



Sefik Ilkin Serengil

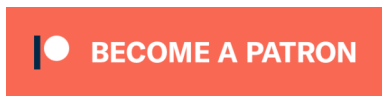
Code wins arguments

Menu ▾

A Step by Step Perceptron Example

January 4, 2020 / Machine Learning





Haven't you subscribe my channel yet?



Sefik Ilkin Serengil

YouTube

6K

Follow me on Twitter

Follow @serengil

Perception is everything. Communication faculty students learn this in their early lessons. Machine learning practitioners learn this in their freshman days as well. perceptron is an early version of modern neural networks. Understanding the logic behind the classical single layer perceptron will help you to understand the idea behind deep learning as well. Because you can image deep neural networks as combination of nested perceptrons. You can also imagine single layer perceptron as legacy neural networks.





Perception is everything!

Camels are the little white lines whereas black lines are shadows in the picture above. This amazing photo is taken by [George Steinmetz](#).



You may consider to enroll my top-rated machine learning course on Udemy



Decision Trees for Machine Learning From Scratch



Sefik Ilkin Serengil

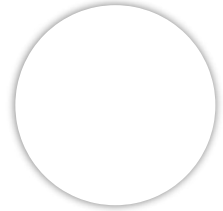
Udemy



Explaining perceptron with metaphors

Explaining perceptron with some metaphors might help you to understand the perceptron better.

Perceptron Explained in 2 Minutes



Vlog

You can either watch the following video or read this blog post. They both cover the perceptron from scratch. Additionally, vlog explains perceptron in python.

Perceptron From Scratch in Python



Supportive problem

Suppose that we are going to work on AND Gate problem. The gate returns 1 if and only if both inputs are true.

X_1	X_2	Y
0	0	0
0	1	0
1	0	0
1	1	1

We are going to set weights randomly. Let's say that $w_1 = 0.9$ and $w_2 = 0.9$

Round 1

We will apply 1st instance to the perceptron. $x_1 = 0$ and $x_2 = 0$.

Sum unit will be 0 as calculated below

$$\Sigma = x_1 * w_1 + x_2 * w_2 = 0 * 0.9 + 0 * 0.9 = 0$$

Activation unit checks sum unit is greater than a threshold. If this rule is satisfied, then it is fired and the unit will return 1, otherwise it will return 0. BTW, modern neural networks architectures do not use this kind of a step function as activation.

Activation threshold would be 0.5.

Sum was 0 for the 1st instance. So, activation unit would return 0 because it is less than 0.5. Similarly, its output should be 0 as well. We will not update weights because there is no error in this case.



Let's focus on the 2nd instance. $x_1 = 0$ and $x_2 = 1$.

$$\text{Sum unit: } \Sigma = x_1 * w_1 + x_2 * w_2 = 0 * 0.9 + 1 * 0.9 = 0.9$$

What about errors?

Activation unit will return 1 because sum unit is greater than 0.5. However, output of this instance should be 0. This instance is not predicted correctly. That's why, we will update weights based on the error.

$$\varepsilon = \text{actual} - \text{prediction} = 0 - 1 = -1$$

We will add error times learning rate value to the weights. Learning rate would be 0.5. BTW, we mostly set learning rate value between 0 and 1.

$$w_1 = w_1 + \alpha * \varepsilon = 0.9 + 0.5 * (-1) = 0.9 - 0.5 = 0.4$$

$$w_2 = w_2 + \alpha * \varepsilon = 0.9 + 0.5 * (-1) = 0.9 - 0.5 = 0.4$$

Focus on the 3rd instance. $x_1 = 1$ and $x_2 = 0$.

$$\text{Sum unit: } \Sigma = x_1 * w_1 + x_2 * w_2 = 1 * 0.4 + 0 * 0.4 = 0.4$$

Activation unit will return 0 this time because output of the sum unit is 0.4 and it is less than 0.5. We will not update weights.

Mention the 4th instance. $x_1 = 1$ and $x_2 = 1$.

$$\text{Sum unit: } \Sigma = x_1 * w_1 + x_2 * w_2 = 1 * 0.4 + 1 * 0.4 = 0.8$$

Activation unit will return 1 because output of the sum unit is 0.8 and it is greater than the threshold value 0.5. Its actual value should 1 as well. This means that 4th instance is predicted correctly. We will not update anything.

Round 2

In previous round, we've used previous weight values for the 1st instance and it was predicted correctly. Let's apply feed forward for the new weight values.



Remember the 1st instance. $x_1 = 0$ and $x_2 = 0$.

$$\text{Sum unit: } \Sigma = x_1 * w_1 + x_2 * w_2 = 0 * 0.4 + 0 * 0.4 = 0.4$$

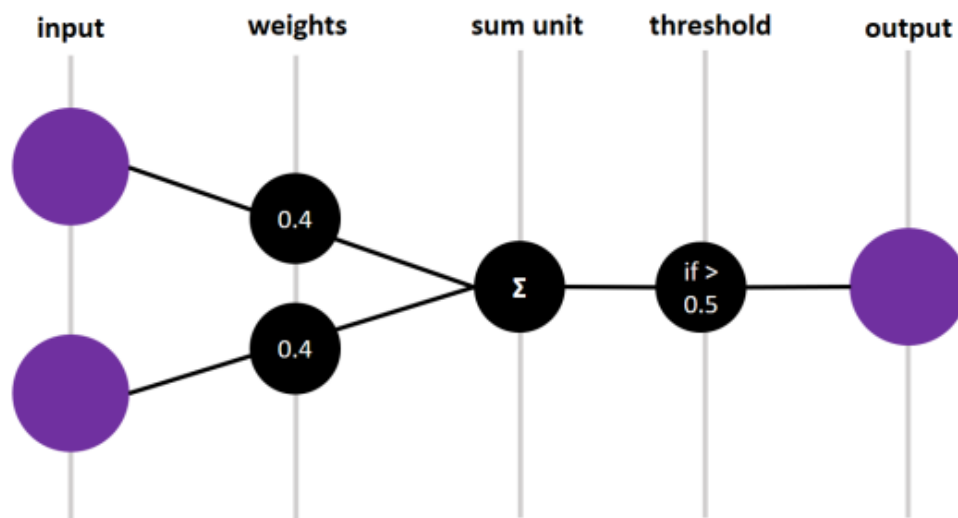
Activation unit will return 0 because sum unit is 0.4 and it is less than the threshold value 0.5. The output of the 1st instance should be 0 as well. This means that the instance is classified correctly. We will not update weights.

Feed forward for the 2nd instance. $x_1 = 0$ and $x_2 = 1$.

Sum unit: $\Sigma = x_1 * w_1 + x_2 * w_2 = 0 * 0.4 + 1 * 0.4 = 0.4$


Activation unit will return 0 because sum unit is less than the threshold 0.5. Its output should be 0 as well. This means that it is classified correctly and we will not update weights.

We've applied feed forward calculation for 3rd and 4th instances already for the current weight values in the previous round. They were classified correctly.



Perceptron for AND Gate

Learning term

We  continue this procedure until learning completed. We can terminate the learning procedure here. Luckily, we can find the best weights in 2 rounds.

Updating weights means learning in the perceptron. We set weights to 0.9 initially but it causes some errors. Then, we update the weight values to 0.4. In this way, we can predict all instances correctly.

I've written the logic of perceptron in python. You can find the source code [here](#).

Homework

To reinforce the perceptron, you should apply learning procedure for OR Gate. The gate returns 0 if and only if both inputs are 0. Do not hesitate to change the initial weights and learning rate values.

X_1	X_2	Y
0	0	0
0	1	1
1	0	1
1	1	1

Basic perceptron can generalize any kind of linear problem. The both AND and OR Gate problems are linearly separable problems. On the other hand, this form cannot generalize non-linear problems such as XOR Gate. Perceptron evolved to [multilayer perceptron](#) to solve non-linear problems and deep neural networks were born.

Feature Importance

Explainable AI and machine learning interpretability are the hottest topics nowadays in the data world. Herein, perceptrons are naturally explainable algorithms. Similar to linear regression, coefficients are directly related to the feature importance values.



Feature Importance of Linear Regression in Python From Scratch



Multilayer perceptron

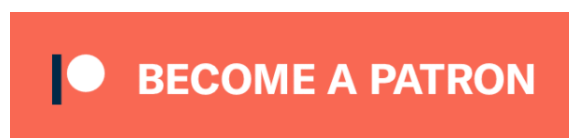
Multilayer perceptron or its more common name neural networks can solve non-linear problems. It is the evolved version of perceptron.



Math Behind Neural Networks and Deep Learning: Backpropagation



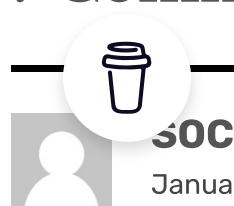
Support this blog if you do like!



#perceptron, #single layer perceptron

« *Previous* / *Next* »

7 Comments



January 11, 2020 at 11:15 pm

Algebraic, Geometric, Statistical aspects of the weighted sum?

**Sefik Serengil**

January 13, 2020 at 1:19 pm

what do you mean exactly?

**kurnaiwan**

April 14, 2020 at 3:22 am

show hyperplane that created each iteration, will help me so much. thanks.

**Sefik Serengil**

April 14, 2020 at 5:39 am

it would be clever! I will study on this.

**leila**

July 5, 2020 at 12:12 pm

thanks a lot

**António**

January 10, 2021 at 11:52 pm

Hey, thx a lot for the post.

In Round 2 you have:

Sum unit:

$$\Sigma = x_1 * w_1 + x_2 * w_2 = 0 * 0.4 + 0 * 0.4 = 0.4$$

which is 0, not 0.4.

Although the conclusion still applies, you might want to correct this minor error, to avoid future confusion.

Che

**Veys**

April 16, 2021 at 10:05 pm

Very helpful post to understand perceptron algorithm, thank you very much

Comments are closed.

Licensed under a Creative Commons Attribution 4.0 International License.



You can use any content of this blog just to the extent that you cite or reference

Please cite this post if it helps your research. Here is an example of BibTeX entry:

```
@misc{sefiks13833,  
  author = {Serengil, Sefik Ilkin},  
  title = { A Step by Step Perceptron Example  
},  
  howpublished = {  
https://sefiks.com/2020/01/04/a-step-by-  
step-perceptron-example/ },  
  year = { 2020 },  
  note = "[Online; accessed 2024-07-02]"  
}
```

