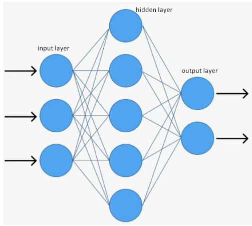
Perceptron, Logistic Regression and Multilayer Perceptron

Shmeleva Mariia 5130203/20101

1 Perceptron

1.1 Model Architecture



1.2 Vector Representation of Data

Input vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ and the target output $y \in \{-1, 1\}$.

1.3 Mathematical Formulation

Linear Combination

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

Activation Function

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

Loss Function
Perceptron loss for misclassified samples:

$$L(\mathbf{w}, b) = -y(\mathbf{w}^T \mathbf{x} + b)$$

1.4 Prediction Calculation

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

1.5 Gradient Descent Algorithm

Update weights only when a sample is misclassified.

1.6 Gradients and Weight/Bias Updates

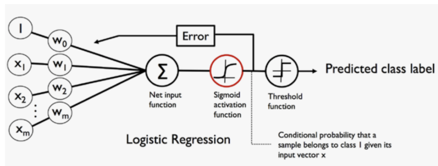
$$\mathbf{w} \leftarrow \mathbf{w} + \eta y \mathbf{x}$$

$$b \leftarrow b + \eta y$$

where η is the learning rate.

2 Logistic Regression

2.1 Model Architecture



2.2 Vector Representation of Data

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

2.3 Mathematical Formulation

Linear Combination

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

Activation Function
Sigmoid function:

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

Loss Function
Binary cross-entropy loss:

$$L(\mathbf{w}, b) = -\left[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\right]$$

2.4 Prediction Calculation

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

2.5 Gradient Descent Algorithm

Iteratively update weights to minimize the loss function.

2.6 Gradients and Weight/Bias Updates

Gradient of the loss w.r.t weights and bias:

$$\nabla_{\mathbf{w}} L = (\hat{y} - y)\mathbf{x}$$

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

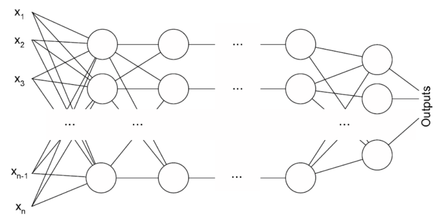
Update rules:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \nabla_{\mathbf{w}} L$$

$$b \leftarrow b - \eta \nabla_b L$$

3 Multilayer Perceptron

3.1 Model Architecture



3.2 Vector Representation of Data

Input vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$, target output y.

3.3 Mathematical Formulation

Linear Combination and Activation in Hidden Layer

For hidden layer neurons:

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

Output Layer

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

Activation function f can be ReLU, sigmoid, etc.

Loss Function

For regression tasks:

$$L = \frac{1}{2}(y - \hat{y})^2$$

For classification tasks:

$$L = -\sum_k y_k \log(\hat{y}_k)$$

3.4 Prediction Calculation

Compute outputs through forward propagation.

3.5 Gradient Descent Algorithm

Backpropagate errors to update weights.

3.6 Gradients and Weight/Bias Updates

Compute gradients using chain rule:

$$\delta^{(2)} = \nabla_{\hat{y}} L \cdot f'(z^{(2)})$$
$$\delta^{(1)} = (\mathbf{W}^{(2)})^T \delta^{(2)} \cdot f'(\mathbf{z}^{(1)})$$

Update weights and biases:

$$\mathbf{W}^{(2)} \leftarrow \mathbf{W}^{(2)} - \eta\delta^{(2)}(\mathbf{h})^T$$
$$\mathbf{b}^{(2)} \leftarrow \mathbf{b}^{(2)} - \eta\delta^{(2)}$$
$$\mathbf{W}^{(1)} \leftarrow \mathbf{W}^{(1)} - \eta\delta^{(1)}\mathbf{x}^T$$
$$\mathbf{b}^{(1)} \leftarrow \mathbf{b}^{(1)} - \eta\delta^{(1)}$$