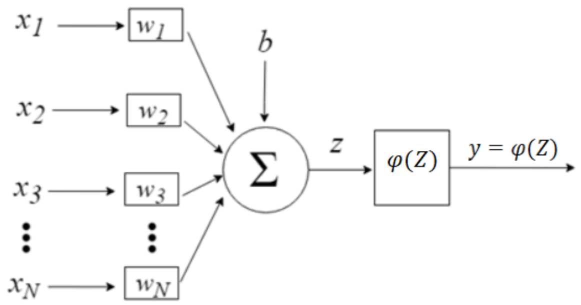
Vashchylko Alexei
group 5130203/20102

Neural Networks Report

1. Perceptron

A single-layer perceptron model consists of a feed-forward network and includes a threshold transfer function for thresholding on the Output. The main objective of the single-layer perceptron model is to classify linearly separable data with binary labels.

- **Model architecture**



- **Vector representation of data (inputs and outputs)**

Inputs: $X = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_N \end{bmatrix}$ Outputs: $Y = [y], y \in \{1,0\}$

Math formulation of linear combination, activation function and loss function

- **Linear combination**

$$Z = \sum_{i=1}^{N} w_i\, x_i + b = [w_1 \quad w_2 \quad \dots \quad w_N] \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_N \end{bmatrix} + [b] = W^T X + B,$$

where

$W = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_N \end{bmatrix}$ – the weight vector, $B = [b]$ - the bias.

- **Activation Function:**

The result of linear combination Z is passed to the activation function.

- threshold function: $\varphi(Z) = \begin{cases} 1\,, Z \geq 0, \\ 0\,, Z < 0. \end{cases}$

- **Loss function**

The loss function is a quantity that reflects the discrepancy between the real value (y) and the predicted value $(\hat{y})$ output.

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

- **A mathematical formulation of how neural networks make predictions**

$$\hat{y} = \varphi(z) = \varphi\left(\sum_{i=1}^{N} w_i\, x_i + b\right)$$
$$\hat{Y} = \varphi(W^T X + B)$$

- **Explanation of gradient descendent algorithm**

Gradient descent is an optimization algorithm used to minimize the loss function by iteratively updating the weights and biases

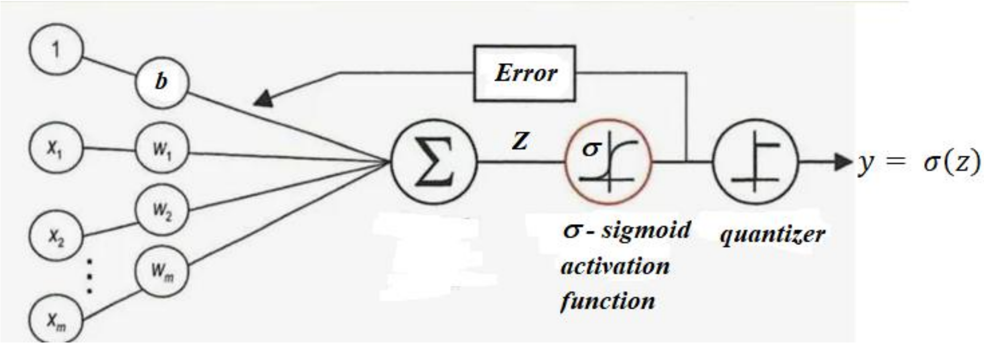
- **Formulas of gradients and weights/biases updates**

$$w_i \leftarrow w_i - \eta \frac{\partial L}{\partial \omega_i} x_i = w_i + \eta(y - \hat{y})x_i, \qquad b \leftarrow b - \eta \frac{\partial L}{\partial b} = b + \eta(y - \hat{y}).$$

where η is the learning rate.

2. Logistic Regression

- **Model architecture**



- **Vector representation of data (inputs and outputs)**

Inputs: $X = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_N \end{bmatrix}$ Outputs: $Y = [y], y \in \{1,0\}$

Math formulation of linear combination, activation function and loss function

- **Linear combination**

$$Z = \sum_{i=1}^{N} w_i\, x_i + b = [w_1 \quad w_2 \quad \dots \quad w_N] \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_N \end{bmatrix} + [b] = W^T X + B,$$

where

$W = \begin{bmatrix} w_1 \\ w_2 \\ \dots \\ w_N \end{bmatrix}$ – the weight vector, $B = [b]$ - the bias.

- **Activation Function:**

The result of linear combination Z is passed to the activation function.

- sigmoid function:

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

- **Loss function**

The loss function is a quantity that reflects the discrepancy between the real value (y) and the predicted value $(\hat{y})$ output.

- binary cross entropy

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

- **A mathematical formulation of how neural networks make predictions**

$$\hat{y} = \sigma(z) = \sigma\left(\sum_{i=1}^{N} w_i\, x_i + b\right)$$

$$\hat{Y} = \sigma(W^T X + B)$$

- **Explanation of gradient descendent algorithm**

The objective of the gradient descent algorithm is to minimize the binary cross-entropy loss $L(y, \hat{y})$ by iteratively updating the weights and bias. The cost function is the average of loss function of the entire training set. We are going to find the parameters w and b that minimize the overall cost function

$$J(w, b) = \frac{1}{m}\sum_{i=1}^{m} L\left(y^{(i)}, \hat{y}^{(i)}\right) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\, log\hat{y}^{(i)} + \left(1 - y^{(i)}\right)\log\left(1 - \hat{y}^{(i)}\right)\right]$$

- **Formulas of gradients and weights/biases updates**

$$w_i \leftarrow w_i - \eta\frac{\partial J(w,b)}{\partial \omega_i}x_i, \qquad b \leftarrow b - \eta\frac{\partial J(w,b)}{\partial b}.$$

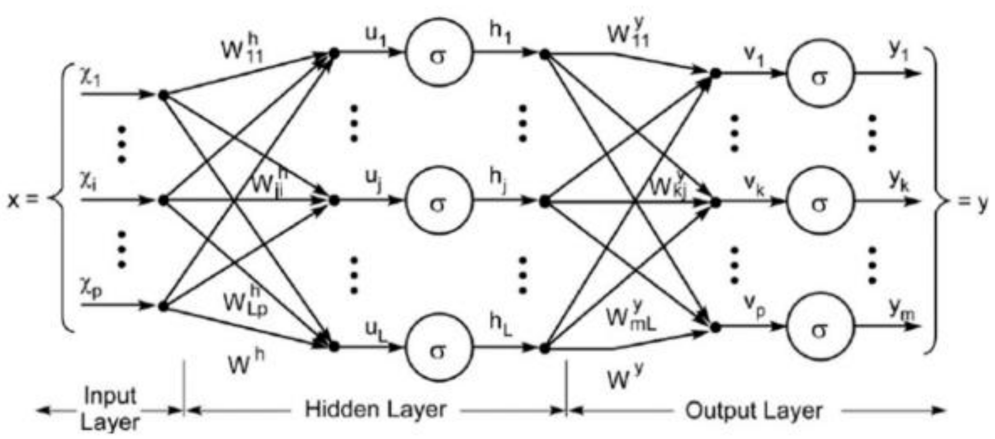
where η is the learning rate,

$$\frac{\partial J(w,b)}{\partial \omega_i} = \frac{1}{m}\sum_{j=1}^{m}\left(y^{(j)} - \hat{y}^{(j)}\right){x_i}^{(j)},$$

$$\frac{\partial J(w,b)}{\partial b} = \frac{1}{m}\sum_{j=1}^{m}\left(y^{(j)} - \hat{y}^{(j)}\right).$$

3. Multilayer Perceptron (MLP)

- **Model architecture**



- **Vector representation of data (inputs and outputs)**

Inputs: $X = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_p \end{bmatrix}$ Outputs: $Y = \begin{bmatrix} y_1 \\ y_2 \\ \dots \\ y_m \end{bmatrix}$

Math formulation of linear combination, activation function and loss function

- **Linear combination**

$$z^{(l)} = w^{(l)}a^{(l)} + b^{(l)},$$

where l – layer.

- **Activation Function:**

Common activation functions:

- sigmoid function:

$$\varphi(Z) = \frac{1}{1 + e^{Z}}$$

- ReLU: $\varphi(\mathrm{Z}) = \max(0, Z)$

- Tanh

$$\varphi(Z) = \frac{e^{Z} - e^{-Z}}{e^{Z} + e^{-Z}}$$

$a^{(l)} = \varphi\left(Z^{(l)}\right)$

- **Loss function**

The loss function is a quantity that reflects the discrepancy between the real value (y) and the predicted value $(\hat{y})$ output.

- Cross entropy (CE) (for multi-class classification):

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

- **A mathematical formulation of how neural networks make predictions**

$$\hat{y} = a^{(l)} = \varphi\left(Z^{(l)}\right) = \varphi\left(w^{(l)} a^{(l)} + b^{(l)}\right),$$

where $l -$ layer.

- **Explanation of gradient descendent algorithm**

Gradient Descent minimizes the loss $L(y, \hat{y})$ by updating weights $w^{(l)}$ and biases $b^{(l)}$ using the gradients of the loss.

Backward Propagation: Gradients are computed for each layer starting from the output layer (using the chain rule).

- **Formulas of gradients and weights/biases updates**

For each layer (l):

$$w^{(l+1)} \leftarrow w^{(l)} - \eta \frac{\partial L}{\partial w^{(l)}}, \qquad b^{(l+1)} \leftarrow b^{(l)} - \eta \frac{\partial L}{\partial b^{(l)}}.$$

where η is the learning rate.