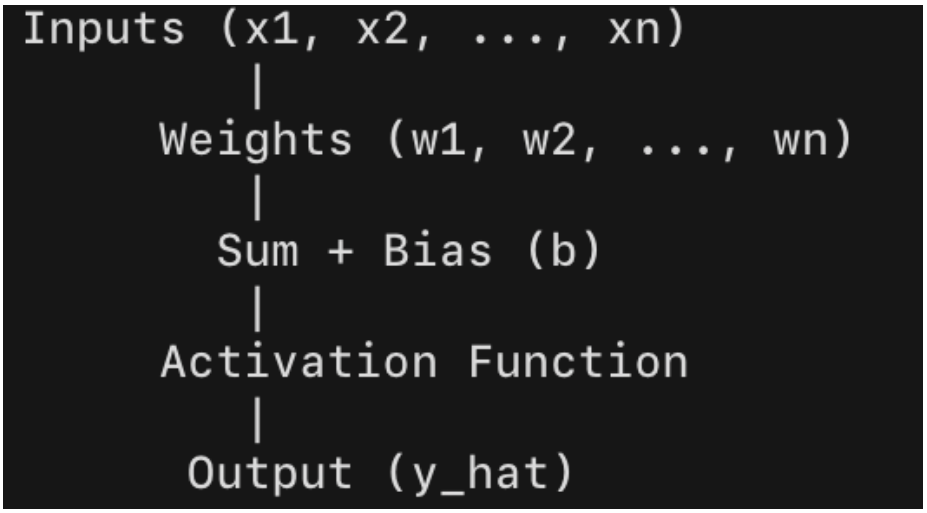
Neural networks are a subset of machine learning models inspired by the structure and function of the brain. They consist of interconnected nodes (neurons) that process input data and produce output. This report focuses on three fundamental models: the Perceptron, Logistic Regression, and Multilayer Perceptron.

Perceptron



Vector Representation

$$W = \begin{pmatrix} w_1^T \\ \cdots \\ w_u^T \end{pmatrix} = \begin{pmatrix} w_{1,1} & \cdots & w_{1,n} \\ & \cdots & \\ w_{u,1} & \cdots & w_{u,n} \end{pmatrix}$$

Linear Combination

$z = \mathbf{w}^T\mathbf{x} + b$ where $\mathbf{w} = [w_1, w_2, \ldots, w_n]^T$

Activation Function

$$y_{\text{hat}} = \begin{cases} 1 & \text{if } z \geq 0 \\ 0 & \text{otherwise} \end{cases}$$

Loss Function

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

Predictions

The prediction \(\hat{y} \) can be represented as: $\hat{y} = \text{Activation}(z)$

Gradient Descent Algorithm

Gradient Descent is used to minimize the loss function. The basic idea is to update the weights to reduce the difference between the predicted output and the actual output.

Formulas of Gradients and Weights/Biases Updates

For a single instance:

Gradient with respect to weights: $\frac{\partial L}{\partial w_i} = (y_{\text{hat}} - y) \cdot x_i$

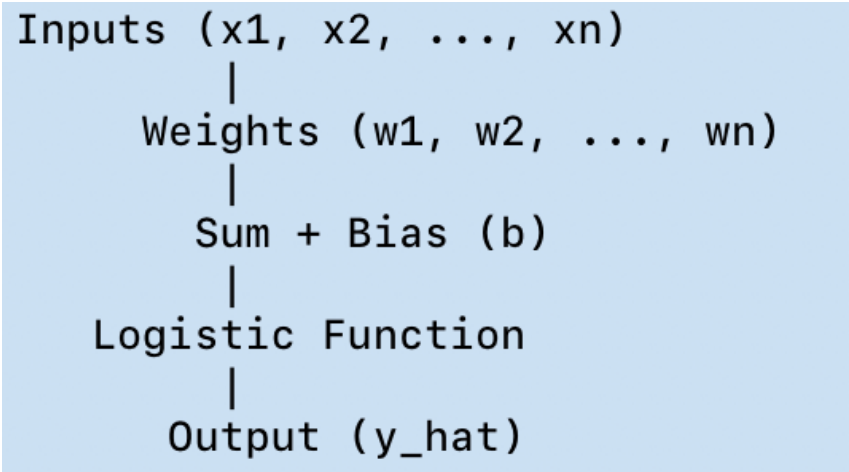
Gradient with respect to bias: $\frac{\partial L}{\partial b} = y_{\text{hat}} - y$

Weight and Bias Update

Using a learning rate η: $w_i := w_i - \eta \frac{\partial L}{\partial w_i}$ $b := b - \eta \frac{\partial L}{\partial b}$ ...

Logistic Regression

Logistic Regression can be visualized similarly to a Perceptron, but with a logistic function used in the activation step.



Vector Representation

Input vector: $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ Output: $y \in \{0,1\}$

Linear Combination

$$z = \mathbf{w}^T\mathbf{x} + b$$

Activation Function

Logistic Function: $y_{\text{hat}} = \sigma(z) = \dfrac{1}{1 + e^{-z}}$

Loss Function

Binary Cross-Entropy Loss: $L(y, y_{\text{hat}}) = -\left[y \log(y_{\text{hat}}) + (1 - y)\log(1 - y_{\text{hat}})\right]$

Predictions

The prediction $\hat{y}$ can be represented as: $\hat{y} = \sigma(z)$

Gradient Descent Algorithm

Similar to the Perceptron, but with a more complex gradient calculation based on the logistic function.

Formulas of Gradients and Weights/Biases Updates

Gradient with respect to weights: $\frac{\partial L}{\partial w_i} = (y_{\text{hat}} - y) \cdot x_i$

Gradient with respect to bias: $\frac{\partial L}{\partial b} = y_{\text{hat}} - y$

Weight and Bias Update

Using a learning rate η

$$w_i := w_i - \eta \frac{\partial L}{\partial w_i}$$

$$b := b - \eta \frac{\partial L}{\partial b}$$

....

Multilayer Perceptron

A Multilayer Perceptron (MLP) consists of an input layer, one or more hidden layers, and an output layer.

```
Input Layer -> Hidden Layer(s) -> Output Layer
```

Vector Representation

Input vector: $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$
Hidden layer outputs: $\mathbf{h} = [h_1, h_2, \ldots, h_m]^T$
Output layer: y

Linear Combination

For the hidden layer: $z^{(1)} = \mathbf{w}^{(1)}\mathbf{x} + b^{(1)}$
For the output layer: $z^{(2)} = \mathbf{w}^{(2)}\mathbf{h} + b^{(2)}$ ....

Activation Function

Commonly used activation function (e.g., ReLU for hidden layers):
$h = \max(0, z^{(1)})$
Logistic activation for output: $y_{\text{hat}} = \sigma(z^{(2)})$

Loss Function

Binary Cross-Entropy Loss: $L(y, y_{\text{hat}}) = -[y \log(y_{\text{hat}}) + (1 - y)\log(1 - y_{\text{hat}})]$

Predictions

The prediction $\hat{y}$ can be represented as: $\hat{y} = \sigma(z^{(2)})$

Gradient Descent Algorithm

Backpropagation is used to compute the gradient efficiently.

Formulas of Gradients and Weights/Biases Updates

For each layer, the gradients can be defined similarly:
For the output layer: $\delta^{(2)} = (y_{\text{hat}} - y) \cdot \sigma'(z^{(2)})$
For the hidden layer: $\delta^{(1)} = \delta^{(2)} \cdot \mathbf{w}^{(2)} \cdot \sigma'(z^{(1)})$

Weight updates:

$$\mathbf{w}^{(2)} := \mathbf{w}^{(2)} - \eta\delta^{(2)}\mathbf{h}^T$$
$$\mathbf{w}^{(1)} := \mathbf{w}^{(1)} - \eta\delta^{(1)}\mathbf{x}^T$$

Bias updates:

$$b^{(2)} := b^{(2)} - \eta\delta^{(2)}$$
$$b^{(1)} := b^{(1)} - \eta\delta^{(1)}$$

....

Conclusion

Neural networks, including Perceptrons, Logistic Regression, and Multilayer Perceptrons, are powerful tools for classification and regression tasks. Understanding their mathematical formulations, architecture, and training algorithms is fundamental for applying these models effectively.