Exercise 1 :

Suppose you want to use a neural network for predicting user preferences based on the data set in Table 1.

Example	Author	Thread	Length	WhereRead	UserAction
e_1	known	new	long	home	skips
e_2	unknown	new	short	work	reads
e_3	unknown	follow Up	long	work	skips
e_4	known	follow Up	long	home	skips
e_5	known	new	short	home	reads
e_6	known	follow Up	long	work	skips
e_7	unknown	follow Up	short	work	skips
e_8	unknown	new	short	work	reads
e_9	known	follow Up	long	home	skips
e_{10}	known	new	long	work	skips
e_{11}	unknown	follow Up	short	home	skips
e_{12}	known	new	long	work	skips
e_{13}	known	follow Up	short	home	reads
e_{14}	known	new	short	work	reads
e_{15}	known	new	short	home	reads
e_{16}	known	follow Up	short	work	reads
e_{17}	known	new	short	home	reads
e_{18}	unknown	new	short	work	reads
e_{19}	unknown	new	long	work	?
e_{20}	unknown	follow Up	long	home	?
e_{21}	unknown	follow Up	short	home	?

Table 1: The user preference data to be used in Exercise 1.

What neural network structure could you use, especially: what would be the input and output units?

Solution:

This is a problem of finding a numerical encoding for discrete input and output attributes. There are several possibilities:

Assuming that all attributes are binary (i.e., only can have the two values that appear in the table, and, e.g., 'Length' can not also have the value 'medium'), one can use a neural network with one input node for each input attribute, and, for each attribute, encode one of the possible values as 0, and the other as 1. For example, assuming that 'unknown', 'follow Up', 'short' and 'home' are the values encoded as 0, one would have that the input in the first example is given as (1,1,1,0). In the same way, the output attribute 'User Action' would be encoded.

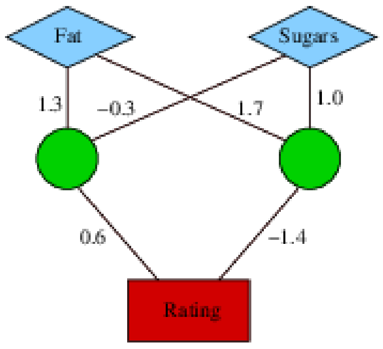
Alternatively (and this also works for attributes with more than 2 values), one can use one input node for each attribute-value combination. Here we would have the 8 different attributes Author_is_known, Author_is_unknown, ..., WhereRead_is_home, WhereRead_is_work. Then we encode the actual inputs as 0/1 values of these new attributes. In the first example, the inputs then are set to Author_is_known=1, Author_is_unknown=0, . . . , WhereRead_is_home=1, WhereRead_is_work=0.

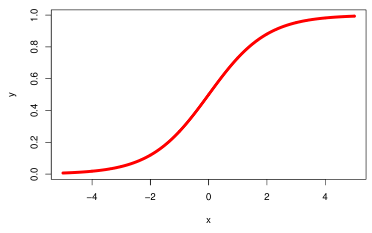
Exercise 2 :

Compute for the neural network below the *Rating* output computed for the two inputs

Fat	Sugars
1	5
0	14

The two hidden units have the sigmoid activation function. Values for this function can be either computed precisely according to the definition $\sigma(x) = 1/(1 + e^{-x})$, or you can read approximate function values off the plot on the right below. The output unit has the identity activation function, i.e. the output is just the weighted sum of the inputs.





Solution:

The computation in the neural network proceeds top-to-bottom, where each node computes its output from the input it receives from the nodes in the preceding layer.

The input nodes don't perform any computations. Their output is just the input, i.e. in the first case, the *Fat* input node outputs a 1, and the *Sugars* input node outputs a 5.

Next, the two nodes in the hidden layer perform their computations. Each node first computes the weighted sum of its input, and then applies the activation function to compute the final output.

The weighted sum of inputs is:

left hidden node: $1.3 \cdot 1 + (-0.3) \cdot 5 = -0.2$
right hidden node: $1.7 \cdot 1 + 1.0 \cdot 5 = 6.7$

Now the activation function is applied to these numbers. The function value can be approximately read off the plot, or computed precisely:

output left hidden node: $1/(1 + e^{0.2}) = 0.45$ *output left hidden node*: $1/(1 + e^{-6.7}) = 0.998$

Next, the 'Rating' output node can compute its output. The weighted input is

$0.6 \cdot 0.45 + (-1.4) \cdot 0.998 = -1.1272$.

Since the output node has the identity activation function, this is also already the output of the 'Rating' node.

Exercise 3 :

Assume that we have the following training examples:

X_1	X_2	T
1	1	1
-1	1	-1
1	-1	1
-1	-1	-1

That is, with input $X_1 = 1$ and $X_2 = -1$ we want the output 1.

Consider a perceptron (single neuron) with threshold input 1 and with initial weights $w_0 = 0$,$w_1 = 0$ and $w_2 = 0$.

Show the first two iterations of gradient descent when learning a perceptron (having the sign function as activation function) using learning rate $\alpha = 0.25$. Indicate the intermediate steps that need to be computed (e.g. the value of forward propagation, error term and how the weights are adjusted).

Solution:

First iteration:

Cases:	$(1,1,1)$	$(1,-1,1)$	$(1,1,-1)$	$(1,-1,-1)$
y	1	-1	1	-1
o	-1	-1	-1	-1
$\delta_o = y - o$	$\underline{2}$	0	2	0

$$\bar{w} := (0,0,0) + \frac{1}{4} \cdot 2 \cdot (1,1,1) = (0.5, 0.5, 0.5)$$

(note that we can focus on elements where there is some error $\delta_o \neq 0$)

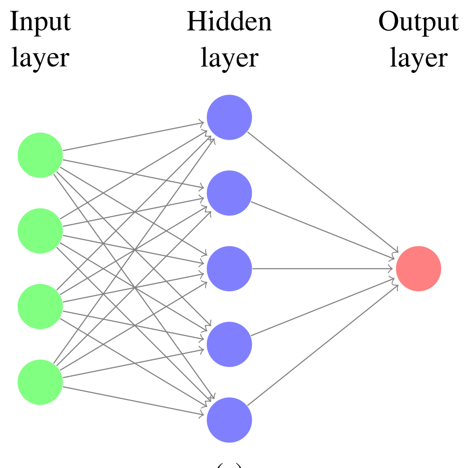
Second iteration:

Cases:	$(1,1,1)$	$(1,-1,1)$	$(1,1,-1)$	$(1,-1,-1)$
t	1	-1	1	-1
o	1	1	1	-1
$\delta_o = y - o$	0	$\underline{-2}$	0	0

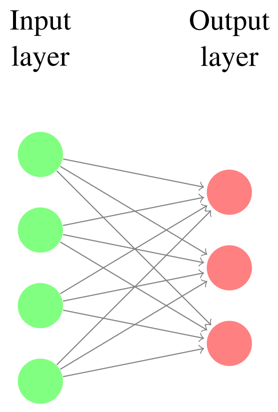
$$\bar{w} := (0.5, 0.5, 0.5) - \frac{1}{4} \cdot 2 \cdot (1,-1,1) = (0,1,0)$$

Exercise 4 :

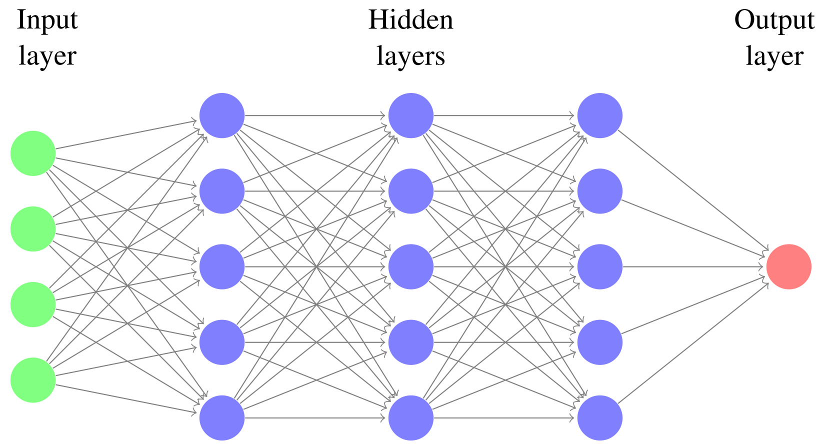
For each of the following neural networks:



(a)



(b)



(c)

indicate:

i) How many parameters need to be adjusted by the learning algorithm?

ii) How many functions are represented by the network?

iii) Can the network represent the Boolean function x_1 xor x_2? Justify your answer.

Solution:

1. The parameters are the weights which are trained for: so (a) has $4 \cdot 5 + 5 = 30$, (b) has $4 \cot 3 = 12$, and (c) has $4 \cdot 5 + 2 \cot 5 \cdot 5 + 5 = 75$.
2. The number of represented functions are the number of output neurons, so 1, 3, and 1.
3. (b) cannot represent it, as it has no hidden layers.

Exercise 5 :

Experiment with the tensorflow playground

https://playground.tensorflow.org

- Use different data sets and neural network structures (varying the number of layers and neurons). How does the different network structures affect the learning ability? Can you derive any useful insights by considering the features learned at the hidden units.
- What is the effect of varying the learning rate?

Optional: You can also take a look at the questions in this course: `https://developers.google.com/machine-learning/crash-course/introduction-to-neural-networks/playground-exer`

Exercise 6 (Optional):

Load the Iris dataset in WEKA (link) and choose the MultilayerPerceptron classifier model. Use Test options: "Use training set". Observe how the performance of the learned model (and the time needed for learning) changes when you modify the following parameters of the learning procedure:

- hidden Layers: this controls the structure of the network (use GUI:true to check).
- trainingTime: controls how many iterations are performed in the weight learning.
- learning rate: controls the stepsize in the gradient descent.

Exercise 7 :

Complete one more iteration of the back propagation algorithm for the example on slide 08.29.
