Introduction to Machine Learning WiSe-2023
Assignment: 04

Group Number - 22

4th January, 2024

1 Gradient Descent

1.1 Name one difference between the perceptron training rule and the gradient descent method.

Solution:
One key difference between the perceptron training rule and the gradient descent method is that the perceptron training rule updates the weights based on the error of each individual training instance, while gradient descent considers the overall error by computing the gradient of the entire dataset.

1.2 Name the difference in the algorithm between batch gradient descent and incremental gradient descent.

Solution:
The main difference in the algorithm between batch gradient descent and incremental gradient descent (also known as stochastic gradient descent) is the way they update the model parameters. Batch gradient descent computes the gradient of the entire dataset and updates the parameters once per epoch, while incremental gradient descent updates the parameters after each training instance or a small subset (mini-batch) of instances.

2 Perceptron Learning

Given two perceptrons $y_0()$ and $y_1()$ defined by heaviside$(\sum_{j=0}^{P} w_j x_j)$ as usual with identical weights except that $w_0 = 0$ for $y_0()$ but $w_0 = 1$ for $y_1()$. Is one of $y_0()$ and $y_1()$ more general than the other, and if yes, which one (or both)? Explain your answer.

Solution:

1. $y_0()$: here, $w_0 = 0$ and $y_0 = \text{Heaviside}(\sum w_j x_j)$
2. $y_1()$: here, $w_0 = 1$ and $y_1 = \text{Heaviside}(\sum w_j x_j)$

Heaviside(x) returns 1 if $x \geq 0$ and returns 0 otherwise.

The y_0 will be 1 if $\left(\sum(w_j x_j) + w_0\right) \geq 0$, but as $w_0 = 0$, its contribution is none. Thus, y_0 is determined solely by other weights.

The y_1 will be 1 if $\left(\sum(w_j x_j) + w_0\right) \geq 0$. Here, $w_0 = 1$, and its contribution is non-zero and positive.

$y_1()$ is influenced by all weights, including w_0, and will always output 1 when the sum of the weighted inputs is greater than or equal to 1.

$y_0()$ is more general in the sense that its output is not influenced by the weight associated with the bias term (w_0).

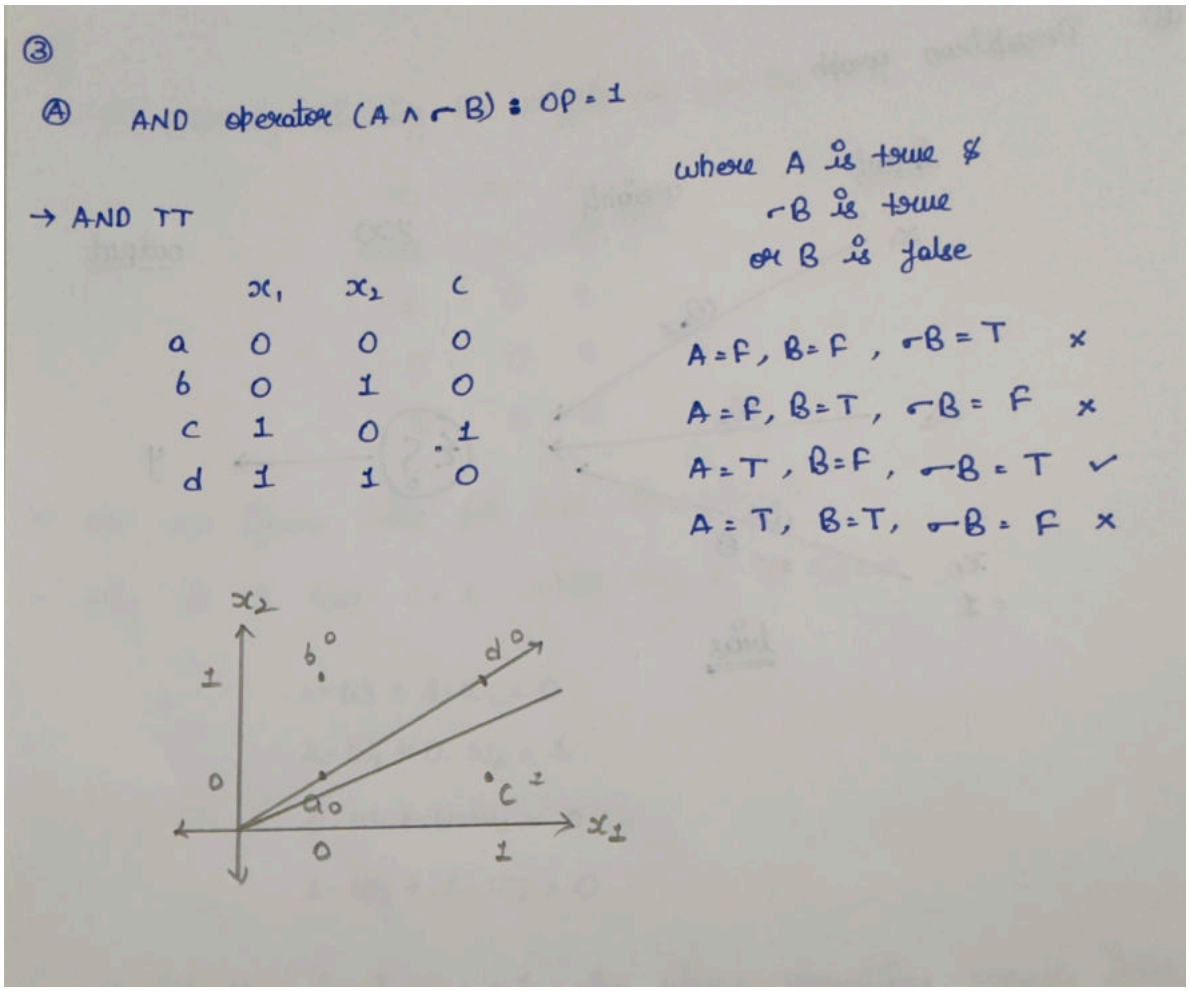
$y_1()$ is less general in this specific scenario as it always considers the contribution of w_0, which is fixed at 1.

3 Perceptron Learning

In this exercise, you design a single perceptron with two inputs x_1 and x_2. This perceptron shall implement the boolean formula $A \wedge \neg B$ with a suitable function $y(x_1, x_2)$. Use the values 0 for false and 1 for true.

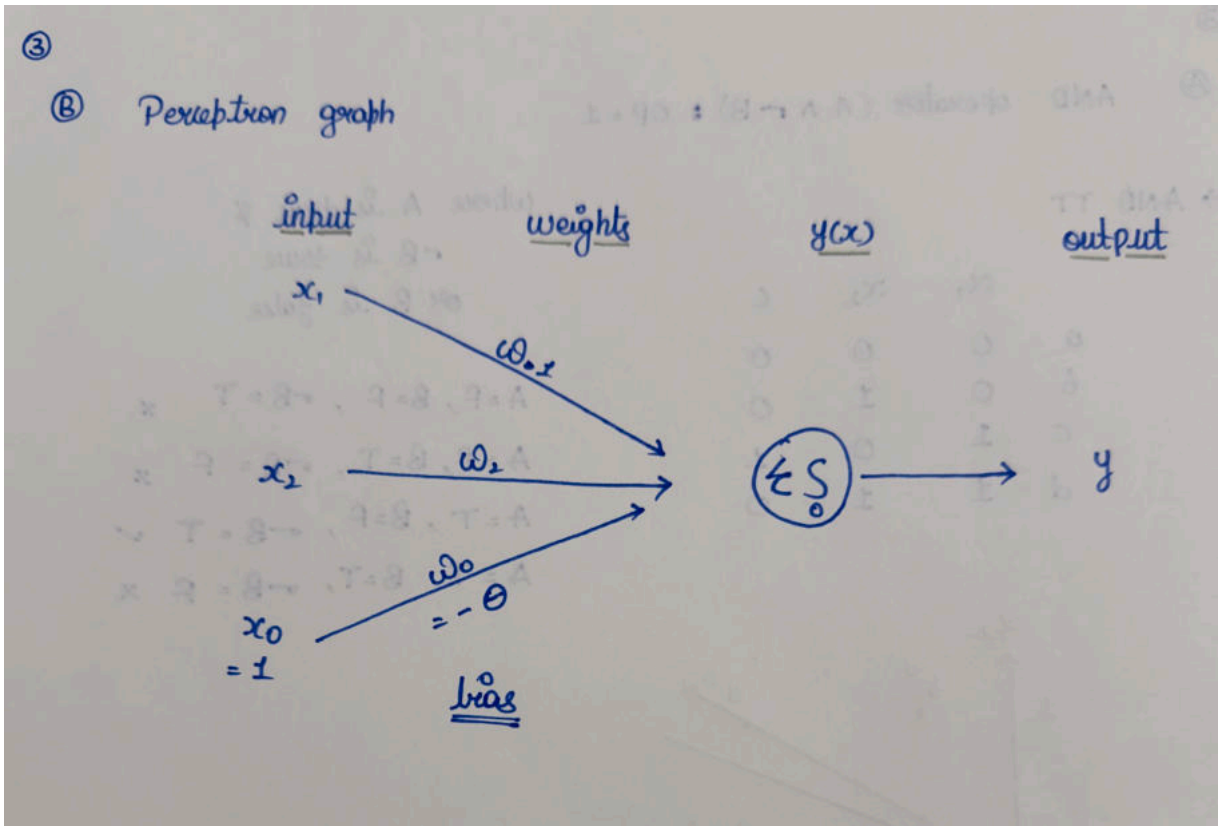
3.1 Draw all possible examples and a suitable decision boundary in a coordinate system.

Solution:



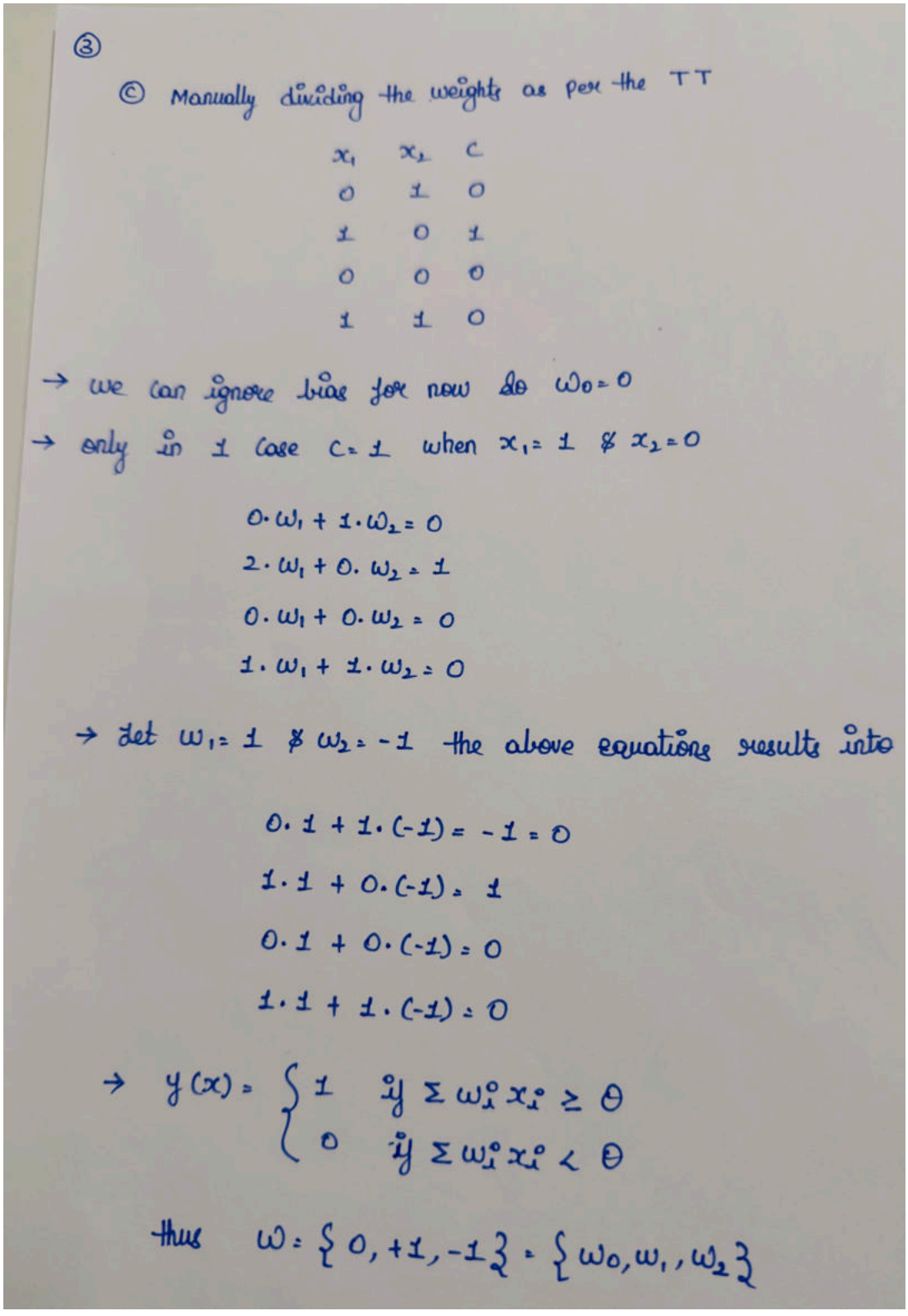
3.2 Draw the graph of the perceptron. The schematic must include x1, x2, and all model weights.

Solution:



3.3 Manually determine a set of suitable weights w = (w0, w1, w2) from your drawings.

Solution:



③

© Manually dividing the weights as per the TT

x_1	x_2	C
0	1	0
1	0	1
0	0	0
1	1	0

→ we can ignore bias for now so $w_0 = 0$

→ only in 1 case $C = 1$ when $x_1 = 1$ & $x_2 = 0$

$$0 \cdot w_1 + 1 \cdot w_2 = 0$$
$$2 \cdot w_1 + 0 \cdot w_2 = 1$$
$$0 \cdot w_1 + 0 \cdot w_2 = 0$$
$$1 \cdot w_1 + 1 \cdot w_2 = 0$$

→ let $w_1 = 1$ & $w_2 = -1$ the above equations results into

$$0 \cdot 1 + 1 \cdot (-1) = -1 = 0$$
$$1 \cdot 1 + 0 \cdot (-1) = 1$$
$$0 \cdot 1 + 0 \cdot (-1) = 0$$
$$1 \cdot 1 + 1 \cdot (-1) = 0$$

→ $$y(x) = \begin{cases} 1 & \text{if } \Sigma w_i x_i \geq \theta \\ 0 & \text{if } \Sigma w_i x_i < \theta \end{cases}$$

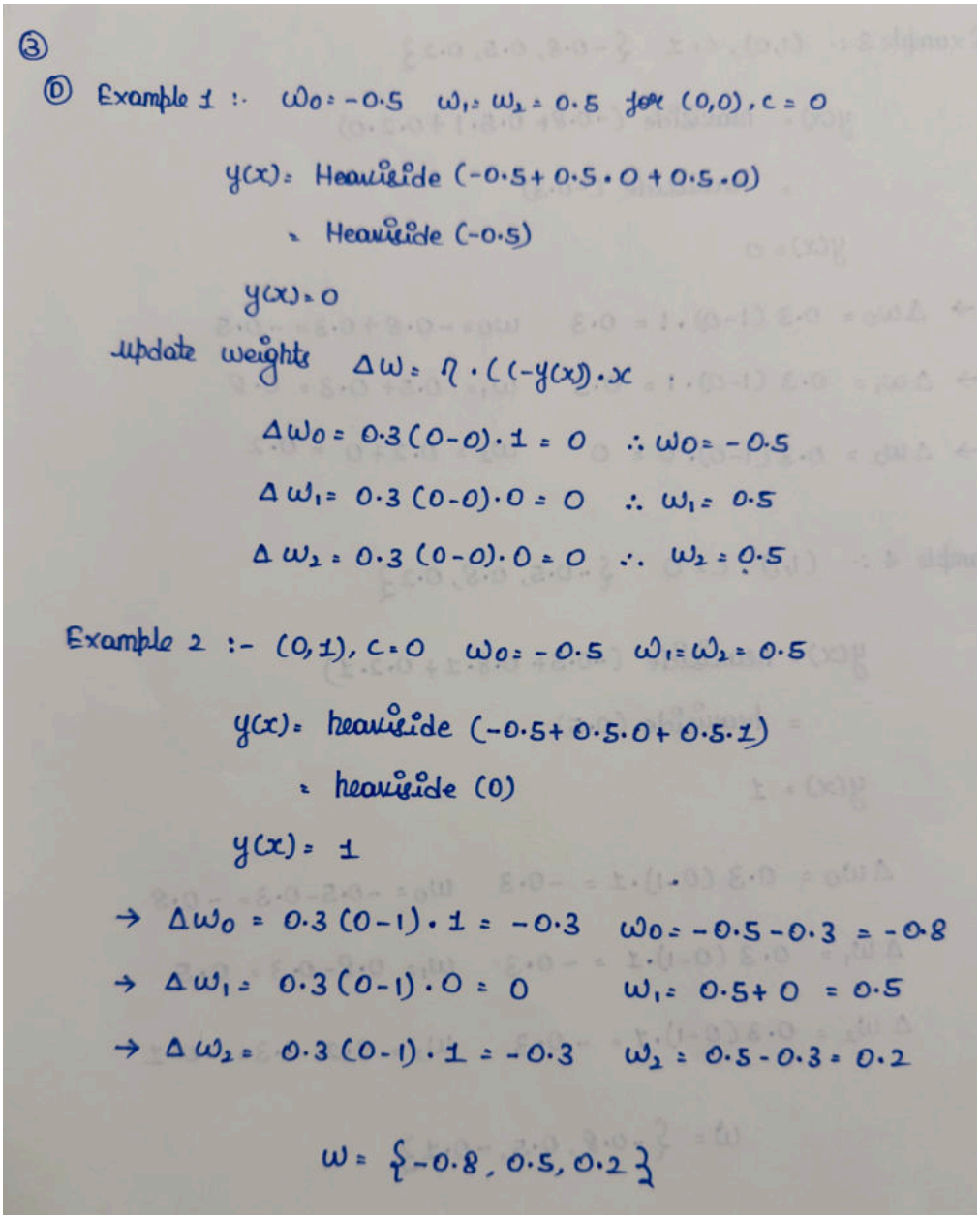
thus $w = \{0, +1, -1\} = \{w_0, w_1, w_2\}$

3.4 Now determine w using the Perceptron Training algorithm (PT). Use a learning rate η of 0.3 and initialize the weights with $w_0 = -0.5$ and $w_1 = w_2 = 0.5$. Instead of selecting examples randomly, use the following examples in the given order (stop after those four examples):

x_1	x_2	c
0	0	0
0	1	0
1	0	1
1	1	0

Draw the decision boundary after every weight update into the coordinate system of (a).

Solution:



③

ⓓ Example 1 :- $w_0 = -0.5$ $w_1 = w_2 = 0.5$ for $(0,0), c = 0$

$y(x) =$ Heaviside $(-0.5 + 0.5 \cdot 0 + 0.5 \cdot 0)$

$=$ Heaviside (-0.5)

$y(x) = 0$

update weights $\Delta w = \eta \cdot ((-y(x)) \cdot x$

$\Delta w_0 = 0.3(0-0) \cdot 1 = 0 \quad \therefore w_0 = -0.5$

$\Delta w_1 = 0.3(0-0) \cdot 0 = 0 \quad \therefore w_1 = 0.5$

$\Delta w_2 = 0.3(0-0) \cdot 0 = 0 \quad \therefore w_2 = 0.5$

Example 2 :- $(0,1), c = 0$ $w_0 = -0.5$ $w_1 = w_2 = 0.5$

$y(x) =$ heaviside $(-0.5 + 0.5 \cdot 0 + 0.5 \cdot 1)$

$=$ heaviside (0)

$y(x) = 1$

→ $\Delta w_0 = 0.3(0-1) \cdot 1 = -0.3$ $w_0 = -0.5 - 0.3 = -0.8$

→ $\Delta w_1 = 0.3(0-1) \cdot 0 = 0$ $w_1 = 0.5 + 0 = 0.5$

→ $\Delta w_2 = 0.3(0-1) \cdot 1 = -0.3$ $w_2 = 0.5 - 0.3 = 0.2$

$w = \{-0.8, 0.5, 0.2\}$

Example 3 :- (1,0), c = 1 $\{-0.8, 0.5, 0.2\}$

$y(x)$ = heaviside $(-0.8 + 0.5 \cdot 1 + 0.2 \cdot 0)$

= heaviside (-0.3)

$y(x) = 0$

→ $\Delta w_0 = 0.3\,(1-0) \cdot 1 = 0.3$ $\quad w_0 = -0.8 + 0.3 = -0.5$

→ $\Delta w_1 = 0.3\,(1-0) \cdot 1 = 0.3$ $\quad w_1 = 0.5 + 0.3 = 0.8$

→ $\Delta w_2 = 0.3\,(1-0) \cdot 0 = 0$ $\quad w_2 = 0.2 + 0 = 0.2$

Example 4 :- (1,1), c = 0 $\{-0.5, 0.8, 0.2\}$

$y(x)$ = heaviside $(-0.5 + 0.8 \cdot 1 + 0.2 \cdot 1)$

= heaviside (0.5)

$y(x) = 1$

$\Delta w_0 = 0.3\,(0-1) \cdot 1 = -0.3$ $\quad w_0 = -0.5 - 0.3 = -0.8$

$\Delta w_1 = 0.3\,(0-1) \cdot 1 = -0.3$ $\quad w_1 = 0.8 - 0.3 = 0.5$

$\Delta w_2 = 0.3\,(0-1) \cdot 1 = -0.3$ $\quad w_2 = 0.2 - 0.3 = -0.1$

$w = \{-0.8, 0.5, -0.1\}$

3.5 Briefly describe one effect of changing the learning rate η on the learning progress.

Solution:

The learning rate (η) is a crucial hyperparameter in training machine learning models, including neural networks. One effect of changing the learning rate on the learning progress is:

Effect on Convergence:

- **High Learning Rate (η):** A high learning rate can cause the model to converge quickly, but it may also lead to overshooting the optimal weights. In such cases, the model might oscillate or diverge, making it difficult to reach the minimum of the loss function. It may skip over the optimal solution, preventing convergence.
- **Low Learning Rate (η):** A low learning rate slows down the convergence process. While it may lead to more stable updates, it requires more iterations for the model to reach the optimal weights. If the learning rate is too low, the model may get stuck in local minima or take a long time to converge.

Selecting an appropriate learning rate is essential to achieve effective convergence during training. It often involves experimentation and tuning to find the right balance between convergence speed and stability.

4 Perceptron Learning

Why can the boolean formula A XOR B not be learned by a single perceptron? Justify your answer with a drawing.

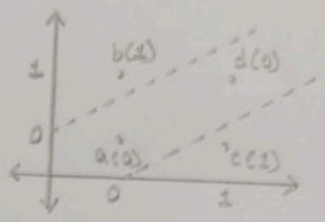
Solution:

④ XOR unsolvable by single layer Perceptron

→ XOR TT

	x_1	x_2	C
a	0	0	0
b	0	1	1
c	1	0	1
d	1	1	0

→ Graph representation



→ Here are two sets which are not linearly separable

→ Consider XOR data points in 2D space

- (0,0) & (1,1) has op (0)
- (0,1) & (1,0) has op (1)

→ No straight line can separate them into two correct op classes

→ Single layer perceptron only learns linearly separable functions, where the decision boundary is a straight line or a hyperplane

→ It requires more complex model like a multi layer perceptron with atleast one hidden layer

→ Hidden layer will allow the model to learn a non-linear decision boundaries. A hidden layer has a non-linear activation function

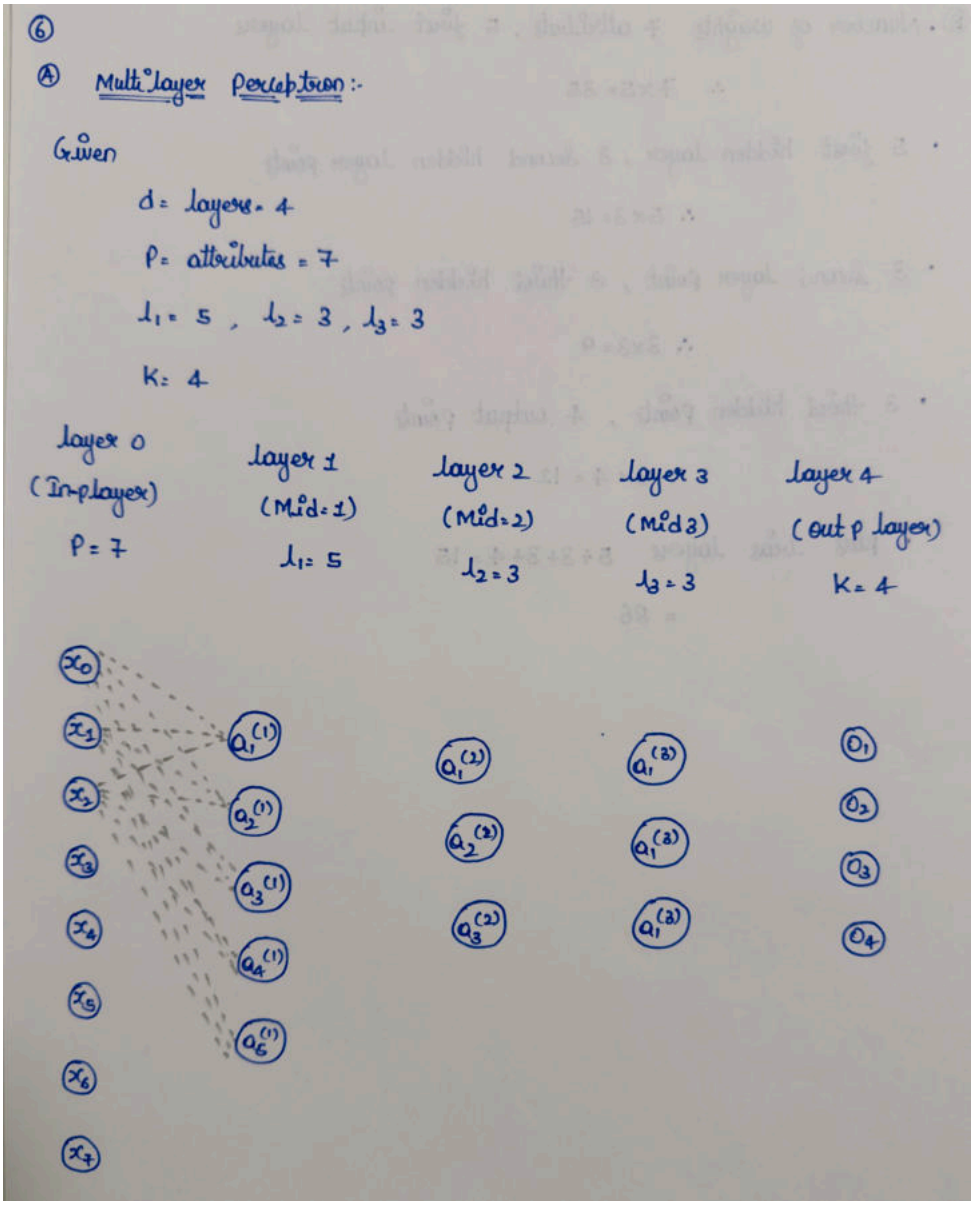
5 Multilayer Perceptron

6 Parameters of the Multilayer Perceptrons

In this exercise, you analyze the number of weights (parameters) of multilayer perceptrons. We use the notation from the lecture (e.g., slide ML:IV-104), where multilayer perceptrons have d layers, p attributes, hidden layer i with l_i units, and an output layer with k units.

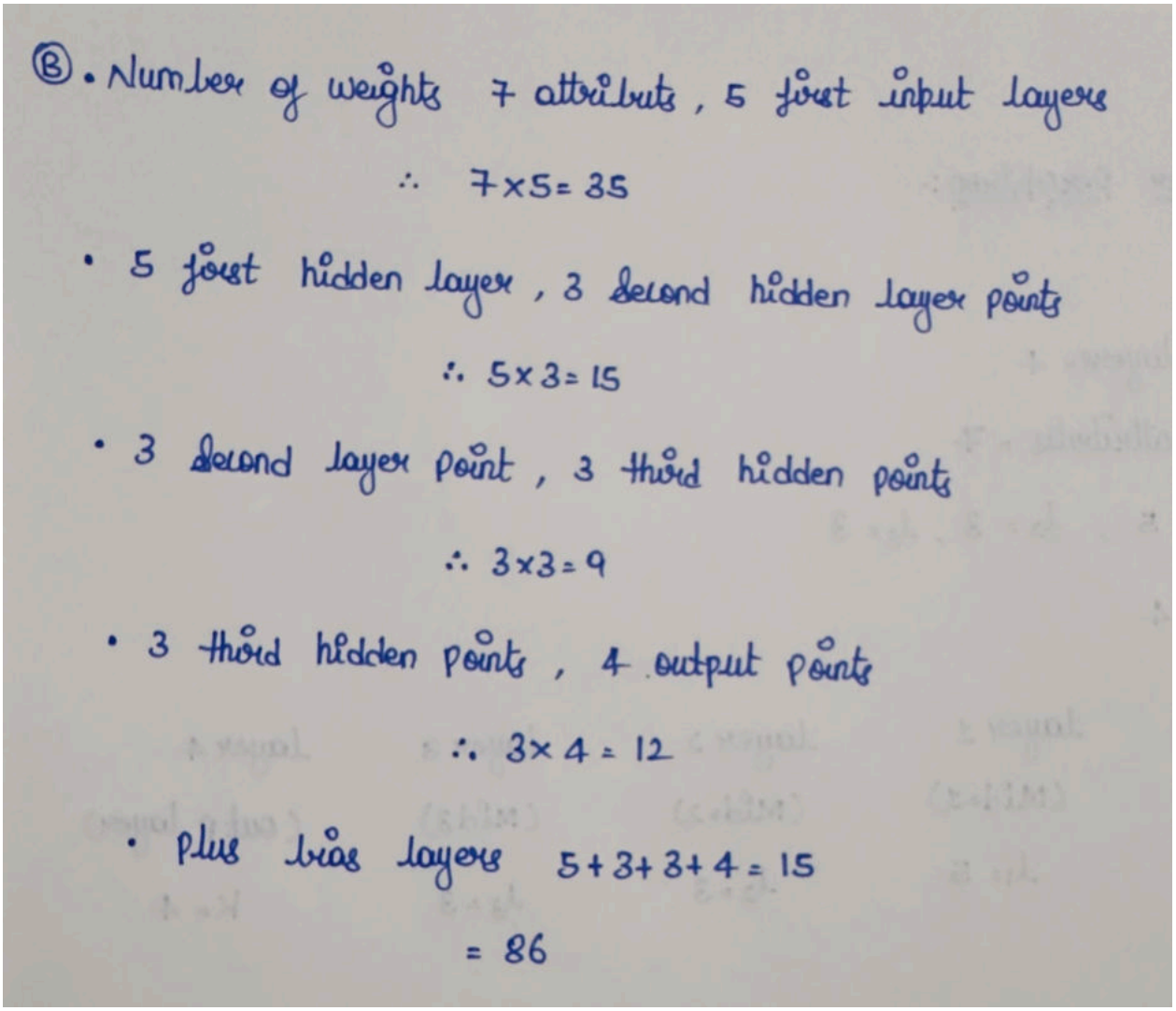
6.1 Let $d = 4$, $p = 7$, $l_1 = 5$, $l_2 = 3$, $l_3 = 3$, and $k = 4$. Draw the graph of the multilayer perceptron.

Solution:



6.2 Calculate the number of weights in the multilayer perceptron of (a).

Solution:



7 Multi-Class Classification with Neural Networks

In this exercise, you will implement a two-layer perceptron for predicting binary argument quality. Download and use these files from Moodle (the tsv files are the same as in the last sheet):

Solution: Implementaton and the solution with the plot added to the attachment. (Note: Please close the plot while running the program to obtained all the results.)