Report about Neural Networks

by Kravchenko Sofya, group 5130203/20102

Perceptron

Perceptron is a type of neural network that performs binary classification that maps input features to an output decision, usually classifying data into one of two categories. Perceptron consists of inputs, weights, bias b , linear combination $z = w^T x + b$ and activation function – a step function $\phi(z)$ that outputs +1 or −1 (or 1 and 0 depending on convention).

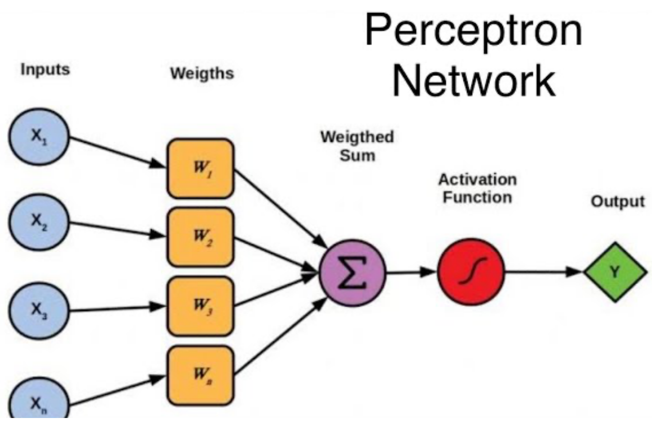

Image 1. Perceptron model architecture

The data can be represented as vectors:

- Inputs: $x = [x_1, x_2, \dots, x_n]^T$
- Weights: $w = [w_1, w_2, \dots, w_n]^T$
- Output: scalar $y \in \{-1,1\}$ $(or\ y \in \{0,1\})$

Mathematical formulation:

1) The linear combination z is computed as:

$$z = w^T x + b = \sum_{i=1}^{n} w_i x_i + b$$

2) The perceptron usually uses a signum function $\phi(z)$ as activation function:

$$\phi(z) = \begin{cases} +1 & if\ z \geq 0, \\ -1 & otherwise \end{cases}$$

3) Loss function:

$$L(w, b; x, y) = \begin{cases} 0 & if\ y * (w^T x + b) \geq 0, \\ -y * (w^T x + b) & otherwise \end{cases}$$

The prediction $\hat{y}$ is $\hat{y} = \phi(z) = \phi(w^T x + b)$. This maps the linear combination z to the discrete output set $\{-1,1\}$.

Gradient descent algorithm:

The Perceptron algorithm updates weights and biases based on misclassified examples only: $if\ y * (w^T x + b) \geq 0$ – no update; otherwise, update using:

$$w \leftarrow w + \eta y x, \qquad b \leftarrow b + \eta y$$

where $\eta > 0$ is the learning rate.

Formulas for gradient and updates:

1) Gradient of loss with respect to weights:
$$\frac{\partial L}{\partial w} = -yx$$
2) Gradient of loss with respect to bias:
$$\frac{\partial L}{\partial b} = -y$$
3) Weight update rule:
$$w^{(t+1)} = w^{(t)} - \eta \frac{\partial L}{\partial w}$$
4) Bias update rule:
$$b^{(t+1)} = b^{(t)} - \eta \frac{\partial L}{\partial b}$$

Logistic regression

Logistic regression is used for binary classification where we use sigmoid function, that takes input as independent variables and produces a probability value between 0 and 1.

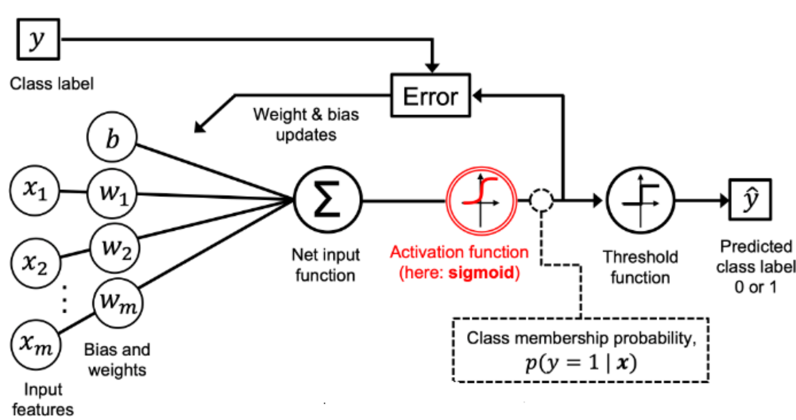

Image 2. Logistic regression model architecture

The data can be represented as vectors:

- Inputs: $x = [x_1, x_2, \dots, x_n]^T$
- Weights: $w = [w_1, w_2, \dots, w_n]^T$
- Output: scalar $y \in \{0,1\}$

  Mathematical formulation:

  1) The linear combination z is computed as:

  $$z = w^T x + b = \sum_{i=1}^{n} w_i x_i + b$$

  2) The sigmoid function maps z to a probability:

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

  The predicted probability $\hat{y}$ is $\hat{y} = \sigma(z)$.

  3) Loss function is Log-Loss/Cross-entropy loss

  $$L(w, b; x, y) = -\frac{1}{m}\sum_{i=1}^{m}[y_i log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)]$$

The predicted probability $\hat{y}$ is $\hat{y} = \sigma(w^T x + b)$. The predicted class $\hat{y}_{class}$ is:

$$\hat{y}_{class} = \begin{cases} 1 & if \;\; \hat{y} \geq 0.5, \\ 0 & otherwise \end{cases}$$

Logistic Regression minimizes the log-loss using gradient descent. The updates are derived from the gradient of the loss with respect to the weights w and bias b.

Formulas for gradient and updates:

1) Gradient with respect to weights:

$$\frac{\partial L}{\partial w} = \frac{1}{m} X^T (\hat{y} - y)$$

2) Gradient with respect to bias:

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

3) Weight update rule using learning rate $\eta > 0$:

$$w^{(t+1)} = w^{(t)} - \eta \frac{\partial L}{\partial w}$$

4) Bias update rule:

$$b^{(t+1)} = b^{(t)} - \eta \frac{\partial L}{\partial b}$$

Multilayer perceptron

An MLP consists of:

- Input Layer
- Hidden Layers: One or more layers, each containing multiple neurons.
- Output Layer: Provides the final prediction, typically for classification or regression tasks.

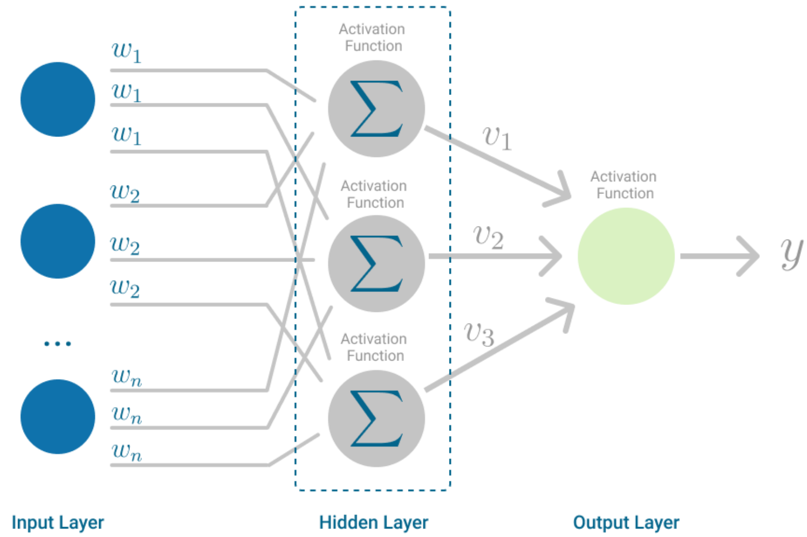

Image 3. Multilayer perceptron model architecture

The data can be represented as vectors:

- Inputs: $x \in \mathbb{R}^n$
- Weights for hidden layer: $W^{(l)} \in \mathbb{R}^{k*m}$ for layer l, where k is the number of neurons in the layer and mmm is the number of inputs
- Bias for hidden layer: $b^{(l)} \in \mathbb{R}^k$ (one bias per neuron in the layer)
- Output vector: $\hat{y}$

Mathematical formulation:

The forward pass in an MLP involves computing outputs at each layer:

1) Input layer: $a^{(0)} = x$
2) Hidden layers: for each layer l: $z^{(l)} = W^{(l)} a^{(l-1)} + b^{(l)}$

   Apply activation function ϕ: $a^{(l)} = \phi(z^{(l)})$
3) Output layer: compute the final output (logist or probabilities): $\hat{y} = \phi_{output}\left(z^{(L)}\right)$, where L is the total number of layers and ϕ_{output} depends on the task: classification – softmax function for multiclass or sigmoid for binary classification; regression – identity function (no non-linearity).

Loss function depends on the task:

- Classification (Cross-Entropy Loss):

$$L(W,b;X,Y) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{j=1}^{c} y_{ij}\log(\hat{y}_{ij})$$

- Regression (Mean Squared Error):

$$L(W,b;X,Y) = \frac{1}{m}\sum_{i=1}^{m} \|y_i - \hat{y}_i\|^2$$

The prediction involves:

1. Forward propagation through all layers as described above.
2. Output: for classification – $\hat{y} = argmax(\phi_{output}(z^{(L)}))$, for regression – $\hat{y} = z^{(L)}$

Gradient descent algorithm:

To optimize the weights and biases, MLP uses backpropagation with gradient descent:

1. Compute gradients of the loss with respect to weights and biases at each layer using the chain rule.
2. Update weights and biases using:

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

where $\eta > 0$ is the learning rate.

Formulas for gradient and updates:

1) Error at the output layer:

$$\delta^{(L)} = \nabla_{\hat{\mathbf{y}}} L \odot \phi'(\mathbf{z}^{(L)})$$

where $\odot$ represents element-wise multiplication.

2) Error propagation for hidden layers: for layer l (from output to input):

$$\delta^{(l)} = (\mathbf{W}^{(l+1)})^\top \delta^{(l+1)} \odot \phi'(\mathbf{z}^{(l)})$$

3) Gradients of weights and biases:

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

4) Weight and bias updates:

$$\mathbf{W}^{(l)} \leftarrow \mathbf{W}^{(l)} - \eta\frac{\partial L}{\partial \mathbf{W}^{(l)}}$$

$$\mathbf{b}^{(l)} \leftarrow \mathbf{b}^{(l)} - \eta\frac{\partial L}{\partial \mathbf{b}^{(l)}}$$