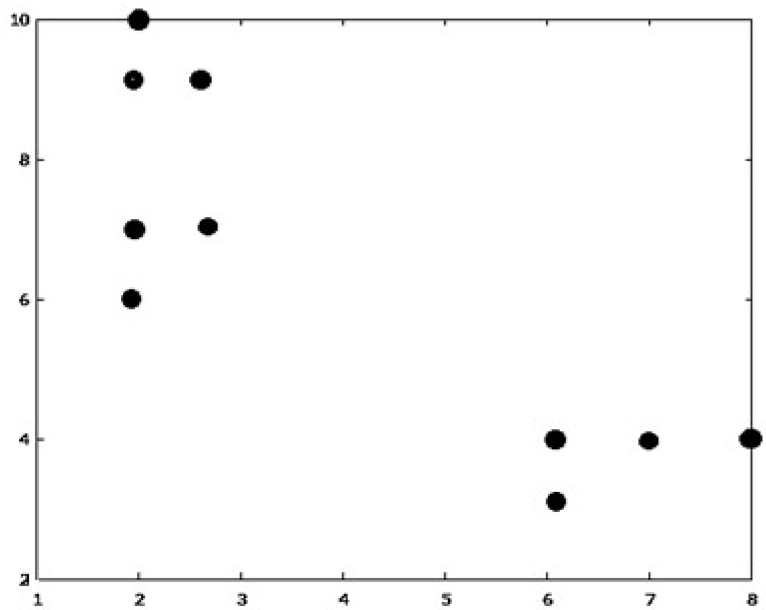
Q No 1. [Clustering].

Part b) **[6 Points]**

For the following points use the k-means clustering algorithm to partition the given points in 3 clusters assuming that the initial cluster centers are: **C_1 = (8, 9), C_2 = (6, 3.5), C_3 = (7.5, 4)**

Points: A_1 = (2, 10), A_2 = (2, 9), A_3 = (2, 7), A_4 = (2, 6), A_5 = (3, 7), A_6 = (3, 9), A_7 = (6, 3), A_8 = (6, 4), A_9 = (7, 4), A_{10} = (8, 4)

For your convenience, the plot of the points is also shown below



Clearly show all distance calculations that are computed by the k-means clustering algorithm in each iteration and show the final cluster centers on a similar figure as shown above.

Iteration 1. Using the Euclidian Distance

Point\Cluster Center	C_1 = (8, 9)	C_2 = (6, 3.5),	C_3 = (7.5, 4)
A_1 = (2, 10)	6.08	7.63	8.14
A_2 = (2, 9)	6.00	6.80	7.43
A_3 = (2, 7)	6.32	5.32	6.26
A_4 = (2, 6)	6.71	4.72	5.85
A_5 = (3, 7)	5.39	4.61	5.41
A_6 = (3, 9)	5.00	6.26	6.73
A_7 = (6, 3)	6.32	0.50	1.80
A_8 = (6, 4)	5.39	0.50	1.50
A_9 = (7, 4)	5.10	1.12	0.50
A_{10} = (8, 4)	5.00	2.06	0.50

New Centers

C_1 = (2.33, 9.33), C_2 = (3.8, 5.4), C_3 = (7.5, 4)

Iteration 2

Point\Cluster Center	C_1 = (8, 9)	C_2 = (6, 3.5),	C_3 = (7.5, 4)
A_1 = (2, 10)	0.75	4.94	8.14
A_2 = (2, 9)	0.47	4.02	7.43
A_3 = (2, 7)	2.35	2.41	6.26
A_4 = (2, 6)	3.35	1.90	5.85
A_5 = (3, 7)	2.42	1.79	5.41
A_6 = (3, 9)	0.75	3.69	6.73
A_7 = (6, 3)	7.32	3.26	1.80
A_8 = (6, 4)	6.47	2.61	1.50
A_5 = (7, 4)	7.09	3.49	0.50
A_{10} = (8, 4)	7.78	4.43	0.50

New Centers

C_1 = (2.25, 8.75), C_2 = (2.5, 6.5), C_3 = (6.75, 3.75)

Iteration 3:

Point\Cluster Center	C_1 = (8, 9)	C_2 = (6, 3.5),	C_3 = (7.5, 4)
A_1 = (2, 10)	1.27	3.54	7.85
A_2 = (2, 9)	0.35	2.55	7.08
A_3 = (2, 7)	1.77	0.71	5.76
A_4 = (2, 6)	2.76	0.71	5.26
A_5 = (3, 7)	1.90	0.71	4.96
A_6 = (3, 9)	0.79	2.55	6.45
A_7 = (6, 3)	6.86	4.95	1.06
A_8 = (6, 4)	6.05	4.30	0.79
A_5 = (7, 4)	6.72	5.15	0.35
A_{10} = (8, 4)	7.46	6.04	1.27

New Centers

$C_1 = (2.33, 9.33)$, $C_2 = (2.33, 6.67)$, $C_3 = (6.75, 3.75)$

Iteration 4:

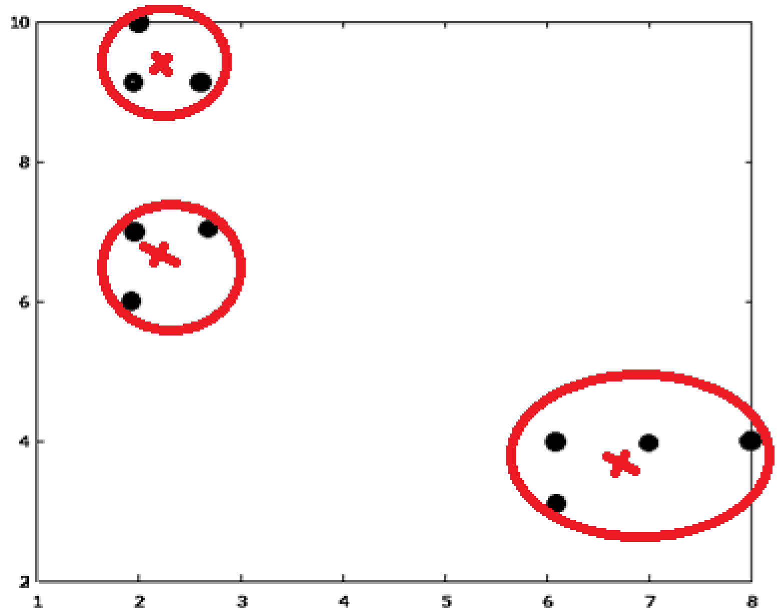
Point\Cluster Center	$C_1 = (8, 9)$	$C_2 = (6, 3.5)$,	$C_3 = (7.5, 4)$
$A_1 = (2, 10)$	0.75	3.35	7.85
$A_2 = (2, 9)$	0.47	2.36	7.08
$A_3 = (2, 7)$	2.36	0.47	5.76
$A_4 = (2, 6)$	3.35	0.75	5.26
$A_5 = (3, 7)$	2.43	0.75	4.96
$A_6 = (3, 9)$	0.75	2.43	6.45
$A_7 = (6, 3)$	7.32	5.19	1.06
$A_8 = (6, 4)$	6.47	4.53	0.79
$A_5 = (7, 4)$	7.09	5.37	0.35
$A_{10} = (8, 4)$	7.78	6.26	1.27

New Centers

$C_1 = (2.33, 9.33)$, $C_2 = (2.33, 6.67)$, $C_3 = (6.75, 3.75)$

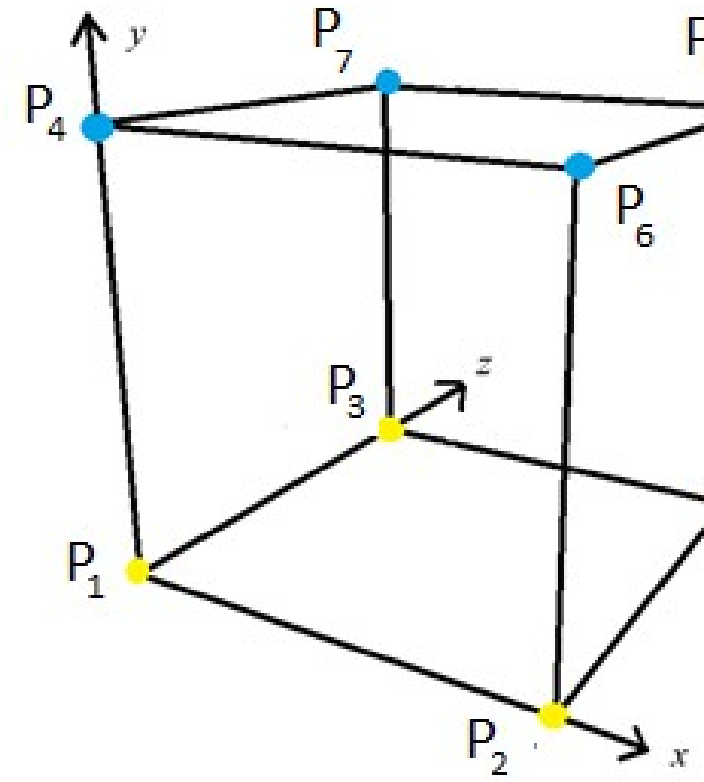
No Change in centers so we stop

Following are the clusters found



Q No 2. [Perceptron Learning] **[10 Points]**

You are given a 2-class, 3-dimensional data set. The data is plotted in the graph below. Yellow and blue dots represent negative class (-1) and positive class (1) respectively. Basically, the samples/points represent the corners of a unit cube. The point P_1 is at the centre of the coordinate plane i.e. (0, 0, 0)



Using a single Perceptron, learn a linear model for this data that correctly classifies all the samples/points. You must start with the initial weights **[-2 2 3 2]** and use the points in order i.e. P_1 then P_2 … P_8 during the learning process. Further, the first weight i.e. -2 is that of the bias term the bias term is always 1 and learning rate is also 1.

Show all the working that you did to calculate the values of the weights. The activation function g(x) is given below:

$$g(x) = \begin{cases} 1, \wedge x \geq 0 \\ -1, \wedge x < 0 \end{cases}$$

SOLUTION

Iteration No 1

With Weights [-2 2 3 2]

Example	Computed Output	Weight
P_1 (0, 0, 0)	x = -2*1 + 2*0 + 3*0 + 2 * 0 = -1 and hence g(x) = -1 CORRECT	NO CHANGE IN WEIGHTS
P_2 (1, 0, 0)	x = -2*1 + 2*1 + 3*0 + 2 * 0 = 0 and hence g(x) = 1 INCORRECT	NEW WEIGHTS [-2 2 3 2] - [1 1 0 0] = [-3 1 3 2]

P_3 (0, 0, 1)	x = -3*1 + 1*0 + 3*0 + 2 * 1 = -1 and hence g(x) = -1 CORRECT	NO CHANGE IN WEIGHTS
P_4 (0, 1, 0)	x = -3*1 + 1*0 + 3*1 + 2 * 0 = 0 and hence g(x) = 1 CORRECT	NO CHANGE IN WEIGHTS
P_5 (1, 0, 1)	x = -3*1 + 1*1 + 3*0 + 2 * 1 = 0 and hence g(x) = 1 CORRECT	NO CHANGE IN WEIGHTS
P_6 (1, 1, 0)	x = -3*1 + 1*1 + 3*1 + 2 * 0 = 1 and hence g(x) = 1 CORRECT	NO CHANGE IN WEIGHTS
P_7 (0, 1, 1)	x = -3*1 + 1*0 + 3*1 + 2 * 1 = 2 and hence g(x) = 1 CORRECT	NO CHANGE IN WEIGHTS
P_8 (1, 1, 1)	x = -3*1 + 1*1 + 3*1 + 2 * 1 = 3 and hence g(x) = 1 CORRECT	NO CHANGE IN WEIGHTS

Iteration No 2

With Weights [-3 1 3 2]

Example	Computed Output	Weight
P_1 (0, 0, 0)	x = -3*1 + 1*0 + 3*0 + 2 * 0 = -3 and hence g(x) = -1 CORRECT	NO CHANGE IN WEIGHTS
P_2 (1, 0, 0)	x = -3*1 + 1*1 + 3*0 + 2 * 0 = -2 and hence g(x) = -1 CORRECT	NO CHANGE IN WEIGHTS
P_3 (0, 0, 1)	x = -3*1 + 1*0 + 3*0 + 2 * 1 = -1 and hence g(x) = -1 CORRECT	NO CHANGE IN WEIGHTS
P_4 (0, 1, 0)	x = -3*1 + 1*0 + 3*1 + 2 * 0 = 0 and hence g(x) = 1 CORRECT	NO CHANGE IN WEIGHTS
P_5 (1, 0, 1)	x = -3*1 + 1*1 + 3*0 + 2 * 1 = 0 and hence g(x) = 1 CORRECT	NO CHANGE IN WEIGHTS
P_6 (1, 1, 0)	x = -3*1 + 1*1 + 3*1 + 2 * 0 = 1 and hence g(x) = 1 CORRECT	NO CHANGE IN WEIGHTS
P_7 (0, 1, 1)	x = -3*1 + 1*0 + 3*1 + 2 * 1 = 2 and hence g(x) = 1 CORRECT	NO CHANGE IN WEIGHTS
P_8 (1, 1, 1)	x = -3*1 + 1*1 + 3*1 + 2 * 1 = 3 and hence g(x) = 1 CORRECT	NO CHANGE IN WEIGHTS

NO CHANGE IN WEIGHTS IN THIS ITERATION HANCE FINAL WEIGHTS ARE

[-3 1 3 2]

Q No 4. [Local Search] **[5 Points]**

In this question we are going to pose the subset sum problem of described in first question as an optimization problem and then use Hill climbing strategy (i.e. a local search algorithm) to solve it.

Once again assume that for a set having n elements, a solution is coded using a bit string of length n with a bit being set to 1 if the element is part of the subset and 0 otherwise. Further, assume that the optimality of a solution is computed using 1/ (|**S** - **∑** | + 1) where S is the required value of sum and ∑ is sum of the subset and | x | represents absolute value of x.

A simple operator to generate a new solution from an existing solution can be defined as follows

NEW_SOLUTION(X) = FLIP A BIT IN THE SOLUTION X

This is equivalent to including an element in the subset or excluding an already chosen element from the subset. It is obvious that for a set of size **n** we can generate **n** new solutions from an existing solution.

Use the above operator for generating new solutions along with the hill climbing search strategy (also known as local search) to find a solution for the following subset sum problem.

Find a subset of the set {2, 3, 4, 8, 16} having sum **17**. Take the solution **00000** as the starting solution in your local search**.**

You must show all intermediate steps in the form of the following table. For each iteration show all intermediate solutions considered/generated and the solution selected at that iteration.

SOLUTION

Iteration No	**Intermediate Solutions**	**Selected Solution**
1	10000(fitness = 1/16) 01000(fitness = 1/15) 00100(fitness = 1/14) 00010(fitness = 1/10) 00001(fitness = 1/2)	00001
2	10001(fitness = 1/2) 01001(fitness = 1/3) 00101(fitness = 1/4) 00011(fitness = 1/8) 00000(fitness = 1/17)	**No Better Solution**

WE STOP HERE AND THE FINAL SOLUTION IS **00001 corresponding to the subset {16}**

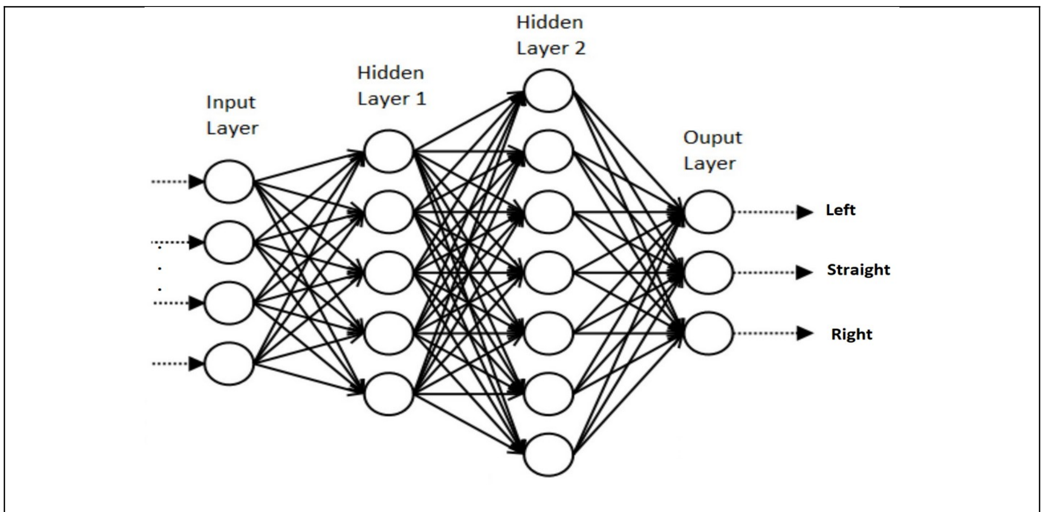
Q No 5. [Neural Networks]

Inspired by the movie WALL-E a teacher at FAST-LHR is designing a line following robot. You might search/google Internet to see various line following robots but don't spend much time surfing the Internet as this is a limited time Exam.

Being aware of the representation and learning power of a feed-forward artificial neural network, he decided to use such a neural network with two hidden layers of neurons for creating the main controller/brain of the robot. The initial prototype of the robot will only follow smooth lines i.e. lines without any sharp turns.

Input to the controller/brain of this robot will be a grey-scale camera image of dimensions **N x N**. The number of neurons in the first hidden layer will be **H1** and the number of neurons in the second hidden layer will be **H2** whereas there will be **3** neurons in the output layer with each output being **between 0 and 1** specifying the direction in which the robot need to turn. The robot will be turned slightly towards left, right or will go straight depending upon the largest value of the output neuron **Left, Straight or Right** respectively.

Such a neural network is depicted in the figure below



In this question we will assume that **N** is 4526, **H1** is 17 and **H2** is 45.

Part a) [Neural Network Weights] **[2 Points]**

Specify the number of weights needed to create such a neural network. Assume that each neuron in this network has a bias term as well. Further if we assume that each weight is a double precision number represented using 8-bytes, approximately how many kilo bytes will be needed to store this neural network on some storage.

Total weights = 348240457

Storage = 2720628.57KB or 2785923656 BYTES

Part b) [Neural Network Computations] [3 Points]

Assuming that the camera is generating 30 frames/Images per second and the neural network will be used to make a decision for each one of the frames, how many floating point multiplications and additions will be computed per second if this network is used for making decision.

Total weights excluding bias = 348240392 total weights = 348240457
Total multiplications = 348240392 * 30 = 10447211760 ---------> 1 mark

Total weights = 348240457
Total Additions = 348240457 * 30 = 10447213710 ---------> 1 mark

Total computations = 20894425470 ---------> 1 mark