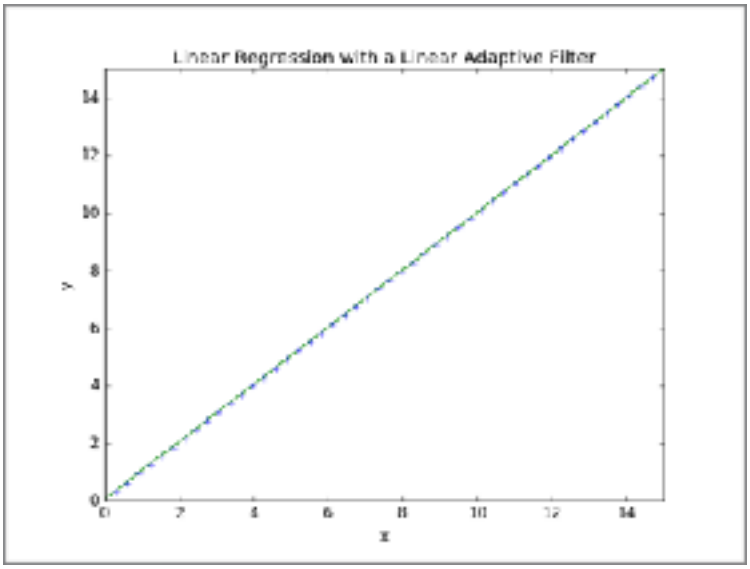
SIMPLE PERCEPTRON AND LINEAR ADAPTIVE FILTERS

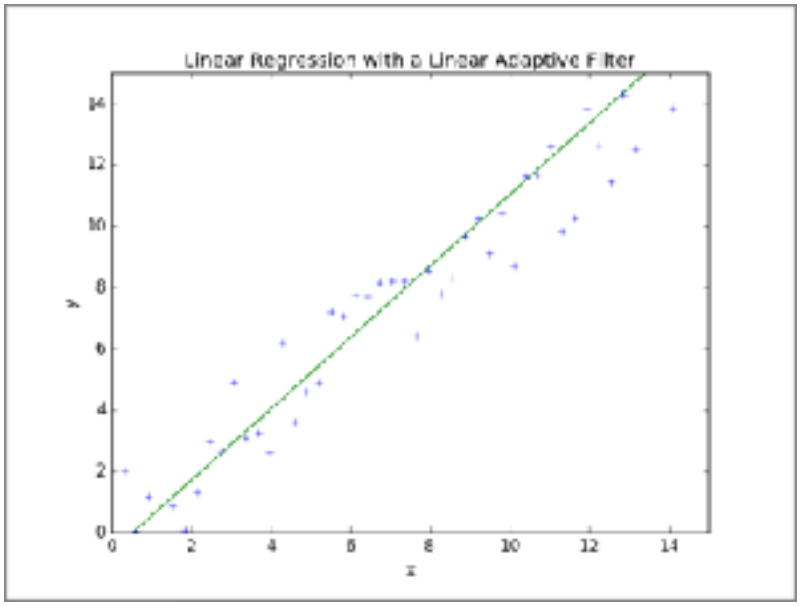
Manas Gaur, Ph.D, Wright State University
U00856829

1. Answers a: An inductive reasoning approach is being followed by linear regression. In linear regression model all the data points in the data sets are fitted to a linear line using a linear activation function with a gradient. The assumption that is used by linear regression model is that points adhere to some linear rule (linearly dependent) but if we consider some random set of points that this assumptions gets falsified many number of time s as a result we get an approximation to curve fitting. Also, if the data points are multi-dimensional then it is impossible to fit them to a linear line.

Ideal set of data points following y=mx + c rule.



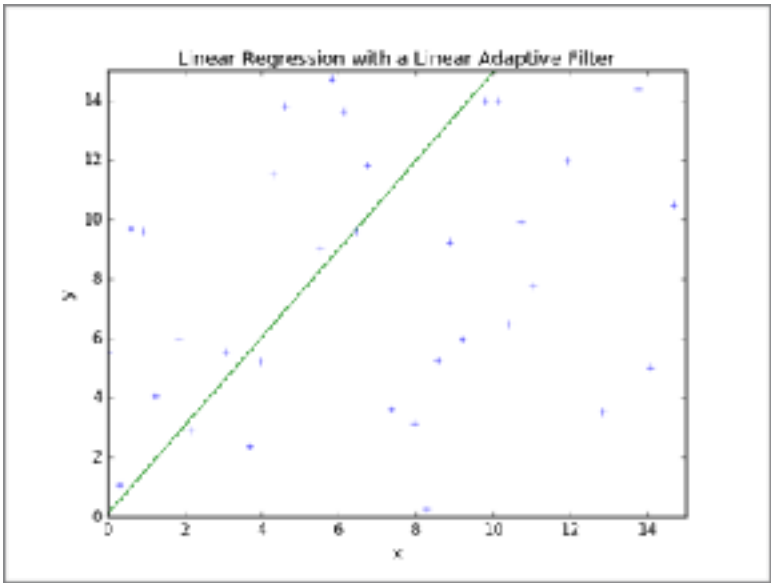
Data points with small randomness



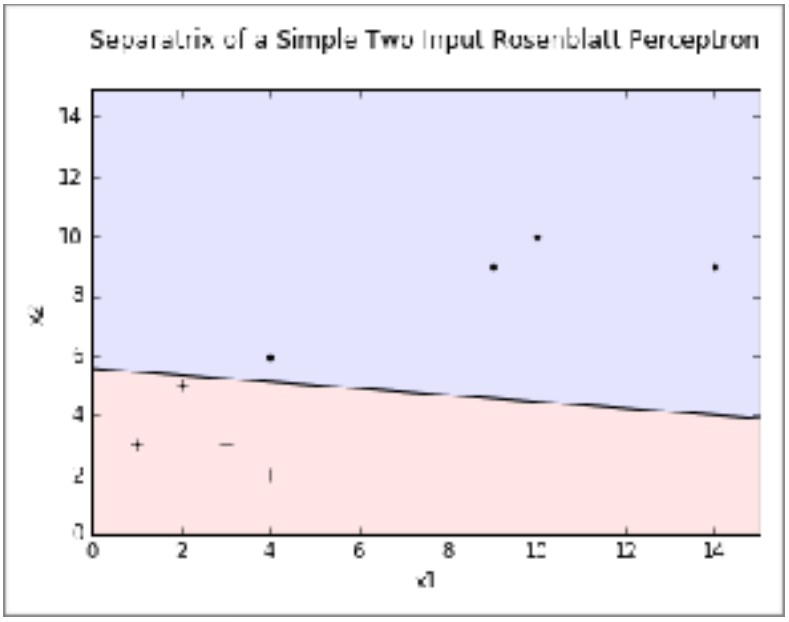
SIMPLE PERCEPTRON AND LINEAR ADAPTIVE FILTERS

Manas Gaur, Ph.D, Wright State University
U00856829

By adding more random samples to the data, it becomes almost impossible to fit the data to the line.



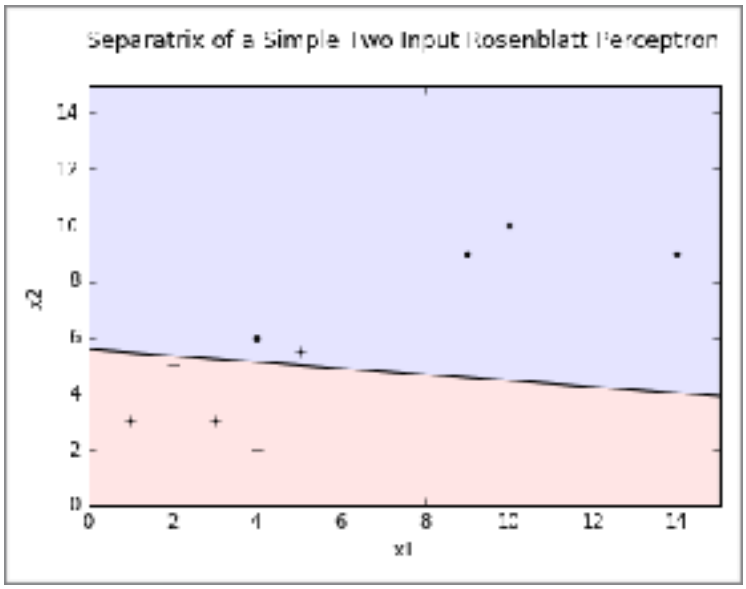
1. Answer b. A Rosenblatt perceptron is a single neuron two class classifier which can classify those data points which are linear separable. If the data points are not linearly separable (which is generally the case in real time problems) rosenblatt perceptron don't perform the job. Moreover, it can classify those data points which can be classified in to two classes. It the data sets required 4 or more classes for classification than they have to relay on multi-layer perceptron that is multiple rosenblatt perceptron needs to be included. This again comes with an exception that the data points needs to be linearly classified.



SIMPLE PERCEPTRON AND LINEAR ADAPTIVE FILTERS

Manas Gaur, Ph.D, Wright State University
U00856829

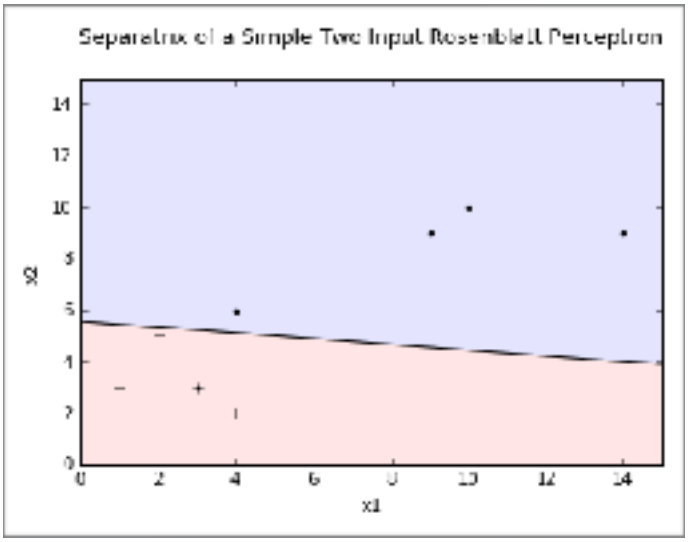
In the above figure , the rosenblatt preceptron can easily segregate the data points in to two classes. But if we increase the randomness of new data points it becomes difficult for the separatrix of the classifier to easily separate the data points into two classes.



2. Answer : Hebb's rule or Hebbian Learning depends on synchronous firing of neurons in said time period. the learning is heavily dependent on input , output and learning parameter. That being said, the learning process does not incorporate the error term which can improve the learning. This error term responsible for supervising the learning process. Since the error term is not present in Hebbian learning, we call it unsupervised learning. On the contrary, the general Delta rule depends on input, learning parameter and the error term. This error term is generated by the calculating the difference between the prediction and the actual output. This error terms assist in providing the necessary direction the learning process. Since the learning is supervised by the error term, we term learning by Delta rule as supervised learning.
3. Answer 3 a : Rosenblatt perceptron separate the data points into two classes. It can be seen in the graphical representation below :

SIMPLE PERCEPTRON AND LINEAR ADAPTIVE FILTERS

Manas Gaur, Ph.D, Wright State University
U00856829

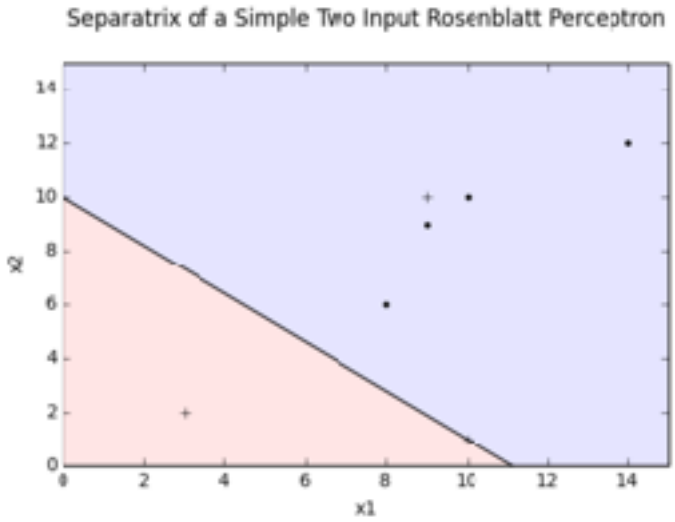



```
Class A input pair [2.0, 5.0] belongs to filter class 1.0
Class A input pair [1.0, 3.0] belongs to filter class 1.0
Class A input pair [4.0, 2.0] belongs to filter class 1.0
Class A input pair [3.0, 3.0] belongs to filter class 1.0
Class B input pair [10.0, 10.0] belongs to filter class -1.0
Class B input pair [4.0, 6.0] belongs to filter class -1.0
Class B input pair [14.0, 9.0] belongs to filter class -1.0
Class B input pair [9.0, 9.0] belongs to filter class -1.0
```

Rosenblatt fails to classify the data point into two classes. This is because, rosenblatt simply throws away points to either classes.

SIMPLE PERCEPTRON AND LINEAR ADAPTIVE FILTERS

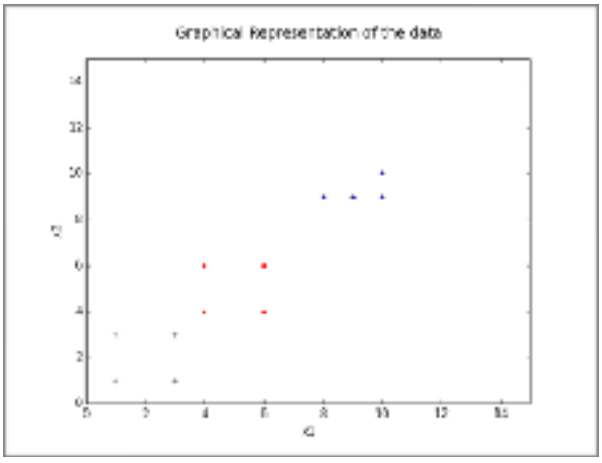
Manas Gaur, Ph.D, Wright State University
U00856829




```
Class A input pair [10.0, 1.0]     belongs to filter class 0.0
Class A input pair [3.0, 2.0]      belongs to filter class -1.0
Class A input pair [-2.0, -5.0]    belongs to filter class -1.0
Class A input pair [9.0, 10.0]     belongs to filter class 1.0

Class B input pair [10.0, 10.0]    belongs to filter class 1.0
Class B input pair [8.0, 6.0]      belongs to filter class 1.0
Class B input pair [14.0, 12.0]    belongs to filter class 1.0
Class B input pair [9.0, 9.0]      belongs to filter class 1.0
```

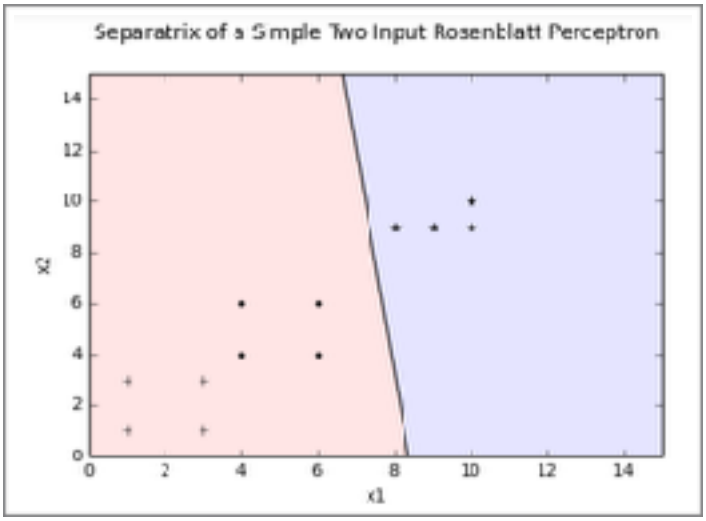
Answer 3 b:



In the above figure, we show a graphical representation of data points belonging to three classes. We subsequently will show how rosenblatt perceptron with two neurons can classify data points belonging to three classes. “+” is Class A, “.” is Class B and “*” is Class C.

SIMPLE PERCEPTRON AND LINEAR ADAPTIVE FILTERS

Manas Gaur, Ph.D, Wright State University
U00856829



In the above figure, we show how one neuron rosenblatt perceptron fails to classify the data points belonging to 2 classes.

```
Class A pair [1.0, 1.0] is filter class NEGATIVE
Class A pair [1.0, 3.0] is filter class NEGATIVE
Class A pair [3.0, 1.0] is filter class NEGATIVE
Class A pair [3.0, 3.0] is filter class NEGATIVE

Class B pair [4.0, 4.0] is filter class NEGATIVE
Class B pair [4.0, 6.0] is filter class NEGATIVE
Class B pair [6.0, 4.0] is filter class NEGATIVE
Class B pair [6.0, 6.0] is filter class NEGATIVE
Class C pair [10.0, 10.0] is filter class POSITIVE
Class C pair [10.0, 9.0] is filter class POSITIVE
Class C pair [8.0, 9.0] is filter class POSITIVE
Class C pair [9.0, 9.0] is filter class POSITIVE
 Neuron         Class
1       Class_A or Class_B
-1      Class_C
bias = 0.501044055357   weight_1 = -0.059845204947   weight_2 = -0.0688014522439
```

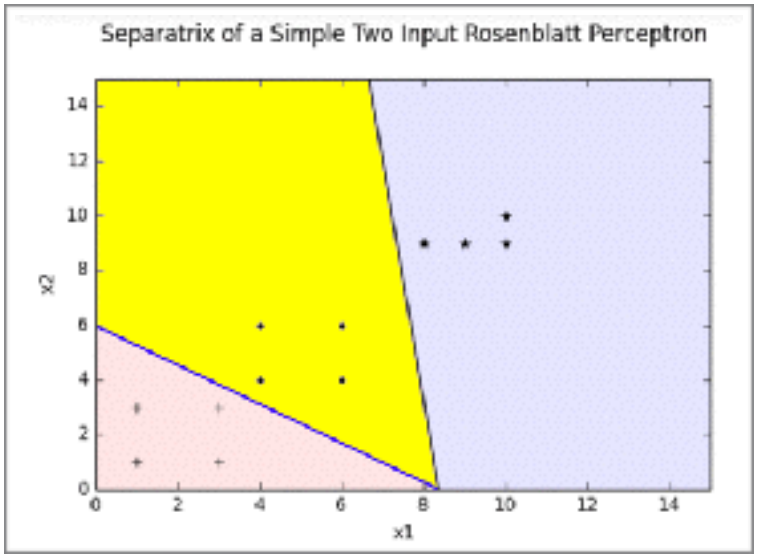
Above snap of the results obtained after modifying Notebook3.ipynb shows one neurons based classification of data points belonging to three classes. The neuron takes Class A and Class B as one class and Class C as another class.
In the above figure we show two separatrix representing two neurons. First neuron classify Class A and Class B whereas second neuron classify Class B and Class C. In order to achieve above classification neurons were configured as follows :

Neuron 1	Neuron 2	Class Type
1	-1	Class A

SIMPLE PERCEPTRON AND LINEAR ADAPTIVE FILTERS

Manas Gaur, Ph.D, Wright State University
U00856829

Neuron 1	Neuron 2	Class Type
-1	-1	Class B
-1	1	Class C



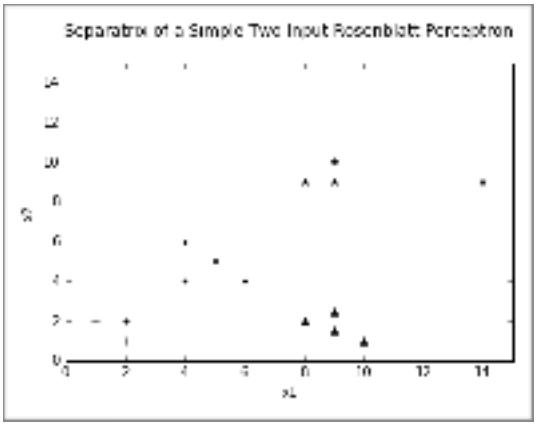

```
Class A pair [1.0, 1.0] is filter class [1.0,-1.0]
Class A pair [1.0, 3.0] is filter class [1.0,-1.0]
Class A pair [3.0, 1.0] is filter class [1.0,-1.0]
Class A pair [3.0, 3.0] is filter class [1.0,-1.0]

Class B pair [4.0, 4.0] is filter class [-1.0,-1.0]
Class B pair [4.0, 6.0] is filter class [-1.0,-1.0]
Class B pair [6.0, 4.0] is filter class [-1.0,-1.0]
Class B pair [6.0, 6.0] is filter class [-1.0,-1.0]
Class C pair [10.0, 10.0] is filter class [-1.0,1.0]
Class C pair [10.0, 9.0] is filter class [-1.0,1.0]
Class C pair [9.0, 9.0] is filter class [-1.0,1.0]
```

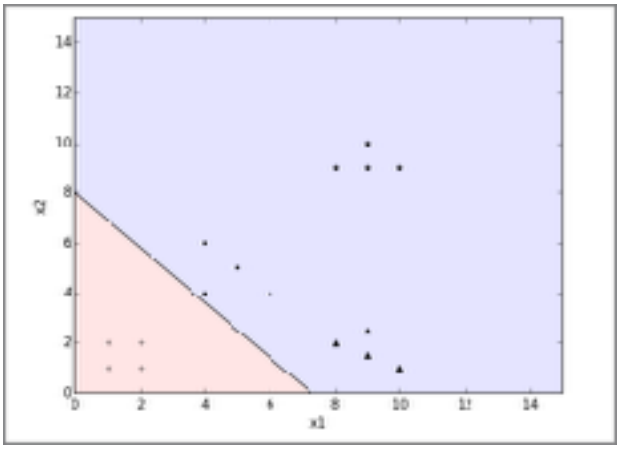
SIMPLE PERCEPTRON AND LINEAR ADAPTIVE FILTERS

Manas Gaur, Ph.D, Wright State University
U00856829

Answer 3 c:



In the above figure, we graphically represent a set of data points belonging to four classes. We subsequently show that rosenblatt perceptron fails to classify the data points. "." is class B, "+" is class A, "Δ" is class D and "*" is class C.

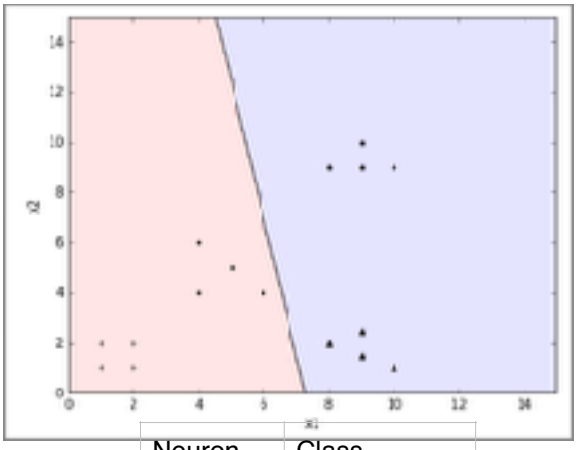


In the above figure there is one separatrix separating Class A from set Class B, Class C and Class D as another set. In order words, the rosenblatt perceptron considers Class A as one set and Class B, Class C and Class D as another set.

Neuron	Class
1	Class A
-1	Class B,C,D

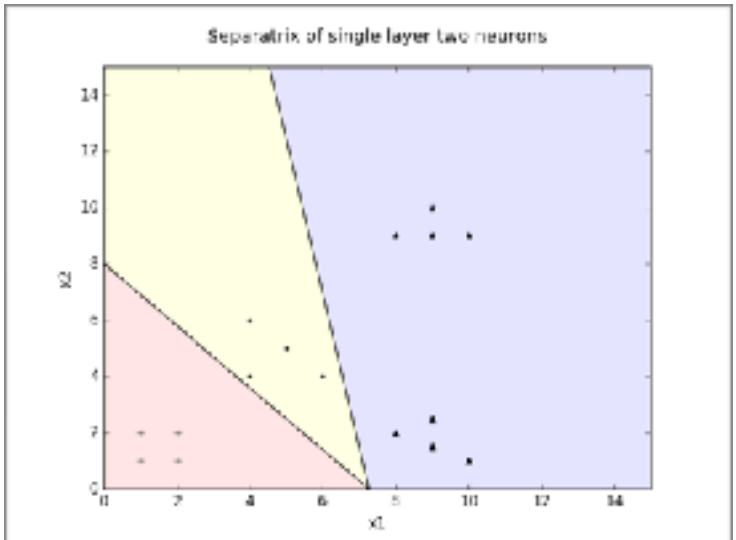
SIMPLE PERCEPTRON AND LINEAR ADAPTIVE FILTERS

Manas Gaur, Ph.D, Wright State University
U00856829



Neuron	Class
1	Class A,B
-1	Class C,D

In the above figure, single separatrix classifies the data points taking Class A,B in one group and Class C,D in another group.



SIMPLE PERCEPTRON AND LINEAR ADAPTIVE FILTERS

Manas Gaur, Ph.D, Wright State University
U00856829

Neuron 1	Neuron 2	Class
1	-1	A
-1	-1	B
-1	1	C & D

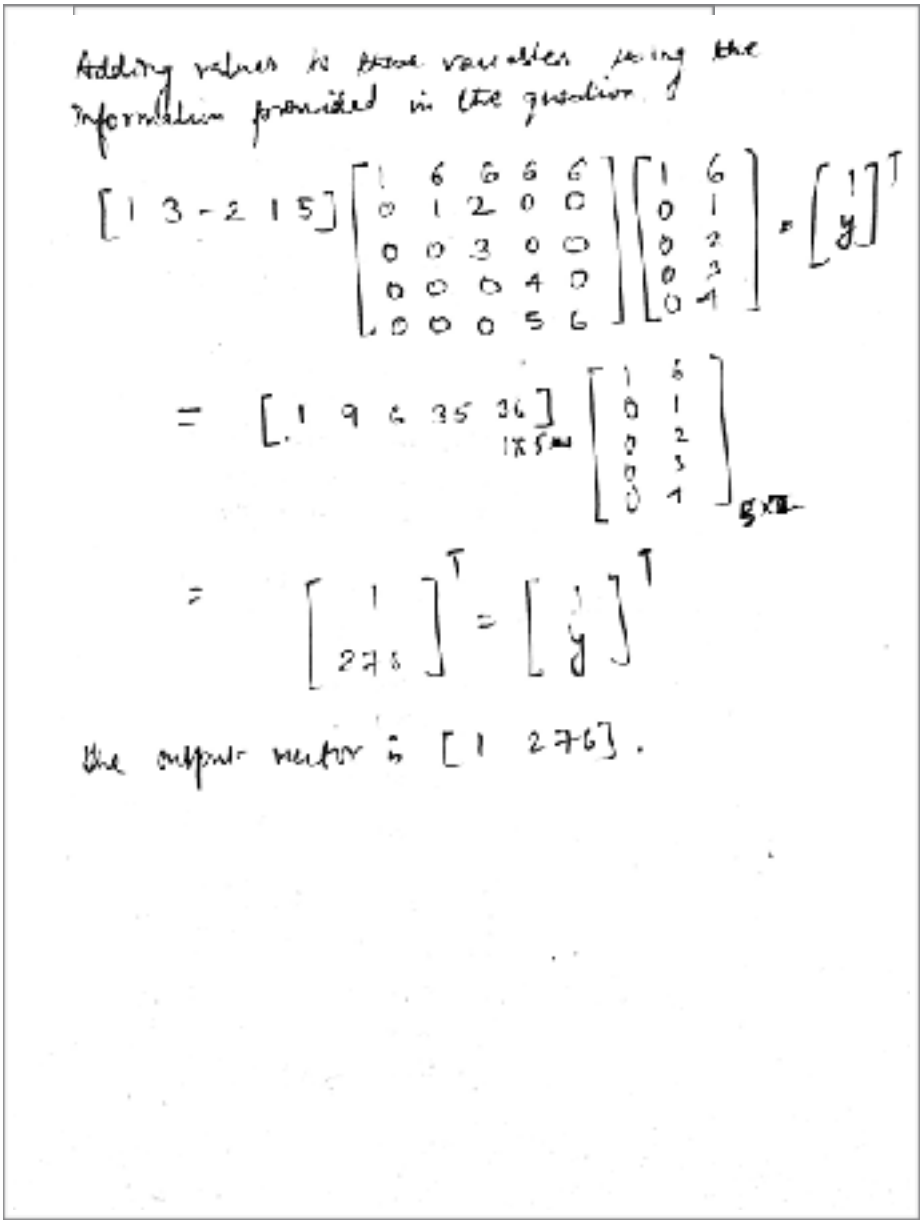
In the above figure, we observe that rosenblatt perceptrons fails to classify data points with perceptron ends in classifying four classes. The one label. Hence we need two Class C and Class D under classifying such data points. layer neural network for have been using single layer Beginning from Q3b, we perceptron.

Answer 4 :

Answer 5: We can represent a single layer neural network with linear activation as y=w.x. Where w is a vector of n+1 weights ([b, w1, w2, w3,…….,wN]) and x is a vector of n+1 inputs ([1, x1, x2, x3, …….., xN]) of d dimensions. The equation y=w.x is a dot product of transpose of the weight vector and a row vector in the data set. The notion of multilayer perceptron (neural network) is originated rosenblatt perceptron. If the activation function used in the multilayer

SIMPLE PERCEPTRON AND LINEAR ADAPTIVE FILTERS

Manas Gaur, Ph.D, Wright State University
U00856829



perceptron is linear on the data points then we can represent multilayer perceptron as rosenblatt single layer perceptron.

Let us consider a K layer perceptron. The input is X and output is Y. Between each layer in the multilayer perceptron there is a weight vector defined randomly (d dimension). Suppose the weight vector between input and layer 0 is w0 , between layer 0 and layer 1 is w1 …. between layer K-1 and layer K is wK.

Output from layer 0 is Y0=w0.X

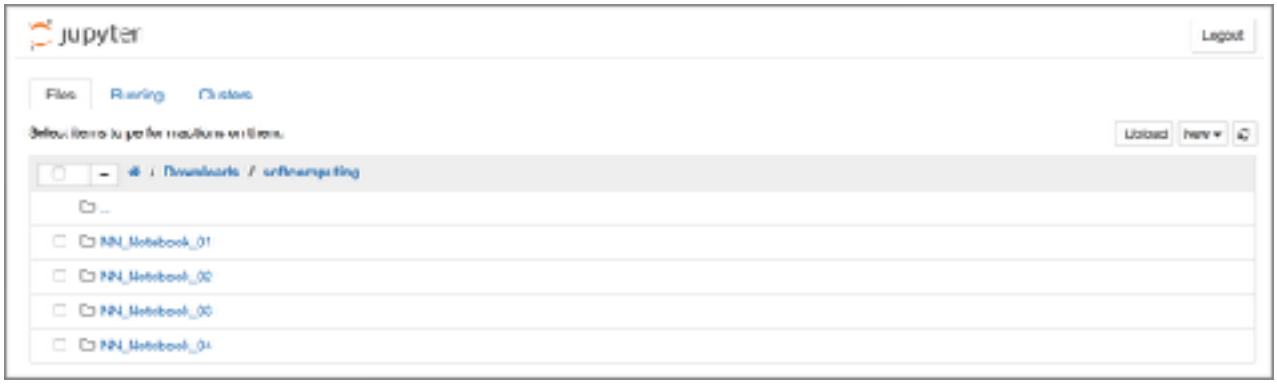
SIMPLE PERCEPTRON AND LINEAR ADAPTIVE FILTERS

Manas Gaur, Ph.D, Wright State University
U00856829

Output from layer 1 is Y1=w1.Y0 = w1.w0.X
Output from layer 2 is Y2=w2.Y1=w2.w1.Y0=w2.w1.w0.X
:
:
:
:
Output from layer K is YK=Y=[wK].[wK-1].[wK-2]……w0.X
Above formulation can be represented as Y=WX, where W is [[wK].[wK-1].[wK-2]….] of dimension 1X d.

Hence we proved that addition of layers to a pre-existing neural network having linear activation function adds no computation power.

Answer 6: I have downloaded the notebooks 1 to 4. I have played around with the notebooks by changing various parameters in it.



Answer 7 a:

```
# Let's make a cloud of randomized data points for us to fit the line to. First we'll
# make two lists of points and then stack them into a single matrix of point pairs
X_points = np.linspace(0.0, 15.0, num = 50)
### i have uncommented this statement
Y_points = [x + random.uniform(-2.0, 2.0) for x in X_points]
#Y_points = [x for x in X_points]
Points   = np.column_stack([X_points, Y_points])
```

SIMPLE PERCEPTRON AND LINEAR ADAPTIVE FILTERS

Manas Gaur, Ph.D, Wright State University
U00856829

Answer 7 b : Y = random and uniform data points in range (-2.0, 2.0)

Initial Error	Epochs	Final Error
20.7582850235	10000	1.17281089882
11.0921324676	10000	1.32931381953
14.1871508753	10000	1.33026351305
32.1757817127	10000	2.46975598519
30.2956016938	10000	1.16837203699
54.9277956393	10000	2.59032397729

Answer 7 c: Y = random and uniform data points in range (-3.0, 3.0)

Initial Error	Epochs	Final Error
8.0272539085	10000	4.09359353273
7.92851291231	10000	2.22202577406
3.6692759305	10000	4.69987413737
42.025414068	10000	2.59769590081
9.44981141816	10000	3.22473068757
30.2235417519	10000	3.84317698355

Answer 7 d: Y= random and uniform data points in range (-10.0, 10.0)

Initial Error	Epochs	Final Error
38.0522813419	10000	30.8217249668
52.3867737283	10000	61.3188501969
77.2294884815	10000	41.9577104437

SIMPLE PERCEPTRON AND LINEAR ADAPTIVE FILTERS

Manas Gaur, Ph.D, Wright State University
U00856829

Initial Error	Epochs	Final Error
30.8441764939	10000	34.7005487511
116.965902343	10000	42.4997245281
45.0008798558	10000	36.4982343934

Answer 7 e: Based on the above experiments, it is fairly conclusive that the training of linear adaptive filter is not appropriate. On analyzing the code, it is visible that a fixed iteration number of 10000 is taken as a constraint which obstructs complete training. Following are the changes made in the code :

Note : I have kept Error threshold of 0.001. Changing it to 0.0001 will also affect the experiment.

1. Based on the above experiment, I have assumed an upper bound on the error. The upper bound is set to 100.
2. At each stage the weight vector is changed. So, the mean square error over the "Points" also changes. Hence, the error per iterations is recorded as the difference between the mean square error calculated on the prior weight and new weights. That is

Error Per Iterations = Mean Square Error on Data Points with W - Mean Square Error on Data Points with W*

W : Prior weight or weight calculated in previous iteration
W*: New weight or weight calculated in current iteration

Answer 7 f : Experiments recorded at Y = random and uniform points in range (-2.0, 2.0)

Initial Error	Epochs	Final Error
1.87254591171	36	1.68201279414
2.60032115559	58	1.78708452181
16.5034701321	15	1.28769833099
19.0574597716	38	2.6359628184
1.5813350494	1	2.4875490712

SIMPLE PERCEPTRON AND LINEAR ADAPTIVE FILTERS

Manas Gaur, Ph.D, Wright State University
U00856829

Initial Error	Epochs	Final Error
64.8577819255	15	1.99150845351

Experiments recorded at Y= random and uniform points in range (-10.0, 10.0)

Initial Error	Epochs	Final Error
33.7405319692	1	58.9881880077
83.6171944283	29	34.3978532799
45.3749983236	1	47.0741378547
89.5927103547	136	76.2849825527
35.6051948949	1	38.4548320731
86.8294569179	91	42.8857998969