Joshua Samuel P. Siy

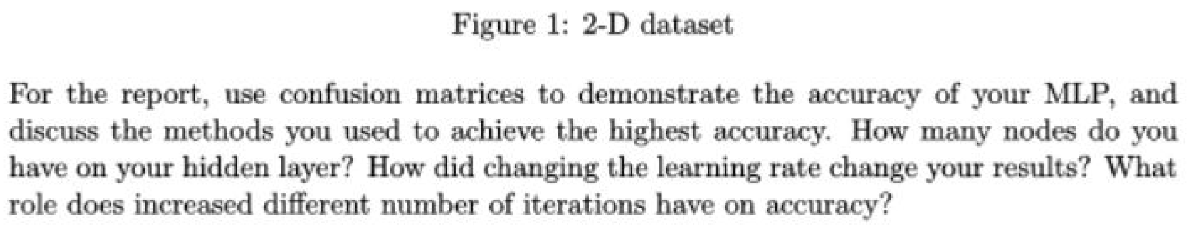


Figure 1: 2-D dataset

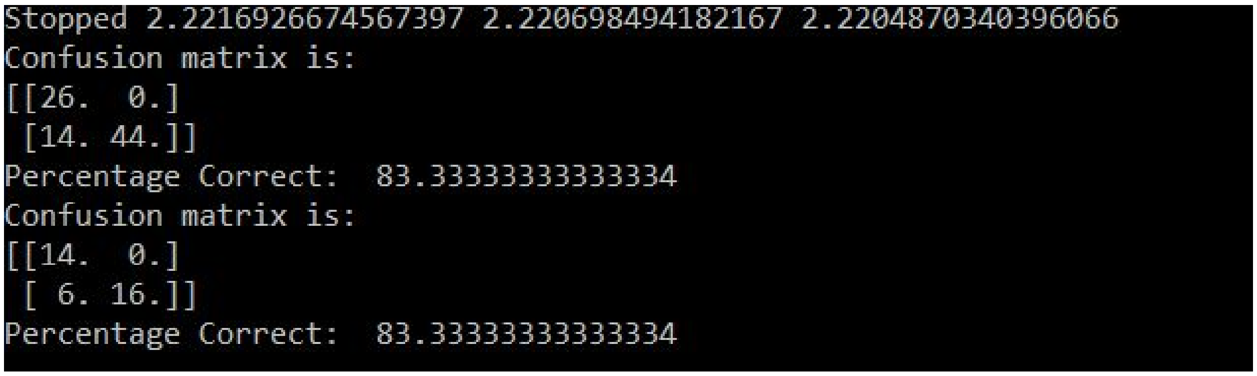
For the report, use confusion matrices to demonstrate the accuracy of your MLP, and discuss the methods you used to achieve the highest accuracy. How many nodes do you have on your hidden layer? How did changing the learning rate change your results? What role does increased different number of iterations have on accuracy?

NOTE: The summary of the answers for the questions given above are stated on the last page.

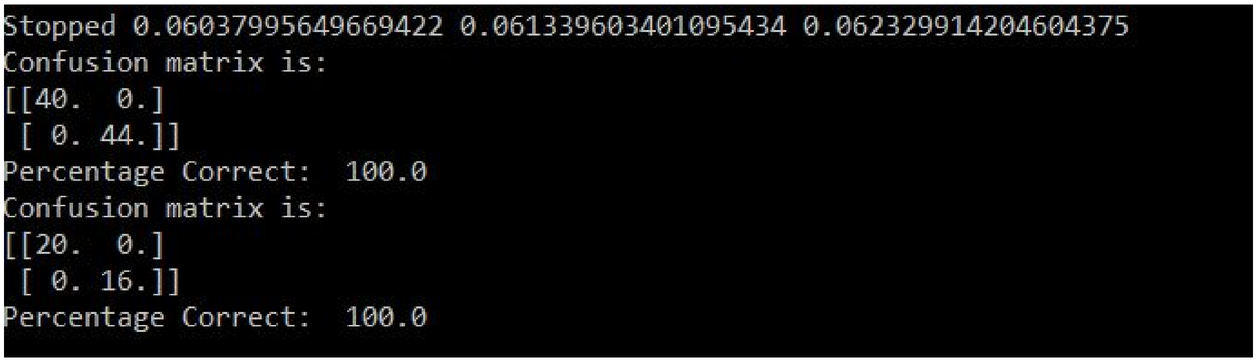
Convention: The above confusion matrix is for the training data and the confusion matrix below is for the Validation data. Since they are in random state, the accuracy and percentage tends to be different from time to time upon repetition, however sometimes increasing the number of iterations and ETA to an acceptable amount (Not less not too much) increases it's chance to be more accurate around 100%. A line starting with the word "Stopped" Indicates the errors where it stopped.

Implementation: From the code given, the earlystopping user defined function is used to give the code a limit the early stopping code uses the mlpfwd and mlptrain in its function s making this the primary code for generating more outputs to the class.. Then after generating those outputs and results, the confmat user defined function is used to check the accuracy of the confusion matrix generated. The input of the confmat is the train, the validation and the whole dataset

3 Hidden layers ,ETA= 0.01, #of iterations : 500

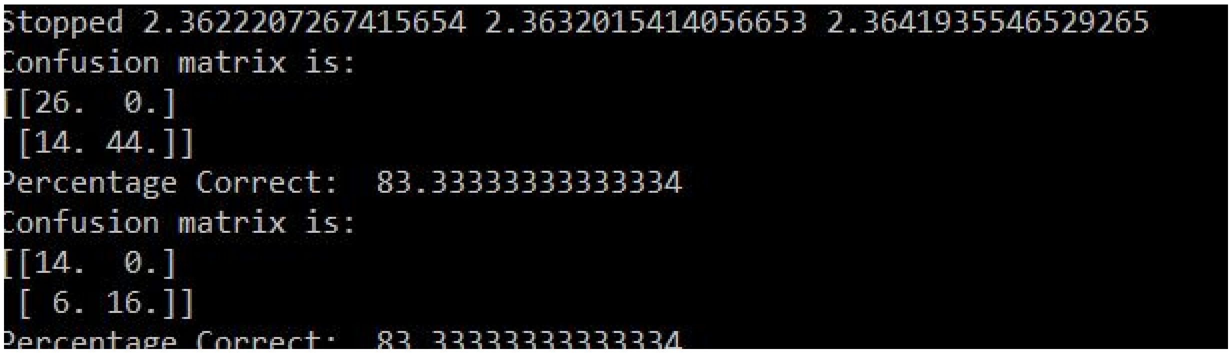

```
Stopped 2.2216926674567397 2.220698494182167 2.2204870340396066
Confusion matrix is:
[[26.  0.]
 [14. 44.]]
Percentage Correct:  83.33333333333334
Confusion matrix is:
[[14.  0.]
 [ 6. 16.]]
Percentage Correct:  83.33333333333334
```

3 Hidden layers ,ETA= 0.07, #of iterations : 1500



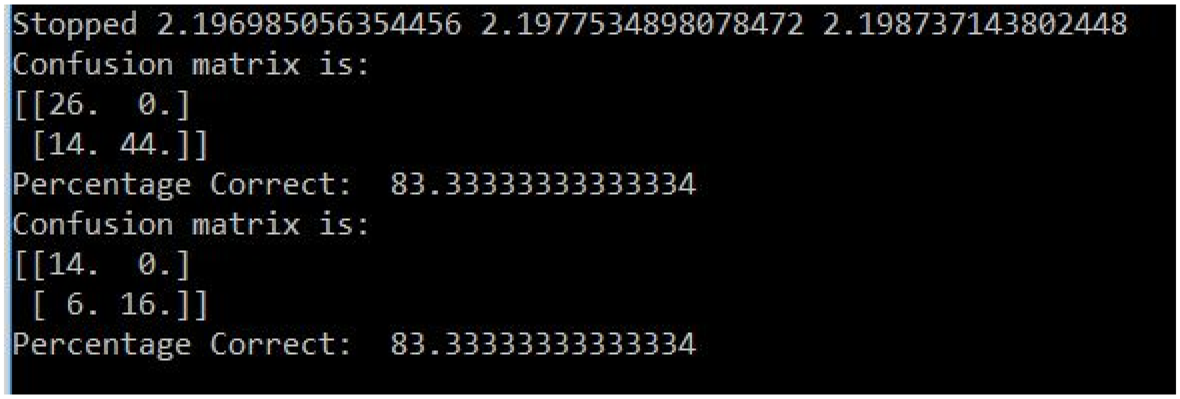
```
Stopped 0.06037995649669422 0.061339603401095434 0.062329914204604375
Confusion matrix is:
[[40.  0.]
 [ 0. 44.]]
Percentage Correct:  100.0
Confusion matrix is:
[[20.  0.]
 [ 0. 16.]]
Percentage Correct:  100.0
```

3 Hidden layers ,ETA=0.01, #of iterations :600



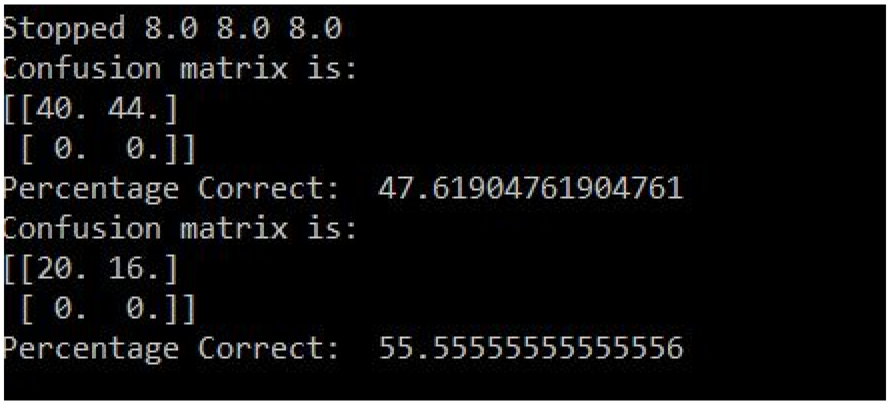
```
Stopped 2.3622207267415654 2.3632015414056653 2.3641935546529265
Confusion matrix is:
[[26.  0.]
 [14. 44.]]
Percentage Correct:  83.33333333333334
Confusion matrix is:
[[14.  0.]
 [ 6. 16.]]
Percentage Correct:  83.33333333333334
```

3 Hidden layers ,ETA=0.1, #of iterations :600



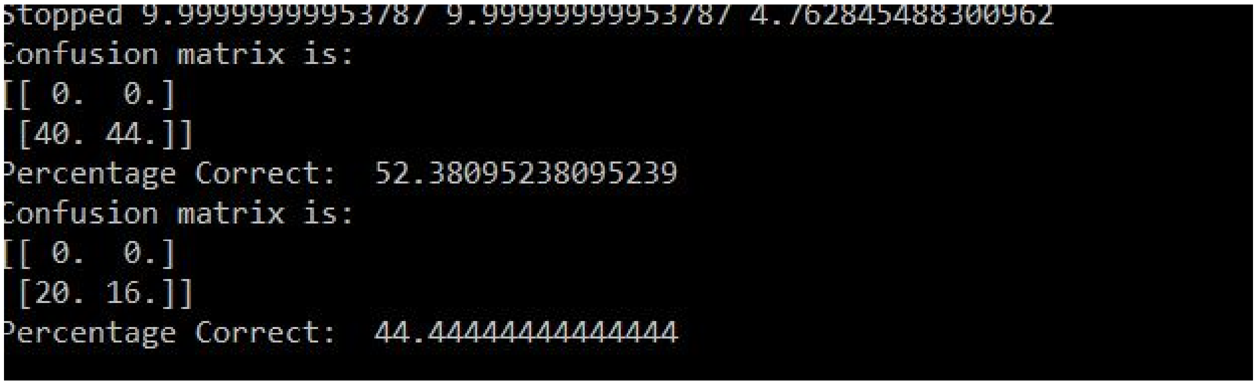
```
Stopped 2.196985056354456 2.1977534898078472 2.198737143802448
Confusion matrix is:
[[26.  0.]
 [14. 44.]]
Percentage Correct:  83.33333333333334
Confusion matrix is:
[[14.  0.]
 [ 6. 16.]]
Percentage Correct:  83.33333333333334
```

Absurdly increasing numbers(especially ETA) to a weird amount
3 hidden layers, ETA = 5 , niterations=900



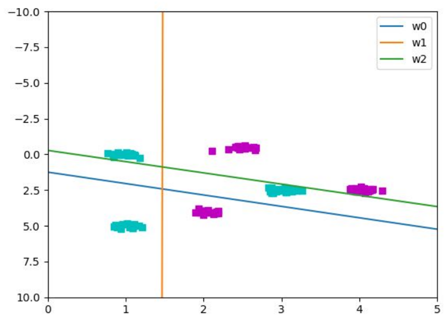
```
Stopped 8.0 8.0 8.0
Confusion matrix is:
[[40. 44.]
 [ 0.  0.]]
Percentage Correct:  47.61904761904761
Confusion matrix is:
[[20. 16.]
 [ 0.  0.]]
Percentage Correct:  55.55555555555556
```

# hidden layers= 150, # of iternations =3, ETA 0.1



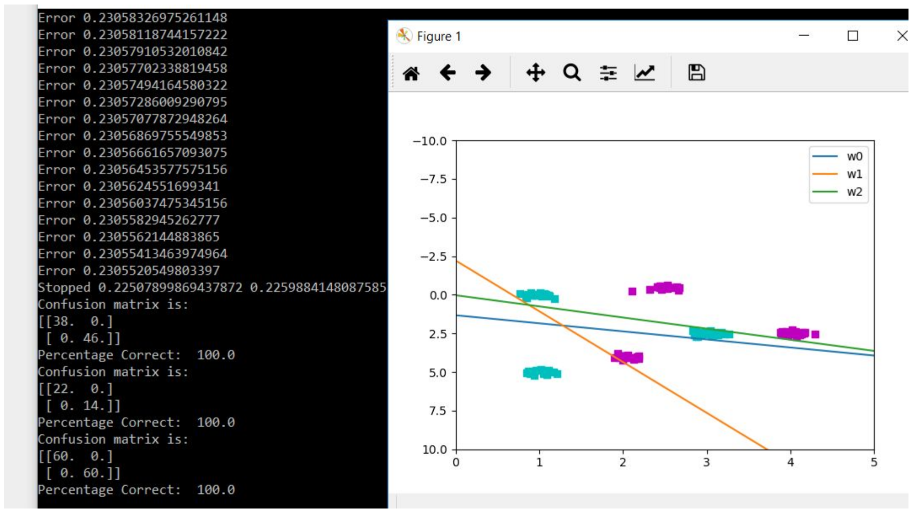
```
Stopped 9.99999999953787 9.99999999953787 4.762845488300962
Confusion matrix is:
[[ 0.  0.]
 [40. 44.]]
Percentage Correct:  52.38095238095239
Confusion matrix is:
[[ 0.  0.]
 [20. 16.]]
Percentage Correct:  44.44444444444444
```

Discussing my results. So far i have 3 hidden layers . Where i normally would assume the first hidden layer would have 3 neurons to draw 3 lines. Well by the code we do not exactly have any option to know how much neurons are there. The learning rate (ETA) largely affects the results as it produces more error if the learning rate is too high or when the learning rate is too small it would not actually train the data set to an error of 0 depending on the number of iterations. The number of iterations say that how much number of epochs will the weights have to adjust itself. So far, depending on the learning rate, the number of iterations can help the weights adjust to an optimal value as long as the ETA is not too large so that the error graph would not oscillate to a result making it erroneous, making the confusion matrix output more errors. So far adjusting the ETA and the number of iterations to a "sweet spot" allows the program to train itself to produce as less error as possible. Where assumptively most of the successful number of iterations is around 1500 and above given you have 3 neurons in this problem as the ETA value is around 0.07 under the logistic output type. These are the results under logistic function. Unfortunately, i cannot provide a fully 100% Correct Percentage as the weight generation in every iteration could vary and is not actually the same compared to the previous runs of codes. So the percentage could potentially vary from time to time. From back propagation the figure below plots the weights in order to visualize how the code performed. WARNING: This is inconsistent because each run produces different percentage correctness and different errors depending on how the backpropagation decided to perform or how random the testing /validation set was generated.



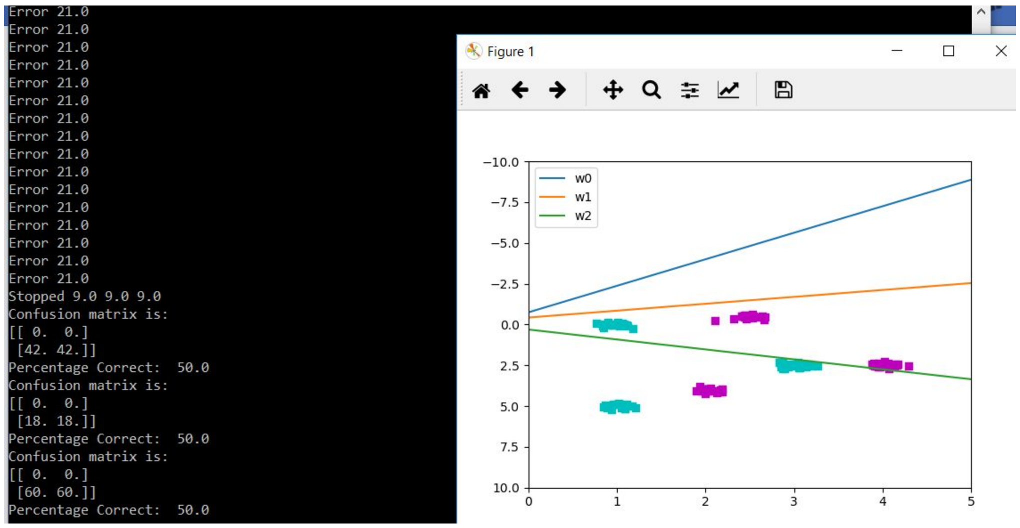
Taking the result of the most successful iteration and comparing the results in linear and softmax we get these results

LINEAR



```
Error 0.23058326975261148
Error 0.23058118744157222
Error 0.23057910532010842
Error 0.23057702338819458
Error 0.23057494164580322
Error 0.23057286009290795
Error 0.23057077872948264
Error 0.23056869755549853
Error 0.23056661657093075
Error 0.23056453577575156
Error 0.2305624551699341
Error 0.23056037475345156
Error 0.2305582945262777
Error 0.2305562144883865
Error 0.23055413463974964
Error 0.2305520549803397
Stopped 0.22507899869437872 0.2259884148087585
Confusion matrix is:
[[38.  0.]
 [ 0. 46.]]
Percentage Correct:  100.0
Confusion matrix is:
[[22.  0.]
 [ 0. 14.]]
Percentage Correct:  100.0
Confusion matrix is:
[[60.  0.]
 [ 0. 60.]]
Percentage Correct:  100.0
```

SOFTMAX

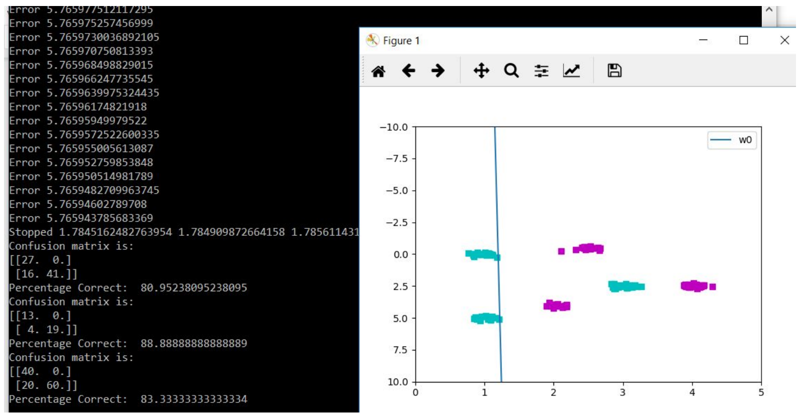


```
Error 21.0
Error 21.0
Error 21.0
Error 21.0
Error 21.0
Error 21.0
Error 21.0
Error 21.0
Error 21.0
Error 21.0
Error 21.0
Error 21.0
Error 21.0
Error 21.0
Error 21.0
Error 21.0
Stopped 9.0 9.0 9.0
Confusion matrix is:
[[ 0.  0.]
 [42. 42.]]
Percentage Correct:  50.0
Confusion matrix is:
[[ 0.  0.]
 [18. 18.]]
Percentage Correct:  50.0
Confusion matrix is:
[[ 0.  0.]
 [60. 60.]]
Percentage Correct:  50.0
```

The results of softmax is not very cooperative due to it's equational structure and its update process. Since the update of the errors of softmax very minimal, something tells me softmax is a good function when you are trying to optimize or correct a certain value of weights that are already accurate however one is trying to adjust it to a better optimal value, not weight value starting from 0 or random.

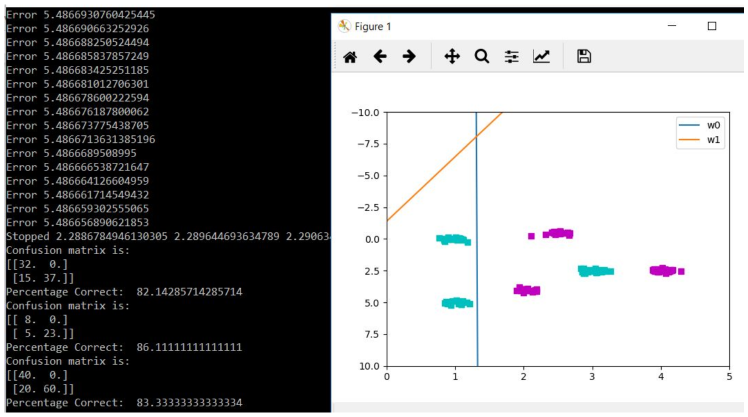
Given that we used a hidden layer of 3 as intuition, normally if we have only 1 or 2 , theoretically it will not give 100% accuracy.

Experimenting on 1 hidden layer with ETA 0.01 with a good value of niterations probably 5000.



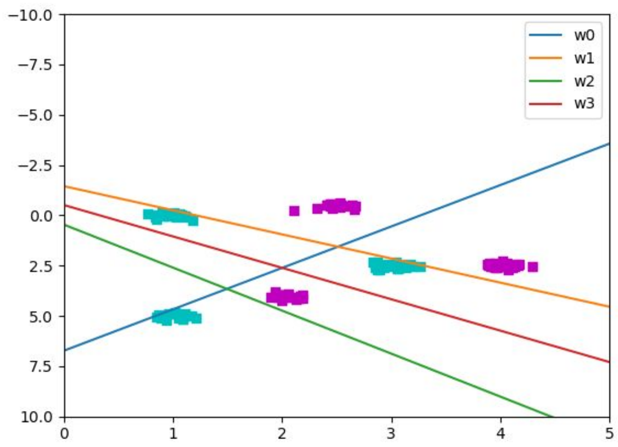
```
Error 5.765977512117295
Error 5.765975257456999
Error 5.7659730036892105
Error 5.765970750813393
Error 5.765968498829015
Error 5.765966247735545
Error 5.7659639975324435
Error 5.76596174821918
Error 5.76595949979522
Error 5.7659572522600335
Error 5.765955005613087
Error 5.765952759853848
Error 5.765950514981789
Error 5.7659482709963745
Error 5.76594602789708
Error 5.765943785683369
Stopped 1.7845162482763954 1.784909872664158 1.785611431
Confusion matrix is:
[[27.  0.]
 [16. 41.]]
Percentage Correct:  80.95238095238095
Confusion matrix is:
[[13.  0.]
 [ 4. 19.]]
Percentage Correct:  88.88888888888889
Confusion matrix is:
[[40.  0.]
 [20. 60.]]
Percentage Correct:  83.33333333333334
```

while having 2 neurons , the best performance it can give is similar to this figure

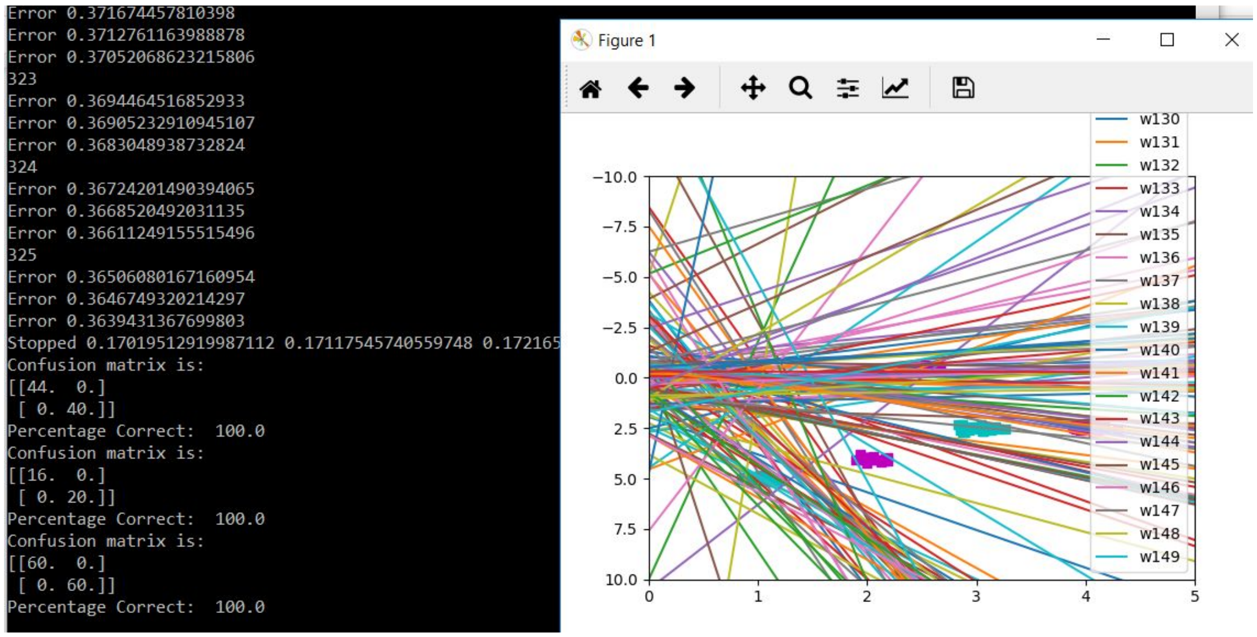


```
Error 5.4866930760425445
Error 5.486690663252926
Error 5.486688250524494
Error 5.486685837857249
Error 5.486683425251185
Error 5.486681012706301
Error 5.486678600222594
Error 5.486676187800062
Error 5.486673775438705
Error 5.4866713631385196
Error 5.4866689508995
Error 5.486666538721647
Error 5.486664126604959
Error 5.486661714549432
Error 5.486659302555065
Error 5.486656890621853
Stopped 2.2886784946130305 2.289644693634789 2.29063
Confusion matrix is:
[[32.  0.]
 [15. 37.]]
Percentage Correct:  82.14285714285714
Confusion matrix is:
[[ 8.  0.]
 [ 5. 23.]]
Percentage Correct:  86.11111111111111
Confusion matrix is:
[[40.  0.]
 [20. 60.]]
Percentage Correct:  83.33333333333334
```

and experimenting with more neurons, the performance of the network can result to a bit of problem and sometimes takes more time to correct itself due to having a new set of weights to affect the system. Despite it did perform, it actually overperform and took more time to train itself compared to the sufficient amount of neurons needed. E.g. 4 neurons

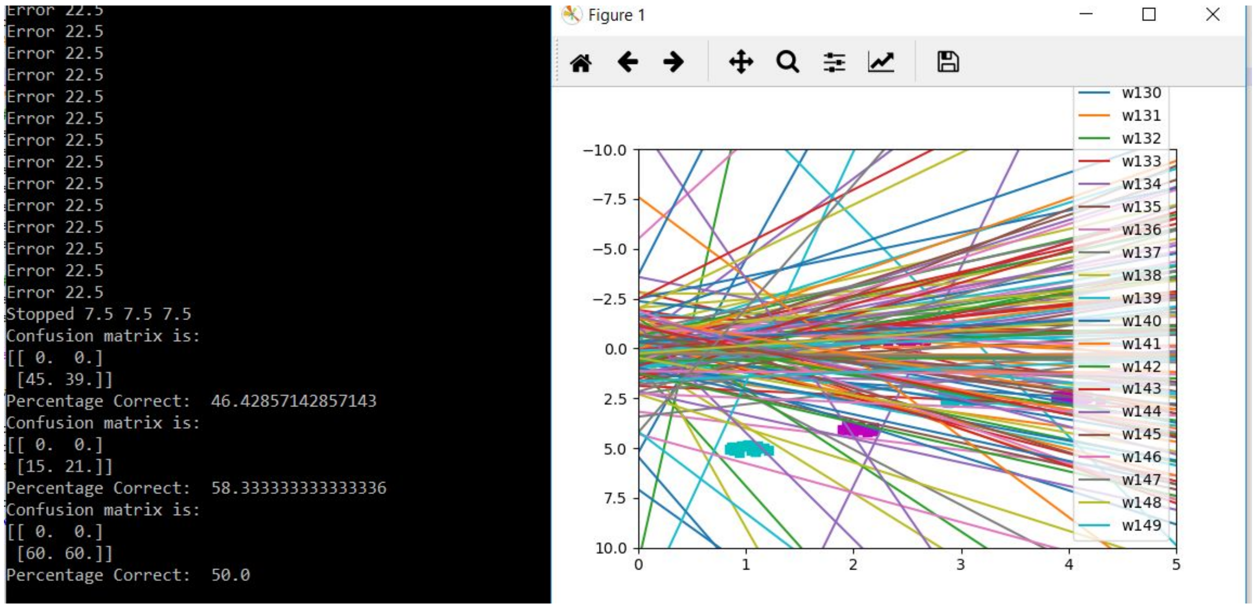


and lastly the experiment results of having 150 neurons and 0.01 ETA done with 3 number of iterations to correct itself but sometimes alot of overlapping information can be found and the neural network configuration is too redundant.



```
Error 0.371674457810398
Error 0.3712761163988878
Error 0.37052068623215806
323
Error 0.3694464516852933
Error 0.36905232910945107
Error 0.3683048938732824
324
Error 0.36724201490394065
Error 0.3668520492031135
Error 0.36611249155515496
325
Error 0.36506080167160954
Error 0.3646749320214297
Error 0.3639431367699803
Stopped 0.17019512919987112 0.17117545740559748 0.172165
Confusion matrix is:
[[44.  0.]
 [ 0. 40.]]
Percentage Correct:  100.0
Confusion matrix is:
[[16.  0.]
 [ 0. 20.]]
Percentage Correct:  100.0
Confusion matrix is:
[[60.  0.]
 [ 0. 60.]]
Percentage Correct:  100.0
```

Checking the results even if we try softmax, the results seems to be still inaccurate when using the softmax function



```
Error 22.5
Error 22.5
Error 22.5
Error 22.5
Error 22.5
Error 22.5
Error 22.5
Error 22.5
Error 22.5
Error 22.5
Error 22.5
Error 22.5
Error 22.5
Error 22.5
Stopped 7.5 7.5 7.5
Confusion matrix is:
[[ 0.  0.]
 [45. 39.]]
Percentage Correct:  46.42857142857143
Confusion matrix is:
[[ 0.  0.]
 [15. 21.]]
Percentage Correct:  58.333333333333336
Confusion matrix is:
[[ 0.  0.]
 [60. 60.]]
Percentage Correct:  50.0
```

Summary of answers to the questions above:

How many nodes?

Check the figures but for simplicity , the basic i used 3 to atleast make the correct percentage of confusion matrix to be 100%.. you need an amount of nodes greater than 3 to achieve 100% accuracy.

How does the learning rate affect the results? The learning rate ETA can affect the correctness of the results in a way that making it too small would need more number of iterations to achieve a good network . While having an ETA that is too big would tend to make your system less stable as if would have a hard time or never even achieve less than 1% error on the system. as your error keeps oscillating or jumping up and down instead of converging to a 0 error network.

What role does the number of iterations do?

It indicates how much and the limit to how many times will the system attempt to train its weights. So far the performance of the system will heavily base on the ETA than the nubmer of iterations so it actually will depend on the combination of both to achieve a good result on neural networks.

Extra: Softmax, Linear or Logistic.

So far from the observed usage of these output types. Linear tends to be straight and botherless compared to logistic or softmax. While logistic has a lesser time to correct itself but tends to be more controllable because of how slow it changes as the error decreases slowly compared to linear. While Softmax, in this application never really changes its percentage correctness and its weights doesn't change at all or maybe changes too small, i'd think the Softmax function is better when you have accurately selected the weights and attempted to use backpropagation just to correct it in very small manners.