Expressiveness

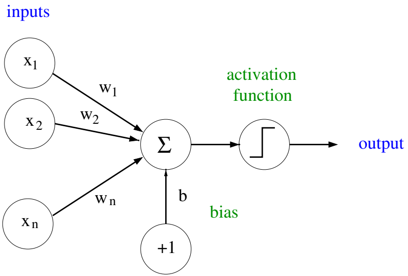
Next arguments

What can we compute with a NN?

- the single layer case

The perceptron

Binary threshold:



$$output = \begin{cases} 1 & if \sum_i w_i x_i + b \geq 0 \\ 0 & otherwise \end{cases} \qquad output = \begin{cases} 1 & if \sum_i w_i x_i \geq -b \\ 0 & otherwise \end{cases}$$

Remark: the bias set the position of threshold.

Hyperplanes

The set of points

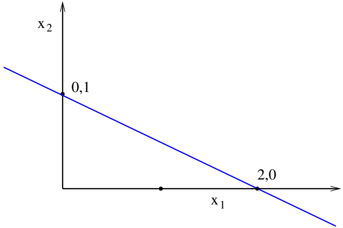
$$\sum_i w_i x_i + b = 0$$

defines a hyperplane in the space of the variables x_i

Example:

$$-\frac{1}{2}x_1 + x_2 + 1 = 0$$

is a line in the bidimensional space



Hyperplanes

The hyperplane

$$\sum_i w_i x_i + b = 0$$

divides the space in two parts: to one of them (above the line) the perceptron gives value 1, to the other (below the line) value 0.

"above" and "below" can be inverted by just inverting parameters:

$$\sum_i w_i x_i + b \leq 0 \iff \sum_i -w_i x_i - b \geq 0$$

Computing logical connectives: NAND

Can we implement this function (NAND) with a perceptron?

x_1	x_2	*output*
0	0	1
0	1	1
1	0	1
1	1	0

Computing logical connectives: NAND

Can we implement this function (NAND) with a perceptron?

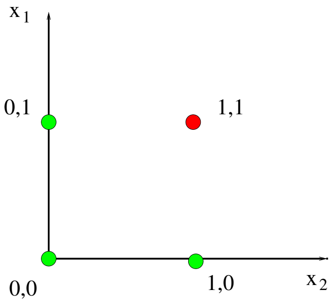
x_1	x_2	*output*
0	0	1
0	1	1
1	0	1
1	1	0

Can we find two weights w_1 and w_2 and a bias b such that

$$nand(x_1, x_2) = \begin{cases} 1 & if \sum_i w_i x_i + b' \geq 0 \\ 0 & otherwise \end{cases}$$

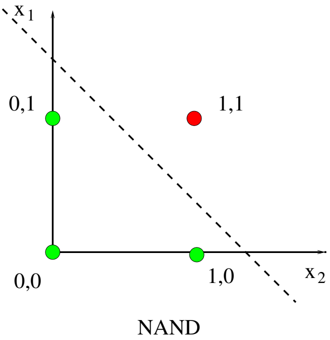
Graphical representation

Same as asking:
can we draw a straight line to separate green and red points?



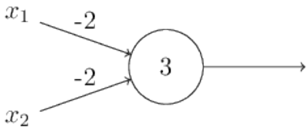
NAND

Yes!



line equation: $1.5 - x_1 - x_2 = 0$ or $3 - 2x_1 - 2x_2 = 0$

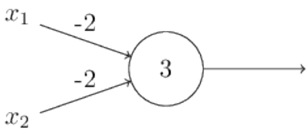
The NAND-perpceptron



$$output = \begin{cases} 1 & if -2x_1 - 2x_2 + 3 \geq 0 \\ 0 & otherwise \end{cases}$$

x_1	x_2	*output*
0	0	1
0	1	1
1	0	1
1	1	0

The NAND-perpceptron



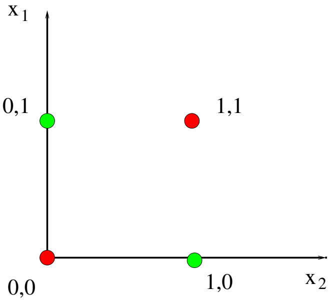
$$output = \begin{cases} 1 & if -2x_1 - 2x_2 + 3 \geq 0 \\ 0 & otherwise \end{cases}$$

x_1	x_2	*output*
0	0	1
0	1	1
1	0	1
1	1	0

Can we compute any logical circuit with a perceptron?

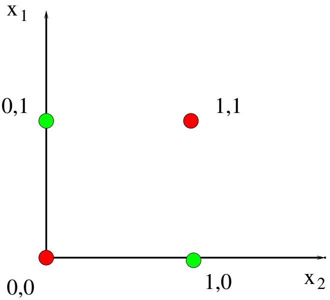
The XOR case

Can we draw a straight line separating red and green points?



The XOR case

Can we draw a straight line separating red and green points?



No way!

Single layer perceptrons are not complete!

XOR in image processing

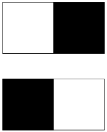
Can we recognize these patterns with a perceptron (aka binary threshold)?

XOR in image processing

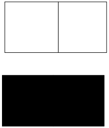
Can we recognize these patterns with a perceptron (aka binary threshold)?

No Each pixel should individually contribute to the classification, that is not the case (more in the next slides)

XOR in image processing



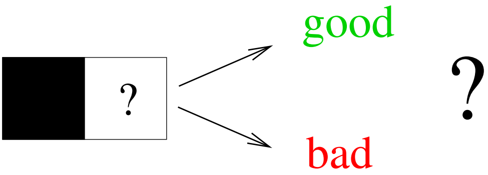
good



bad

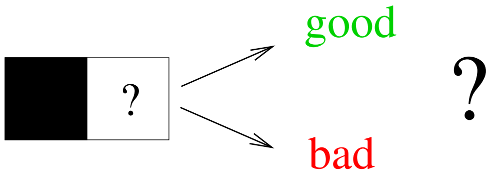
XOR in image processing

Let us e.g. consider the first pixel, and suppose it is **black** (the white case is symmetric)

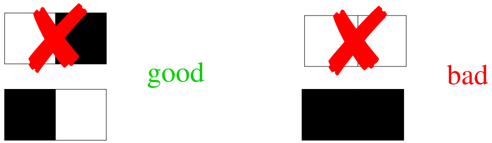


does this improve our knowledge for the purposes of classification?

XOR in image processing



we can say nothing

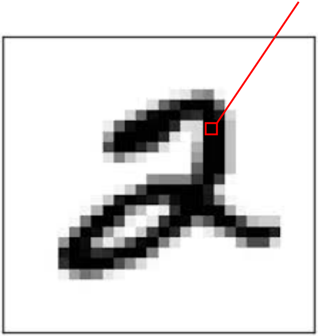


we have still the same probability to have a good or a bad example.

Example MNIST

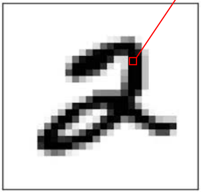
Can we address digit recognition with linear tools? (perceptrons, logistic regression, . . .)

Does the intensity of each pixel contribute to classify digits?



Example MNIST

Does the intensity of each pixel contribute to classify digits?



- + weighted sum over a large number of features
- - need of preproceesing (centering, rotating, normalizing, etc)
- - different ways to write a same digit (e.g. 1,4,7,. . .)

classification results are modest: error rate 7-8 %

Multi-layer perceptrons

Question

- we know we can compute nand with a perceptron
- we know that nand is logically complete
  (i.e. we can compute any connective with nands)

so:

why perceptrons are not complete?

Question

- we know we can compute nand with a perceptron
- we know that nand is logically complete
  (i.e. we can compute any connective with nands)

so:

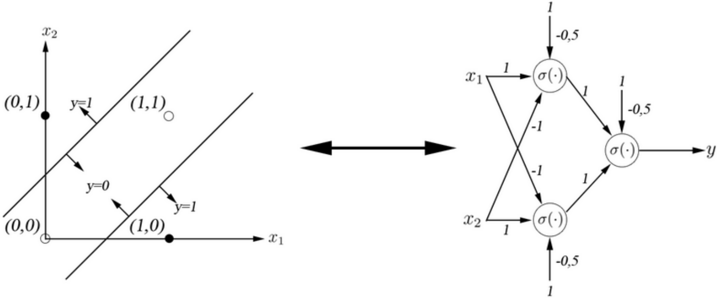
why perceptrons are not complete?

answer:

because we need to compose them and consider
Multi-layer perceptrons

Example: Multi-layer perceptron for XOR

Can we compute XOR by **stacking** perceptrons?



Multilayer perceptrons are logically complete!

Important Points

- shallow nets are already complete

  Why going for deep networks?

  With deep nets, the same function may be computed with less neural units (Cohen, et al.)

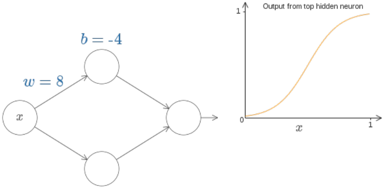
- Activation functions play an essential role, since they are the only source of nonlinearity, and hence of the expressiveness of NNs.

Formal expressiveness in the continuous case

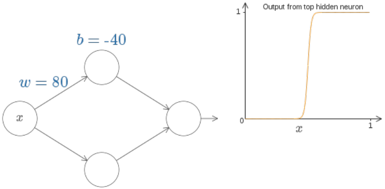
approximating functions with logistic neurons

Approximation by step functions

Single variable case: $\sigma(wx + b)$



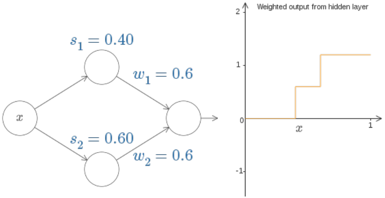
The "step" is located at the inflection point, $x = -\frac{b}{w}$



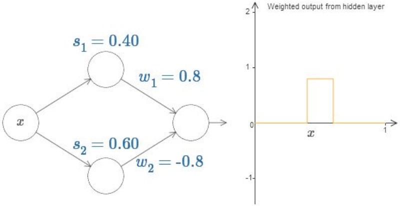
steepness varies with w

Sum of step functions

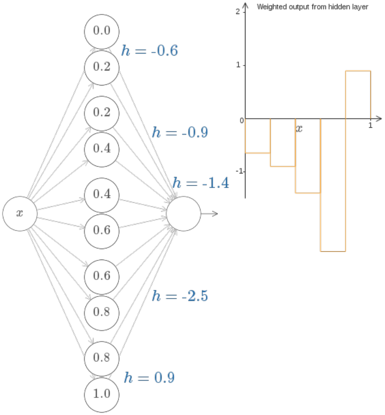
$s = -\frac{b}{w}$ per $w >> 0$



We can thus form "bumps" of arbitrary height and width



Approximations via bumps



Espressiveness

- Every continuous function $\mathcal{R} \to [0, 1]$ can be approximated by neural networks
- a single hidden layer is enough (shallow net)

Why using deep nets?

Espressiveness

- Every continuous function $\mathcal{R} \to [0, 1]$ can be approximated by neural networks
- a single hidden layer is enough (shallow net)

Why using deep nets?

fewer neurons suffice. There are known examples of deep narrow ReLU-networks that cannot be approximated by a shallow network unless it has exponentially many neurons (see e.g. formal expressivenes)

Demo: approximating functions

[demo]