Neural Networks

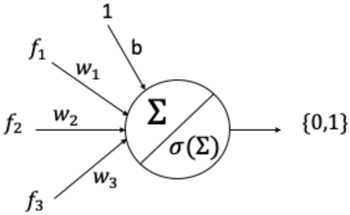
Dr. Víctor Uc Cetina

Universidad Autónoma de Yucatán

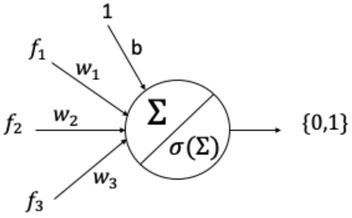
Content

1. Perceptron
2. Multilayer Perceptron

A neuron for animal classification (Elephant or Dog)



A neuron for animal classification (Elephant or Dog)

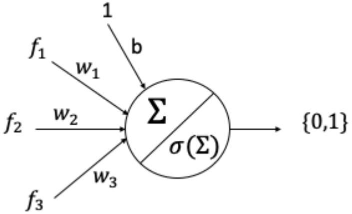


$$\Sigma = b + w_1 f_1 + w_2 f_2 + w_3 f_3$$

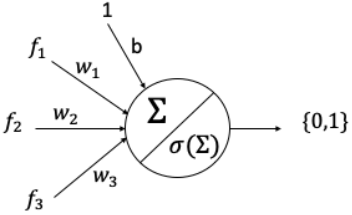
where:

f_1 : Weight of the animal
f_2 : Height of the animal
f_3 : Color of the animal

A neuron for animal classification (Elephant or Dog)



A neuron for animal classification (Elephant or Dog)

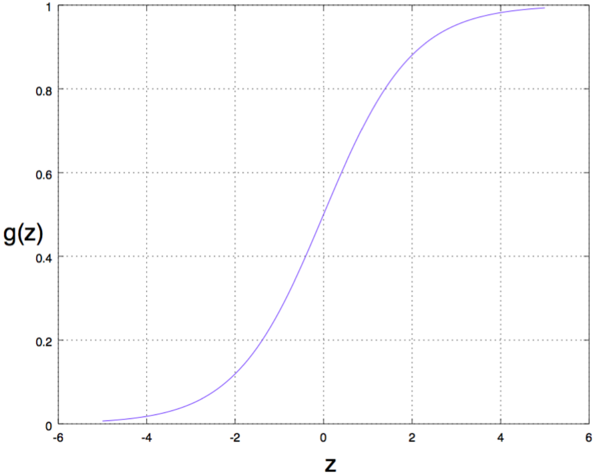


$$\Sigma = b + w_1 f_1 + w_2 f_2 + w_3 f_3$$

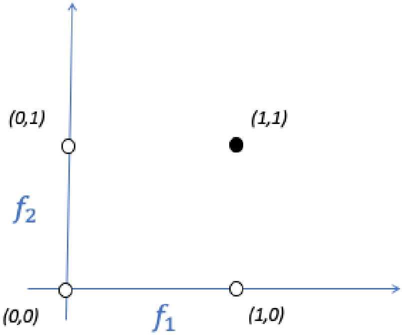
$$\sigma(\Sigma) = \frac{1}{1+e^{-\Sigma}} \qquad \sigma : \mathbb{R} \to (0, 1)$$

Sigmoid function

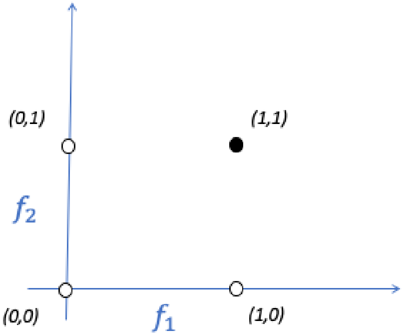
$$g(z) = \frac{1}{1+e^{-z}} \qquad g : \mathbb{R} \to (0, 1)$$



A neuron for the AND function

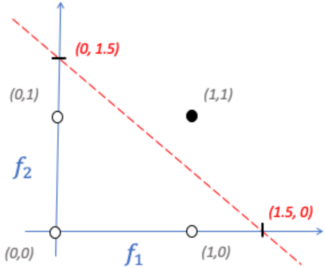


A neuron for the AND function

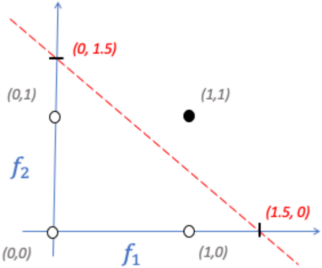


f_1	f_2	Output
0	0	0
0	1	0
1	0	0
1	1	1

A neuron for the AND function

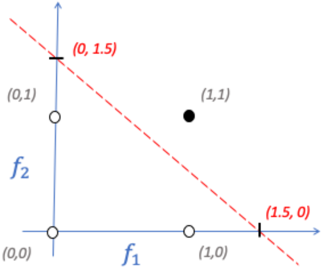


A neuron for the AND function



We can write $w_1 f_1 + w_2 f_2 + b = 0$

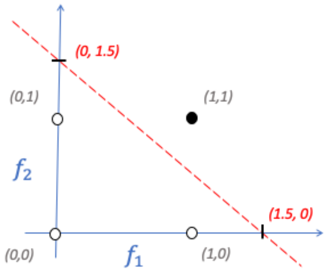
A neuron for the AND function



We can write $w_1 f_1 + w_2 f_2 + b = 0$

Solving for $(1.5, 0)$, we have
$1.5 w_1 = -b$

A neuron for the AND function

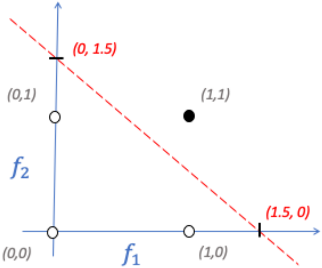


We can write $w_1 f_1 + w_2 f_2 + b = 0$

Solving for $(1.5, 0)$, we have
$1.5 w_1 = -b$

Solving for $(0, 1.5)$, we have
$1.5 w_2 = -b$

A neuron for the AND function



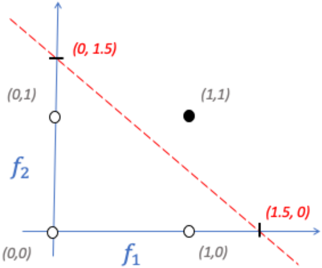
We can write $w_1 f_1 + w_2 f_2 + b = 0$

Solving for $(1.5, 0)$, we have
$1.5w_1 = -b$

Solving for $(0, 1.5)$, we have
$1.5w_2 = -b$

This implies that $w_1 = w_2$

A neuron for the AND function



We can write $w_1 f_1 + w_2 f_2 + b = 0$

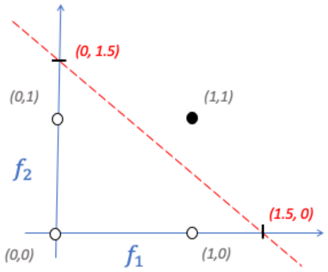
Solving for $(1.5, 0)$, we have $1.5w_1 = -b$

Solving for $(0, 1.5)$, we have $1.5w_2 = -b$

This implies that $w_1 = w_2$

Making $w_1 = 1$, we can see that $b = -1.5$

A neuron for the AND function



We can write $w_1 f_1 + w_2 f_2 + b = 0$

Solving for $(1.5, 0)$, we have $1.5w_1 = -b$

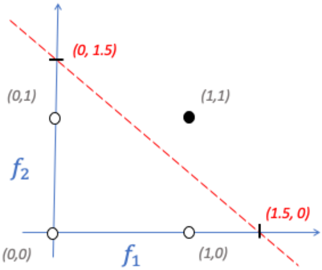
Solving for $(0, 1.5)$, we have $1.5w_2 = -b$

This implies that $w_1 = w_2$

Making $w_1 = 1$, we can see that $b = -1.5$

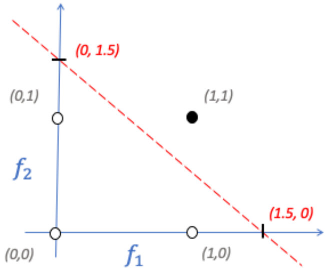
Therefore, our equation is $f_1 + f_2 - 1.5 = 0$

A neuron for the AND function



Classifying new points with
$\Sigma = f_1 + f_2 - 1.5$

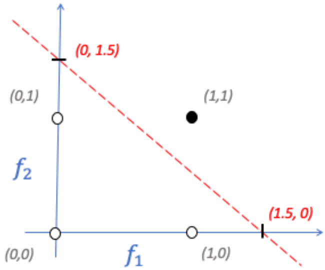
A neuron for the AND function



Classifying new points with
$\Sigma = f_1 + f_2 - 1.5$

Evaluating $(0, 0)$, we have $\Sigma = -1.5$
Since $\Sigma < 0$, the label is 0 (Neg.)

A neuron for the AND function

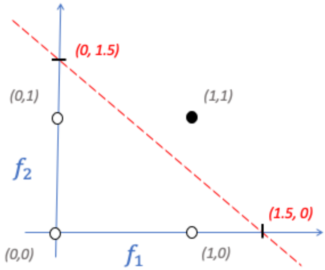


Classifying new points with
$\Sigma = f_1 + f_2 - 1.5$

Evaluating $(0, 0)$, we have $\Sigma = -1.5$
Since $\Sigma < 0$, the label is 0 (Neg.)

Evaluating $(0, 1)$, we have $\Sigma = -0.5$
Since $\Sigma < 0$, the label is 0 (Neg.)

A neuron for the AND function



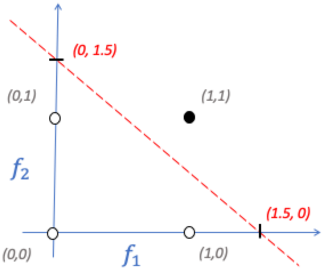
Classifying new points with
$\Sigma = f_1 + f_2 - 1.5$

Evaluating $(0,0)$, we have $\Sigma = -1.5$
Since $\Sigma < 0$, the label is 0 (Neg.)

Evaluating $(0,1)$, we have $\Sigma = -0.5$
Since $\Sigma < 0$, the label is 0 (Neg.)

Evaluating $(1,0)$, we have $\Sigma = -0.5$
Since $\Sigma < 0$, the label is 0 (Neg.)

A neuron for the AND function



Classifying new points with
$\Sigma = f_1 + f_2 - 1.5$

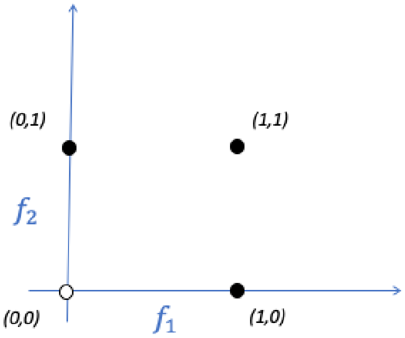
Evaluating $(0, 0)$, we have $\Sigma = -1.5$
Since $\Sigma < 0$, the label is 0 (Neg.)

Evaluating $(0, 1)$, we have $\Sigma = -0.5$
Since $\Sigma < 0$, the label is 0 (Neg.)

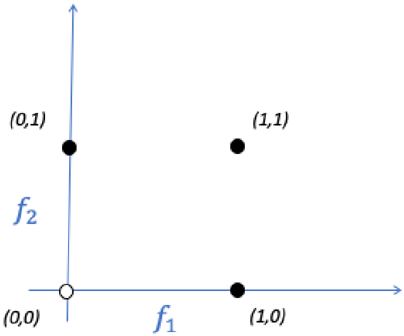
Evaluating $(1, 0)$, we have $\Sigma = -0.5$
Since $\Sigma < 0$, the label is 0 (Neg.)

Evaluating $(1, 1)$, we have $\Sigma = 0.5$
Since $\Sigma > 0$, the label is 1 (Pos.)

A neuron for the OR function

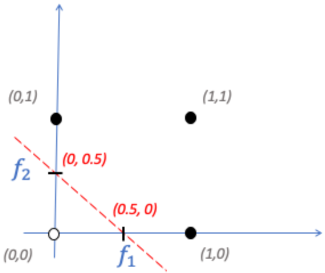


A neuron for the OR function



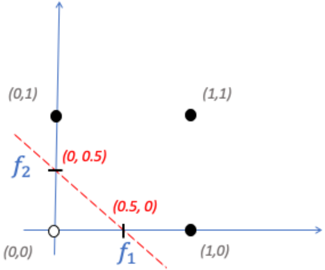
f_1	f_2	Output
0	0	0
0	1	1
1	0	1
1	1	1

A neuron for the OR function



Classifying new points with
$\Sigma = f_1 + f_2 - 0.5$

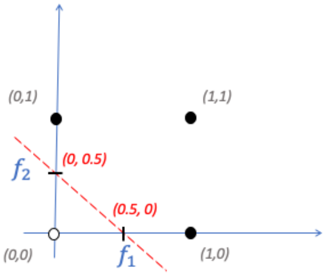
A neuron for the OR function



Classifying new points with
$\Sigma = f_1 + f_2 - 0.5$

Evaluating $(0, 0)$, we have $\Sigma = -0.5$
Since $\Sigma < 0$, the label is 0 (Neg.)

A neuron for the OR function

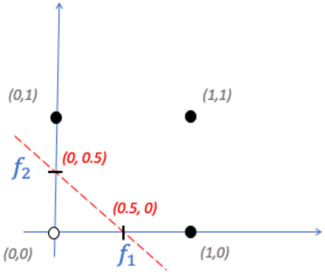


Classifying new points with
$\Sigma = f_1 + f_2 - 0.5$

Evaluating $(0, 0)$, we have $\Sigma = -0.5$
Since $\Sigma < 0$, the label is 0 (Neg.)

Evaluating $(0, 1)$, we have $\Sigma = 0.5$
Since $\Sigma > 0$, the label is 1 (Pos.)

A neuron for the OR function



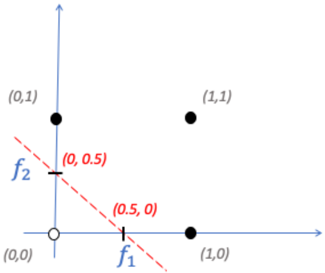
Classifying new points with
$\Sigma = f_1 + f_2 - 0.5$

Evaluating $(0, 0)$, we have $\Sigma = -0.5$
Since $\Sigma < 0$, the label is 0 (Neg.)

Evaluating $(0, 1)$, we have $\Sigma = 0.5$
Since $\Sigma > 0$, the label is 1 (Pos.)

Evaluating $(1, 0)$, we have $\Sigma = 0.5$
Since $\Sigma > 0$, the label is 1 (Pos.)

A neuron for the OR function



Classifying new points with
$\Sigma = f_1 + f_2 - 0.5$

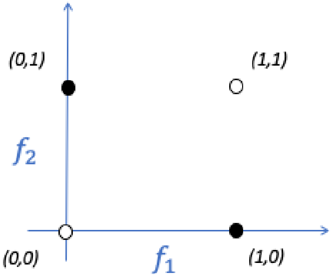
Evaluating $(0, 0)$, we have $\Sigma = -0.5$
Since $\Sigma < 0$, the label is 0 (Neg.)

Evaluating $(0, 1)$, we have $\Sigma = 0.5$
Since $\Sigma > 0$, the label is 1 (Pos.)

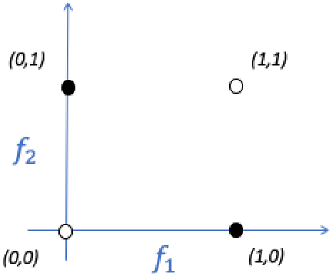
Evaluating $(1, 0)$, we have $\Sigma = 0.5$
Since $\Sigma > 0$, the label is 1 (Pos.)

Evaluating $(1, 1)$, we have $\Sigma = 1.5$
Since $\Sigma > 0$, the label is 1 (Pos.)

A neuron for the XOR function

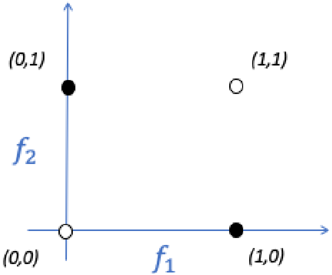


A neuron for the XOR function



f_1	f_2	Output
0	0	0
0	1	1
1	0	1
1	1	0

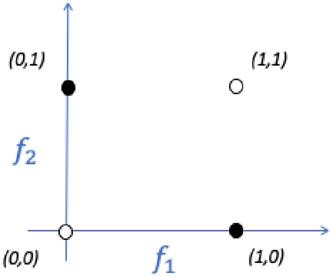
A neuron for the XOR function



f_1	f_2	Output
0	0	0
0	1	1
1	0	1
1	1	0

- XOR problem with perceptron (Minsky and Papert, 1969).

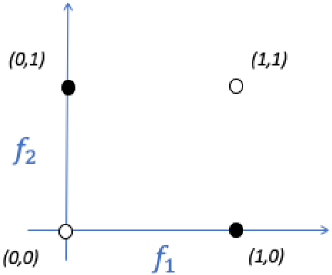
A neuron for the XOR function



f_1	f_2	Output
0	0	0
0	1	1
1	0	1
1	1	0

- XOR problem with perceptron (Minsky and Papert, 1969).
- Proposal of multilayer perceptrons (MLP) (Werbos,1982).

A neuron for the XOR function

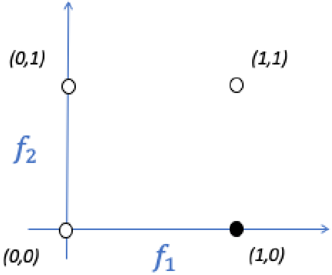


f_1	f_2	Output
0	0	0
0	1	1
1	0	1
1	1	0

- XOR problem with perceptron (Minsky and Papert, 1969).
- Proposal of multilayer perceptrons (MLP) (Werbos,1982).
- MLPs are popularized (Hinton et al., 1986).

Solving the XOR problem, with 3 neurons

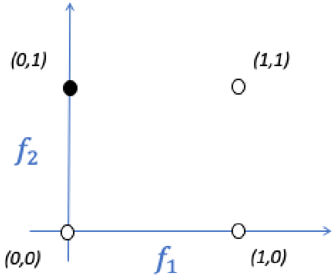
Neuron 1



f_1	f_2	Output
0	0	0
0	1	0
1	0	1
1	1	0

Solving the XOR problem, with 3 neurons

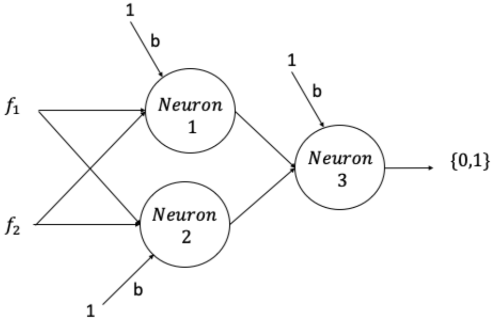
Neuron 2



f_1	f_2	Output
0	0	0
0	1	1
1	0	0
1	1	0

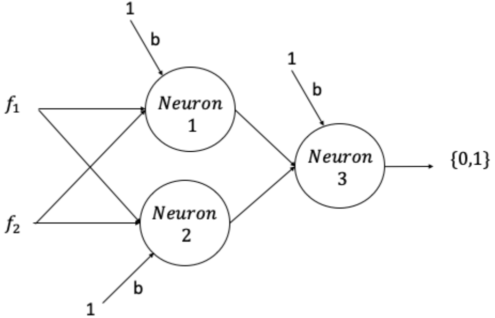
Solving the XOR problem, with 3 neurons

Neuron 3



Solving the XOR problem, with 3 neurons

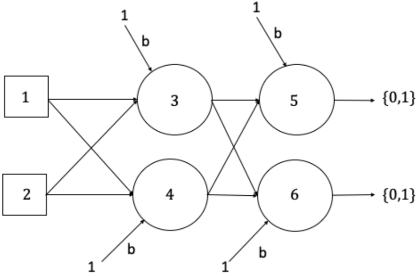
Neuron 3



Neuron 1	Neuron 2	Neuron 3 (XOR)
0	0	0
0	1	1
1	0	1
1	1	0

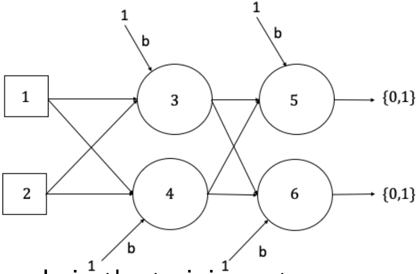
Backpropagation algorithm

MLP with topology 2 x 2 x 2:



Backpropagation algorithm

MLP with topology 2 x 2 x 2:

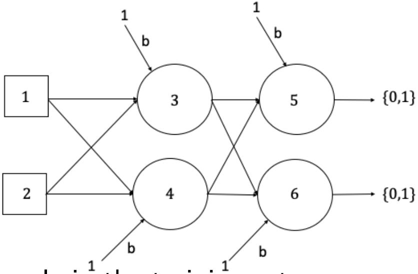


Repeat for every example in the training set:

1. Take an example and propagate the input signal, going forward: from the input neurons to the output neurons, passing through all the hidden neurons.

Backpropagation algorithm

MLP with topology 2 x 2 x 2:

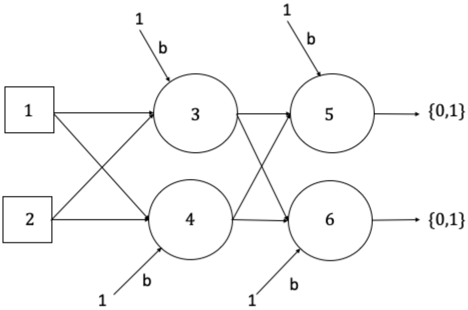


Repeat for every example in the training set:

1. Take an example and propagate the input signal, going forward: from the input neurons to the output neurons, passing through all the hidden neurons.
2. Backpropagate the error of the network with respect to the current example, going backward: from the output neurons to the input neurons, passing through all the hidden neurons.

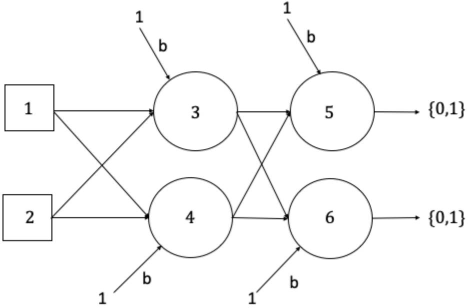
Backpropagation algorithm

MLP with topology 2 x 2 x 2:



Backpropagation algorithm

MLP with topology 2 x 2 x 2:

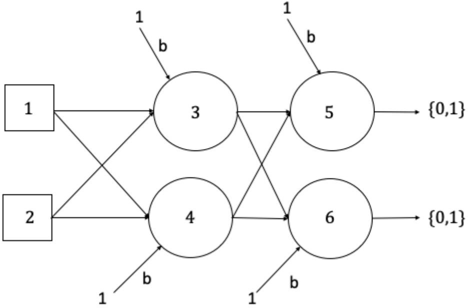


Learning rule for an output neuron:

$$\delta_5 := o_5(1 - o_5)(y_5 - o_5)$$

Backpropagation algorithm

MLP with topology 2 x 2 x 2:

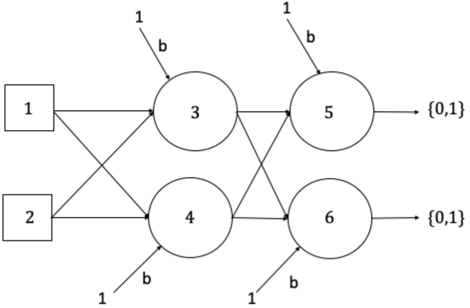


Learning rule for an output neuron:

$$\delta_5 := o_5(1 - o_5)(y_5 - o_5)$$
$$w_{5,3} := w_{5,3} + \alpha \delta_5 o_3$$

Backpropagation algorithm

MLP with topology 2 x 2 x 2:



Learning rule for an output neuron:

$$
\begin{aligned}
\delta_5 &:= o_5(1 - o_5)(y_5 - o_5) \\
w_{5,3} &:= w_{5,3} + \alpha \delta_5 o_3 \\
w_{5,4} &:= w_{5,4} + \alpha \delta_5 o_4
\end{aligned}
$$

Backpropagation algorithm

Learning rule for an output neuron:

$$\delta_5 := o_5(1 - o_5)(y_5 - o_5)$$

$$w_{5,3} := w_{5,3} + \alpha\delta_5 o_3$$

$$w_{5,4} := w_{5,4} + \alpha\delta_5 o_4$$

Backpropagation algorithm

Learning rule for an output neuron:

$$\delta_5 := o_5(1 - o_5)(y_5 - o_5)$$

$$w_{5,3} := w_{5,3} + \alpha \delta_5 o_3$$

$$w_{5,4} := w_{5,4} + \alpha \delta_5 o_4$$

$$\delta_k : \text{error of neuron } k$$

$$o_k : \text{output of neuron } k, \text{ for instance } o_k = \sigma(\Sigma_k)$$

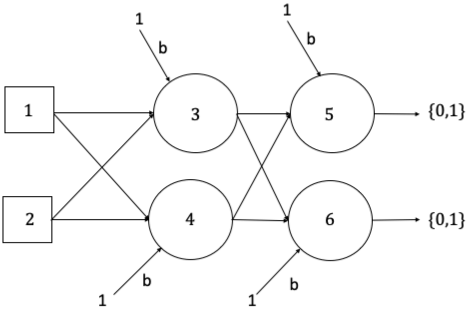
$$y_k : \text{desired output of neuron } k$$

$$w_{i,j} : \text{weight of connection going from } j \text{ to } i$$

$$\alpha : \text{learning rate}$$

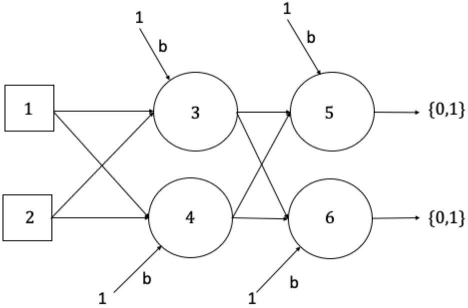
Backpropagation algorithm

MLP with topology 2 x 2 x 2:



Backpropagation algorithm

MLP with topology 2 x 2 x 2:

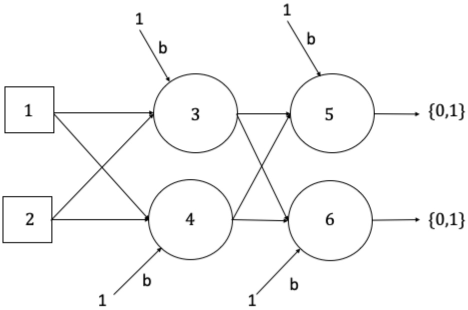


Learning rule for a hidden neuron:

$$\delta_3 := o_3(1 - o_3)(w_{5,3}\delta_5 + w_{6,3}\delta_6)$$

Backpropagation algorithm

MLP with topology 2 x 2 x 2:

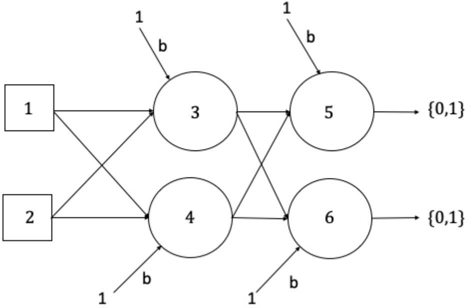


Learning rule for a hidden neuron:

$$\delta_3 := o_3(1 - o_3)(w_{5,3}\delta_5 + w_{6,3}\delta_6)$$
$$w_{3,1} := w_{3,1} + \alpha\delta_3 o_1$$

Backpropagation algorithm

MLP with topology 2 x 2 x 2:

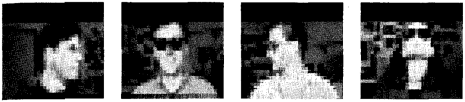


Learning rule for a hidden neuron:

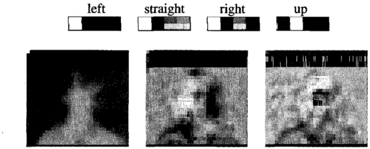
$$\delta_3 := o_3(1 - o_3)(w_{5,3}\delta_5 + w_{6,3}\delta_6)$$
$$w_{3,1} := w_{3,1} + \alpha\delta_3 o_1$$
$$w_{3,2} := w_{3,2} + \alpha\delta_3 o_2$$

MLP example

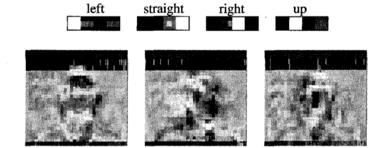
Topology 960 x 3 x 4, trained on 260 grey-level images, it achieves 90% performance. (Taken from Tom Mitchell's book)



30 × 32 resolution input images



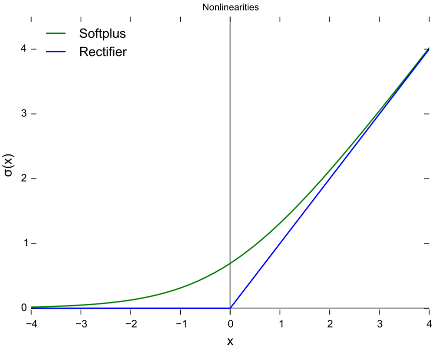
Network weights after 1 iteration through each training example



Network weights after 100 iterations through each training example

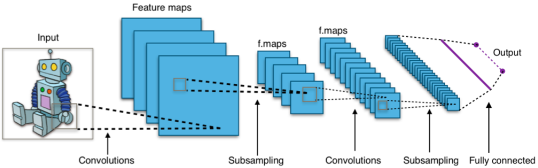
From shallow neural networks to deep neural networks

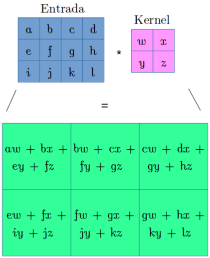
- The vanishing gradient problem that suffered the first MLP has now been solved with different approaches. One of them is by using activation functions such as the rectified linear unit (ReLu).



CNNs

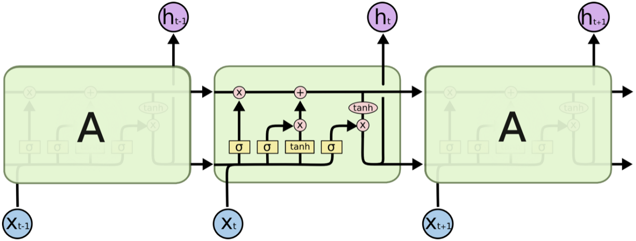
- Convolutional neural network (CNN): One important kind of network specially powerful for learning patterns on images and video. Some of the most popular are: LeNet, AlexNet and GoogleNet.





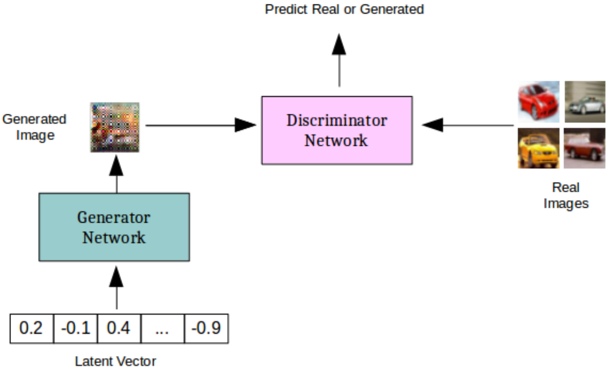
RNNs

- Recurrent neural network (RNN): Neural network specially tailored for memorizing sequences of data such as audio and text: the long-short term memory (LSTM) and the gated recurrent unit (GRU) are by far the most popular.



GANs

- Generative adversarial networks (GAN): These models use 2 networks that learn a distribution of the data.



GANs

- Nvidia researchers introduced in 2018 StyleGAN, a GAN for producing an unlimited number of portraits of fake human faces. It was trained using Nvidia's GPU processors.

Thank you!

Dr. Víctor Uc Cetina
victoruccetina@gmail.com