \cdot x$$

The gradient with respect to bias is:

$$\nabla_b L = \hat{y} - y$$

The weight and bias updates are:

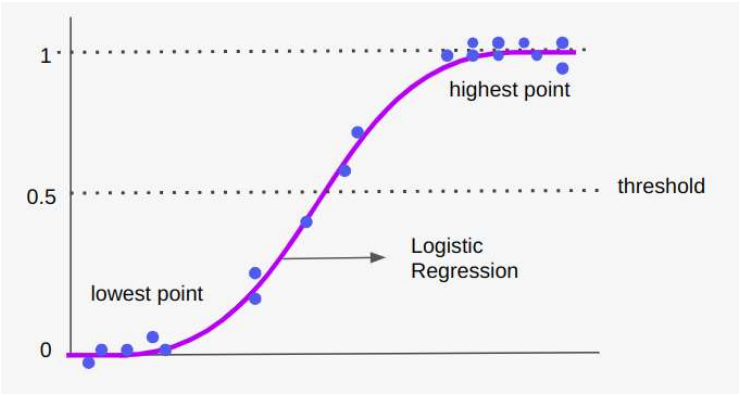
$$w_i \leftarrow w_i + \eta \cdot (y - \hat{y}) \cdot x_i$$

$$b \leftarrow b + \eta \cdot (y - \hat{y})$$

Where η is the learning rate.

Logistic Regression

Logistic regression is a probabilistic model used to describe the probability of discrete outcomes given input variables. It computes the probability of the result using the sigmoid function.



Vector Representation of Data:

Inputs: $\boldsymbol{x} = [x_1, x_2, \ldots, x_n]$, a vector of features.

Output: $y \in 0,1$, binary class label.

Mathematical Formulation:

Linear Combination:

Logistic Regression computes a linear combination of the inputs:

$$z = \boldsymbol{w}^T\boldsymbol{x} + b = w_1x_1 + w_2x_2 + \cdots + w_nx_n + b$$

Where b is bias.

Activation Function:

$$\hat{y} = \sigma(z) = \frac{e^z}{e^z + 1} = \frac{1}{1 + e^{-z}}$$

Loss Function:

$$L(y, \hat{y}) = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})]$$

Prediction Calculation:

$$\hat{y} = \sigma(\mathbf{w}^T\mathbf{x} + b)$$

$$\hat{y} = \begin{cases} 1, \; if \; \sigma(z) \geq 0.5 \\ 0, \; if \; \sigma(z) < 0.5 \end{cases}$$

Gradient Descent Algorithm:

Logistic Regression uses stochastic gradient descendent algorithm (SGD) to optimize the weights and bias by minimizing the log loss.

Gradients and Weight Updates:

The gradient of the loss function with respect to weights is:

$$\nabla_w L = (\hat{y} - y) \cdot x$$

The gradient with respect to bias is:

$$\nabla_b L = \hat{y} - y$$

The weight and bias updates are:

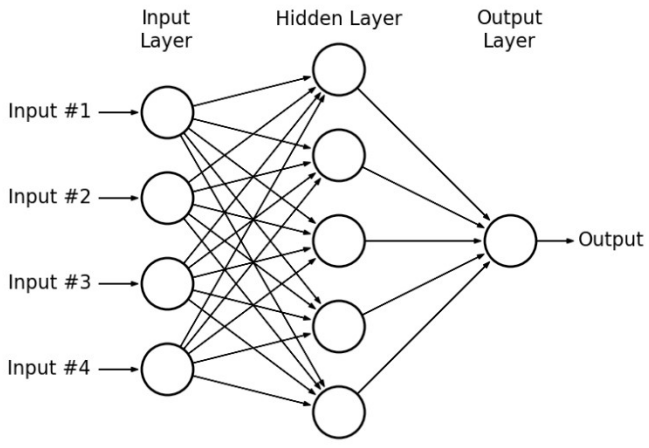
$$w_i \leftarrow w_i + \eta \cdot (y - \hat{y}) \cdot x_i$$

$$b \leftarrow b + \eta \cdot (y - \hat{y})$$

Where η is the learning rate.

Multilayer Perceptron

A multi-layer perceptron (MLP) is a type of artificial neural network that is composed of multiple layers of interconnected "neurons". These neurons are modeled after the neurons in the human brain, which are used to learn complex data and to make meaningful predictions.



Vector Representation of Data:

Inputs:

$$\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$$

Hidden Layer Outputs:

$$\mathbf{h} \in \mathbb{R}^m$$

Final Output:

$$\hat{y} \in [0, 1]$$

Mathematical Formulation:

Linear Combination for Hidden Layer:

$$\mathbf{z}^{(1)} = \mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)}$$

Activation Function:

For hidden layer, commonly ReLU:

$$h_i = \max(0, z_i^{(1)})$$

Sigmoid for output:

$$\hat{y} = \sigma(z^{(2)})$$

Output Layer Linear Combination:

$$z^{(2)} = \mathbf{W}^{(2)}\mathbf{h} + b^{(2)}$$

Loss Function:

Binary Cross-Entropy:

$$L(y, \hat{y}) = -[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})]$$

Prediction Calculation:

Forward pass:

$$\mathbf{h} = f(\mathbf{W}^{(1)}\mathbf{x} + \mathbf{b}^{(1)})$$

$$\hat{y} = \sigma(\mathbf{W}^{(2)}\mathbf{h} + b^{(2)})$$

Gradient Descent Algorithm:

Utilizes backpropagation to compute gradients for all weights and biases through the network.

Gradients and Weight Updates:

For each layer:

$$\Delta_W = -\eta \cdot \nabla L$$

$$\Delta_b = -\eta \cdot \nabla L$$

This report has provided a detailed mathematical foundation for the Perceptron, Logistic Regression, and MLP, covering essential aspects from data representation to gradient updates.