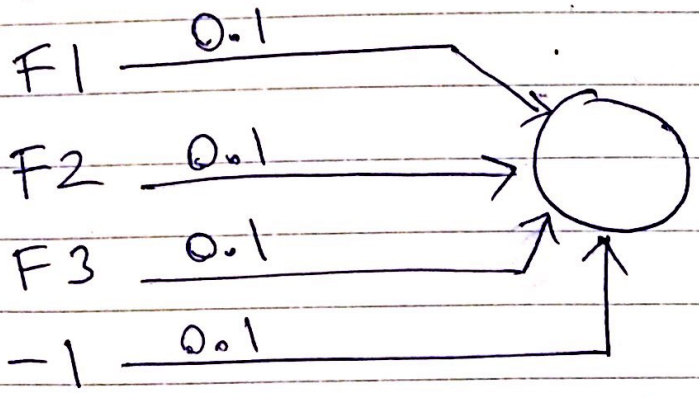
Problem 1 - Perceptron

(i)



for ex1; F1 = 1 ; F2 = 1 ; F3 = 0 ; y = 1

Weighted sum = (1 · 0.1) + (1 · 0.1) + (0.0) + (-1 · 0.1) ⩾ 0.1

$h_w(x)$ ⩾ 0.1 = threshold [weighted sum = threshold]

hence $\boxed{h_w(x) = 1}$

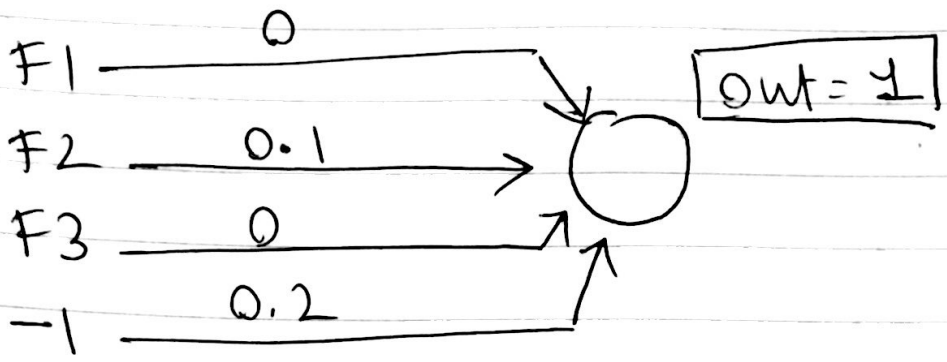
y = $h_w(x)$ hence weights don't change.

(ii) for ex2; F1 = 1 ; F2 = 1 ; F3 = ϕ ; y = ϕ

weighted sum = (1 · 0.1) + (1 · 0.1) + (0.0) + (-1 · 0.1) = 0.1

$h_w(x)$ = 0.1 (threshold = weighted sum)

but y = ϕ so need to update weights.



using perception learning rule

$$W_i = W_i + \alpha (y - h_w(x)) \times x_i$$

$$W_i = W_i + 0.1(-1) \times x_i$$

$$W_{F1} = 0.1 + 0.1(-1) \times 1 = 0$$

$$W_{F2} = 0.1 + 0.1(-1) \times 0 = 0.1$$

$$W_{F3} = 0.1 + 0.1(-1) \times 1 = 0$$

$$W_{(-1)} = 0.1 + 0.1(-1) \times -1 = 0.2$$

(see perception above)

(ii) For ex3 ; F1 = 0, F2 = 1, F3 = 1, y = 1

weighted sum = $(0 \cdot 0) + (1 \cdot 0.1) + (1 \cdot 0) + (-1 \cdot 0.2) = -0.1$

weighted sum < threshold (0.2)

hence $h_w(x) = 0$ but $y = 1$

so need to update weights

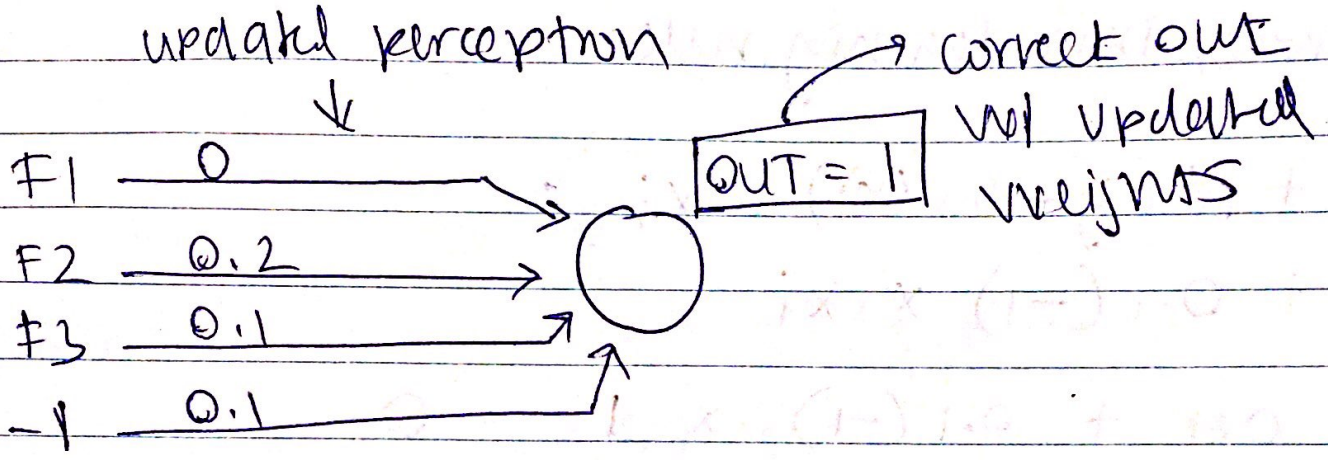
$W_i = W_i + 0.1(1) \times X_i$

$W_{F1} = 0 + (0.1 \cdot 0) = 0$

$W_{F2} = 0.1 + (0.1 \cdot 1) = 0.2$

$W_{F3} = 0 + (0.1 \cdot 1) = 0.1$

$W_{(-1)} = 0.2 + (0.1 \cdot -1) = 0.1$



(iv)

ex NEW) F1 = 0

F2 = 1

F3 = 0

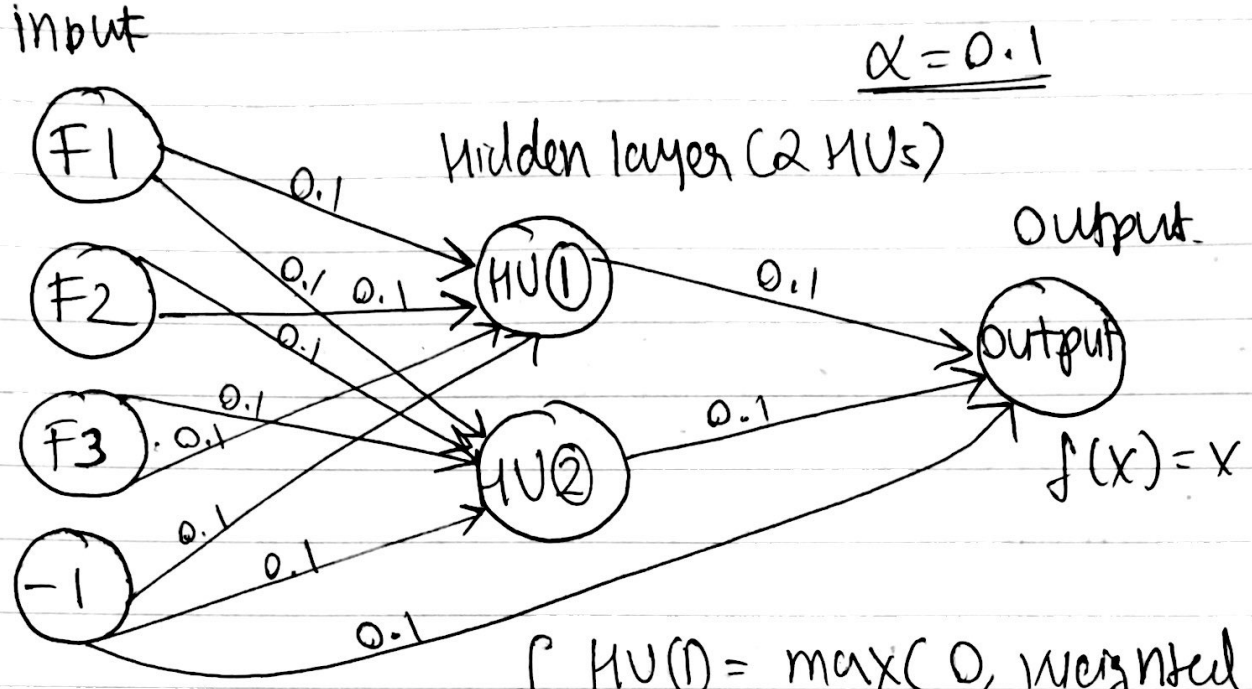
- weighted sum = (0.0) + (1.0.2) + (0.0.1) + (-1.0.1) = 0.1
- threshold (0.1) = weighted sum

hence $h_w(x) = 1$

hence **category = 1** Ans) 1

Problem 2 - Hidden Units

i)



$$\text{HU①} = \max(0, \text{weighted sum}_1)$$
$$\text{HU②} = \max(0, \text{weighted sum}_2)$$

ReLU output = [HU① · 0.1 + HU② · 0.1]

$f(x) = x$

ii) Put i) inputs are:

F1 = 1
F2 = 1
F3 = 0
F(bias)(-1)

Weighted sum for HU① =
$(0.1 \cdot 1) + (1 \cdot 0.1) + (0.1 \cdot 0) + (-1 \cdot 0.1) = 0.1$

→ HU① = max(0, 0.1) = 0.1 //

Weighted sum for HU② = 0.1

→ HU② = max(0, 0.1) = 0.1 //

output [f(x) = x]

= HU① · 0.1 + HU② · 0.1 + (-1 · 0.1

= **-0.08**

iii) $\Delta[j] = g'(in_j) \cdot (y_j - a_j)$

$\Delta[j]$ for output layer $= 1 \times (1 + 0.08)$

$\Delta_{j\,output} = 1.08$

~~$\Delta[j]$ for hidden unit 1 and 2 [same for this case]~~

~~$(g'(in) = 1 \quad = 1 \times (0.1 \times$ as $in > 0)$~~

$\Delta[i] = g'(in_i) \sum_j w_{i,j} \Delta[j]$

$\Delta[i]$ for hidden units 1 and 2 [same for this case]

$\Rightarrow g'(in_i) = 1$ as $in > 0$

$\Delta[i]_{HU1,HU2} = 1 \cdot (0.1 \cdot 1.08)$

$= 0.108$

for output layer : $\Delta[j] = 1.08$

for hidden layer $\Delta[i]_{HU1} = 0.108$; $\Delta[i]_{HU2} = 0.108$

updating weight in network using deltas

$W_{ij} \leftarrow W_{ij} + \alpha \times a_i \times \Delta[j]$; $\alpha = 0.1$

$W_{F1,HU1} = 0.1 + 0.1 \times 1 \times 0.108 = 0.1108$

$W_{F1,HU2} = 0.1 + 0.1 \times 1 \times 0.108 = 0.1108$

$W_{F2,HU1} = 0.1 + 0.1 \times 1 \times 0.108 = 0.1108$

$W_{F2,HU2} = 0.1 + 0.1 \times 1 \times 0.108 = 0.1108$

$W_{-1,HU1} = 0.1 + 0.1 \times -1 \times 0.108 = 0.0892$

$W_{-1,HU2} = 0.1 + 0.1 \times -1 \times 0.108 = 0.0892$

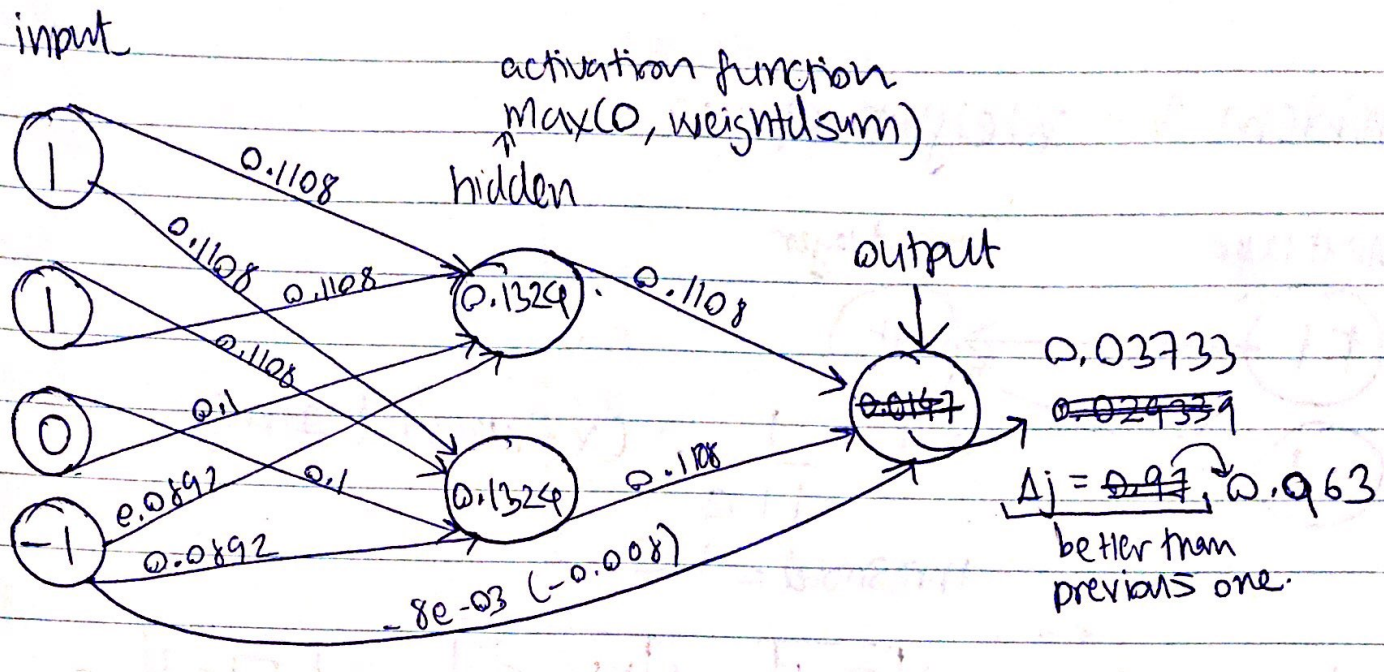
$W_{F3,HU1} = 0.1 \ldots \times 0 = 0.1$

$W_{F3,HU2} = 0.1 \ldots \times 0 = 0.1$

$W_{HU1,output} = 0.1 + 0.1 \times 0.1 \times 1.08 = 0.1108$

$W_{HU2,output} = 0.1 + 0.1 \times 0.1 \times 1.08 = 0.1108$

$W_{-1,output} = 0.1 + 0.1 \times -1 \times 1.08 = -8e-03$



* resulting neural network after applying the back propagation algorithm to Part (ii) (corresponding a_i (weighted sum) ~~case~~ case shown.

* Additional work

Weighted sum for HU1 and HU2

$= (0.1108 \times 1) + (0.1108 \times 1) + (0 \cdot 0.1) + (-1 \times 0.0892) = \max(0, 0.1324)$

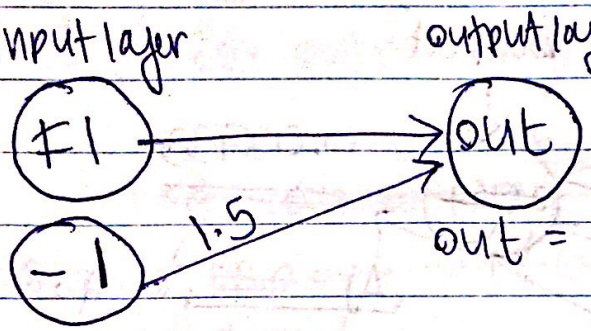
$\hookrightarrow = 0.1324$

① a_i for output layer would be then

$(0.1324 \cdot 0.1108) + (0.1324 \cdot 0.1108) + (-1 \cdot -8e-03) =$ ~~0.97~~ //

0.037339 //

Problem 3 - Weight space



$out = \frac{1}{1+e^{-x}}$ (x = weighted sum)

threshold = 1.5

~~ex1: F1=1, y=1~~	F1=1	F1=2	F1=3
weight	error(Ex1) +	error(Ex2) +	error(Ex3)
-4	0.9918 +	~~3.9995~~ 5.6e-09 +	1.87e-12
-2	0.94223 +	1.656×10^{-5} +	3.056e-7
-1	0.8540 +	8.592×10^{-4} +	1.207e-4
0	0.6689 +	0.03327 +	0.033279
1	0.3879 +	0.3874 +	0.66843
2	0.1425 +	0.854 +	0.97815
4	5.75×10^{-3} +	0.997 +	0.999994

Weight	Total error
-4	0.99
-2	0.942
-1	0.855
0	0.73494
1	1.4428
2	1.9745
4	2.002

simple calculation

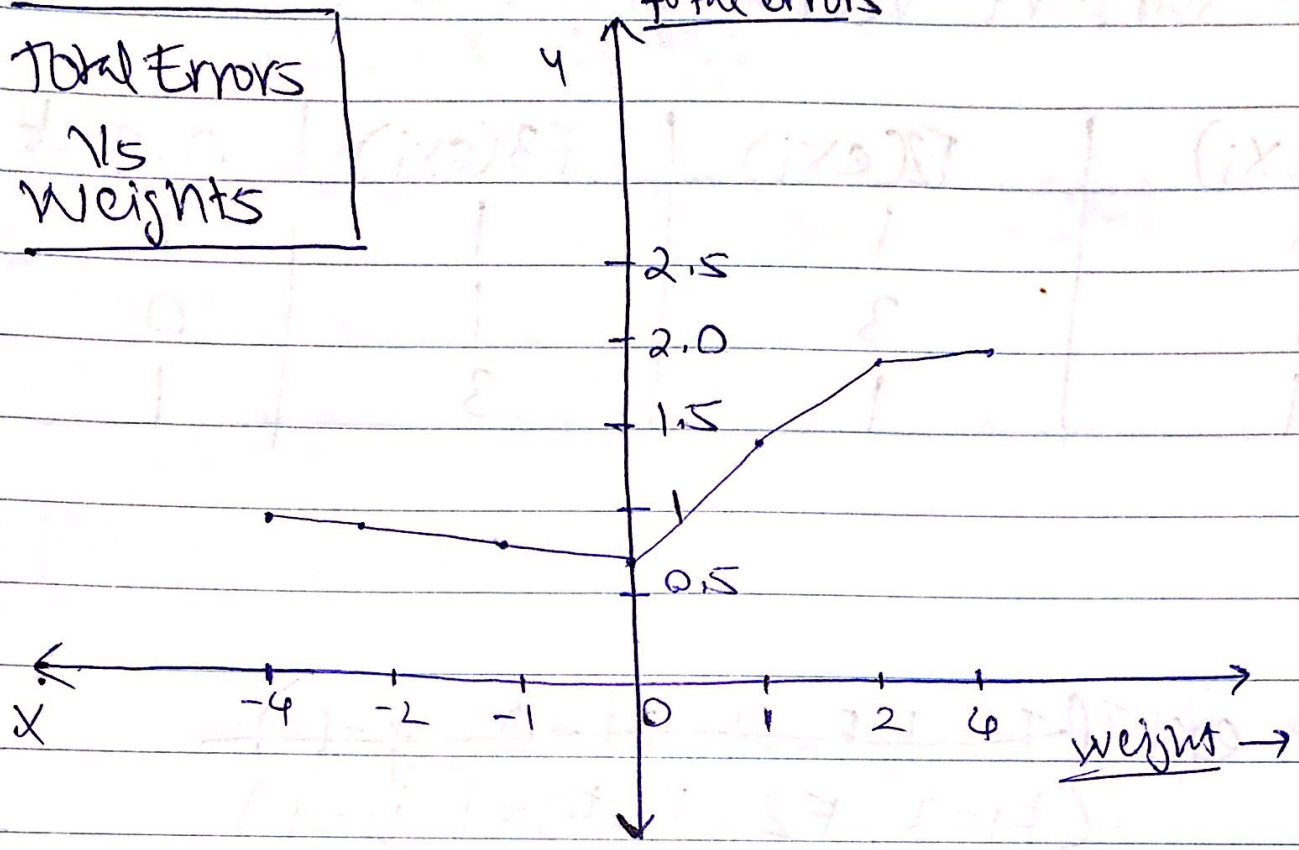
$out = \frac{1}{1+e^{-(F1 \cdot w_i + (-1.5))}}$

and

$error = [(\text{category of } ex_i) - out]^2$

total errors are populated on left

Total Errors
Vs
Weights



- graph from the data on the previous page

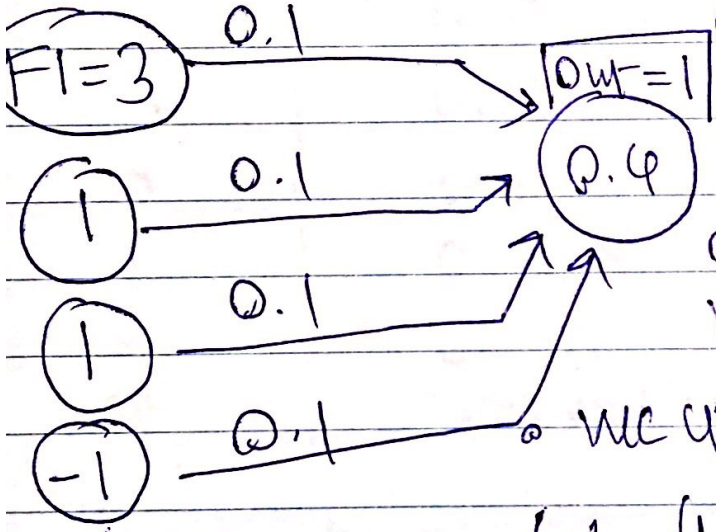
Problem 4 - Support Vector Machines

i)

	F1(exi)	F2(exi)	F3(exi)	output
x1	3	1	1	1
x2	1	3	1	0
x3	1	1	3	1

ii)

1) Training on ex1 ~~(F1=1, F2=1, F3=0, y=1)~~
(F1=3, F2=1, F3=1, y=1)



out = (3·0.1) + (1·0.1) + (1·0.1) + (-1·0.1)

0.4 > threshold [0.1]

hence output = 1 //

- We update weights regardless

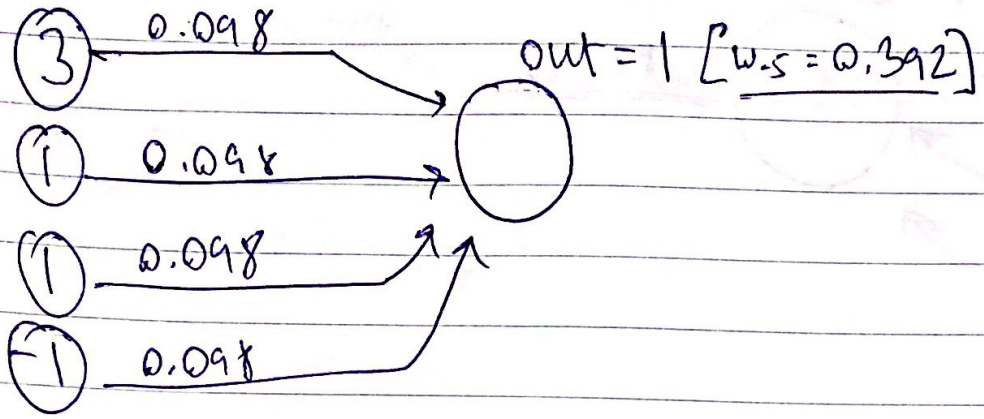
① $W_1 = W_1 + (\alpha \cdot (1-1) \cdot 3) - [\alpha \cdot \lambda \cdot W_1]$
$= 0.1 + [0.1 \cdot (0 \cdot 3)] - [0.1 \cdot 0.2 \cdot 0.1]$
$= 0.098$

① $W_2 = 0.1 + [0.1 \cdot (0 \cdot 1)] - [0.1 \cdot 0.2 \cdot 0.1]$
$= 0.098$

① $W_3 = 0.098$

① $W_4 = 0.098$

Updated perception



ii) training on ex2 [F1 = 1, F2 = 3, F3 = 1, y = ∅]

out ⇒

weighted sum = (3·0.098) + (1·0.098) + (1·0.098) + (-1·0.098)

= 0.392

out = 1 // (y = ∅)

* we update weights again:

$W_1 = 0.098 + [0.1 \times (-1) \cdot 1] - (0.1 \cdot 0.2 \cdot 0.098)$

$= -3.96 \times 10^{-3}$

$W_2 = 0.098 + [0.1 \times (-1) \cdot 3] - (0.1 \cdot 0.2 \cdot 0.098)$

$= -0.20396$

$W_3 = 0.098 + [0.1 \times (-1) \cdot 1] - (0.1 \cdot 0.2 \cdot 0.098)$

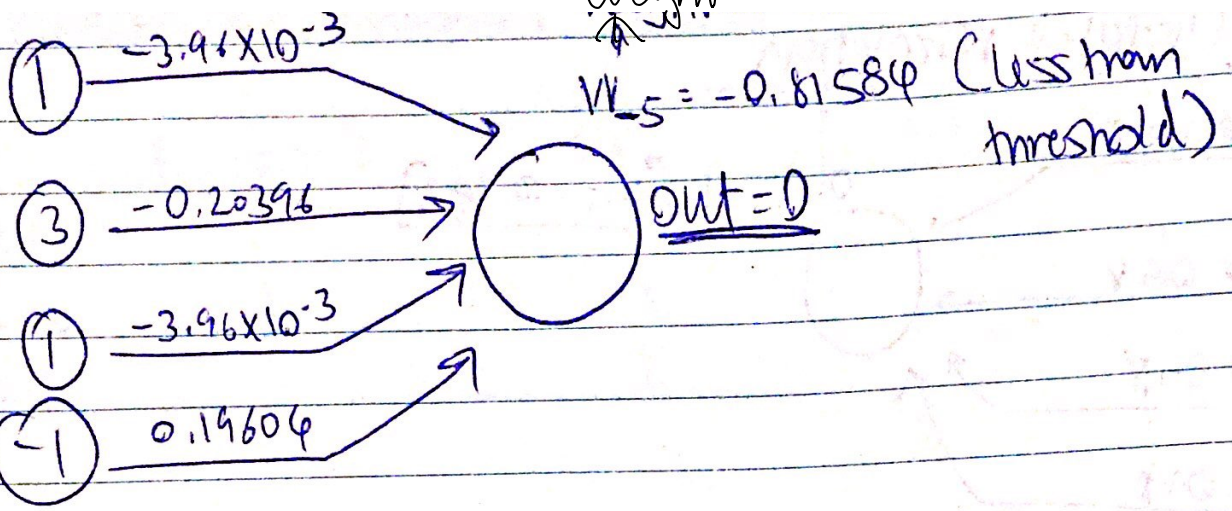
$= -3.96 \times 10^{-3}$

$W_4 = 0.098 + [0.1 \times (-1) \cdot (-1)] - (0.1 \cdot 0.2 \cdot 0.098)$

$= 0.19604$

(updated perception on next page)

weighted sum



(iii) ex2 followed by ex3

ex3 : F1 = 1 ; F2 = 1 ; F3 = 3 ; y = 1

Out = stepfunction (1 · -3.96 X10^{-3} +
1 · -0.20396 + 3 · -3.96X10^{-3}
+ (-1 · 0.19604)

= stepfunction (-0.41558)

= 0 (threshold > y)

y = 1 hypothesis

hence need to update weight

W_1 = -3.96X10^{-3} + [0.1 X 1 X 1] - [0.1 X 0.2 X -0.00396]

= 0.0961192

W_2 = -0.20396 + [0.1 X 1 X 3] - [0.1 X 0.2 X -0.20396]

= -0.09988

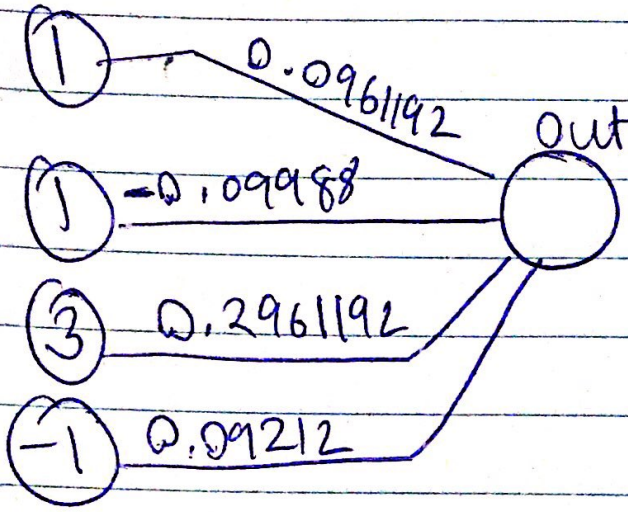
W_3 = -0.00396 + [0.1 X 1 X 3] - [0.1 X 0.2 X -0.00396]

= 0.2961192

W_4 = 0.19604 + [0.1 X 1 X -1] - (0.1 X 0.2 X 0.19604)

= 0.09212

updated perceptron after training on example 3



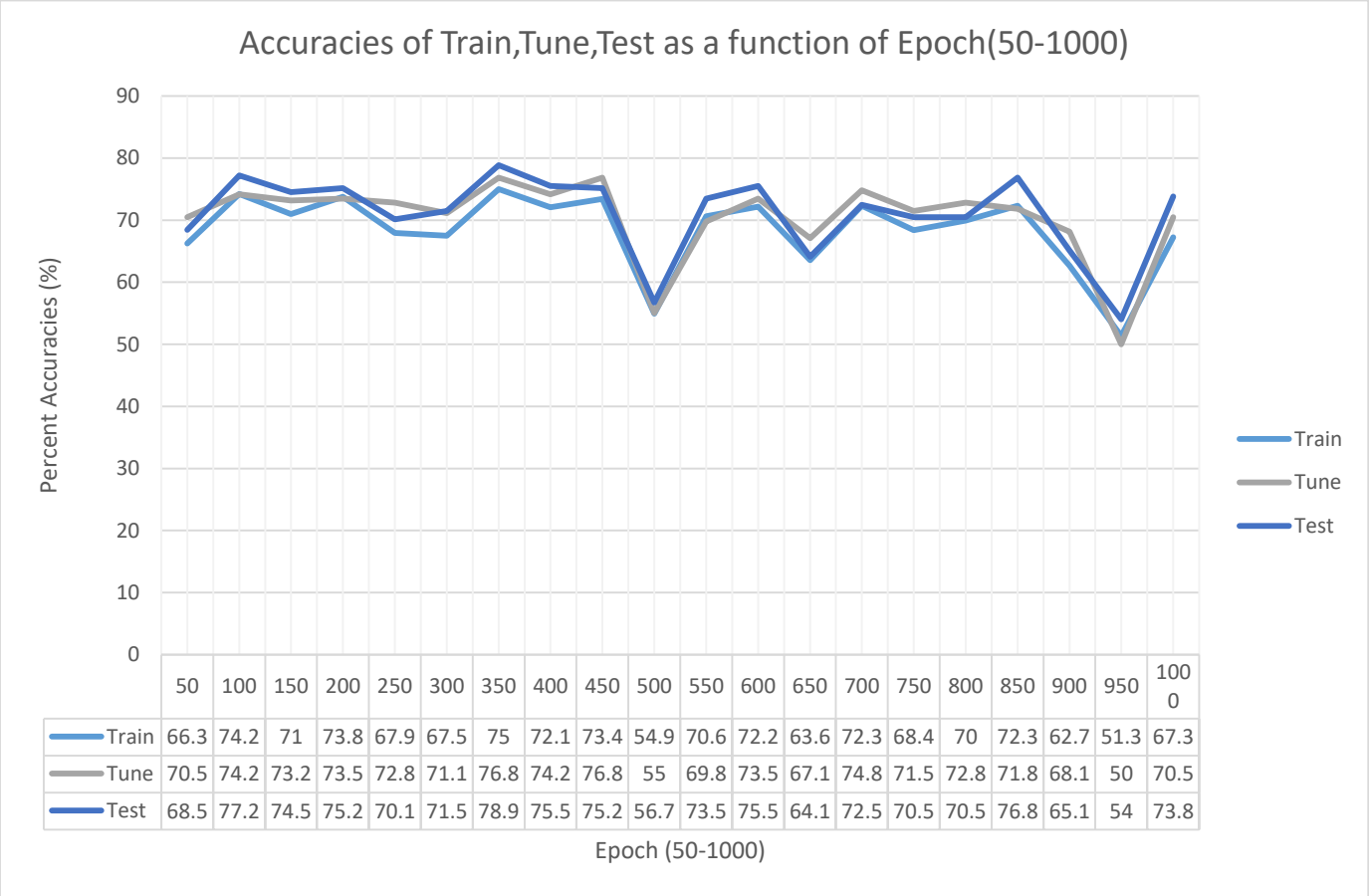
(vii) exNew

F1=0	F3=0
F2=1	out=?

- out = stepfunction(0 + (1 · -0.09988) + (0 × 0.2961192) + (-1 · 0.09212))
  = stepfunction(-0.192) = 0 //

Output to exNew = 0

HW4 Perceptron Lab Report



	50	100	150	200	250	300	350	400	450	500	550	600	650	700	750	800	850	900	950	1000
Train	66.3	74.2	71	73.8	67.9	67.5	75	72.1	73.4	54.9	70.6	72.2	63.6	72.3	68.4	70	72.3	62.7	51.3	67.3
Tune	70.5	74.2	73.2	73.5	72.8	71.1	76.8	74.2	76.8	55	69.8	73.5	67.1	74.8	71.5	72.8	71.8	68.1	50	70.5
Test	68.5	77.2	74.5	75.2	70.1	71.5	78.9	75.5	75.2	56.7	73.5	75.5	64.1	72.5	70.5	70.5	76.8	65.1	54	73.8

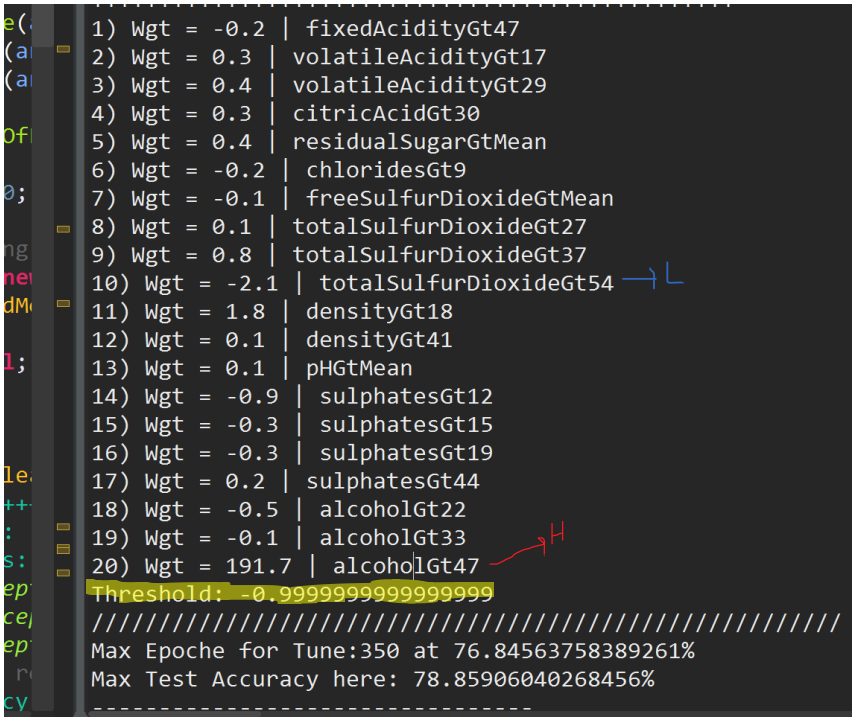
From quantitative analysis of the results, we see that after 1000 epoche (snapshot of some stats in the figure below), the max tune set accuracy is 76.84% at epoche 350, while the test set accuracy there is 78.859%. From the graph above, we can see the trend in the train, tune, and test set accuracy over different increments of epoche up to 1000. We could have factored the tune set accuracy and implemented the early stopping to avoid the fall in accuracies at around epoche 500. We can also tell from the graph that as epoche increase, that is as we make our perceptron learn repeatedly, we do not see strong indication of overfitting (or better performance on training than test data).

Qualitative analysis of results: from the results, we can say that there was no strong indication of overfitting and it looks like the train, tune, and test accuracies are strongly correlated. Also, to note that the nature of this perceptron is not that likely to overfit the data as the total number of inputs or parameters in the wine dataset is 21 and the linearity and less degrees of freedom makes is less likely to overfit. We can also say that stopping early (using the tune set) is not a significant aspect of this perceptron on the wine dataset, but the tune set did a good job of determining a stopping point at epoche 350 with the max accuracy.

In ensemble learning, we noticed higher degree of overfitting as there were greater degrees of freedom with which the bagging data could overfit. In ensemble learning, we had noticed that the best tune set accuracy was around 65% while the best test set accuracy was around 55-60%, and comparing the learning models, we clearly

see that perceptron performed better and did not overfit. I feel like a perceptron is well suited in a cross-validation selection area when you want to choose the best model, while ensemble learning can help choose the best L value that can give us the best model/results. One thing to also note the similarity of early stopping in perceptron to the best L value in bagging.

One thing that I would like to investigate in future is to determine which is a more stable learning model in the wine dataset, where I would see how much a small change to the training data would lead to significant changes in the decision. (it wouldn't be fair to test both on a data which is not linearly separable).



Snapshot of sample statistics – rounded off- and the max tune set accuracy

We see that totalSulfurDioxideGt54 has the lowest weight while alcoholGt47 has the highest weight in the best perceptron model. The threshold is -0.9999. The features closest to zero (there are multiple) include ones that are 0.1, -0.1 in the snapshot above : pHGtMean, totalSulfur…Gt27, alcohol...33, densityGt41. From my results, it looks like alcoholGt47 is a really important feature in determining the output. Not sure if the weights make sense, I'm not a wine expert ☺.

Interesting fact: I ran it **for 100000 epoche** and I got the following results:
(Note that accuracy on tune set increased by about 2%)

```
HW4 [Java Application] C:\Program Files\Java\jdk1.8.0_91\bin\javaw.exe (Nov 21, 2018, 11:30:04 PM)
Accuracy for test set: 66.10738%
++++++++++++++++++++++++++++++++++++++++++++++++
1) Wgt = -0.5 | fixedAcidityGt47
2) Wgt = 0.5 | volatileAcidityGt17
3) Wgt = 0.3 | volatileAcidityGt29
4) Wgt = 0.2 | citricAcidGt30
5) Wgt = 0.1 | residualSugarGtMean
6) Wgt = 0.0 | chloridesGt9
7) Wgt = 0.0 | freeSulfurDioxideGtMean
8) Wgt = 0.2 | totalSulfurDioxideGt27
9) Wgt = 0.9 | totalSulfurDioxideGt37
10) Wgt = -2.4 | totalSulfurDioxideGt54
11) Wgt = 2.3 | densityGt18
12) Wgt = 0.1 | densityGt41
13) Wgt = 0.1 | pHGtMean
14) Wgt = -0.5 | sulphatesGt12
15) Wgt = -0.5 | sulphatesGt15
16) Wgt = -0.1 | sulphatesGt19
17) Wgt = 0.1 | sulphatesGt44
18) Wgt = -0.3 | alcoholGt22
19) Wgt = -0.2 | alcoholGt33
20) Wgt = 22318.7 | alcoholGt47
Threshold: -1.7000000000000002
////////////////////////////////////////////////////
Max Epoche for Tune:19650 at 78.85906040268456%
Max Test Accuracy here: 79.19463087248322%
---------------------------------

Hit <enter> when ready to exit.
```