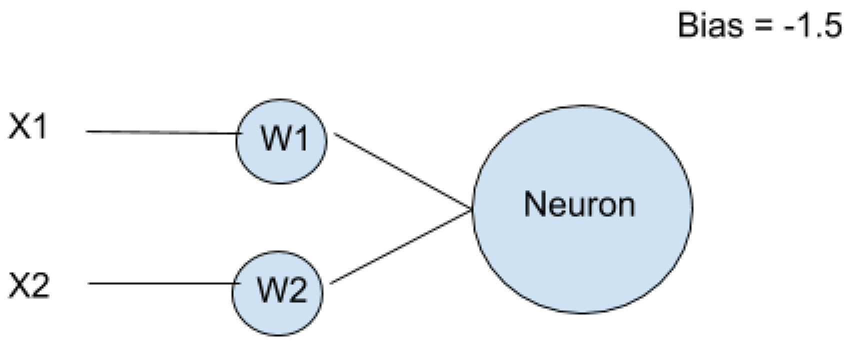
Ishmael Contreras
Kevin Lovell

1.



The below table is calculated assuming the perceptron model where the output is calculated by the following formula:

$$Decision = \left(\sum_{n=0}^{N} X_n * weights_n + bias\right) > 0$$

INPUT	SUM OUTPUT	CLASS DECISION
(0,0)	-1.5	0
(1,0)	-0.5	0
(0,1)	-0.5	0
(1,1)	0.5	1

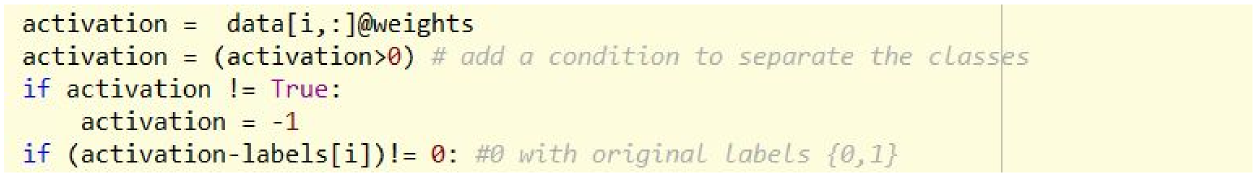
2. Eta is the learning rate. It is independent of the perceptron completing and converging (unless the rate approaches infinity, where it can diverge), but it is a hyperparameter that can be tuned for optimal performance.

As eta increases: There are many oscillations and the line is not robust
As eta decreases: Takes too long and many under-approximations

3. To create overlapping classes, you must simply alter the values of mu1 and mu2 to be equal (completely overlapping) or very close. The value of mu (the mean) centers the data in that cluster according to an (x,y) coordinate, so as the two mu's approach one another, so will the centers of those clusters, thus causing overlap.

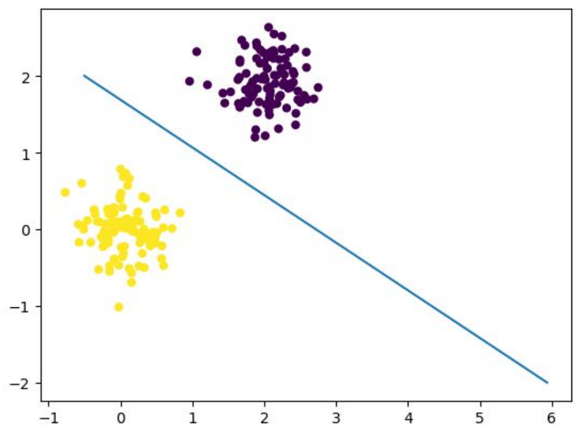
4. With overlapping classes, the algorithm runs through many more iterations and frequently overcompensates. Since the boundary of overlapping classes is continuously changing, this can pose problems to the algorithms' convergence, which could potentially approach infinity.

5. When changing the labels from {0,1} to {1,-1} special care needs to be taken to modifying the activation condition for each class. Lines 27 and 28 in the perceptron script need to be modified so they contain the possible levels the new labels can produce:

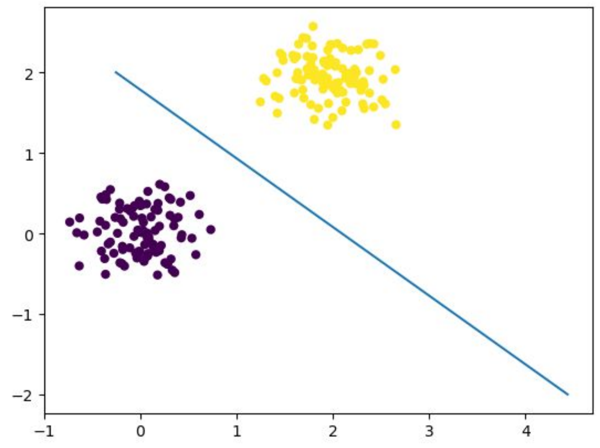


```
activation =  data[i,:]@weights
activation = (activation>0) # add a condition to separate the classes
if activation != True:
    activation = -1
if (activation-labels[i])!= 0: #0 with original labels {0,1}
```

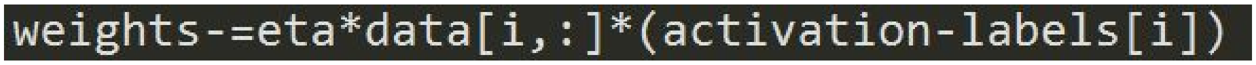
PLOT with Labels S1 = {0,1}



PLOT with Labels S2 = {1,-1}



6. The bias is calculated as -weights[0]/weights[1] in the plotline function. The weights are updated on the below line:



```
weights-=eta*data[i,:]*(activation-labels[i])
```

In the process of learning the weights, the bias is thus intrinsically learned since it is directly a function of the weights. More specifically, changing the weights effectively updates the prior biases and has the effect of shifting the decision boundary after each update. This decision boundary shift will ensure that the weights are updated if the output is still incorrect, and this cycle will continue.