Our own Neural Network

Ens'IA

Ensimag 2022-2023

14 novembre 2022

Presentation de la séance

The presentation :

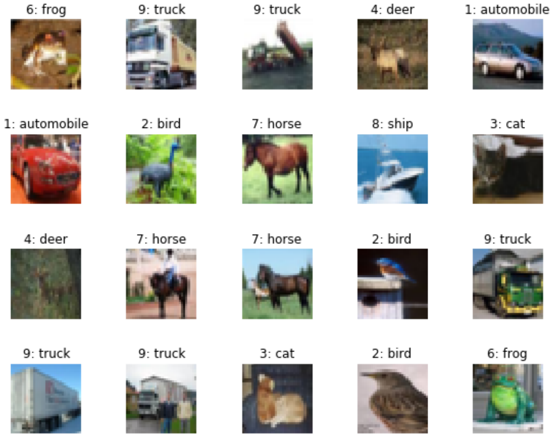
- Reminders from last session
- Generalizing to layers of neurons
- Finding the formulas for each propagation
- Session presentation

Outline

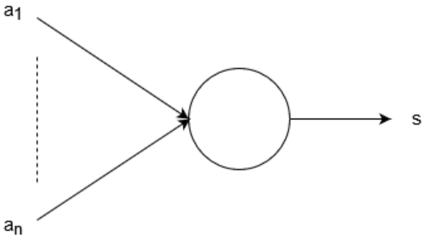
Outline

Motivation

We want to create a program capable of, for example, classifying images...



As a reminder, here's how a sigmoid neuron works



$a_1, ..., a_n \in [0, 1]$
$s = \sigma(\sum_{i=0}^{n} a_i * w_i + b)$ with $\sigma(x) = \frac{1}{1+e^{-x}}$

Training

In order to train our Network :
For each parameter p in the Network :

$$p' = p - \eta \frac{\partial L}{\partial p}$$

Training

In order to train our Network :
For each parameter p in the Network :

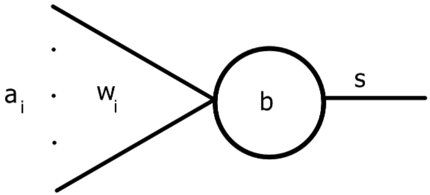
$$p' = p - \eta \frac{\partial L}{\partial p}$$

Objective : Computing $\frac{\partial L}{\partial p}$ for any p in our network.

Outline

Neurons to Layers

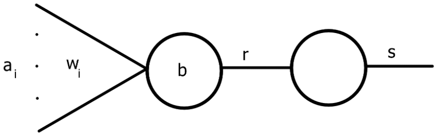
First, we'll split the neuron's forward calculation...



Here : $s = \sigma(\sum_{i=0}^{n} w_i * a_i + b)$.

Neurons to Layers

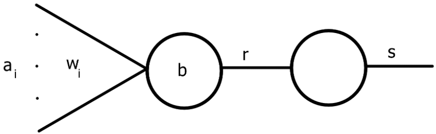
...in two consecutive neurons!



Here : $r = \sum_{i=0}^{n} w_i * a_i + b$ and $s = \sigma(r)$.

Neurons to Layers

...in two consecutive neurons !

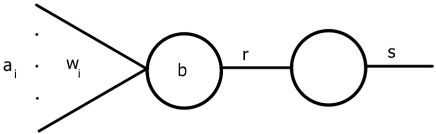


Here : $r = \sum_{i=0}^{n} w_i * a_i + b$ and $s = \sigma(r)$.
By writing $A = (a_i)_{i \in [1,n]}$ and $W = (w_i)_{i \in [1,n]}$:

$$r = W^T.A + b$$

Neurons to Layers

...in two consecutive neurons !



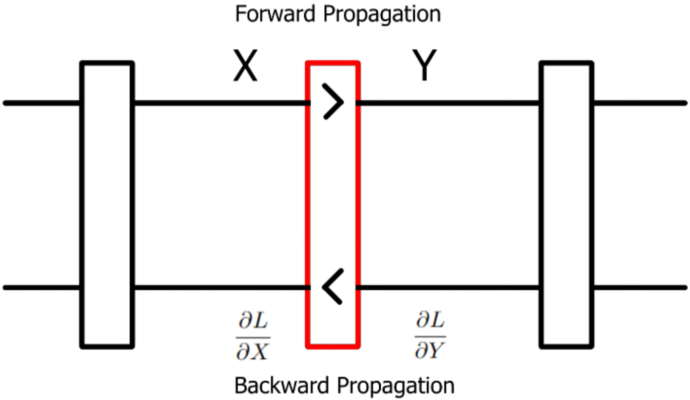
Here : $r = \sum_{i=0}^{n} w_i * a_i + b$ and $s = \sigma(r)$.
By writing $A = (a_i)_{i \in [1,n]}$ and $W = (w_i)_{i \in [1,n]}$:

$$r = W^T.A + b$$

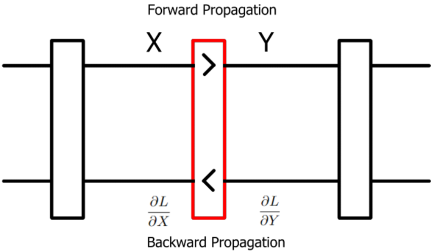
We get different types of neurons... and thus different types of neuron layers !

One layer

We'll consider a layer in a network.



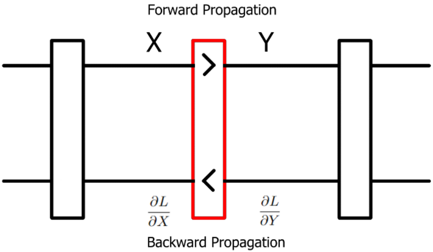
One layer



It receives X from the previous layer, and computes Y for the next layer. The layer's backpropagation will compute :

$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y}\frac{\partial Y}{\partial X}$$

One layer



It receives X from the previous layer, and computes Y for the next layer. The layer's backpropagation will compute :

$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y}\frac{\partial Y}{\partial X}$$

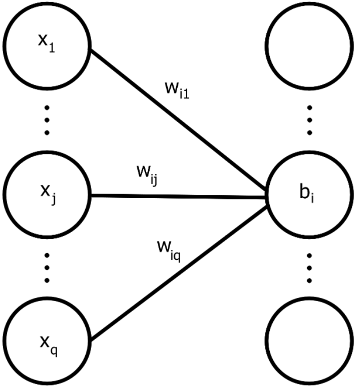
If the layer has a parameter p, then the backpropagation will also compute $\frac{\partial L}{\partial p}$ and update p.

Outline

The Different Layers

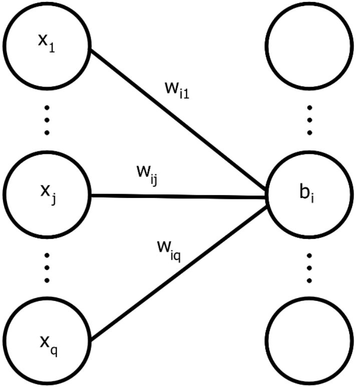
We'll look at how to compute each of those values with a Dense Layer, and a Sigmoid Layer.

Dense Layer



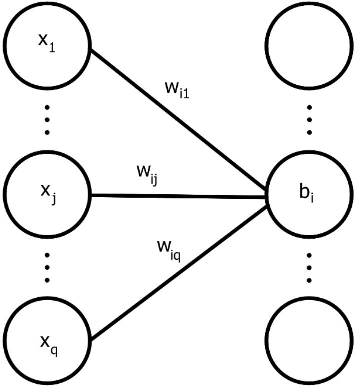
A dense layer is a layer made of perceptrons :
Each input is connected to each neuron with a weight.
We note w_{ij} the weight connecting output i with input j.

Dense Layer - Forward



the input is $X = (x_j)_{j \in [1,q]}$ and we want to compute $Y = (y_i)_{i \in [1,p]}$.
The output y_i of the i-th neuron is $y_i = \sum_{j=0}^{q} w_{ij} x_j + b_i$.

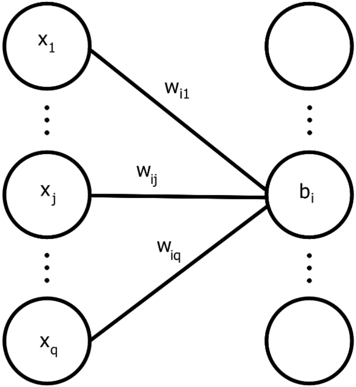
Dense Layer - Forward



By writing $W_i = (w_{ij})_{j \in [1,q]}$, we get :

$$y_i = W_i^T.X + b_i$$

Dense Layer - Forward



Then writing $W = (w_{ij})_{i,j \in [1,p] \times [1,q]}$,
and $B = (b_i)_{i \in [1,q]}$, we get :

$$Y = W.X + B$$

Dense Layer - Backward

For the backward propagation, we need to compute :

$$\forall i, j \in [1, p] \times [1, q]$$

$$\frac{\partial L}{\partial w_{ij}},\ \frac{\partial L}{\partial b_i} \text{ and } \frac{\partial L}{\partial X}$$

Dense Layer - Backward

By noting :

$$\frac{\partial L}{\partial W} = \left(\frac{\partial L}{\partial w_{ij}}\right)_{i,j}$$

$$\frac{\partial L}{\partial B} = \left(\frac{\partial L}{\partial b_i}\right)_{i}$$

We can derive similar equations using the chain rule...

Dense Layer - Backward

Everything can be written easily thanks to matrices!

$$\frac{\partial L}{\partial W} = \frac{\partial L}{\partial Y}.X^T$$

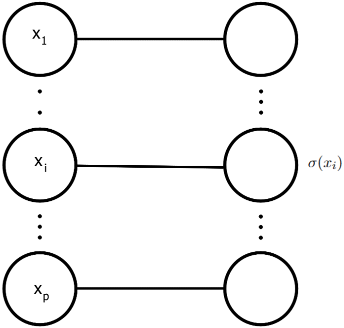
$$\frac{\partial L}{\partial B} = \frac{\partial L}{\partial Y}$$

$$\frac{\partial L}{\partial X} = W^T.\frac{\partial L}{\partial Y}$$

For the math nerds, an explanation of these formulas are detailled in the notebook ;)

Sigmoid Layer - Forward

And now for the Sigmoid layer :



The output of the layer is $\forall i \in [1, p], y_i = \sigma(x_i)$, so :

$$Y = \sigma(X)$$

Sigmoid Layer - Backward

The sigmoid layer has no parameters.
For the backward propagation, we need to compute : $\frac{\partial L}{\partial X}$

Sigmoid Layer - Backward

The sigmoid layer has no parameters.
For the backward propagation, we need to compute : $\frac{\partial L}{\partial X}$
Using the chain rule, we get :

$$\frac{\partial L}{\partial X} = \frac{\partial L}{\partial Y}\sigma'(X)$$

Outline

In today's session, we'll code :

- A Dense layer
- A Sigmoid layer
- A Neural Network !

Discord

Join us on Discord !

Useful to ask questions, contact us or to pass on information ! → https ://discord.gg/UgTRbRFqNv

And add us on Instagram ! @ensia_ensimag