Report on Neural Networks

Mironov Vasiliy
5130203/20102

November 29, 2024

Contents

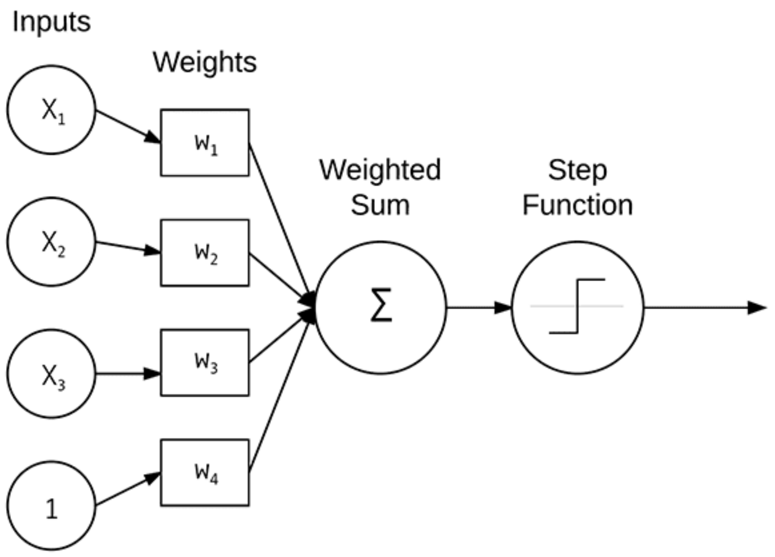
1 Perceptron 1

2 Logistic Regression 2

3 Multilayer Perceptron 3

1 Perceptron

Model architecture



Vector representation of data (inputs and outputs)

- Inputs: $x = [x_1, x_2, \ldots, x_n]$
- Output: $y \in \{0, 1\}$

Math formulation of linear combination, activation function and loss function

- Linear Combination:

$$z = wx + b, \text{where w is the weight vector and b is the bias.}$$

- Activation Function:

$$\hat{y} = f(z) = \begin{cases} 1, z \geq 0 \\ 0, \text{otherwise} \end{cases}$$

- Loss Function:

$$L(y, \hat{y}) = \begin{cases} 0, y = \hat{y} \\ 1, y \neq \hat{y} \end{cases}$$

Math formulation of how neural nets calculate the predictions (y_hat)

Predictions:

$$\hat{y} = f(wx + b)$$

Explanation of gradient descendent algorithm

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

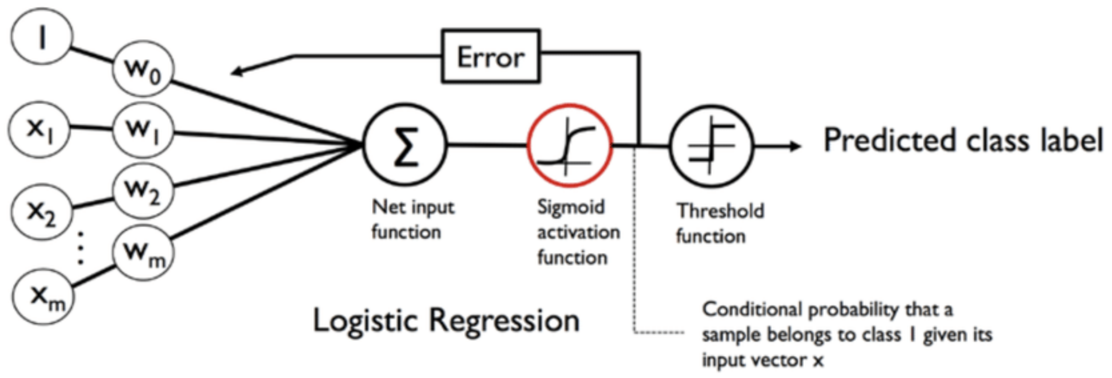
Formulas of gradients and weights/biases updates

Formulas:

$$w_i \hookleftarrow w_i + \eta(y - \hat{y})x_i, b \hookleftarrow b + \eta(y - \hat{y}), \text{where } \eta \text{ is the learning rate.}$$

2 Logistic Regression

Model architecture


1
x1
x2
xm
w0
w1
w2
wm
Error
Net input function
Sigmoid activation function
Threshold function
Predicted class label
Logistic Regression
Conditional probability that a sample belongs to class 1 given its input vector x

Vector representation of data (inputs and outputs)

- Inputs: $x = [x_1, x_2, \ldots, x_n]$
- Output: $y \in \{0, 1\}$

Math formulation of linear combination, activation function and loss function

- Linear Combination:

$$z = wx + b, \text{where w is the weight vector and b is the bias.}$$

- Activation Function:

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

- Loss Function (binary cross entropy):

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

Math formulation of how neural nets calculate the predictions (y_hat)

Predictions:

$$\hat{y} = \sigma(wx + b)$$

Explanation of gradient descendent algorithm

Explanation: Similar to the Perceptron, gradient descent is used to minimize the loss function by updating weights and biases.

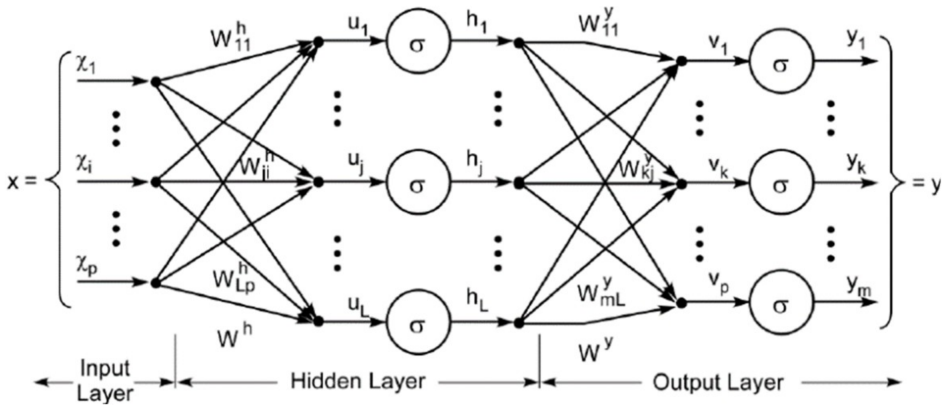
Formulas of gradients and weights/biases updates

Formulas:

$$w_i \hookleftarrow w_i - \eta\frac{\delta L}{\delta w_i}, b \hookleftarrow \eta\frac{\delta L}{\delta b}, \text{where } \frac{\delta L}{\delta w_i} = (\hat{y} - y)x_i, \frac{\delta L}{\delta b} = \hat{y} - y$$

3 Multilayer Perceptron

Model architecture



Vector representation of data (inputs and outputs)

- Inputs: $x = [x_1, x_2, \ldots, x_n]$
- Output: $y = [y_1, y_2, \ldots, y_n]$

Math formulation of linear combination, activation function and loss function

- Linear Combination: For the l layer:

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

- Activation Function: Choices:

  ReLU (Rectified Linear Unit): $a^l = max(0, z^l$

  Sigmoid: $a^(l) = \sigma(z^{(l)}$

  Softmax (for output layer in multi-class classification):

$$a_i^{(L)} = \frac{e^{z_i^{(L)}}}{\sum_{j=1}^{m} e^{z_j^{(L)}}}$$

- Loss Function (multi-class classification):

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

Math formulation of how neural nets calculate the predictions (y_hat)

Predictions: The output of the MLP can be expressed as:

$$\hat{y} = a^{(L)} = f(z^{(L)}) = f(W^{(L)}a^{(L-1)} + b^{(L)}), \text{where L is the number of layers.}$$

,

Explanation of gradient descendent algorithm

Explanation: The MLP uses backpropagation to compute gradients of the loss function with respect to weights and biases, allowing for efficient updates during training.

Formulas of gradients and weights/biases updates

Formulas: For each weight $w^{(l)}$ and bias $b^{(l)}$:

$$w^{(l)} \hookleftarrow w^{(l)} - \eta \frac{\delta L}{\delta w^{(l)}}, b^{(l)} \hookleftarrow b^{(l)} - \eta \frac{\delta L}{\delta b^{(l)}}$$