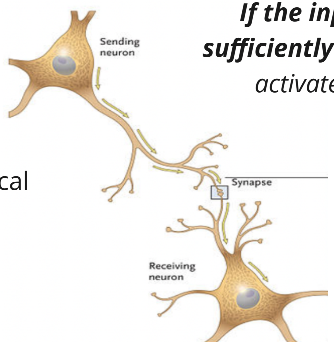
Behind the scenes

BNN

Multiple biological neurons propagate an electrochemical signal



If the input signal is sufficiently powerful *to activate a neuron, it transmits*

A receiving biological neuron receives an electrochemical signal

https://teachmephysiology.com/nervous-system/synapses/synaptic-transmission/

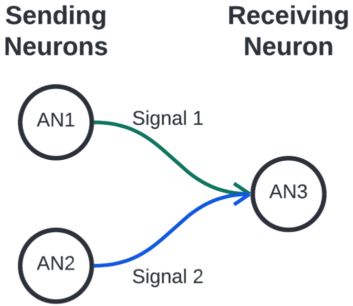
Enough neurons connected together would lead to a neural circuit (aka BNN)

https://www.forbes.com/sites/forbestechcouncil/2020/11/11/how-the-future-of-deep-learning-could-resemble-the-human-brain/?sh=3939c7cb415c

ANN

An ANN resembles this behaviour in a beautiful manner

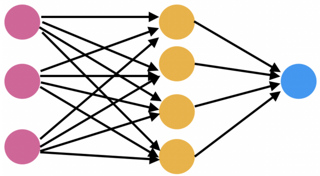
Multiple artificial neurons propagate a signal



A receiving artificial neuron receives a signal from previous AN

If the input signal is sufficiently powerful *to activate the neuron, it transmits*

Enough neurons connected together would lead to an artificial neural circuit (aka ANN)



Perceptron

- This is the simplest form of an ANN.
- A student wants to decide whether to go home after classes or stay and have a beer with friends (binary outcome: **1 if the student goes home and 0 if stays**).

$$f(\text{Input } 1, \ldots, \text{Input } n) = \text{Output Decision}$$

$$x_1 = \begin{cases} 1, & \text{Have to work tomorrow} \\ 0, & \text{Don't have to work tomorrow} \end{cases} \quad x_2 = \begin{cases} 1, & \text{Have assignment} \\ 0, & \text{Don't have assignment} \end{cases} \quad x_3 = \begin{cases} 1, & \text{Can't catch the train} \\ 0, & \text{Can catch the train} \end{cases}$$

Perceptron

*The student weights different factors before making a decision: the larger the weight, the more it would **cost** on the final decision.*

If the student has an assignment but next class is three days from today, it might not be that relevant for the decision. There is some **tolerance t** in the decision.

$$\text{Output Decision} = \begin{cases} 1, & \sum_i x_i w_i > t \\ 0, & \sum_i x_i w_i \leq t \end{cases} = \begin{cases} 1, & -t + \sum_i x_i w_i > 0 \\ 0, & -t + \sum_i x_i w_i \leq 0 \end{cases}$$

As we increase the t (decrease b=-t), the 'tolerance' to go for a beer with friends gets smaller.

Perceptron

With a perceptron, we can mimic a given simple human thought process: the synaptic transmission only happens when the total signal received exceeds a certain level:

$$\sigma(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

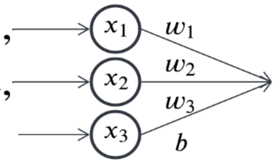
$f($

Input 1,

Input 2,

Input 3

$)$



$$\text{Output} = \sigma(b + \textstyle\sum_i x_i w_i)$$

Perceptron

15 min

EXERCISES: FROM 2.1 TO 2.3:

- Install the packages and restart the runtime in colab
- Remember that

$$w_1 = 3, w_2 = 2 \text{ and } w_3 = 6.$$

$$x_1 = \begin{cases} 1, & \text{Have to work tomorrow} \\ 0, & \text{Don't have to work tomorrow} \end{cases} \quad x_2 = \begin{cases} 1, & \text{Have assignment} \\ 0, & \text{Don't have assignment} \end{cases} \quad x_3 = \begin{cases} 1, & \text{Can't catch the train} \\ 0, & \text{Can catch the train} \end{cases}$$

Linear separation

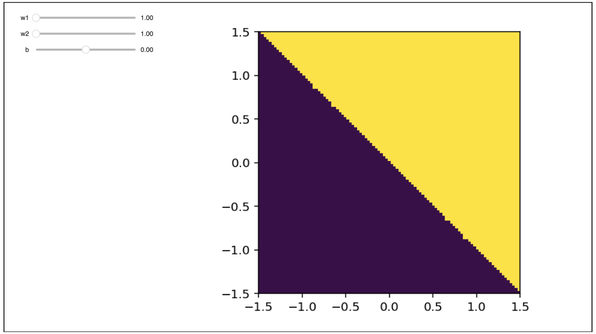
- Simple example in which our perceptron only has two input variables.
- Not only the input on the perceptron affects its output, its parameters (bias and weights) can also change the output. But how? Let us review the decision:

$$\sigma(b + \sum_i x_i w_i) = \sigma(b + x_1 w_1 + x_2 w_2)$$

- **Region 1 (R1)** $b + x_1 w_1 + x_2 w_2 > 0 \equiv x_1 w_1 + x_2 w_2 > t$
- **Region 2 (R2)** $b + x_1 w_1 + x_2 w_2 \leq 0 \equiv x_1 w_1 + x_2 w_2 \leq t$

$$b + x_1 w_1 + x_2 w_2 = 0$$

$$x_2 = -\frac{w_1}{w_2} \cdot x_1 - \frac{1}{w_2} \cdot b$$



Linear separation

5 min

EXERCISE: 2.4

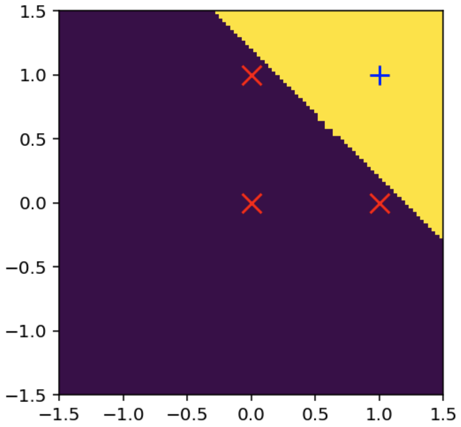
Play around with the visualization provided to get to the answer, you do not have to code anything

Linear separation

Given the previous analysis:

- Depending on the perceptron's parameters we have a different decision boundary, and this allows flexibility in our decision process.
- This boundary implicitly classifies each point from the input space: mapping a point (x1;x2) to either a 0 or 1 value.

Figure 59



linearly separable dataset

Linear separation

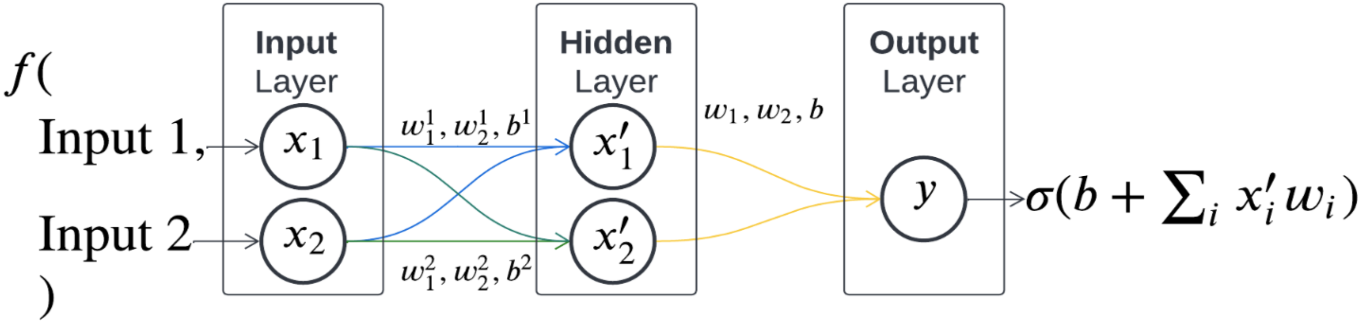
10 min

EXERCISES: FROM 2.5 & 2.6

Play around with the visualization provided to get to the answer, you do not have to code anything

Multilayer perceptron

A BNN is composed by a set of interconnected neurons, this allows us to perform difficult tasks. The same works for more complicated decisions on a perceptron:



It has the same spirit that the single perceptron had: function f which takes an input and throws an output that depend on a set of parameters

Multilayer perceptron

5 min

EXERCISE: 2.7

Play around with the visualization provided to get to the answer, you do not have to code anything.

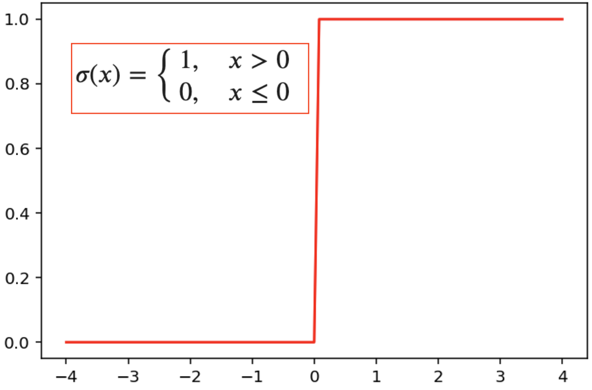
Share with others: Which accuracy were you able to get on the XOR problem?

- 0/4
- 1/4
- 2/4
- 3/4
- 4/4

Multilayer perceptron - Activation functions and neural networks

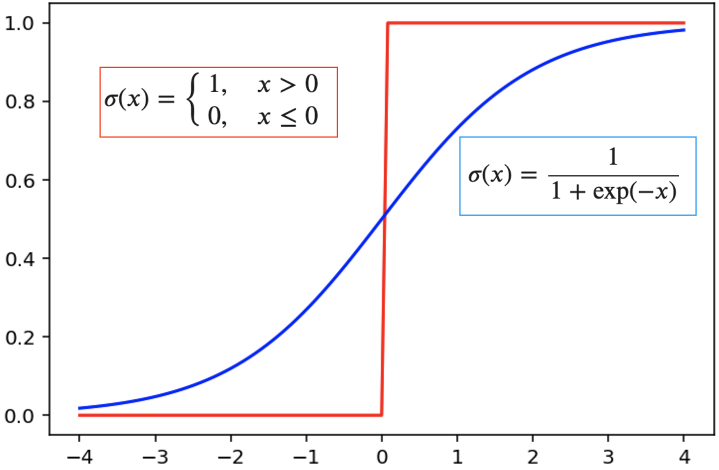
For both the single & multilayer perceptron we have used the step activation function. This presents at least a couple of challenges:

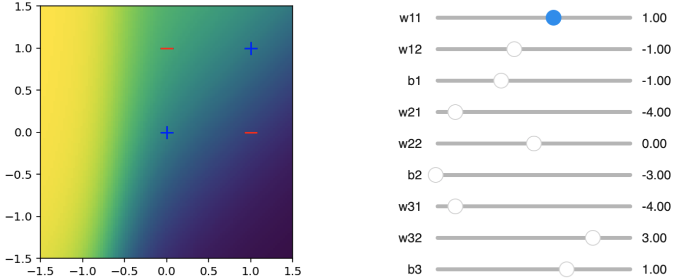
- The perceptron can drastically change with small changes in weights or bias.
- Perceptrons (having a linear activation function) only allow for linearity in the decision boundaries.



Multilayer perceptron - Activation functions and neural networks

If we use the **sigmoid activation function** instead, we will have a **sigmoid neuron**, and the output would be in the [0,1] interval, not only 0 or 1.

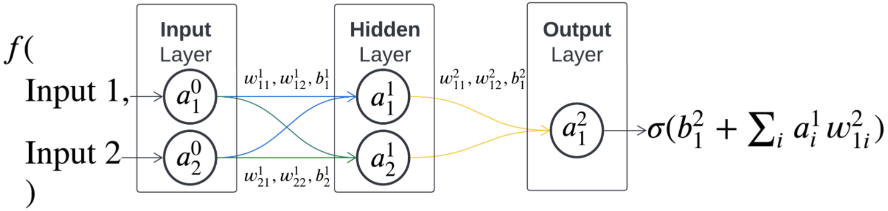




Homework: Play around with the plot and try to find the parameter combination that accurately distinguish both classes.

Multilayer perceptron - Math behind ANN: vectors and matrices

We will use the previous multilayer perceptron as an example to write down the math behind a general ANN: particularly on its hidden layer activation



$$\sum_i a_i^0 w_{1i}^1 = a_1^0 w_{11}^1 + a_2^0 w_{12}^1 = [a_1^0, a_2^0]\begin{bmatrix} w_{11}^1 \\ w_{12}^1 \end{bmatrix} = \mathbf{a}^0\mathbf{w}_1^1$$

$$\mathbf{b}^1 = [b_1^1, b_2^1]$$

$a^0 w_1^1$

$\mathbf{a}^0\mathbf{w}_1^1$

$\mathbf{a}^0\mathbf{w}_2^1$

$$\mathbf{W}^1 = [\mathbf{w}_1^1, \mathbf{w}_2^1] = \begin{bmatrix} w_{11}^1 & w_{21}^1 \\ w_{12}^1 & w_{22}^1 \end{bmatrix}$$

$$\mathbf{a}^1 = \sigma(\mathbf{a}^0\mathbf{W}^1 + \mathbf{b}^1)$$

$$= \sigma([a_1^0 w_{11}^1 + a_2^0 w_{12}^1 + b_1^1, a_1^0 w_{21}^1 + a_2^0 w_{22}^1 + b_2^1])$$

$$= ([\sigma(a_1^0 w_{11}^1 + a_2^0 w_{12}^1 + b_1^1), \sigma(a_1^0 w_{21}^1 + a_2^0 w_{22}^1 + b_2^1)])$$

Perceptron

*The student weights different factors before making a decision: the larger the weight, the more it would **cost** on the final decision.*

- If the student has to work on the next day, the student would probably want to be fresh in the morning.
- The student has an assignment but next class is three days from today.
- If the student can't catch the train back home, the student might not even get back home until next day!

$w_1 = 3, w_2 = 2$ and $w_3 = 6. \longrightarrow C = \sum_i x_i w_i = x_1 w_1 + x_2 w_2 + x_3 w_3$

If the student has to work tomorrow (x1=1), but no assignment (x2=0) and can catch the train (x3=0) $\longrightarrow C = 3$

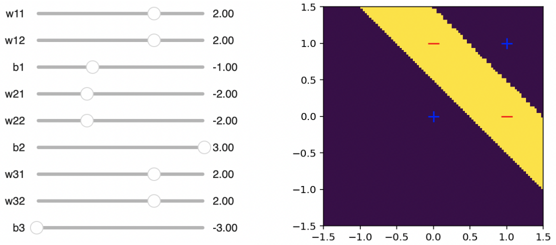
If the student has an assignment (x1=1) and works tomorrow (x2=1) but can catch the train (x3=0). $\longrightarrow C = 5$

*But those are my weights, you can have different ones and even with the **same input** get a **different cost***

Training a NN

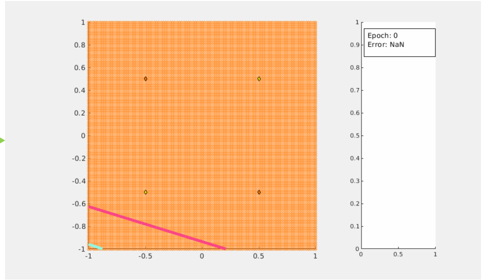
Coming up with the correct weights for the XOR problem is not an easy task: Requires to explore 9 grid values simultaneously!

Manual trial and error



Training a NN is a process in which we **learn** this parameters in a way that "best fits" the data. We learn the parameters with a **training dataset**.

Automated NN training

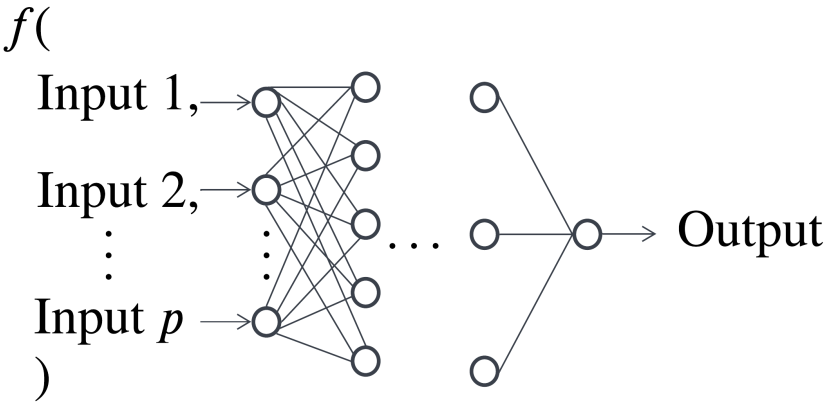


Training a NN

Remember, even complex NN are nothing but a functions that depend on parameters: they receive an input and return an output.

With our training data, we can "plug in" the features and **if parameters are well selected**, expect that the output follows the response.

$$f(\mathbf{x}) = f(x_1, x_2, \ldots, x_p) = y$$



Training a NN

The training dataset has instances of **features** and **responses**. For example, the XOR problem has the following training dataset:

FEATURES | RESPONSES

Value for x1	Value for x2	Class = XOR(x1,x2)
1	1	1
1	0	0
0	1	0
0	0	1

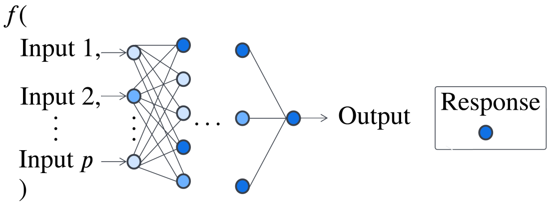
For features (x1,x2)=(1,0) we have a 0 response

The training dataset is crucial in learning which parameters from the NN actually follow the response variable

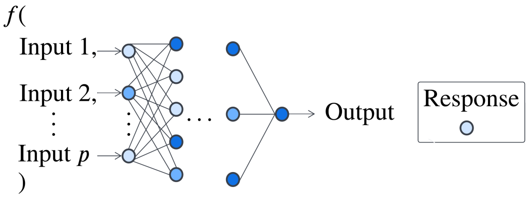
Training a NN

Let us say that we have 4 **training points**. We can see how good the NN is performing when we compare the NN output & the correct response:

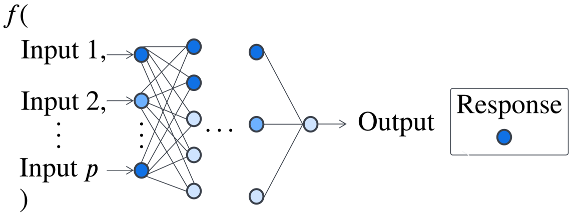
Good work NN! :)



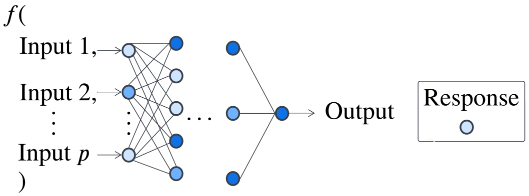
You need to learn more :(



You need to learn more :(



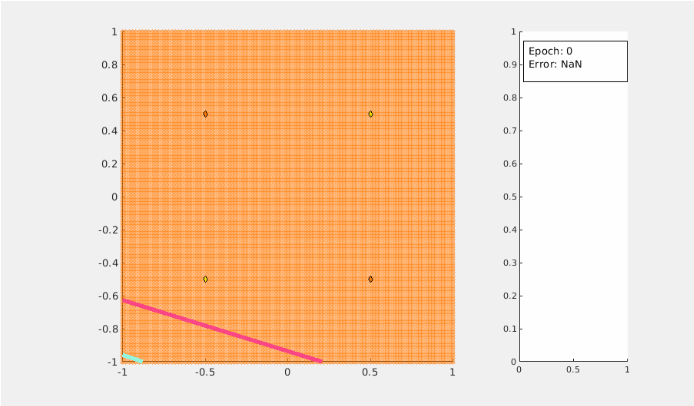
You need to learn more :(



The network error **on its current parameters** is 3/4 = 75% and its accuracy is 25%.

Training a NN

As we change the parameters in each step in a smart way to minimize the error, we can see that the network error rapidly decays (and its accuracy increases).



So we can use trained NN to **parametrize decision making**, and we do not have to find the weights and biases by ourselves.

Conclusions

- An ANN resembles the BNN behaviour: it propagates information from one artificial neuron to another when enough signal activates it.

- A (single/multiple) perceptron can be thought as a decision making process, where inputs correspond to binary information that support a given yes/no decision. The decision can be either simple (AND operator) or rather complex (XOR operator).

- Decision boundaries in the feature space are defined by a given set of parameters from an ANN. This implicitly scores each point from the input space (in {0,1} with a perceptron or [0,1] with a sigmoid neuron).

- Multiple activation functions could be used to "fire" a neuron, this changes several aspects on a given ANN (decision boundaries being one of those aspects). Two instances of activation functions are the stepwise function and the sigmoid function.

References

- Nielsen, M. A. (2015). *Neural networks and deep learning* (Vol. 25). San Francisco, CA, USA: Determination press.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). *An introduction to statistical learning* (Vol. 112, p. 18). New York: springer.
- But what is a neural network? | Chapter 1, Deep learning. Available at:
    - https://www.youtube.com/watch?v=aircAruvnKk&t=105s