Artificial Intelligence
Exercises week 2 - ANN - **Solutions**

COMP9414

Question 1: McCulloch-Pitts' model

Consider the logic function AND with 4 inputs. Implement a McCulloch-Pitts' model that produces this result in its output. Assume $w_i = 1$.

Answer: The logic function AND with 4 inputs is represented as shown in Table 1. If we assume $w_1 = w_2 = w_3 = w_4 = 1$ then the value of $\sum w_i x_i$ is shown in the fifth column.

As the AND function is 1 (or *true*) only if all inputs are 1, we need to set up a threshold $\theta \in (3, 4))$. For instance $\theta = 3.5$.

Question 2: Forward propagation

Consider a single-layer perceptron with 2 inputs and 4 neurons with an activation function sgn or threshold logic function. Moreover, take into account inputs, weight matrix, and thresholds shown next. What would be the network output for these inputs?

$$W_{ij} = \begin{bmatrix} 0.5 & 1.0 & 0.6 \\ -0.5 & -0.7 & 0.1 \\ 0.25 & 0.1 & 0.0 \\ 0.55 & -1.1 & -0.2 \end{bmatrix} \quad X = \begin{bmatrix} 12 & -6 \end{bmatrix} \quad \theta = \begin{bmatrix} 0.5 & 0 & 1.2 & 0.2 \end{bmatrix}$$

Answer: In this case, the weight matrix dimension is 4×3 as each column includes the 4 weights from each input (X_1, X_2 and bias) to the 4 neurons. Thus the actual input X is $X = [12 \quad -6 \quad 1]$.

Table 1: Function AND with 4 inputs.

x_1	x_2	x_3	x_4	$\sum$	AND
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	2	0
0	1	0	0	1	0
0	1	0	1	2	0
0	1	1	0	2	0
0	1	1	1	3	0
1	0	0	0	1	0
1	0	0	1	2	0
1	0	1	0	2	0
1	0	1	1	3	0
1	1	0	0	2	0
1	1	0	1	3	0
1	1	1	0	3	0
1	1	1	1	4	1

Following, we need to compute $X \cdot W^t$ and apply to each value the activation function g or threshold logic function taking into account the values in θ.

$$X \cdot W^t = [0.6 \quad -1.7 \quad 2.4 \quad 13.0] \tag{1}$$

As $\theta = [0.5 \quad 0 \quad 1.2 \quad 0.2]$, therefore, $(0.6 > 0.5), (-1.7 < 0), (2.4 > 1.2), \texttt{and} \ (13.0 > 0.2)$:

$$Y(X) = g(X \cdot W^t) = [1 \quad -1 \quad 1 \quad 1]. \tag{2}$$

Question 3: Perceptron learning

Consider the training data shown in Table 2 to divide the space as shown in Fig. 1. Demonstrate the single-layer perceptron learning rule using a learning rate $\alpha = 1.0$ and initial weights $w_1 = 0$, $w_2 = -1$, and $b = -2.5$.

Table 2: Training examples and classes.

Training example	x_1	x_2	Class
a	0.0	1.0	-1
b	2.0	0.0	-1
c	1.0	1.0	1

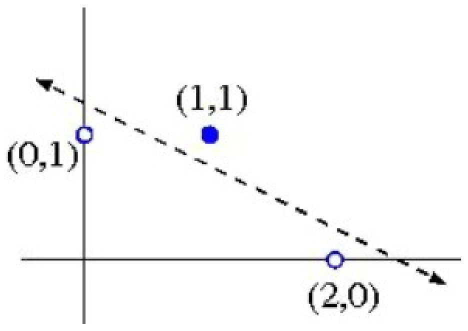


Figure 1: Space division of the training examples.

Answer: To train a single-layer perceptron we use the learning rule shown in Fig. 2. We need to be sure to round the predicted results to the closest class value available (-1 or 1 in this case).

The are only three training examples (a, b, c). We keep iterating over them until no corrections are required for the network to produce the correct output. The result of each iteration is shown in Fig 3.

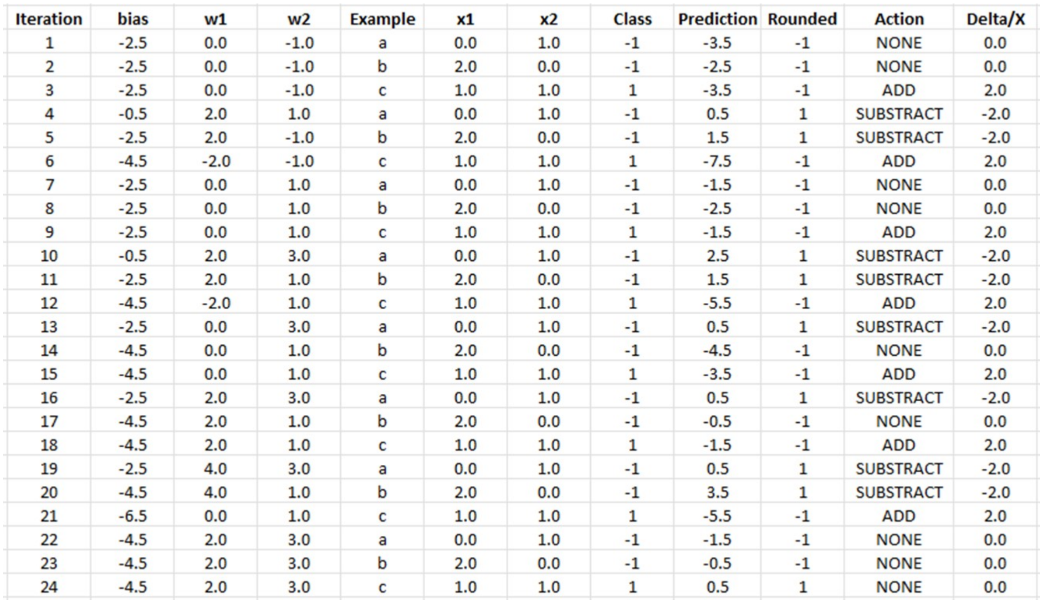
$$\hat{y}(x) = g(X \cdot W^t) \rightarrow \text{perceptron output}$$
$$y(x) \rightarrow \text{target output} \in \{-1, 1\}$$

$$\Delta W_k = \alpha(y(x_k) - \hat{y}(x_k)) \cdot x_k$$
$$W_{k+1} = W_k + \Delta W_k$$

$$\Delta W_k = 0 \rightarrow \text{if the perceptron output is correct}$$
$$\Delta W_k = +2\alpha x_k \rightarrow \text{if } y(x_k) = 1 \text{ and } \hat{y}(x_k) = -1$$
$$\Delta W_k = -2\alpha x_k \rightarrow \text{if } y(x_k) = -1 \text{ and } \hat{y}(x_k) = 1$$

$$\alpha > 0; \{x_1, ..., x_k\} \text{ input sequence}$$

Figure 2: Single-layer perceptron learning rule.



Iteration	bias	w1	w2	Example	x1	x2	Class	Prediction	Rounded	Action	Delta/X
1	-2.5	0.0	-1.0	a	0.0	1.0	-1	-3.5	-1	NONE	0.0
2	-2.5	0.0	-1.0	b	2.0	0.0	-1	-2.5	-1	NONE	0.0
3	-2.5	0.0	-1.0	c	1.0	1.0	1	-3.5	-1	ADD	2.0
4	-0.5	2.0	1.0	a	0.0	1.0	-1	0.5	1	SUBSTRACT	-2.0
5	-2.5	2.0	-1.0	b	2.0	0.0	-1	1.5	1	SUBSTRACT	-2.0
6	-4.5	-2.0	-1.0	c	1.0	1.0	1	-7.5	-1	ADD	2.0
7	-2.5	0.0	1.0	a	0.0	1.0	-1	-1.5	-1	NONE	0.0
8	-2.5	0.0	1.0	b	2.0	0.0	-1	-2.5	-1	NONE	0.0
9	-2.5	0.0	1.0	c	1.0	1.0	1	-1.5	-1	ADD	2.0
10	-0.5	2.0	3.0	a	0.0	1.0	-1	2.5	1	SUBSTRACT	-2.0
11	-2.5	2.0	1.0	b	2.0	0.0	-1	1.5	1	SUBSTRACT	-2.0
12	-4.5	-2.0	1.0	c	1.0	1.0	1	-5.5	-1	ADD	2.0
13	-2.5	0.0	3.0	a	0.0	1.0	-1	0.5	1	SUBSTRACT	-2.0
14	-4.5	0.0	1.0	b	2.0	0.0	-1	-4.5	-1	NONE	0.0
15	-4.5	0.0	1.0	c	1.0	1.0	1	-3.5	-1	ADD	2.0
16	-2.5	2.0	3.0	a	0.0	1.0	-1	0.5	1	SUBSTRACT	-2.0
17	-4.5	2.0	1.0	b	2.0	0.0	-1	-0.5	-1	NONE	0.0
18	-4.5	2.0	1.0	c	1.0	1.0	1	-1.5	-1	ADD	2.0
19	-2.5	4.0	3.0	a	0.0	1.0	-1	0.5	1	SUBSTRACT	-2.0
20	-4.5	4.0	1.0	b	2.0	0.0	-1	3.5	1	SUBSTRACT	-2.0
21	-6.5	0.0	1.0	c	1.0	1.0	1	-5.5	-1	ADD	2.0
22	-4.5	2.0	3.0	a	0.0	1.0	-1	-1.5	-1	NONE	0.0
23	-4.5	2.0	3.0	b	2.0	0.0	-1	-0.5	-1	NONE	0.0
24	-4.5	2.0	3.0	c	1.0	1.0	1	0.5	1	NONE	0.0

Figure 3: Iterations of learning rule.

Question 4: Neural desing

- What would be the MLP architecture to approximate a non-linear function of 3 inputs and 2 outputs if you have available 3,000 samples? Consider 70% data for training and 30% for validation.

  Answer: What it is known from the previous description is the following:

  $Ni = 3$
  $No = 2$
  $N_{samples} = 2,100$ as we use only 70% for training.
  $Nh = ?$

  But we also know that $Nw = (Ni + 1) \times Nh + (Nh + 1) \times No$ and $Nw < N_{samples}/10$, then:

  $210 = (3 + 1) \times Nh + (Nh + 1) \times 2$
  $210 = 4Nh + 2Nh + 2$
  $208 = 6Nh$
  $Nh = 208/6$
  $Nh = 34.6 \rightarrow 34$ as the number of neurons in the hidden layer is an integer. It might be 35, however, with 34 we keep true the relationship $Nw < N_{samples}/10$.

- What would be the MLP architecture to approximate a non-linear function of 6 inputs and 3 outputs if you have available 100 samples? Consider 80% data for training and 20% for validation.

  Answer: What it is known from the previous description is the following:

  $Ni = 6$
  $No = 3$
  $N_{samples} = 80$ as we use only 80% for training.
  $Nh = ?$

  But we also know that $Nw = (Ni + 1) \times Nh + (Nh + 1) \times No$ and $Nw < N_{samples}/10$, then:

$8 = (6 + 1) \times Nh + (Nh + 1) \times 3$
$8 = 7Nh + 3Nh + 3$
$5 = 10Nh$
$Nh = 5/10$
$Nh = 0.5 \rightarrow$ it is possible that there is not enough data to correctly train the neural model as the number of neurons in the hidden layer would be less than 1. Although this depends on the complexity of the underlying function, an alternative solution would be to capture more data, if possible.