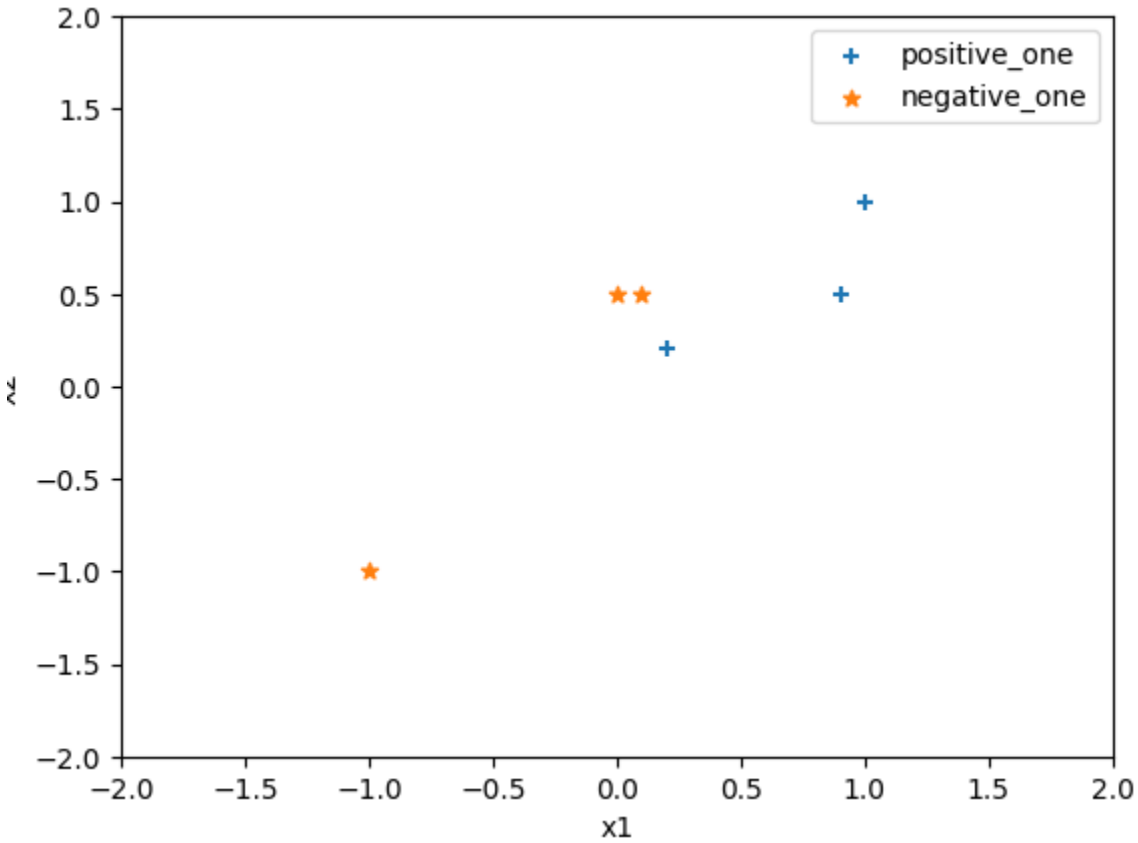
Problem 1: Perceptron

Colab link:- https://colab.research.google.com/drive/1qgwagQF-mFQ5_Mx2rZJY6J1xajnkXzEW#scrollTo=HvNg2OKiRW_3

1. **In how many steps perception learning algorithm will converge.**

The perceptron algorithm has been implemeneted here using python with some basic libraries like numpy, pandas etc. First the the input data is used asis without performing any normalization. Given tabulated data has been segregated between positive and negative classes and put into an array.Then the datapoints have been plotted to show the positive and negative points like below:-



Then the weight vector is initialized as [1 1 0] as w1=1,w2=1 and since w0 is not provided it has been assumed as 0.Then the actual algorithm which is depiced below is coded with python:-

Perceptron Learning Algorithm

While !Convergence

Do

$for\ x \in P \cup N\{$

If $x \in P\ and\ \sum_{i=0}^{n} w_i x_i < 0\ then$//Positive misclassified as negative

$$w = w + x$$

If $x \in N\ and\ \sum_{i=0}^{n} w_i x_i >= 0\ then$ //Negative misclassified as positive

$$w = w - x$$

}

Done

The output shows the algorithm converges in 18 steps with this scenario.

```
Weight vector: [ 3.3 -0.9  0. ]
Number of steps required for convergence: 18
    x1   x2   w1            w2
0  0.0  0.5  1.0  5.000000e-01
0  0.2  0.2  1.2  7.000000e-01
0  0.0  0.5  1.2  2.000000e-01
0  0.2  0.2  1.4  4.000000e-01
0  0.0  0.5  1.4 -1.000000e-01
0  0.2  0.2  1.6  1.000000e-01
0  0.0  0.5  1.6 -4.000000e-01
0  0.2  0.2  1.8 -2.000000e-01
0  0.1  0.5  1.7 -7.000000e-01
0  1.0  1.0  2.7  3.000000e-01
0  0.0  0.5  2.7 -2.000000e-01
0  0.2  0.2  2.9  5.551115e-17
0  0.0  0.5  2.9 -5.000000e-01
0  0.2  0.2  3.1 -3.000000e-01
0  0.1  0.5  3.0 -8.000000e-01
0  0.2  0.2  3.2 -6.000000e-01
0  0.1  0.5  3.1 -1.100000e+00
0  0.2  0.2  3.3 -9.000000e-01
```

Whereas while the same steps are performed by making the input data as mean centered the number of steps reduces drastically to 8.

```
Sample is negative_one
Weight vector: [ 1.7 -0.2  0. ]
Number of steps required for convergence: 8
             x1        x2   w1        w2
0 -2.775558e-17 -0.083333  1.0  0.916667
0 -2.000000e-01  0.216667  1.2  0.700000
0 -1.000000e-01  0.216667  1.3  0.483333
0 -2.775558e-17 -0.083333  1.3  0.400000
0 -2.775558e-17 -0.083333  1.3  0.316667
0 -2.000000e-01  0.216667  1.5  0.100000
0 -2.775558e-17 -0.083333  1.5  0.016667
0 -2.000000e-01  0.216667  1.7 -0.200000
```

2. What will be the final decision boundary? Show step-wise-step update of weight vector using computation as well as hand-drawn plot.

The decision boundary with data without making them mean centered is:-

3.3x1 – 0.9x2 = 0

And with data by making them mean centered is:-

1.7x1 – 0.2x2 = 0

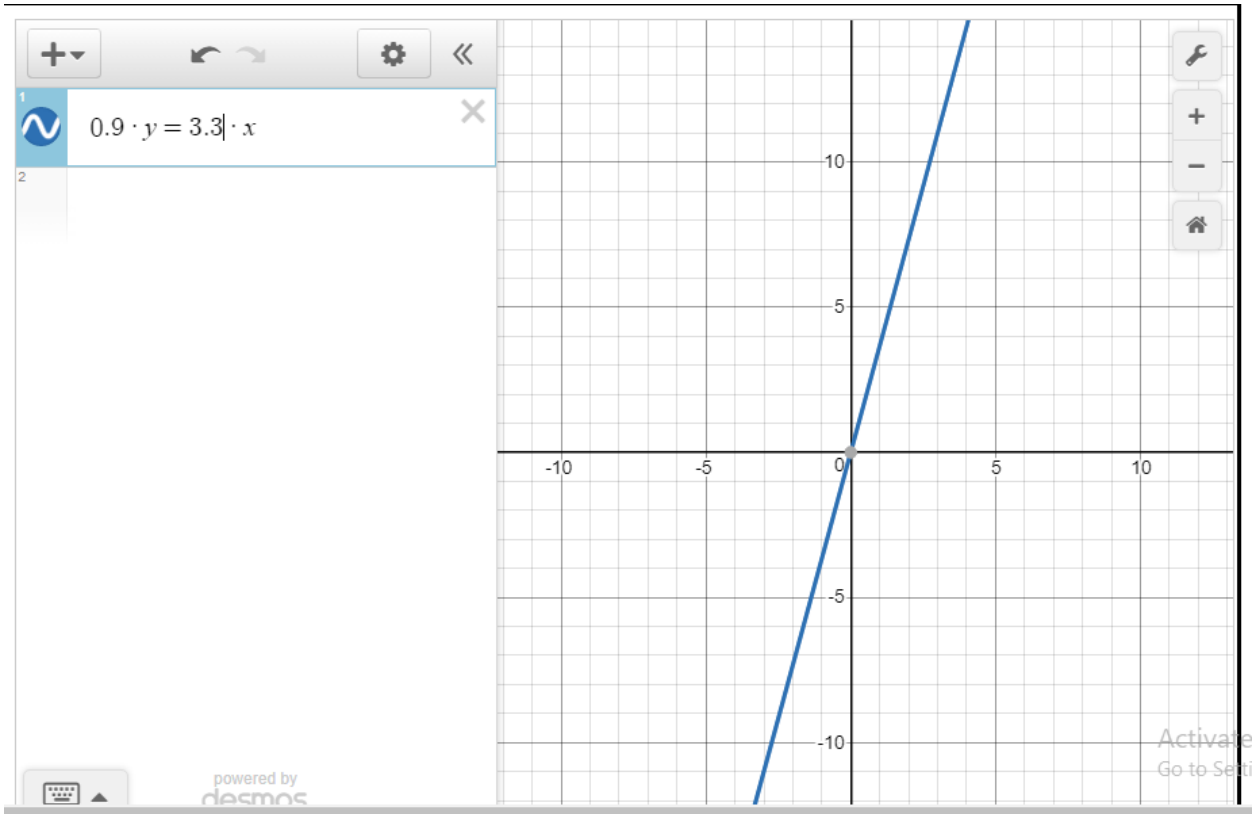
Step wise update of weight vector for 1st scenario:-

```
    x1   x2   w1            w2
0  0.0  0.5  1.0  5.000000e-01
0  0.2  0.2  1.2  7.000000e-01
0  0.0  0.5  1.2  2.000000e-01
0  0.2  0.2  1.4  4.000000e-01
0  0.0  0.5  1.4 -1.000000e-01
0  0.2  0.2  1.6  1.000000e-01
0  0.0  0.5  1.6 -4.000000e-01
0  0.2  0.2  1.8 -2.000000e-01
0  0.1  0.5  1.7 -7.000000e-01
0  1.0  1.0  2.7  3.000000e-01
0  0.0  0.5  2.7 -2.000000e-01
0  0.2  0.2  2.9  5.551115e-17
0  0.0  0.5  2.9 -5.000000e-01
0  0.2  0.2  3.1 -3.000000e-01
0  0.1  0.5  3.0 -8.000000e-01
0  0.2  0.2  3.2 -6.000000e-01
0  0.1  0.5  3.1 -1.100000e+00
0  0.2  0.2  3.3 -9.000000e-01
```

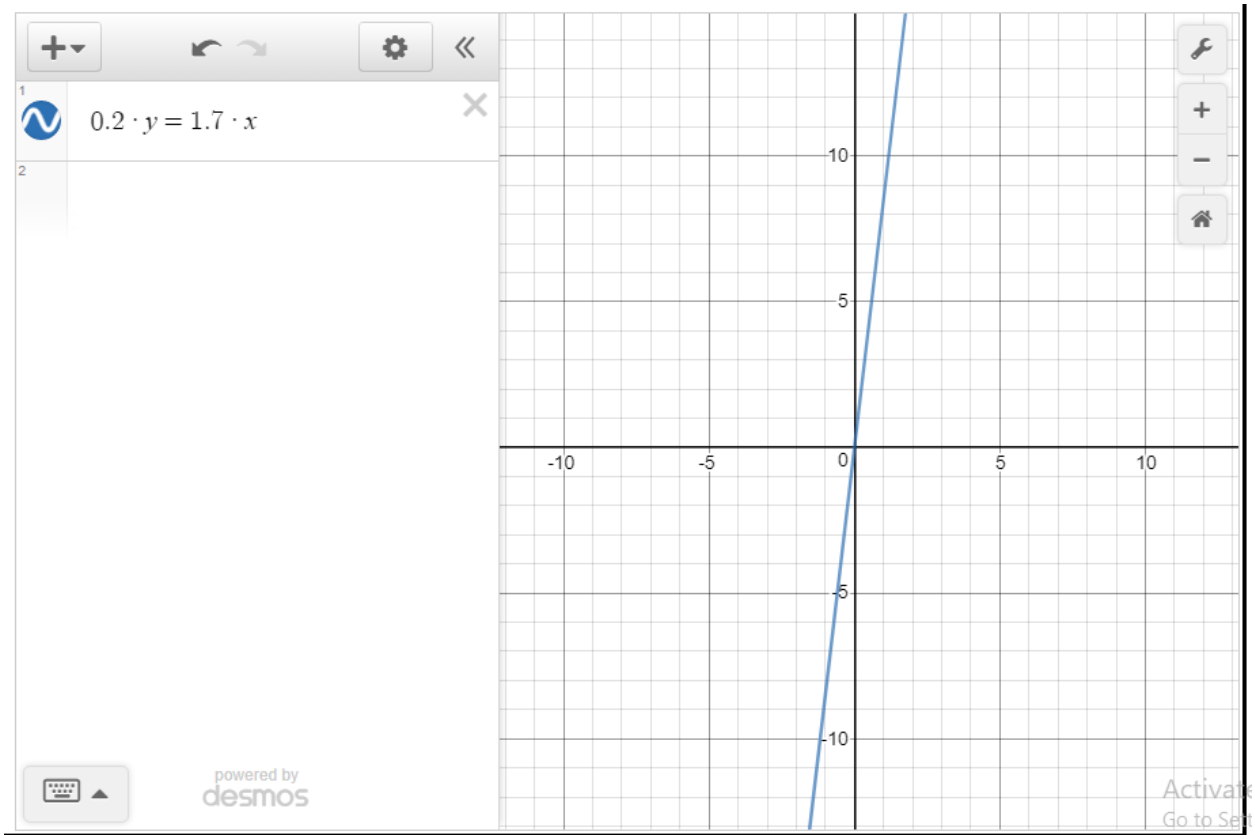
And for second scenario:-

	x1	x2	w1	w2
0	-2.775558e-17	-0.083333	1.0	0.916667
0	-2.000000e-01	0.216667	1.2	0.700000
0	-1.000000e-01	0.216667	1.3	0.483333
0	-2.775558e-17	-0.083333	1.3	0.400000
0	-2.775558e-17	-0.083333	1.3	0.316667
0	-2.000000e-01	0.216667	1.5	0.100000
0	-2.775558e-17	-0.083333	1.5	0.016667
0	-2.000000e-01	0.216667	1.7	-0.200000

By plotting the decision boundary for 1st scenario we get:-



By plotting the decision boundary for 2nd scenario we get:-


1
$0.2 \cdot y = 1.7 \cdot x$
2
10
5
-10
-5
0
5
10
-5
-10
powered by
desmos