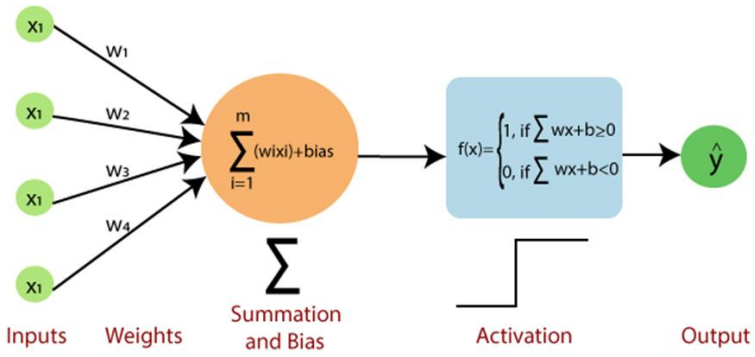
Abutalipov Alexandr 5130203/20101

Report about Neural Networks

1. Perceptron

1. Draw of model architecture



2. Vector representation of data (inputs and outputs)

Input vector: $x = [x_1, x_1 \dots x_n]^T$

Output: $y \in \{-1, 1\}$

Vector of weights: $w = [w_1, w_1 \dots w_n]^T$

3. Math formulation of linear combination, activation function and loss function

- Linear combination:

$$z = \sum_{i=1}^{n} w_i x_i + b = w^T x + b,$$ where b – is offset (bias)

- Activation function:

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

- Loss function:

$$L(x, y, w) = \max(0, -yw^T x + b)$$

4. Math formulation of how neural nets calculate the predictions

$$\hat{y} = \phi(z) = \phi(w^T x + b)$$

5. Explanation of gradient descendent algorithm

The goal of the gradient descent algorithm is to minimize the loss function. At each step, the algorithm updates the weights based on the derivative of the loss function

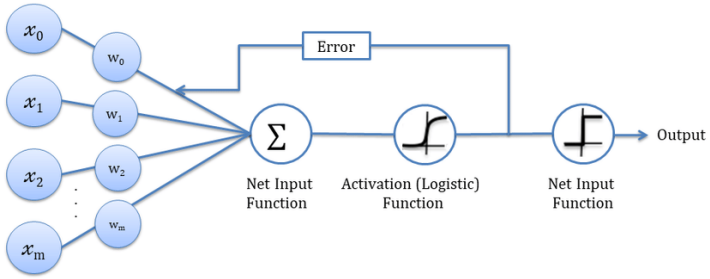
6. Formulas of gradients and weights/biases updates

$$w_i \leftarrow w_i + \eta(y - \hat{y})x_i$$
$$b \leftarrow b + \eta(y - \hat{y}),$$

where η – learning rate, y – real result, $\hat{y}$ – predicted result

2. Logistic Regression

1. Draw of model architecture



2. Vector representation of data (inputs and outputs)

Input vector: $x = [x_1, x_1 \dots x_n]^T$

Output: $y \in \{0, 1\}$

Vector of weights: $w = [w_1, w_1 \dots w_n]^T$

3. Math formulation of linear combination, activation function and loss function

- Linear combination:

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

- Activation function: Sigmoid function

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

- Loss function

$$L(y, \hat{y}) = -\frac{1}{n} \sum_{i=1}^{n} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

4. Math formulation of how neural nets calculate the predictions

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

5. Explanation of gradient descendent algorithm

Gradient descent in logistic regression is used to minimize the loss function, updating the weights at each step depending on the gradient of the loss function.

6. Formulas of gradients and weights/biases updates

Updates of weights:

$$\frac{\partial L}{\partial w_i} = (y - \hat{y}) x_i$$

$$w_i \leftarrow w_i + \eta \frac{\partial L}{\partial w_i}$$

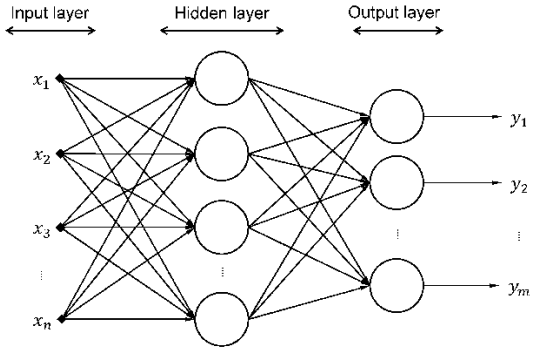
Updates of bias:

$$\frac{\partial L}{\partial w_i} = (y - \hat{y})$$

$$b \leftarrow b + \eta \frac{\partial L}{\partial w_i}$$

3. Multilayer Perceptron

1. Draw of model architecture



2. Vector representation of data (inputs and outputs)

Input vector: $x = [x_1, x_1 \dots x_n]^T$

Output: $y = [y_1, y_1 \dots y_m]^T$

$W^{(l)}$is the weight matrix, and $b^{(l)}$ is the bias vector.

3. Math formulation of linear combination, activation function and loss function

- Linear combination for layer l, the linear combination of inputs x and weights $W^{(l)}$ is:

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

  where $a^{(l-1)}$ is the activation from the previous layer
- Activation function:

  ReLU: $ReLU(z) = \max(0, z)$

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

  Softmax: $Softmax(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{c} e^{z_j}}$
- Loss function

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

4. Math formulation of how neural nets calculate the predictions

$$\hat{y} = \phi(z),$$

where $\phi(z)$ is the activation function at the output of the last layer

5. Explanation of gradient descendent algorithm

Gradient descent in multilayer perceptron calculates the gradients of the loss function starting at the output layer and moving back to the input layer.

6. Formulas of gradients and weights/biases updates

$$\frac{\partial L}{\partial W^{(l)}} = \delta^{(l)} * \left(a^{(l-1)}\right)^T$$

$$w_i \leftarrow w_i + \eta \frac{\partial L}{\partial W^{(l)}}$$

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