Neural Networks from scratch 3

Workshop Tokyo Python Society Club

Yann LE GUILLY

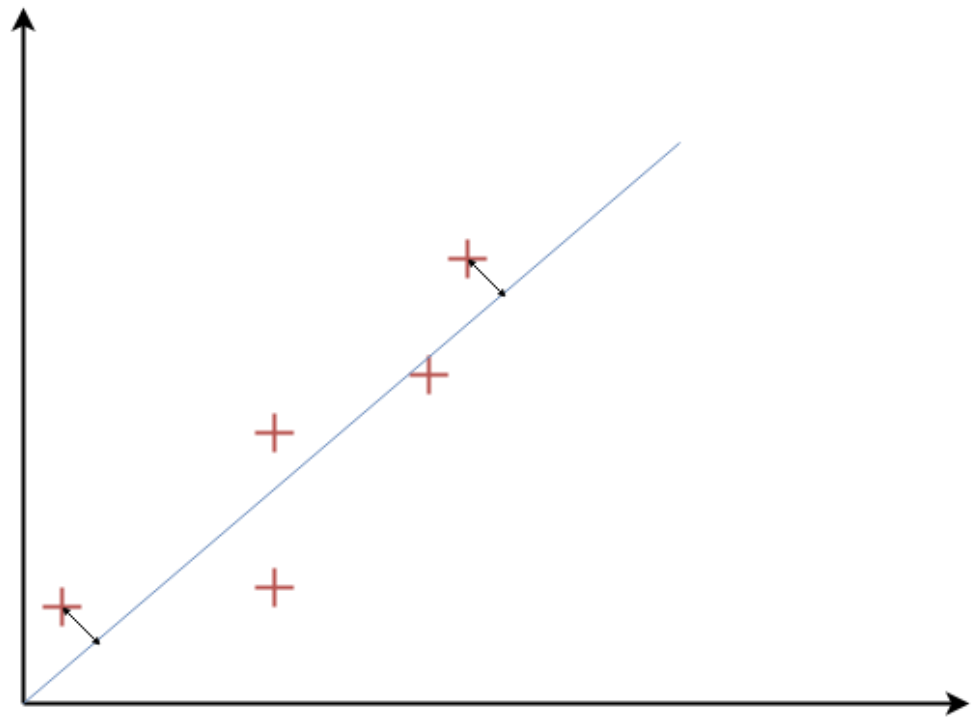
How to train your Neural Network

Accuracy

We want to approximate the function that describes Y.

- Y: is our data
- $\widehat{Y}$: is what we predicted

Intuitively, if $Y - \widehat{Y} = 0$ our accuracy is perfect!



Evaluating the accuracy

We can create a function that describe the accuracy. We call this the **loss function**. In our example, we will use the very common cross-entropy:

$$L(y, \hat{y}) = -\frac{1}{N} \sum_{n \in N} \sum_{i \in C} y_{n,i} \log \widehat{y}_{n,i}$$

The concept is the same as the MSE that we saw in the notebook 1_fct_approx.ipynb

Updating the model

If you remember, $\widehat{y}$ depends on our model because $\widehat{y}$ is predicted FROM our model.

So intuitively, we can think:

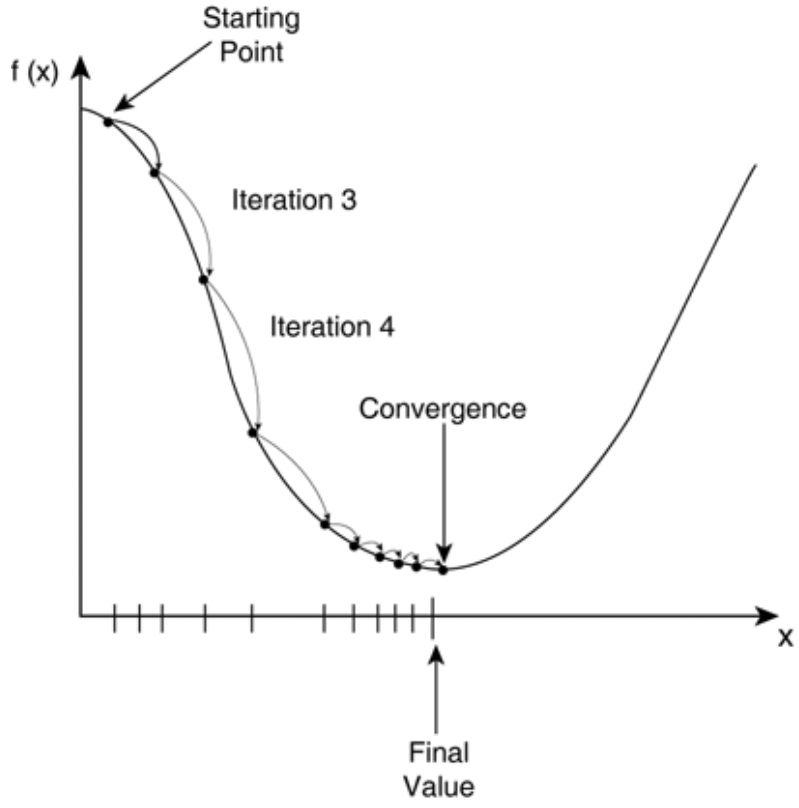
"**Probably there is a way to find a model that making** $L(y, \hat{y}) = 0$".

Optimization 1/4

You just invented Optimization! We want to find $\widehat{y}$ parameters that can decrease $L(y, \hat{y}) = 0$ as much as possible (=it will increase the accuracy of our model = it will approximate the function as much as possible)

Optimization 2/4

For this, we have many tools but we will use: gradient descent.



The gradient descent is looking for the minimum of a given function.

Optimization 3/4

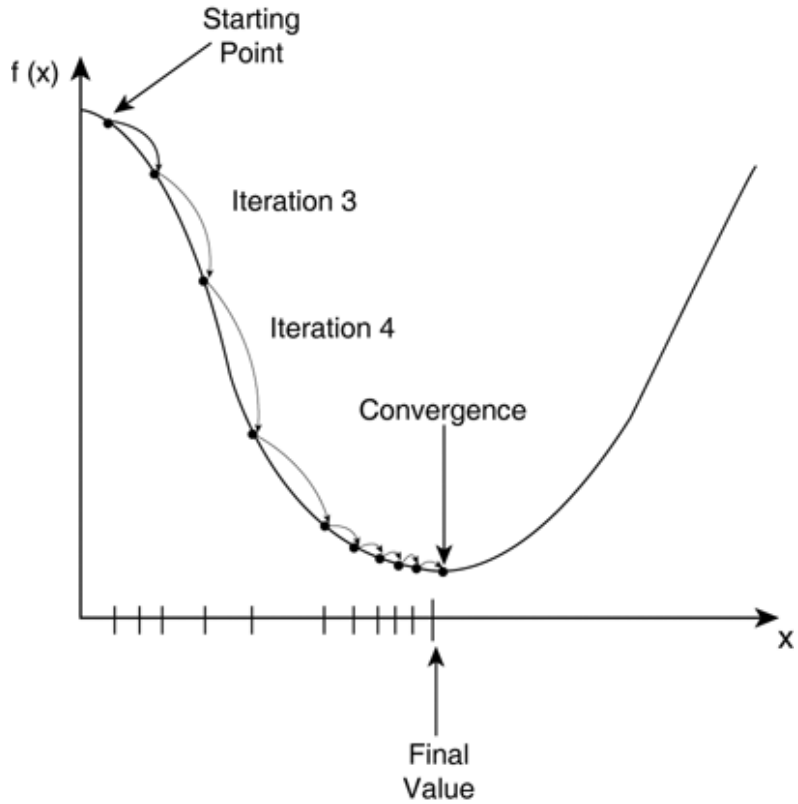
Gradient? Yes! The gradient is a vector of derivation! But what is derivation?! Let's say you in the bus.

- If the driver is accelerating, in 5 seconds, probably the bus speed will increase, right?
- But if he is using breaks, he is decelerate, probably the speed will decrease.

This is the intuition behind derivation. The acceleration is the derivation of the speed. But in the real life, we don't have only $f(x)$ but more often $f(x, y, z)$ or $f(x_1, \ldots, x_{4000})$. So we want to study the behaviour or f from different variable, so we need "a vector of derivations": the gradient.

Optimization 4/4

So with the gradient, we can say whether we are in a slope or not. If the gradient is 0 (no acceleration at this point), probably we reached what we call a "minimum". Our accuracy should be the best that we can have.



Let’s sum up!

1. We have our neural network and it’s working
2. But our model is not accurate
3. So we defined a way to estimate the accuracy: the loss function
4. And now we know a way to improve our model: we have to optimize our loss function

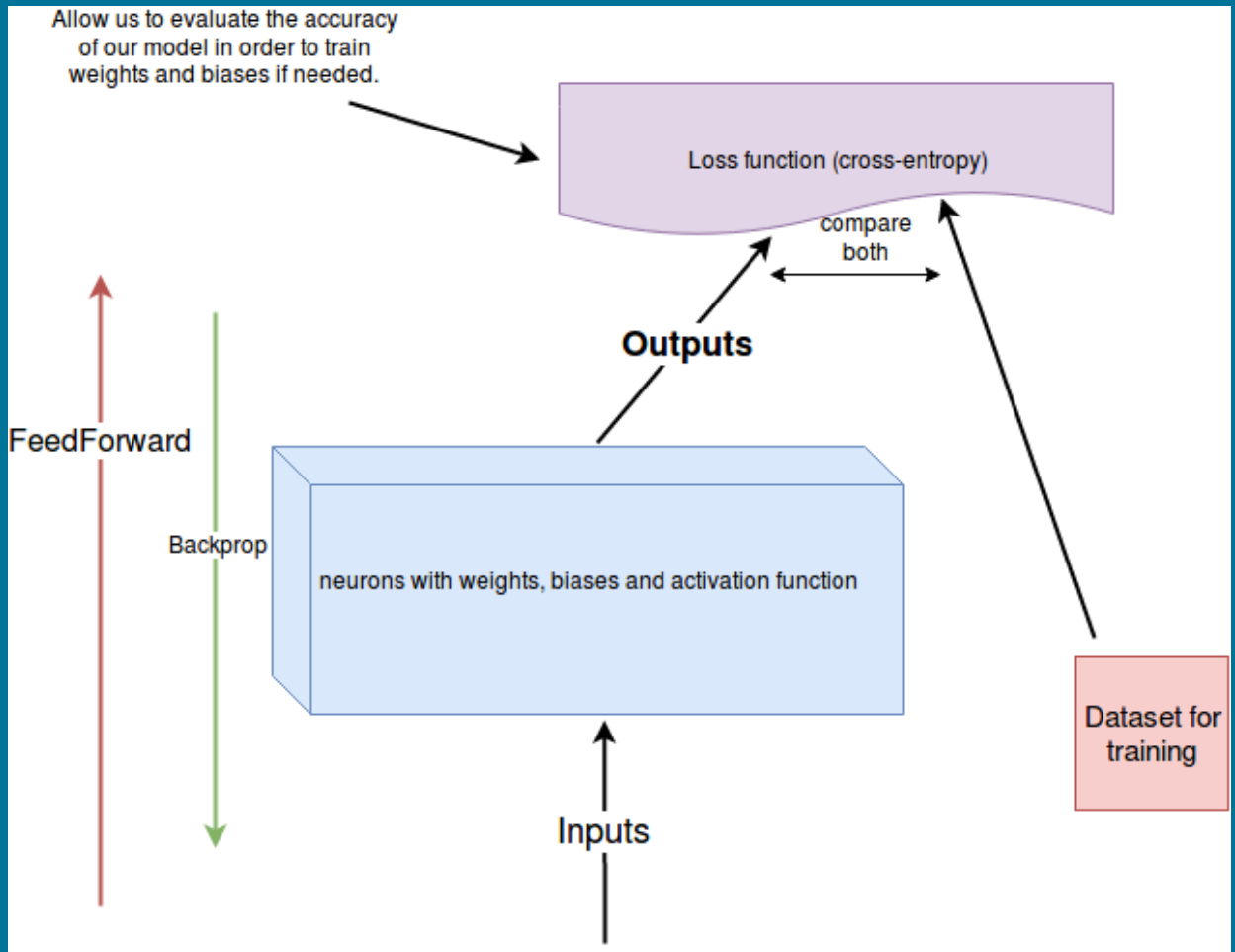
How to train your Neural Network

1. What is the problem?
    - 1.1 Accuracy
    - 1.2 Evaluating the accuracy
    - 1.3 Updating the model
    - 1.4 Optimization
    - 1.5 Let’s sum up!

2. Backpropagation
    - 2.1 Opposite of Feedforward
    - 2.2 Chain rule
    - 2.3 Let’s finish our NN!

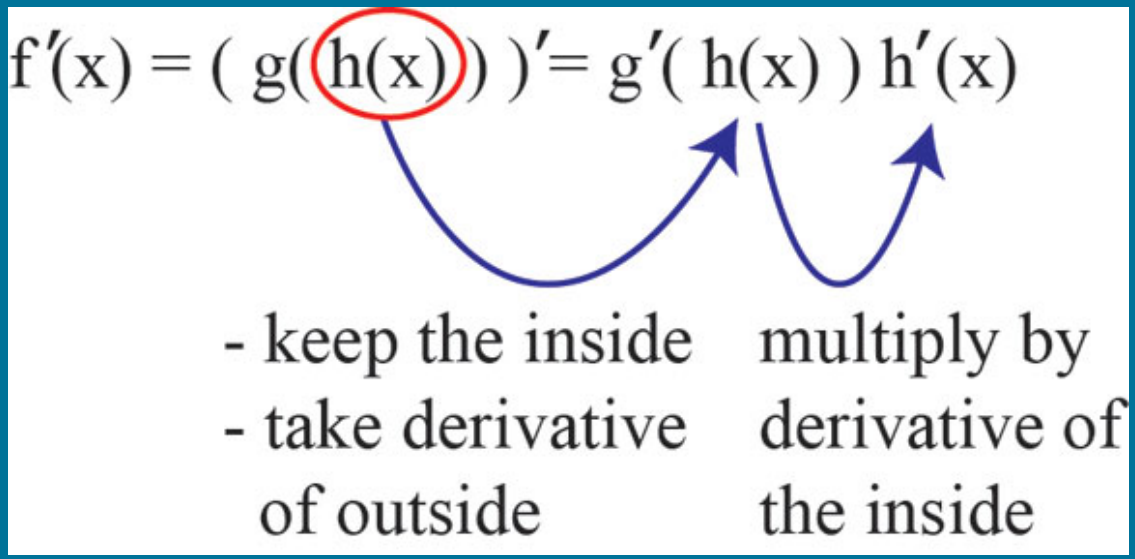
Opposite of Feedforward

FeedForward was inputs->outputs. Loss function is estimating the model accuracy then outputs->inputs by updating all weights and biases in order to improve the model and converge to a minimum.



Chain rule

In order to compute the gradient and update weights and biases, we are using chain rule. It's a convenient way to compute the gradient by using the graph structure. Instead of considering the whole function with every neurons, we are computing step by step.



Let’s finish our NN!

Now we can add last pieces of our NN. Please switch on the notebook 3_backprop_and_finish.ipynb