Machine Learning Models: Perceptron, Logistic Regression, and Multilayer Perceptron

Maximov Ivan

December 7, 2024

1. Perceptron

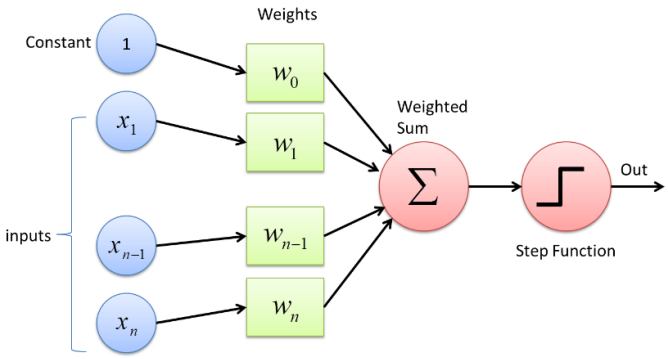

Figure 1: Perceptron Model Architecture

The Perceptron is a simple linear classifier consisting of an input and output layer. The model computes a weighted sum of the input features and passes it through an activation function to produce the output.
The input data is represented as:

$$\mathbf{x} = (x_1, x_2, \ldots, x_n)$$

with output $y \in \{0, 1\}$.
The linear combination is:

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

where $\mathbf{w}$ is the weight vector, and b is the bias. The activation function is the step function:

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

The loss function is the misclassification error:

$$L = \sum_{i=1}^{m} 1(y_i \neq \hat{y}_i)$$

The gradient descent algorithm minimizes the loss by adjusting the weights and bias. The update rule is:

$$w_j = w_j - \eta \frac{\partial L}{\partial w_j}$$

where η is the learning rate and $\frac{\partial L}{\partial w_j}$ is the gradient of the loss with respect to w_j. The gradients are:

$$\frac{\partial L}{\partial w_j} = -\sum_{i=1}^{m} 1(y_i \neq \hat{y}_i) x_{ij}, \quad \frac{\partial L}{\partial b} = -\sum_{i=1}^{m} 1(y_i \neq \hat{y}_i)$$

Then, the weights and bias are updated as:

$$w_j = w_j - \eta \frac{\partial L}{\partial w_j}, \quad b = b - \eta \frac{\partial L}{\partial b}$$

2. Logistic Regression

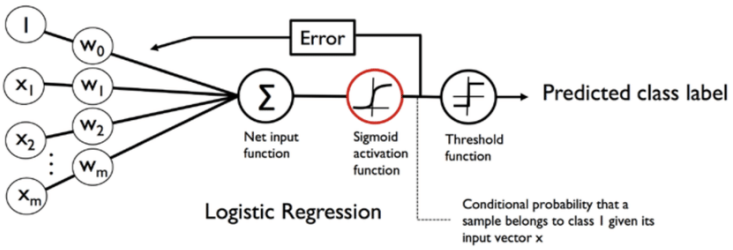

Figure 2: Logistic Regression Model Architecture

Logistic regression is a linear model for binary classification that computes the probability of a sample belonging to the positive class.
The input vector is:

$$\mathbf{x} = (x_1, x_2, \ldots, x_n)$$

and the output is the probability:

$$\hat{y} = P(y = 1|\mathbf{x})$$

The linear combination is:

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

The activation function is the sigmoid:

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

where $\sigma(z)$ maps the linear combination to a value between 0 and 1. The loss function is binary cross-entropy:

$$L = -\sum_{i=1}^{m} (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$$

The gradient descent algorithm follows:

$$w_j = w_j - \eta \frac{\partial L}{\partial w_j}, \quad \frac{\partial L}{\partial w_j} = -\sum_{i=1}^{m} (y_i - \hat{y}_i) x_{ij}$$

$$\frac{\partial L}{\partial b} = -\sum_{i=1}^{m} (y_i - \hat{y}_i)$$

3. Multilayer Perceptron

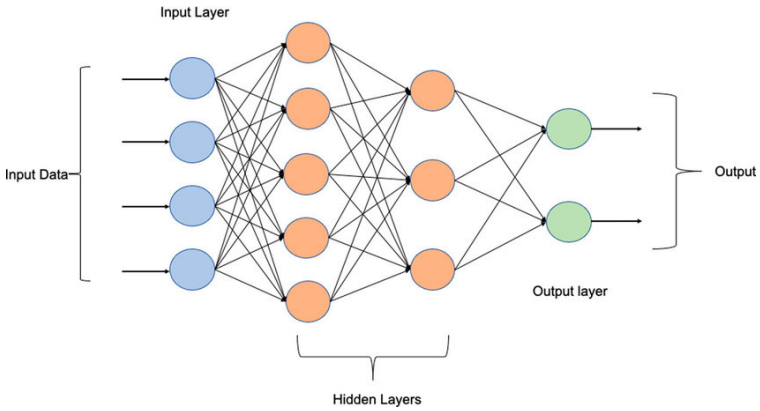

Figure 3: Multilayer Perceptron Model Architecture

The Multilayer Perceptron (MLP) consists of an input layer, hidden layers, and an output layer. It uses a non-linear activation function in the hidden layers to model complex relationships.

The input vector is:

$$\mathbf{x} = (x_1, x_2, \ldots, x_n)$$

For a network with one hidden layer, the linear combinations are:

$$z_1 = \mathbf{w}_1^T \mathbf{x} + b_1 \quad \text{(hidden layer)}, \quad z_2 = \mathbf{w}_2^T \mathbf{h} + b_2 \quad \text{(output layer)}$$

where $\mathbf{h} = \sigma(z_1)$ is the activation of the hidden layer, and σ is a non-linear activation function (e.g., ReLU or sigmoid).

The activation function for each layer is:

$$\hat{y} = \sigma(z_2)$$

The loss function is cross-entropy loss:

$$L = -\sum_{i=1}^{m} \left(y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right)$$

The gradient descent algorithm for MLP involves backpropagation. The gradients of the loss with respect to the weights and biases are computed recursively, starting from the output layer and moving backward through the hidden layers.

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

$$\frac{\partial L}{\partial w_j} = -\sum_{i=1}^{m} (y_i - \hat{y}_i) x_{ij}$$

and for the hidden layers:

$$\frac{\partial L}{\partial w_1} = \frac{\partial L}{\partial h_1} \frac{\partial h_1}{\partial z_1} \mathbf{x}$$

Then, the weights and biases are updated using:

$$w_j = w_j - \eta \frac{\partial L}{\partial w_j}, \quad b_j = b_j - \eta \frac{\partial L}{\partial b_j}$$