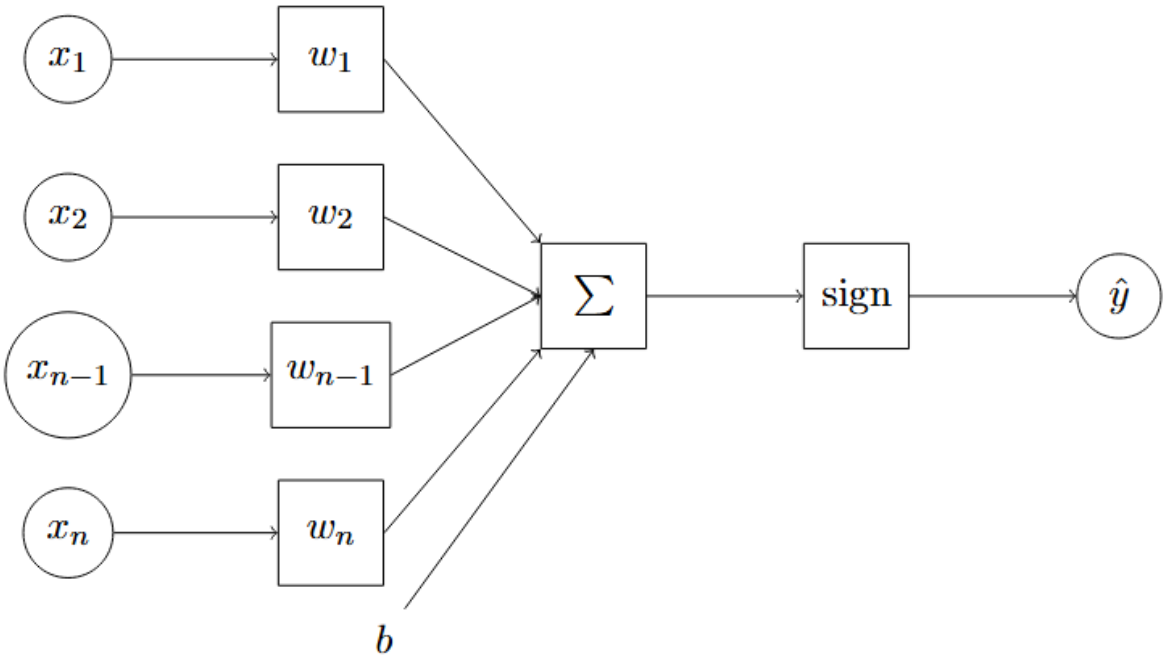
Report about Neural Networks

Karakeschishyan Andronik 5130203/20102

Perceptron Algorithm

Model Architecture: The perceptron solves linearly separable classification tasks by iteratively updating parameters until all examples are correctly classified. Perceptron consists of inputs, weights, a bias, and an output. The model structure is illustrated in the diagram below.



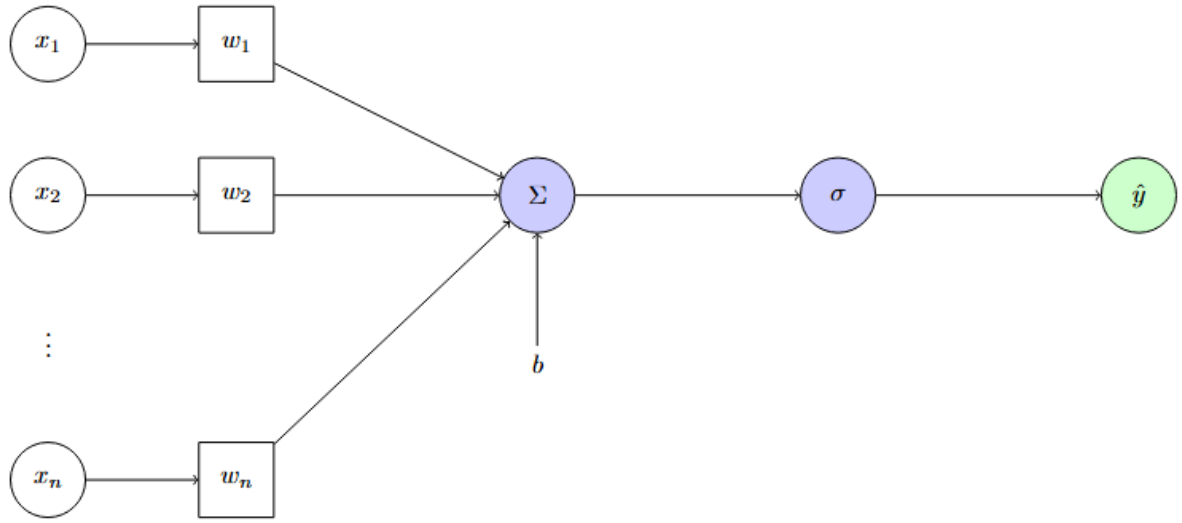
Vector Representation of Data: Input vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^\top$, weights $\mathbf{w} = [w_1, w_2, \ldots, w_n]^\top$, and bias b. Output $y \in \{-1, 1\}$.
Linear Combination: $z = \mathbf{w}^\top \mathbf{x} + b$.
Activation Function: $y = \text{sign}(z) = \begin{cases} +1, & z \geq 0, \\ -1, & z < 0. \end{cases}$
Loss Function: $\mathcal{L} = -\sum_{i=1}^{m} y^{(i)}(\mathbf{w}^\top \mathbf{x}^{(i)} + b)$, where m is the number of training examples.
Prediction: $\hat{y} = \text{sign}(\mathbf{w}^\top \mathbf{x} + b)$.
Gradient Descent: If $y \neq \hat{y}$, update weights and bias:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta y \mathbf{x}, \quad b \leftarrow b + \eta y,$$

where η is the learning rate.
Gradients: $\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = -y\mathbf{x}, \quad \frac{\partial \mathcal{L}}{\partial b} = -y.$

Logistic Regression

Model Architecture: Logistic regression is a linear model for binary classification tasks. It maps input features through a linear combination and applies a sigmoid activation function to predict probabilities.
Diagram:.



Vector Representation of Data: Input vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^\top$, weights $\mathbf{w} = [w_1, w_2, \ldots, w_n]^\top$, and bias b. Output $y \in \{0, 1\}$. Predicted probability $\hat{y} \in [0, 1]$.
Linear Combination: $z = \mathbf{w}^\top \mathbf{x} + b$.
Activation Function: $\sigma(z) = \frac{1}{1+e^{-z}}$.
Loss Function: Binary cross-entropy:

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

where m is the number of training examples.
Prediction: $\hat{y} = \sigma(\mathbf{w}^\top \mathbf{x} + b)$, where $\hat{y}$ is interpreted as the probability of $y = 1$. A threshold (e.g., 0.5) is applied to decide the class.
Gradient Descent: Gradients are calculated for each parameter, and weights/biases are updated as follows:

$$\mathbf{w} \leftarrow \mathbf{w} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{w}}, \quad b \leftarrow b - \eta \frac{\partial \mathcal{L}}{\partial b},$$

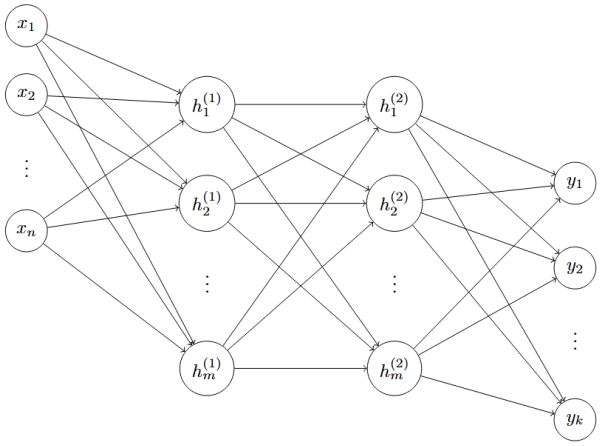
where η is the learning rate.
Gradients: Partial derivatives of the loss function:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{w}} = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)} \right) \mathbf{x}^{(i)}, \quad \frac{\partial \mathcal{L}}{\partial b} = \frac{1}{m} \sum_{i=1}^{m} \left(\hat{y}^{(i)} - y^{(i)} \right).$$

Multilayer Perceptron

Model Architecture: A multilayer perceptron (MLP) is a feedforward neural network with hidden layers enabling non-linear classification. Each layer comprises neurons with weights, biases, and activation functions.

Diagram:



Vector Representation: Input vector $\mathbf{x} = [x_1, x_2, \ldots, x_n]^\top$, layer l parameters $\mathbf{W}^{(l)}, \mathbf{b}^{(l)}$, and output $\mathbf{y}$.

Linear Combination: For layer l, pre-activation:

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

Activation: $\text{ReLU}(z) = \max(0, z)$ or $\sigma(z) = \frac{1}{1+e^{-z}}$. Output layer:

$$\hat{\mathbf{y}} = \text{softmax}(\mathbf{z}^{(L)}) = \frac{e^{z_i^{(L)}}}{\sum_{j=1}^{k} e^{z_j^{(L)}}}.$$

Loss Function: Cross-entropy for m examples and k classes:

$$\mathcal{L} = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{k} y_j^{(i)} \log(\hat{y}_j^{(i)}).$$

Prediction: Class with maximum probability:

$$\text{Class} = \arg\max_j \hat{y}_j.$$

Gradient Descent: Weight and bias updates:

$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \eta\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}}, \quad \mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} - \eta\frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}}.$$

Gradients: For layer l:

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} = \frac{1}{m}\sum_{i=1}^{m} \delta^{(l)}\mathbf{a}^{(l-1)\top}, \quad \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}} = \frac{1}{m}\sum_{i=1}^{m} \delta^{(l)},$$

where $\delta^{(l)}$ is the error term.