# MAP586 project

Deep Neural Network

2022-01-27

You see a lot of people around you who are interested in deep neural networks and you think that it might be interesting to start thinking about creating a software that is as flexible as possible and allows novice users to test this kind of methods.

We recall here the key elements found in deep neural networks. We will not go into the mathematical details as this is not the purpose of this project.

A deep neural network is composed of an input, an output and several hidden layers.

A neuron is illustrated by the following figure

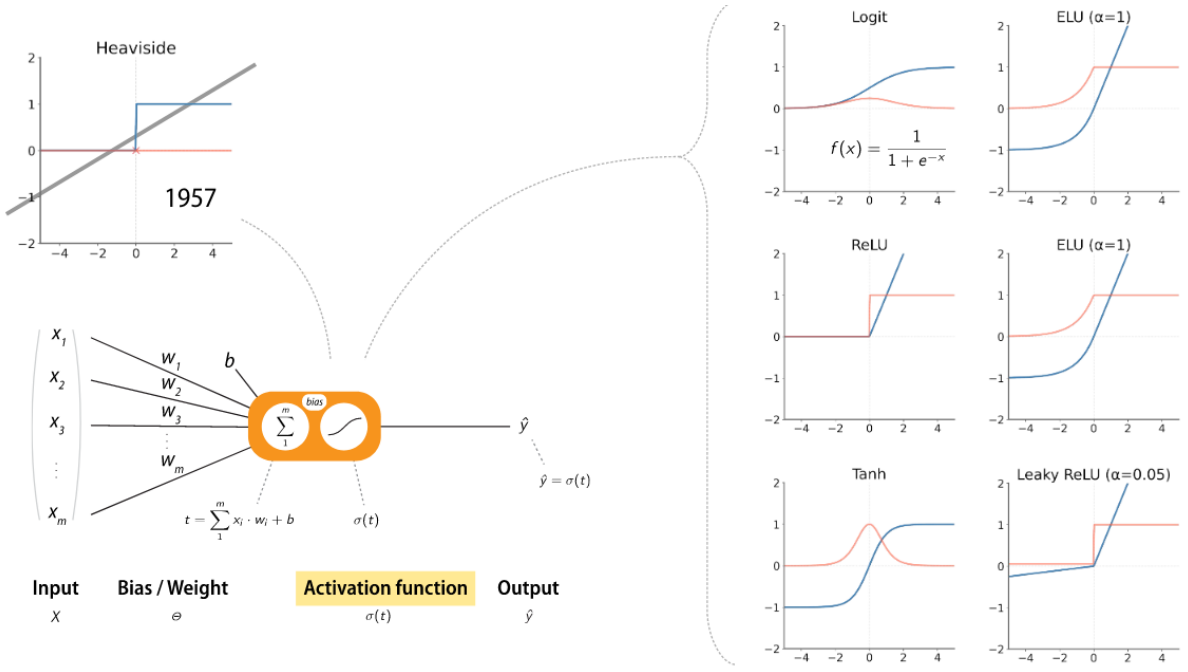


Figure 1: How a neuron works

This figure comes from a CNRS course called fiddle (https://gricad-gitlab.univ-grenoble-alpes.fr/talks/fidle).

We can observe that a neuron is made of weights, a bias and an activation function. The activation function can be a sigmoid, reLU, tanh, ...

A deep neural network is composed of several hidden layers with several neurons as illustrated in the following figure

This figure also comes from the CNRS course fiddle.

In the following, we will use these notations:

- $w^l_{j,i}$ is the weight of the layer $l$ for the neuron $j$ and the input entry $i$.
- $z^l_j$ is the aggregation: $\sum_i x^l_i w^l_{j,i} + b^l_j$ where $x_i$ is the input.

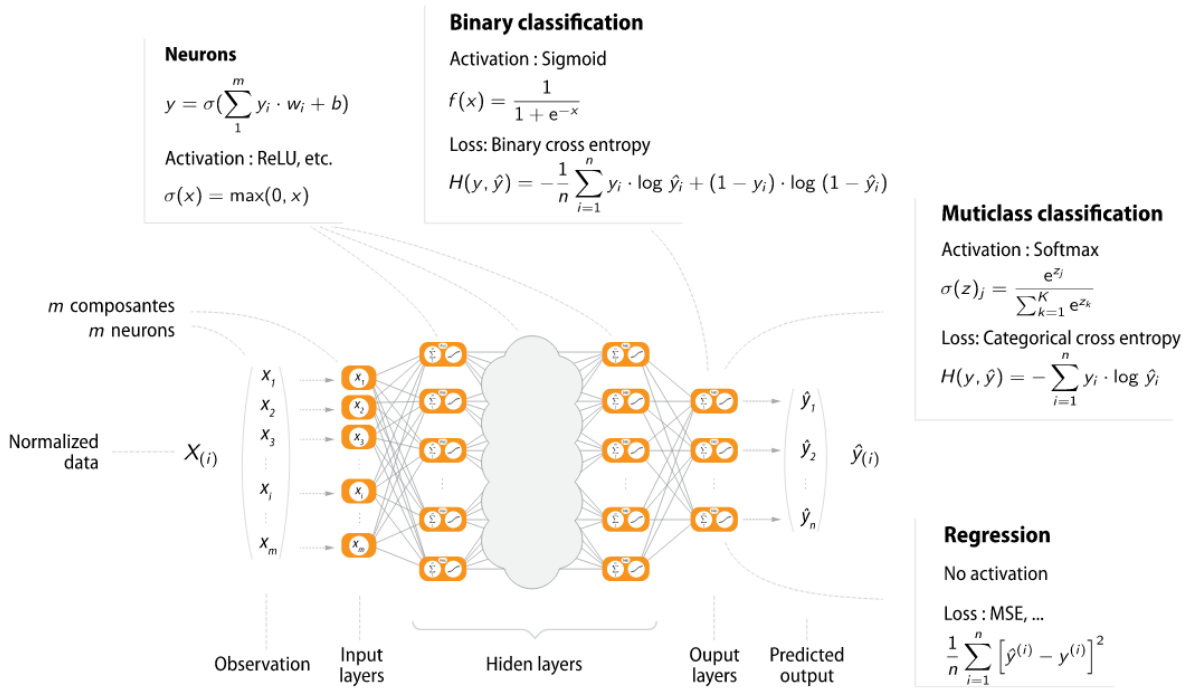


Figure 2: A complete overview of a deep neural network.

- $\sigma$ is the activation function.
- $a_j^l$ is the output of the neuron $j$ for the layer $l$.
- $L$ is the index of the last layer.
- $C(a^L, y)$ is the cost function where $a^L$ is the predict value and $y$ is the expected result.

The algorithm has three steps:

- The forward propagation: for a given input, cross all the layers until the output and fill $z^l$ and $a^l$.
- Change the weights and biases to minimize the cost function using a descent gradient. This is called backward propagation.
- iterate until reaching the maximum number of iterations or a given tolerance.

The gradient descent can be written as

$$w_{j,i}^l = w_{j,i}^l - \mu \frac{\partial C}{\partial w_{j,i}^l},$$

where $\mu$ is the learning rate.

The equations of the backward propagation are

- $\delta_j^L = \frac{\partial C}{\partial a_j^L} \sigma'(z_j^L)$
- $\delta_j^l = \sum_i w_{i,j}^{l+1} \delta_i^{l+1} \sigma'(z_j^l)$
- $\frac{\partial C}{\partial b_j^l} = \delta_j^l$
- $\frac{\partial C}{\partial w_{j,i}^l} = a_i^{l-1} \delta_j^l$

We need a set of datas: datas for training the neural network and datas for testing the final weights and biases.

You can find more information on the following links

- https://gricad-gitlab.univ-grenoble-alpes.fr
- http://neuralnetworksanddeeplearning.com/index.html

## Work to be done

For this project, you have to

- Propose a flexible and user friendly data structure able to manage several hidden layers and describe for each of them the activation function. A list of cost functions should also be available.
- Propose a way to load data from the website openML. You could use for example this csv parser
- Experiment your library on at least the two well-known datasets iris and MNIST

The source code must be well documented and the repository of the project must contain several examples and tests. In the report, you must justify your choices and clearly explain the data structure.