Peter the Great Saint Petersburg Polytechnic University
Institute of Computer Science and Cybersecurity
Higher Education of Intelligent Systems and Supercomputers Technology

Report about Neural Networks

Completed by:
Mikhail Bocharov
group 5130203/20001

Fall 2024

1 Perceptron

1.1 Overview

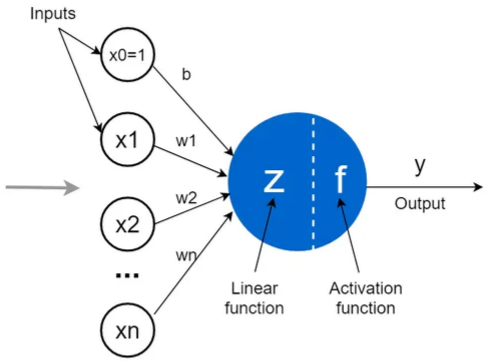

Figure 1 Perceptron schematics

- $x_1, x_2, ...x_n$ are inputs;
- $w_1, w_2, ...w_n$ are weights;
- Z is a linear combination of $x_i s$: $Z = w_0 + w_1 * x_1 + w_2 * x_2 + ...w_n * x_n$;
- ϕ is an activation function;
- w_0 is a bias;
- $\hat{y}$ is output in binary form.

1.2 Linear function

We can represent our x_is and w_is as vectors X and W respectively(1xN matrixes). Then linear function can be represented as $Z = X^T W + w_0$

1.3 Activation function and predictions

$$\phi = \begin{cases} 1, & \text{z>=0} \\ -1, & \text{z<0} \end{cases}$$

Prediction $(\hat{y}) = \phi(z)$, where z - linear combination.

1.4 Loss function

Using perceptron we update weights based on a number of wrong predictions. This model has the same input and output, but uses a different activation function and loss calculation.

$$L = \sum_{i=1}^{n} \delta(\hat{y} \neq y)$$

2 Logistic Regression

2.1 Overview

The schematic representation of a logistic regression is depicted in figure 2.

- $x_1, x_2, ...x_n$ are inputs;
- $w_1, w_2, ...w_n$ are weights;
- Z($\sum$) is a linear combination of $x_i s$: $Z = w_0 + w_1 * x_1 + w_2 * x_2 + ...w_n * x_n$;

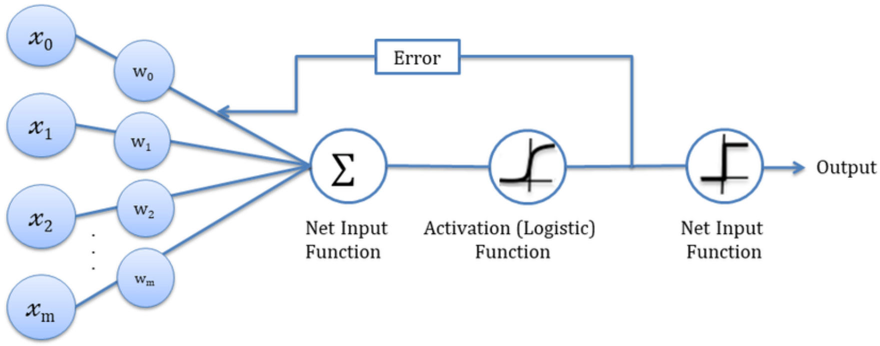

Figure 2 Logisctic regression

- ϕ is an activation function;
- w_0 is a bias;
- $\hat{y}$ is output in binary form.

2.2 Activation function

For activation function we use sigmoid:

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

Graphic of sigmoid function is depicted in figure 3.

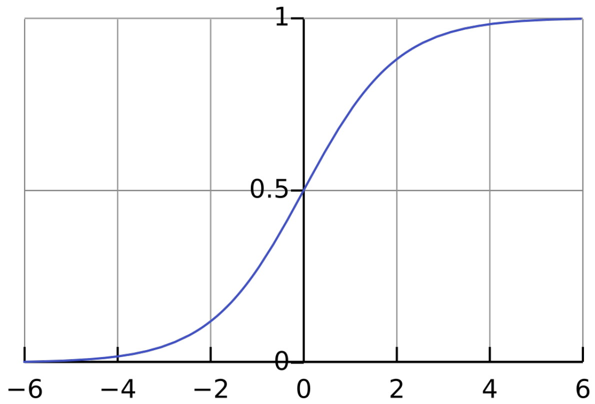

Figure 3 Sigmoid function

2.3 Loss function

For loss function we use cross-entropy loss:

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

Here $\hat{y}$ are our predictions $= \phi(Z)$

2.4 Updating weights

$$\frac{\delta L}{\delta w_j} = \frac{1}{m}\sum_{i=1}^{m}(\hat{y} - y_i)x_j^{(i)}$$

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

Updating weights:

$$w_j := w_j - \eta \frac{\delta L}{\delta w_j}$$

Here η represents learning rate.
For calculating we use so-called rule of chain:

$$\frac{\delta L}{\delta w_j} = \frac{\delta L}{\delta \hat{y}} \frac{\delta \hat{y}}{\delta z} \frac{\delta z}{\delta w_j}$$

3 Multilayer perceptron

3.1 Overview

Multilayer perceptron is a combination of regression model and a perceptron on a bigger scale. Instead of one single neuron, we can have multiple layers of many neurons. The schematics is shown in figure 3.

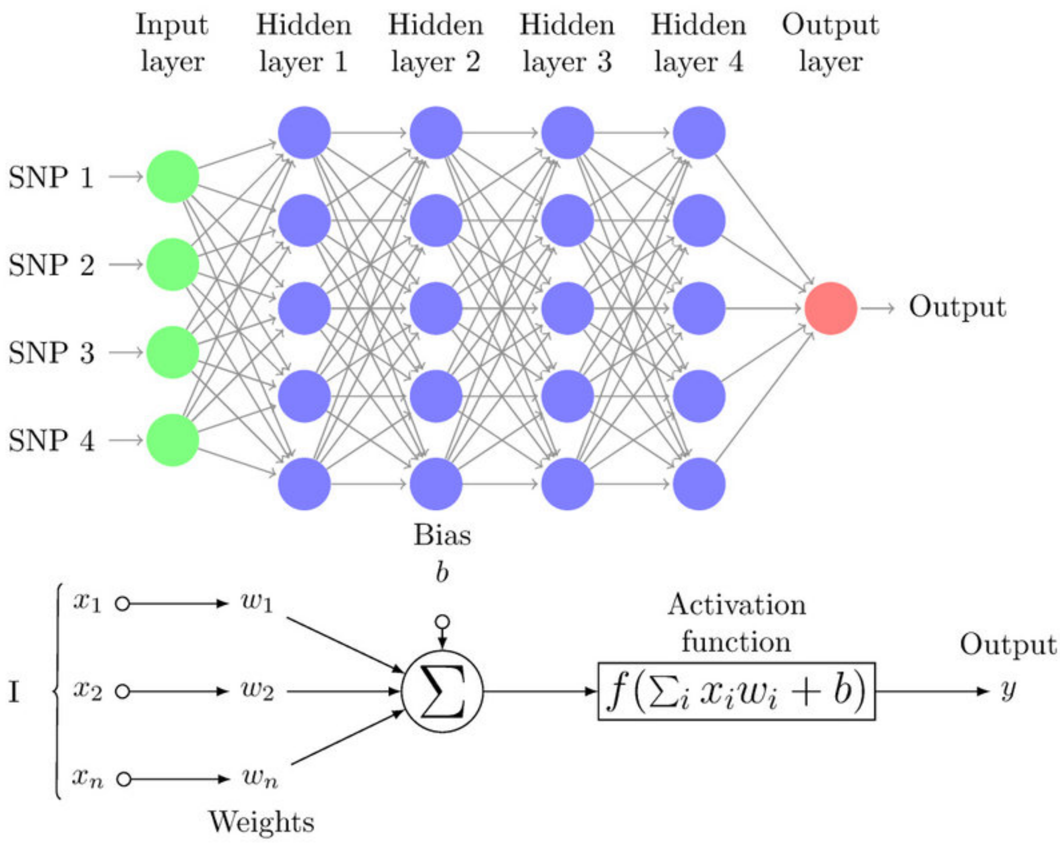

Figure 3 Multilayer perceptron

- $x_1, x_2, ...x_a$ are input values;
- $h_1, h_2, ...h_b$ are hidden layers;
- ϕ is an activation function;
- $\hat{y}_1, \hat{y}_c$ are predicted values.

Since we have a lot of neurons and many layers, we calculate linear combination for every single neuron. All the layers between the input and output layer are called "hidden" layers.

3.2 Linear combination

In a regular perceptron linear combination can be represented as a multiplication of a column of x_is and a vector of weights w_is :

$$Z = X^T W + w_0 = w_1 * x_1 + w_2 * x_2 + ...w_n * x_n + w_0$$

Since now we have multiple layers and neurons, linear combination $\sum$ for layer h can be stored as vector (column):

$$z^{[h]} = W^{[h]} \Phi^{[h-1]} + B^{[h]}$$

where:

$$W^h = \begin{pmatrix} w_{11} & w_{12} & \dots & w_{1a} \\ w_{21} & w_{22} & \dots & w_{2a} \\ \vdots & \vdots & \ddots & \vdots \\ w_{b1} & w_{b2} & \dots & w_{ba} \end{pmatrix}$$

Here we have a weight matrix for one layer from a MLP with b layers and a neurons in each layer;

$$\Phi^{h-1} = \begin{pmatrix} \phi_1 \\ \phi_2 \\ \vdots \\ \phi_b \end{pmatrix}$$

Here ϕ_i represents a vector of activation function's results for the previous layer.
B represents a vector of biases for neurons.

3.3 Activation function

For activation function we use sigmoid:

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

3.4 Prediction

$\Phi^h = \phi(z^h)$

3.5 Loss function

We can use cross-entropy loss from perceptron:

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

3.6 Weight updates

$$W^{[h]} := W^{[h]} - \eta \frac{\delta L}{\delta W^{[h]}}$$

$$B^{[h]} := B^{[h]} - \eta \frac{\delta L}{\delta B^{[h]}}$$