SHIVAM KHARE
201505547
SMAI-ASSIGNMENT-2

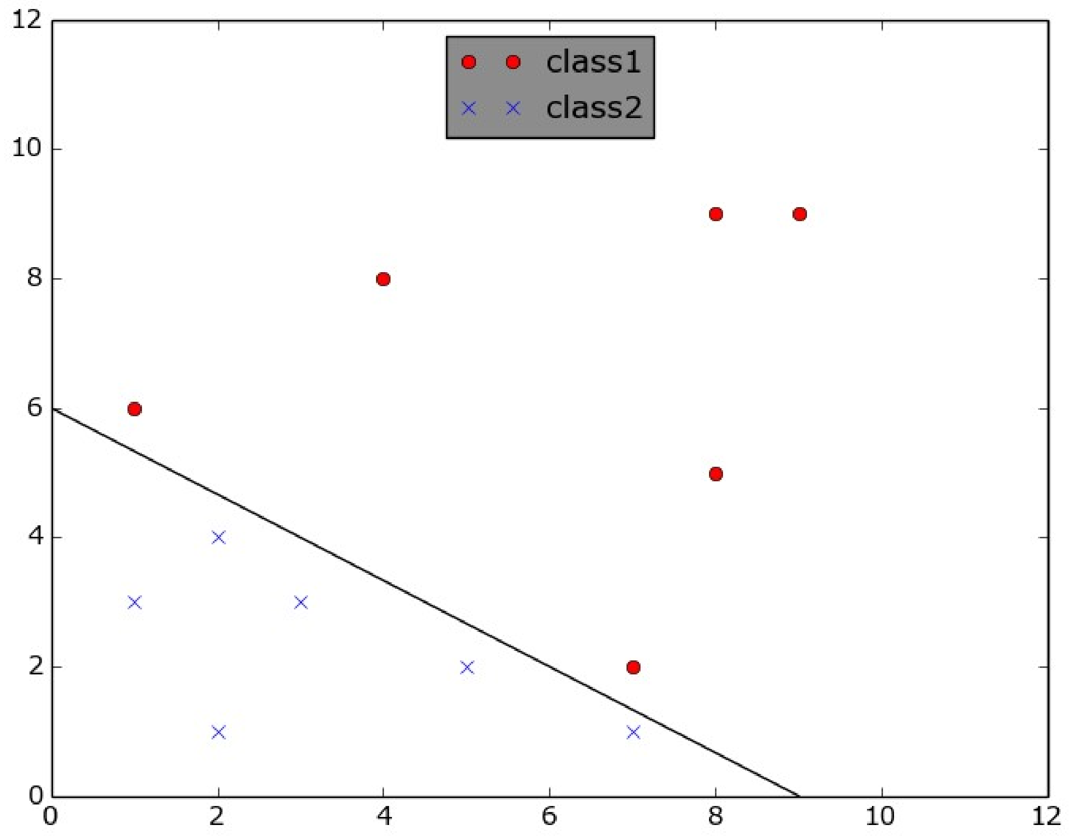
## CODE:

```
#!/usr/bin/python
import numpy as np
import matplotlib.pyplot as plt

class1 = [[1, 6],[7, 2],[8, 9],[9, 9],[4, 8],[8, 5]]
xcord1=[1,7,8,9,4,8]
ycord1=[6,2,9,9,8,5]

class2 = [[2, 1],[3, 3],[2, 4],[7, 1],[1, 3],[5, 2]]
xcord2=[2,3,2,7,1,5]
ycord2=[1,3,4,1,3,2]

class1nm = [[1, 6, 1],[7, 2, 1],[8, 9, 1],[9, 9, 1],[4, 8, 1],[8, 5, 1]]

class2nm = [[-2, -1, -1],[-3, -3, -1],[-2, -4, -1],[-7, -1, -1],[-1, -3, -1],[-5, -2, -1]]

dataset=np.array(class1nm+class2nm)

def plotfunc(listvector):
        linex=[]
        liney=[]
        for x in range (0,10,1):
                y=(-listvector[0]*x - listvector[2])/listvector[1]
                liney.append(y)
                linex.append(x)
        axes = plt.gca()
        axes.set_xlim([0,12])
        axes.set_ylim([0,12])
        plt.plot(xcord1,ycord1,".r",label='class1',marker="o")
        plt.plot(xcord2,ycord2,".b",label='class2',marker="x")
        plt.plot(linex,liney,'-',color='black')
        legend=plt.legend(loc='upper center')
        frame=legend.get_frame()
        frame.set_facecolor('.55')
        plt.show()


def single_sample_perceptron():
        alpha = np.array([1 , 1 , 1])
        eta=.5
        b=2
        size_data = len(dataset)
        counter=0
        while(1):
                flag=0
                counter=0
                for i in range(size_data):
                        dist=np.dot(alpha,dataset[i])
                #print dist
                if(dist<=0):
                    flag=1
                    counter=counter+1
                    alpha=alpha+eta*dataset[i]
            #print alpha
            if counter==0 :
                break

        plotfunc(alpha)

def Single_sample_perceptron_with_margin():
        alpha = np.array([1 , 1 , 1])
        eta=.5
        b=2
        size_data = len(dataset)
        counter=0
        while(1):
            counter=0
            for i in range(size_data):
                dist=np.dot(alpha,dataset[i])
                #print dist
                if(dist<=b):
                    flag=1
                    counter=counter+1
                    alpha=alpha+eta*dataset[i]
            #print alpha
            if counter==0 :
                break
        plotfunc(alpha)

def Relaxation_algorithm_with_margin():
        alpha=np.random.rand(3)
        eta=2
        b=10
        size_data = len(dataset)
        counter=0
        while(1):
            counter=0
            for i in range(size_data):
                dist=np.dot(alpha,dataset[i])
                value=np.dot(dataset[i],dataset[i])
                #print dist
                if dist <=b :
                    #print dist
                    counter=counter+1
                    alpha=alpha+eta*dataset[i]*((b - dist)/(value))

            if counter==0 :
                break
        plotfunc(alpha)
```

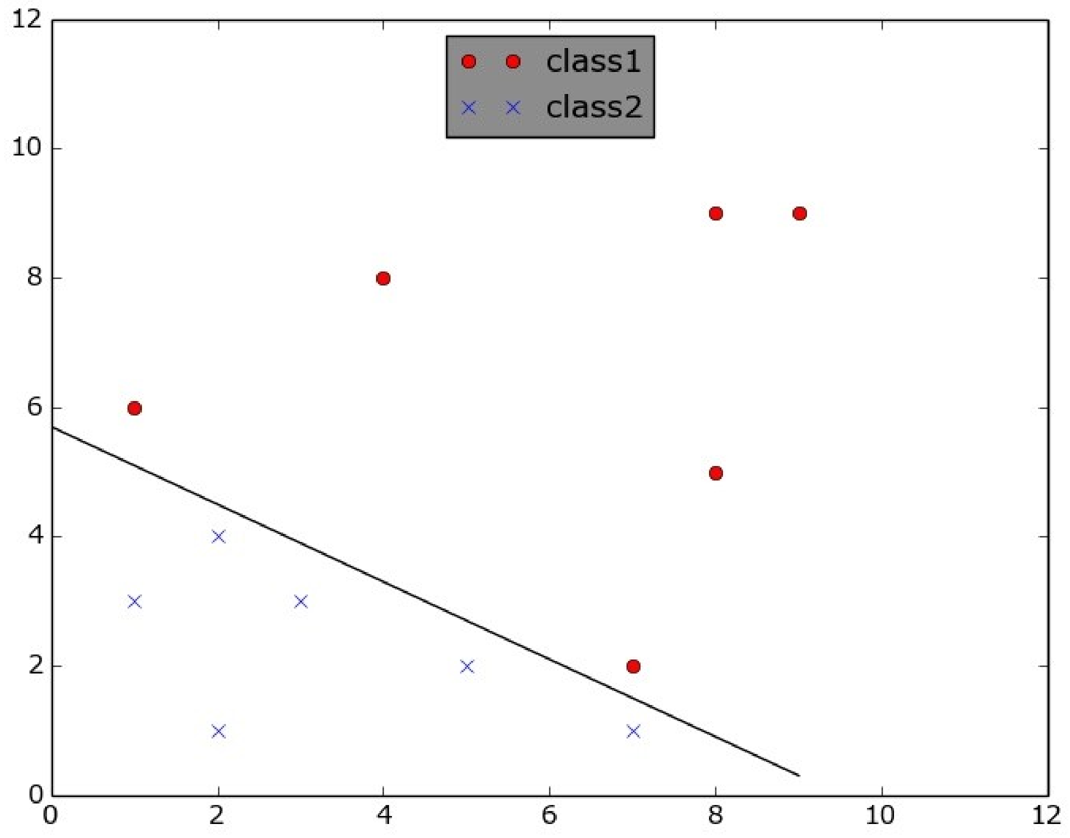
```
def Widrow_Hoff_or_Least_Mean_Squared_Rule():
        alpha=np.random.rand(3)
        eta=.05
        b=2
        size_data = len(dataset)
        counter=0
        #counter1=0
        while(1):
                counter=0
                for i in range(size_data):
                        dist=np.dot(alpha,dataset[i])
                        value=np.dot(dataset[i],dataset[i])
                        #print dist
                        if dist <=b :
                                #print dist
                                counter=counter+1
                                alpha=alpha+eta*dataset[i]*(b - dist)
                #print alpha
                #counter1=counter1+1
                if counter==0 :
                        break

        plotfunc(alpha)

def main():

        single_sample_perceptron()
        Single_sample_perceptron_with_margin()
        Relaxation_algorithm_with_margin()
        Widrow_Hoff_or_Least_Mean_Squared_Rule()


main()
```

## GRAPH:

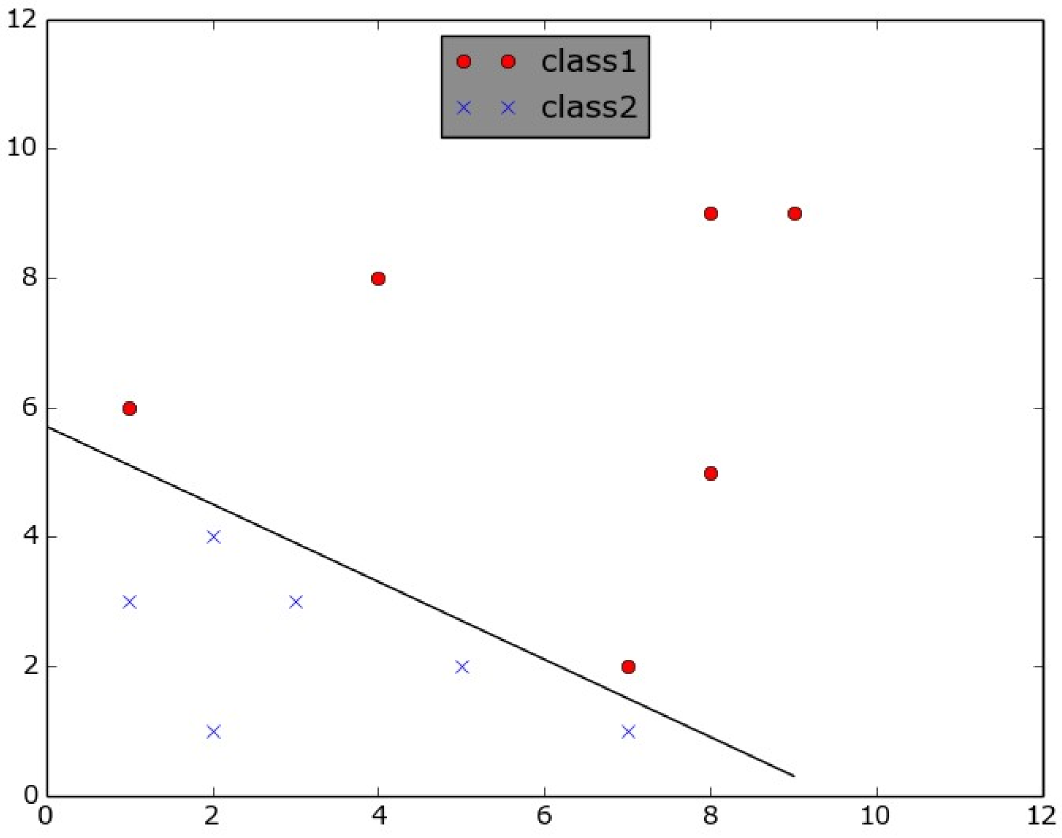
### Single-sample perceptron:



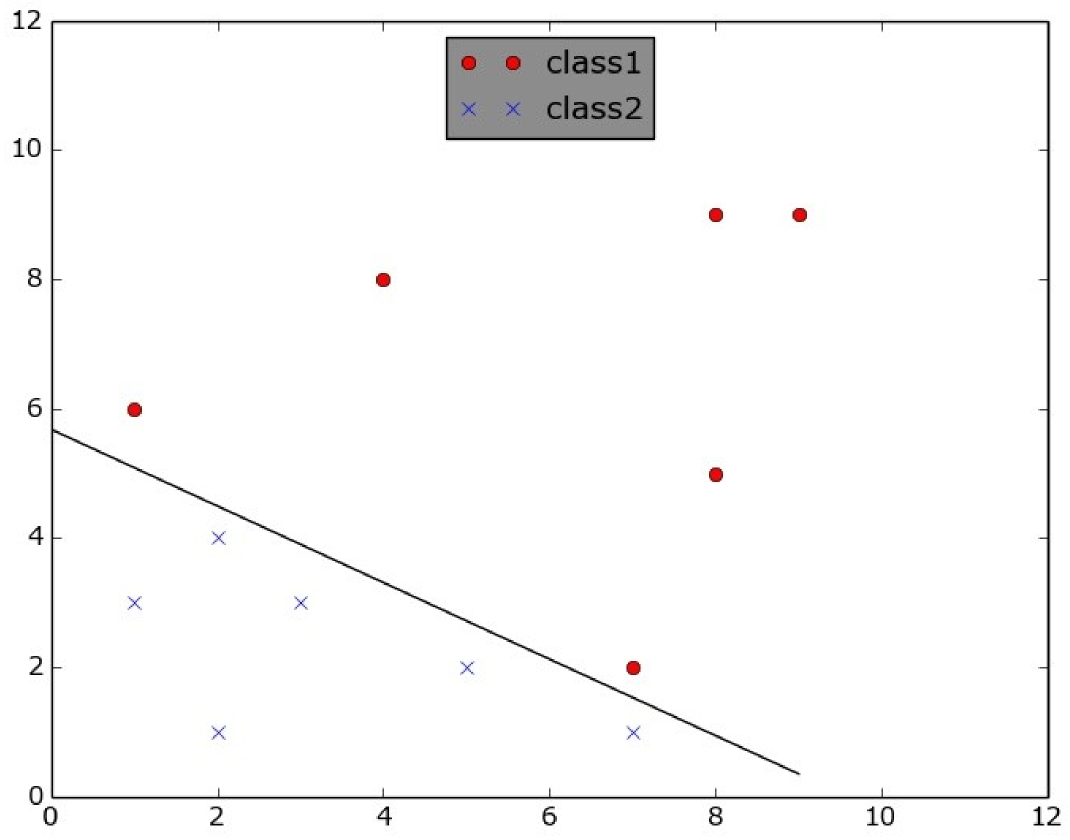
**Single-sample perceptron with margin:**



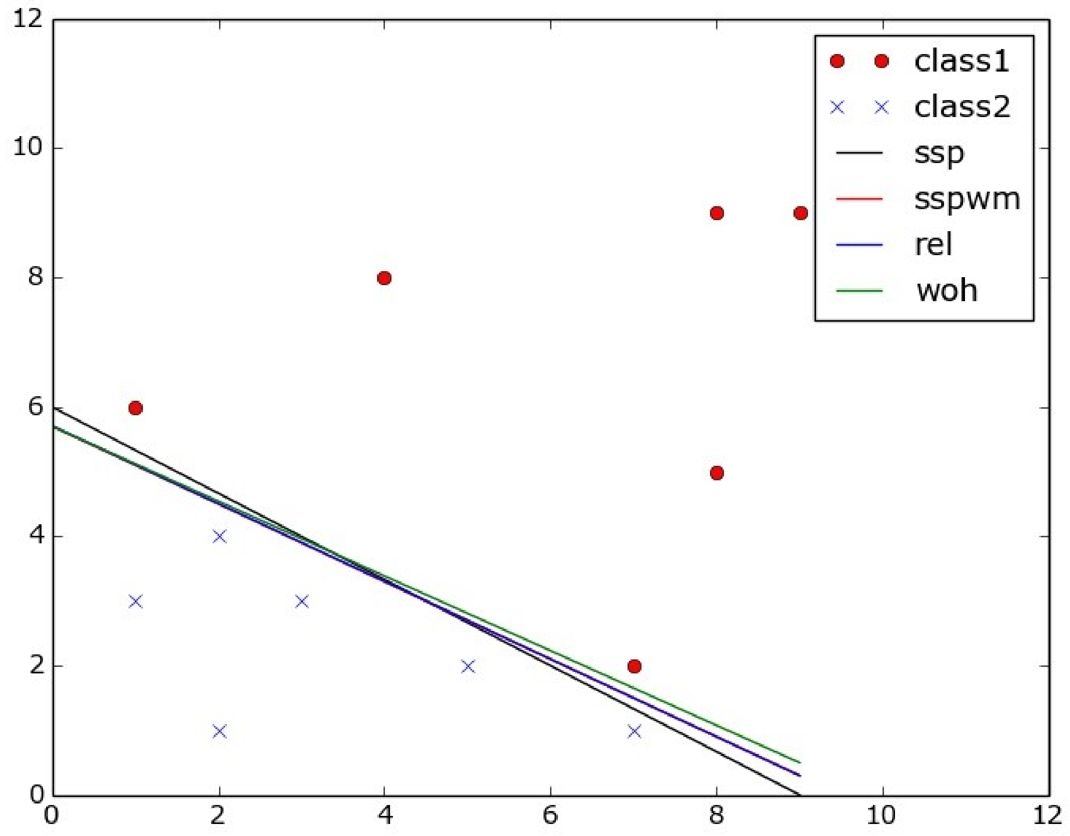
**Relaxation algorithm with margin**



## Widrow-Hoff or Least Mean Squared (LMS) Rule



**ALL GRAPH**



**Question 1:**

convergence proof of single sample perceptron
(if a solution exist)

let $\hat{a}$ be the solution vector

$\Rightarrow$ $a(k+1) = a(k) + y$ [$y \in y_m$ ; set of miss classified one]

$\Rightarrow$ Subtract $\alpha \hat{a}$ from both sides ; $\alpha \in R$.

$\Rightarrow$ $a(k+1) - \alpha\hat{a} = a(k) - \alpha\hat{a} + y$

$\Rightarrow$ square both sides

$\Rightarrow$ $|a(k+1) - \alpha\hat{a}|^2 = |a(k) - \alpha\hat{a}|^2 + |y|^2 + 2(a(k) - \alpha\hat{a})y$

$\Rightarrow$ $|a(k+1) - \alpha\hat{a}|^2 = |a(k) - \alpha\hat{a}|^2 + |y|^2 + \underbrace{2a(k)y}_{\text{-ve as } y \text{ is misclassified}} - \underbrace{2\alpha\hat{a}y}_{\text{+ve always as } \hat{a} \text{ is solution}}$

$\therefore$ $|a(k+1) - \alpha\hat{a}| \leq |a(k) - \alpha\hat{a}|^2 - 2\alpha\hat{a}y + |y|^2$

we want the minimum possible term to be substracted.

so let $|y|^2 = \beta^2$ for maximum $y$

$y = \min$ of all $\hat{a}y$.

$\therefore$ R.H.S. $= |a(k) - \alpha\hat{a}|^2 - 2\alpha y + \beta^2$

let $\alpha = \frac{\beta^2}{y}$

R.H.S. $= \underline{|a(k) - \alpha\hat{a}| - \beta^2}$

$\therefore$ at each iteration atleast $\beta^2$ will reduce.

then algo will terminate in

$$\boxed{\frac{|a(k) - \alpha\hat{a}|}{\beta^2}}$$ steps

**Q2 :**
**Exercise C:**

***Relation between the initial weight vector and the convergence time***

**Single-sample perceptron**
**Single-sample perceptron with margin**
**Relaxation algorithm with margin**
For the above algorithms, if the data is linearly separable the algorithm convergence time is found to be finite irrespective of the initial values of the weight vectors. However, if the data is not linearly separable the convergence time is not finite.

***Widrow-Hoff***
The algorithm convergence time always changes with respect to the initial values of the weight vectors. If the initial values of the weight vectors in closer to the actual values of the weight vector(that we are supposed to obtain), the algorithm converges faster. On the other hand if the initial values are far away from the actual value of the weight vector, the algorithm takes significantly large time to converge.

**Exercise D:**

***Effect of adding different margins on the final solution and the convergence-time for algorithms***

**Single-sample perceptron**
**Single-sample perceptron with margin**
**Relaxation algorithm with margin**
As the margin increases the convergence time is significantly affected for the above algorithms, provided the algorithm always terminates with correct decision boundary.
If we impose a rule stating that the algorithm must terminate after certain number of iterations, the execution time will be still high, however we might not get the accurate or correct decision boundary.

**Widrow-Hoff**
As the margin changes, the algorithm always terminates with almost same decision boundary without any significant change in the convergence time.

**Question 6:**

***Thus there exists any data-set for which the LMS and perceptron always turns out to be in-line(aligned) or different?***

---

No.
In case of linearly separable data, the solution obtained by both the algorithms will be a ligned with each other.
In case of linearly non-separable data, the perceptron algorithm will never converge. So, the depending on the point at which we break the algorithm, the solution could be aligned or different from that of the one obtained from the LMS rule.

**Question 7:**
**The nature of the solution found in**

**Single-sample perceptron**
**Single-sample perceptron with margin**
**Relaxation algorithm with margin**
*As the data is not linearly separable, all the above three algorithms go into an infinite loop. The cause for this behavior is that the above algorithms are trying to find a straight line or an hyperplane that classifies the two classes ω1 and ω2. While trying to do so, always at least one training sample of either or both the classes remains miss-classified. In order to correct it the above algorithm updates the weight vector, which consequently results in the mis-classification of other points. Thus, the above algorithms enter into an infinite loop. The loop is programatically braked after a fixed number of iterations.*

***Widrow-Hoff***
*This algorithm converges to a weight vector which produces minimal error with respect to the classification. The convergence time depends on the initial weight vector and also the learning rate that is chosen.*

| **Comparison Table** | |
|---|---|
| Single-sample perceptron | 90% |
| Single-sample perceptron with margin | 85% |
| Relaxation algorithm with margin | 87% |
| *Widrow-Hoff* | 91% |