<u>KALULU GARVEN, 5130203-20101.</u>

FOUNDATIONS AND IMPLEMENTATION OF NEURAL NETWORK ALGORITHMS: PERCEPTRON, LOGISTIC REGRESSION, AND MULTILAYER PERCEPTRON.

Introduction

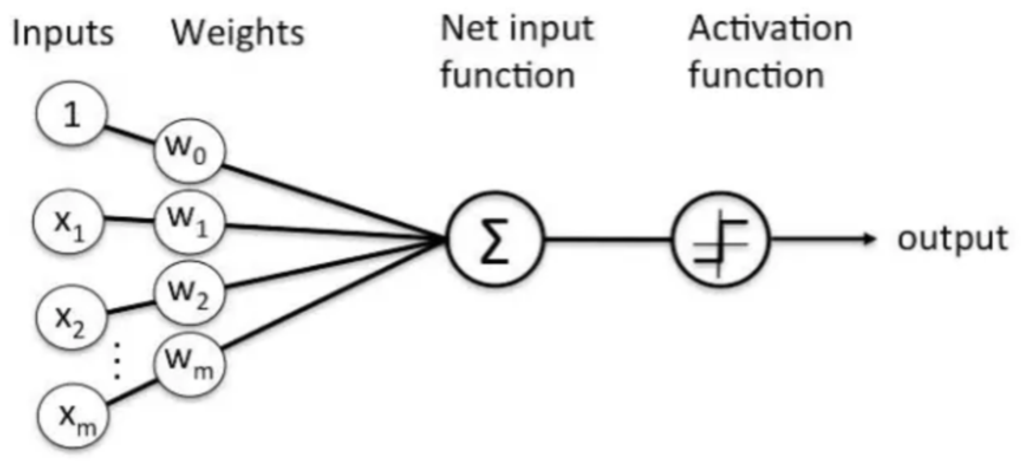
Neural networks as a whole have revolutionized the field of artificial intelligence by enabling machines to perform complex tasks such as image recognition, natural language processing, and decision-making with remarkable accuracy. In this report I tackled the foundational algorithms underpinning neural networks, thus the Perceptron, Logistic Regression, and Multilayer Perceptron (MLP).

1. PERCEPTRON

Model Architecture

The perceptron is a single-layer neural network designed for binary classification. It consists of input features connected to a single output through weighted connections.

Picture 1.1 Model Architecture of Perceptron.



Vector Representation of Data

- Input vector: $x = [x_1, x_2, \dots, x_n]^T \epsilon\ \mathbb{R}^n$
- Weight vector : $w = [w_1, w_2, \dots, w_n]^T \epsilon\ \mathbb{R}^n$
- Bias : $b\ \epsilon\ \mathbb{R}^n$
- Output: y ∈{-1, 1} (binary classification)

Mathematical Formulation

1. Linear Combination:

$$z = w^T x + b$$

2. Activation Function:

$$y = \begin{cases} 1 & if\ z \geq 0 \\ -1 & if\ z < 0 \end{cases}$$

Prediction Formula ($\hat{y}$)

The perceptron computes predictions based on the sign of the weighted sum:

$$\hat{y} = \begin{cases} 1 & w^T x + b \geq 0 \\ -1 & otherwise. \end{cases}$$

Gradient Descent Algorithm

The perceptron uses a simple update rule for each misclassified sample:

1. **Loss Function:**
   The perceptron loss for a misclassified point is:

$$\mathcal{L} = -y_i \cdot z_i$$

2. **Gradient of Loss:**

   For weights:

$$\nabla_w \mathcal{L} = -y_i \cdot z_i$$

   For bias:

$$\nabla_b \mathcal{L} = -y_i$$

3. Weight/Bias Updates:

   Update the weights and bias as follows

$$w \leftarrow w + \eta \cdot y_i \cdot x_i$$

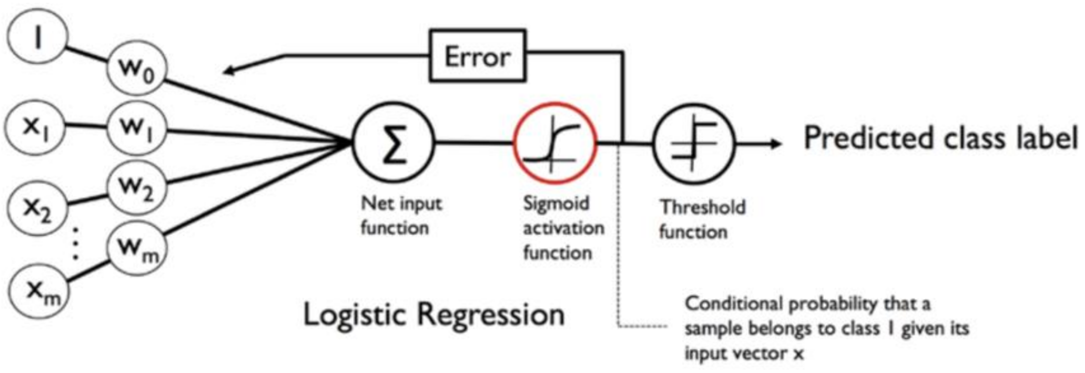
$$b \leftarrow b + \eta \cdot y_i$$

2. LOGISTIC REGRESSION

Model Architecture

Logistic regression is a linear classifier that predicts probabilities through a sigmoid activation function.

Picture 2.1 Model Architecture of Logistic Regression.



1
w0
x1
w1
x2
w2
xm
wm
Error
Σ
Net input function
Sigmoid activation function
Threshold function
Predicted class label
Logistic Regression
Conditional probability that a sample belongs to class 1 given its input vector x

Vector Representation of Data

- Input vector: $x = [x_1, x_2, \dots, x_n]^T \epsilon\, \mathbb{R}^n$
- Weight vector : $w = [w_1, w_2, \dots, w_n]^T \epsilon\, \mathbb{R}^n$
- Bias: $b\ \epsilon\, \mathbb{R}^n$
- Output: y ∈{0, 1}

Mathematical Formulation

1. Linear Combination:

$$z = w^T x + b$$

2. Activation Function:

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

3. Loss Function (Binary Cross-Entropy):

$$\mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}[y_i\, log(\hat{y}_i) + (1 - y_i)\, log(1 - \hat{y}_i)]$$

Prediction Formula ($\hat{y}$)

Logistic regression predicts the probability of y=1:

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

For binary classification:

$$\hat{y} = \begin{cases} 1 & if\ \sigma(z) \geq 0.5 \\ -1 & otherwise. \end{cases}$$

Gradient Descent Algorithm

1. Gradients of Loss:
   For weights:

$$\nabla_w \mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)\, x_i$$

For bias:

$$\nabla_b \mathcal{L} = -\frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)$$

2. Weight/Bias Updates:
   Update using gradient descent:

$$w \leftarrow w - \eta \cdot \nabla_w \mathcal{L}$$

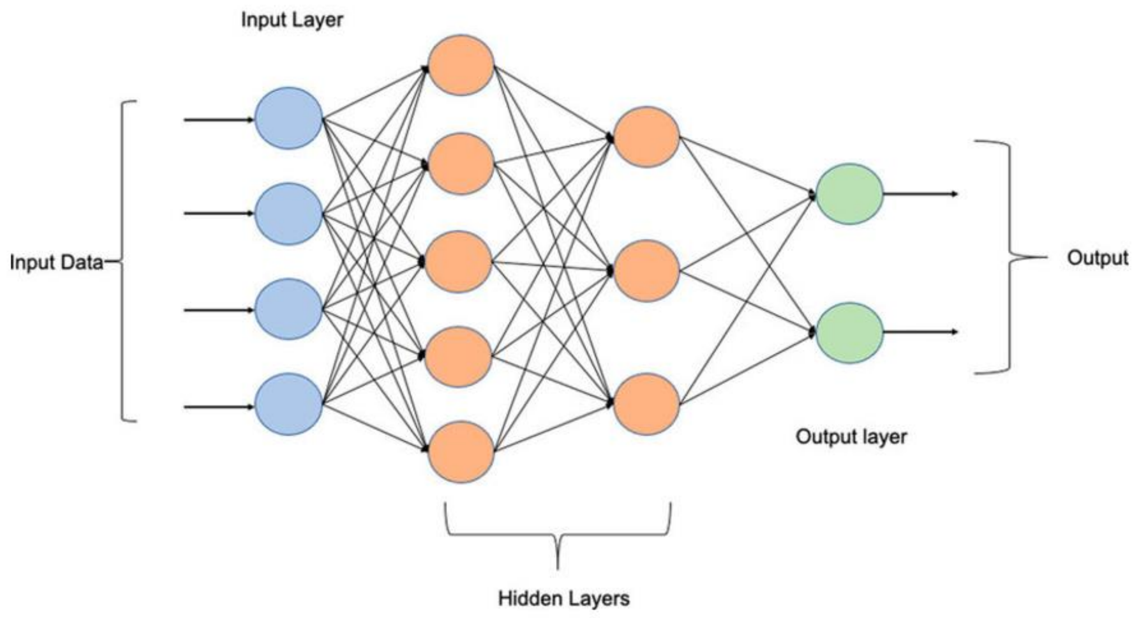
$$b \leftarrow b - \eta \cdot \nabla_b \mathcal{L}$$

3. MULTILAYER PERCEPTRON (MLP)

Model Architecture

An MLP consists of an input layer, one or more hidden layers with nonlinear activations, and an output layer.

Picture 3.1 Model Architecture of Multilayer Perceptron (MLP).



Multi-layer perceptron (MLP-NN) basic Architecture.

Vector Representation of Data

- Input: $x \in \mathbb{R}^n$
- Weights and biases per layer:
- Layer l: $w^{(l)}, b^{(l)}$
- Output $\hat{y} \in \mathbb{R}^m$ for m-class classification

Mathematical Formulation

1. Linear Combination at Layer l:

$$z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$$

   Where $a^{(0)} = x$

2. Activation Function:

$$a^{(l)} = f\left(z^{(l)}\right)$$

   Common activations include ReLU, Sigmoid, and Tanh

3. Output Layer Activation:

- For classification: Softmax

$$softmax\left(z_j\right) = \frac{e^{z_j}}{\Sigma_k e^{z_k}}$$

4. Loss Function (Cross-Entropy):

$$\mathcal{L} = -\frac{1}{N} \sum_{i=1}^{N} \sum_{k=1}^{C} y_i\,, k\, log(\hat{y}_i, k)$$

Prediction Formula ($\hat{y}$)

$$\hat{y} = f\left(z^{(L)}\right)$$

where f is the activation function of the output layer (e.g., softmax for classification).

Gradient Descent Algorithm

1. **Backpropagation:** Gradients are computed layer by layer:

Output Layer:

$$\delta^{(L)} = \hat{y} - y$$

Hidden Layer l:

$$\delta^{(l)} = \left(w^{(l+1)}\right)^T \delta^{(l+1)} \odot f'\left(z^{(l)}\right)$$

2. Weight/Bias Updates:
   For each layer l:

$$W^{(l)} \leftarrow W^{(l)} - \eta \cdot \left(\delta^{(l)} \cdot a^{(l-1)^T}\right)$$
$$b^{(l)} \leftarrow b^{(l)} - \eta \cdot \delta^{(l)}$$

Lastly, the foundational algorithms of neural networks, including the Perceptron, Logistic Regression, and Multilayer Perceptron, demonstrate how mathematical principles drive machine learning models to make accurate predictions. Through linear combinations, activation functions, and gradient-based optimization, these models progressively learn from data. The exploration and implementation of these mechanisms highlight their critical role in powering modern artificial intelligence applications.