Deep Learning course

Edgar **Francisco** Roman-Rangel
edgar.roman@alumni.epfl.ch

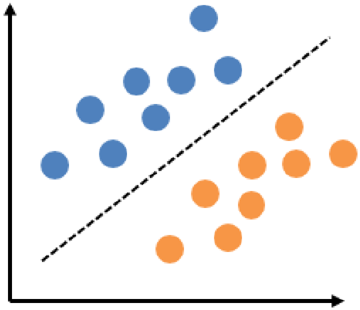
Session 4 – Perceptron

CInC-UAEM. Cuernavaca, Mexico. September 8th, 2018.

Perceptron

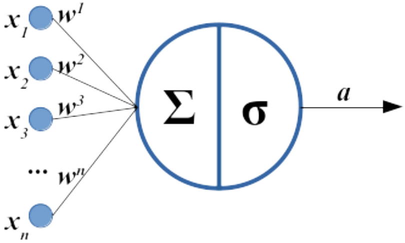
The simplest artificial neural network.

- by Frank Rosenblatt in the 1950s.
- It is a linear classifier used for binary prediction.
- Inspired in the neuron.



Step perceptron

A linear combination. It "fires" if its result is above a threshold.



$$z = \sum_i \omega_i x_i$$

$$a = \begin{cases} 1, & z > b \\ 0, & otherwise. \end{cases}$$

Include bias term

We can add the threshold as the bias term b into the model, so we can estimate it automatically.

$$z = \sum_{i=1}^{n} \omega_i x_i + b$$

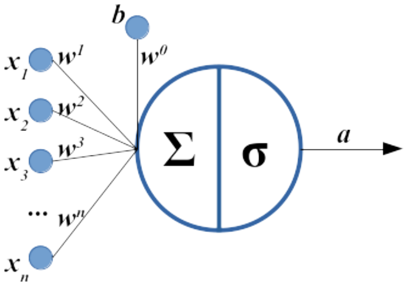
We must redefine the activation as

$$a = sgn(z) = \begin{cases} +1, & z \geq 0 \\ -1, & z < 0. \end{cases}$$

Simplifying bias

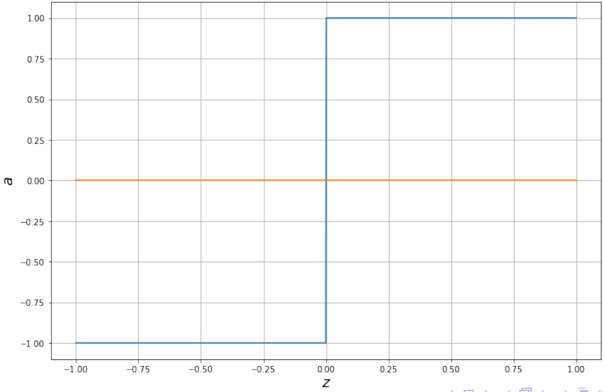
By making $x_0 = b = 1$, we can add the bias term to the summation as

$$z = \sum_{i=0}^{n} \omega_i x_i$$



Step function

$$a = \begin{cases} +1, & z \geq 0 \\ -1, & otherwise. \end{cases}$$



Update rule

$$\omega_i = \omega_i + \delta\omega_i$$

What is a good $\delta\omega$?

$$\omega_i = \omega_i + \eta(y - a)x$$

where,

- η is a learning rate $(0, 1)$.
- y is the expected output associated to x_i.
- x_i is the derivative of $\frac{dE}{d\omega_i}$
- $(y - a)$ indicates direction.

- If classifies a '0' as a '1', subtract the feature from the weight.
- If classifies a '1' as a '0', add the feature to the weight.

Algorithm 1 Pseudocode for perceptron

$\omega = 0, b = 0$

for epoch in Epochs **do**

    for all $(x, y)^j$ **do**

        $a = \sigma(\omega^T x)$

        $e = y - a$

        $\omega = \omega + \eta(e)x$

    end for

end for

Epoch: one iteration over the complete training set.

Sigmoid perceptron

A smooth function is applied after the linear combination.

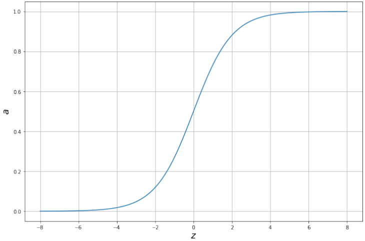
It gives the probability of belonging to the positive class.

$$z = \omega^T x$$

$$y = \sigma(z)$$

sigmoid function

$$\sigma(z) = \frac{1}{1 + \exp^{-z}}$$



Update rule

$$\omega_i = \omega_i + \delta\omega_i$$

What is a good $\delta\omega$?

$$\omega_i = \omega_i + \eta(y - a)x_i$$

Derivative of sigmoid

$$\sigma(z) \cdot (1 - \sigma(z))$$

$$\omega_i = \omega_i + \eta(y - a)(a(1 - a))x$$

Suggested activities

- Check up python tutorial, specially: lists and print.
- Check numpy tutorial.
- Try sigmoid perceptron.
- Extend it to input of n-dimensions.
- Test it on the iris dataset.

Thank you.

Q&A