ML ASSIGNMENT 3

1. a

Page No.

Considering a NN with one hidden layer consisting of one neuron & output layer with one neuron

Initial condition

$$\text{I} \longrightarrow \text{H} \xrightarrow{\text{Relu}} \text{O} \xrightarrow{\text{Linear}}$$

Input $\qquad w_1, b_1 \qquad w_2, b_2$

$w_1 = 1$, $w_2 = +1$, ϕ = Relu func

$b_1 = -1$ $\qquad b_2 = 1$) Input $= [x_1, x_2, x_3]$

Target $= [t_1, t_2, t_3]$

Iteration/ sample $\quad$ Iteration 1 Sampl 1 $\quad x = 1.2, t = 3$

At hidden layer

$$Z_1 = w_1 x + b_1$$
$$= 1.2 - 1$$
$$= 0.2$$

Applying activation function

$$O_1 = \phi(Z_1)$$
$$= 0.2$$

At output layer

$$Z_2 = W_2 O_1 + b_2$$
$$= 1(0.2) + 1$$
$$= 1.2$$

$$O_2 = Z_2 \quad [\text{Linear}]$$

error $e = \frac{1}{2}(O_2 - t)^2$

Date:
Page No.

BPA

Weight updation, first we will calculate gradients

$$\frac{\partial E}{\partial w_2} = \left(\frac{\partial E}{\partial O_2}\right)\left(\frac{\partial O_2}{\partial Z_2}\right)\left(\frac{\partial Z_2}{\partial w_2}\right)$$

$$= (O_2 - t)(1)(O_1)$$

$$\frac{\partial E_1}{\partial b_2} = \left(\frac{\partial E}{\partial O_2}\right)\left(\frac{\partial O_2}{\partial Z_2}\right)\left(\frac{\partial Z_2}{\partial b_2}\right)$$

$$= (O_2 - t)(1)(1)$$

$$\frac{\partial E}{\partial w_1} = \left(\frac{\partial E}{\partial O_2}\right)\left(\frac{\partial O_2}{\partial Z_2}\right)\left(\frac{\partial Z_2}{\partial O_1}\right)\left(\frac{\partial O_1}{\partial Z_1}\right)\left(\frac{\partial Z_1}{\partial w_1}\right)$$

$$\frac{\partial E}{\partial w_1} = (O_2 - t_1)(1)(w_2)\left(1 \text{ if } z_1 > 0 \text{ or } 0 \text{ otherwise}\right)(x)$$

$$\frac{\partial E}{\partial b_1} = \left(\frac{\partial E}{\partial O_2}\right)\left(\frac{\partial O_2}{\partial Z_2}\right)\left(\frac{\partial Z_2}{\partial O_1}\right)\left(\frac{\partial O_1}{\partial Z_1}\right)\left(\frac{\partial Z_1}{\partial b_1}\right)$$

$$\frac{\partial E}{\partial b_1} = (O_2 - t)(1)(w_2)\left(1 \text{ if } z_1 > 0 \text{ or } 0 \text{ otherwise}\right)(1)$$

Weight updation equations

$$w_1 = w_1 - \eta\frac{\partial E}{\partial w_1}, \quad w_2 = w_2 - \eta\frac{\partial E}{\partial w_2}$$

$$b_1 = b_1 - \eta\frac{\partial E}{\partial b_1}, \quad b_2 = b_2 - \eta\frac{\partial E}{\partial b_2}$$

Calculating gradients for sample1

$$\frac{\partial E}{\partial w_2} = (1.2-3)(1)(0.2) = -0.36$$

$$\frac{\partial E}{\partial b_2} = (1.2-3)(1)(1) = -1.8$$

$$\frac{\partial E}{\partial w_1} = (1.2-3)(1)(1)(1)(1.2) = -2.16$$

$$\frac{\partial E}{\partial b_1} = (1.2-3)(1)(1)(1)(1) = -1.8$$

$w_2 = 1 - (0.01)(-0.36) = 1.0036 \approx 1$

$b_2 = 1 - (0.01)(-1.8) = 1.018 \approx 1.01$

$w_1 = 1 - (0.01)(-2.16) = 1.0216 \approx 1.02$

$b_1 = -1 - (0.01)(-1.8) = -0.982 \approx -0.98$

sample 2 $x = 0.8$, $t = 2.5$

$$z_1 = w_1 x + b$$
$$= (1.02)(0.8) + (-0.98)$$
$$= -0.164$$

$$o_1 = \phi(z_1)$$
$$= 0$$

At output layer

$$z_2 = w_2 o_1 + b_2$$
$$= (1) 0 + 1.01$$
$$= 1.01$$

$o_2 = z_2 = 1.01$

$e = \frac{1}{2}(o_2 - t)^2$

$\frac{\partial E}{\partial w_2} = (o_2 - t)(1)(o_1)$
$= 0$

$\frac{\partial E}{\partial b_2} = (o_2 - t)(1)(1) = (1.01 - 2.5)$
$= -1.49$

$\frac{\partial E}{\partial w_1} = 0 = \frac{\partial E}{\partial b_1}$ as $z_1 < 0$

Updating weights

$w_2 = w_2 - (0.01)(0) = w_2$

$w_1 = w_1$, $b_1 = b_1$ as their gradients are 0

$b_2 = b_2 - (0.01)(-1.49)$
$= 1.01 + 0.0149$
$= 1.0249 \approx 1.024$

For sample 3

$x = 2$, $t = 4$

$z_1 = w_1 x + b_1$
$= (1.02)(2) - 0.98$
$= 1.06$

$o_1 = \phi_1(z_1) = 1.06$

$z_2 = w_2 o_1 + b_2$
$= 1(1.06) + \cancel{1.02} \; 1.024$
$= 2.08$

$$O_2 = z_2$$
$$= 2.07$$

$$e = \frac{1}{2}(O_2 - t)^2$$

$$\frac{\partial E}{\partial w_2} = (O_2 - t)(1)(O_1)$$
$$= (2.08 - 4)(1)(1.06)$$
$$= -2.045$$

$$\frac{\partial E}{\partial b_2} = (O_2 - t)(1)(1) = -1.92$$

$$\frac{\partial E}{\partial w_1} = (-1.93)(1)(1)(1)(2) = -3.84$$

$$\frac{\partial E}{\partial b_1} = (-1.93)(1)(1)(1)(1) = -1.92$$

$$w_2 = w_2' - (0.01)(-2.045) = 1.02045$$

$$b_2 = b_2 - (0.01)(-1.92) = \cancel{1.0192}\ 1.021$$
$$= \cancel{2.0432}$$

$$w_1 = w_1 - (0.01)(-3.84) = 1.0584$$

$$b_1 = b_1 - (0.01)(-1.92) = -0.9608$$

1. b)

The classification rule can be expressed as follows

$$= \sum_{i=1}^{N} \alpha_i y_i K(x_i, x_{test}) + b$$

where

→ b is bias

→ α_i are lagrangian multipliers

→ y_i is true label for ith sample

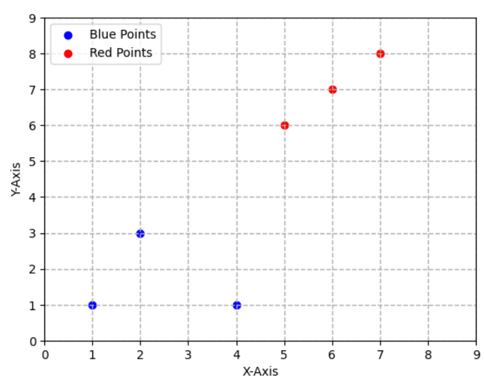
→ N is number of support vectors

$$K(x_i, x_{test}) = \exp\left(\left(\frac{-\|x_i - x_{test}\|^2}{2\sigma^2}\right)\right)$$

↓

gaussian kernel

c



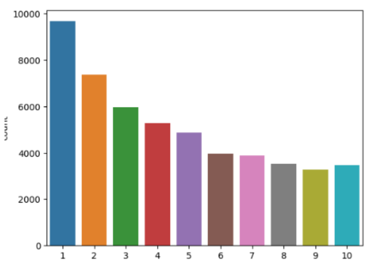
a) From the graph it is clear that points are linearly separable.
b) Line joining the point (4, 1) and (2, 3) is y + x = 5 and the line passing through (5, 6) parallel to it is y+x = 11 so the optimal boundary will be parallel to these lines and midway between them. So optimal boundary equation is y+x - 8 = 0
c) The support vectors will be (4, 1), (2, 3) and (5, 6).
d) Margin of the decision boundary will be $|(y + x - 8)/\sqrt{2}|$. It will be $3/\sqrt{2}$ for the support vectors.

e) If we remove only (4, 1) or (2, 3) then the decision boundary will not change but if we remove both of them then the decision boundary will change as (1, 1) will become the support vector. If we remove (5, 6) then the decision boundary will change.

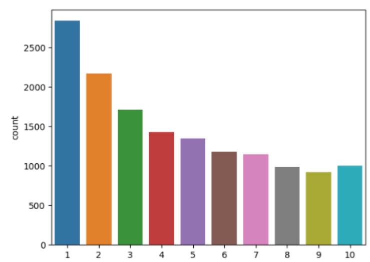
Section C

1. Splitting the data into training, testing and validation

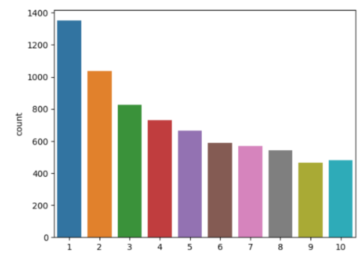
```
X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.3, random_state=42)
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.33, random_state=42)
```

2. 

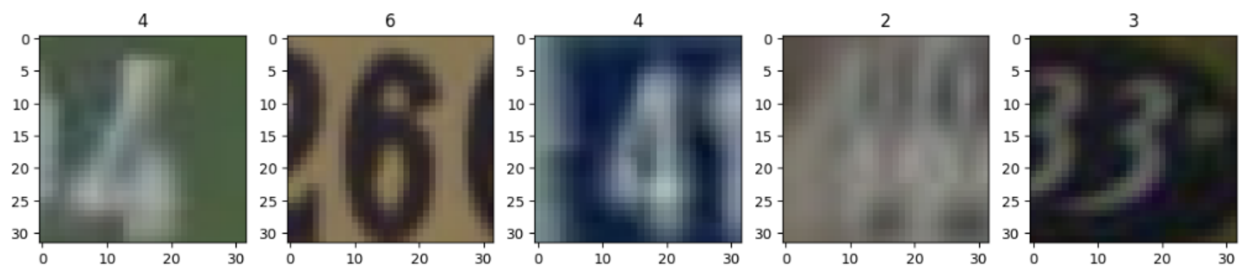
Train Distribution



Val Distribution



Test Distribution

3. 

4. Constructed a neural network with 2 hidden layers and used grid search to find the optimal parameters.

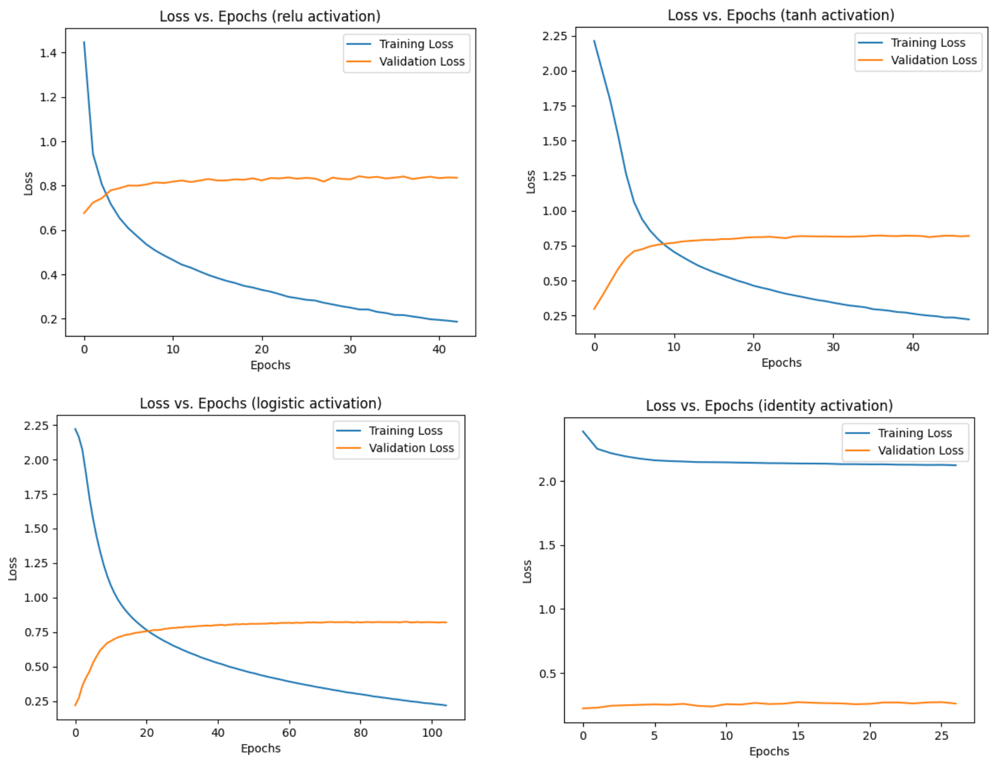
```
param_grid = {
    'hidden_layer_sizes': [(128, 64), (256, 128), (512, 256)],
    'solver': ['adam'],
    'batch_size' : [32, 64, 128],
    'learning_rate_init' : [0.1, 0.01, 0.001, 0.0001]
}

model = MLPClassifier(shuffle = True, early_stopping = True, max_iter=300)
grid_search = GridSearchCV(model, param_grid, cv=5, verbose = 5)
```

5. The optimal parameters were hidden_layer_size = (128, 64), batch_size = 64, learning_rate_init = 1e-4 with a cross validation accuracy of 80 percent and test accuracy of 83.11 percent. The validation accuracy was 82 percent.

```
print(accuracy_score(y_pred, y_test))
```

```
0.8311043706052668
```

6.

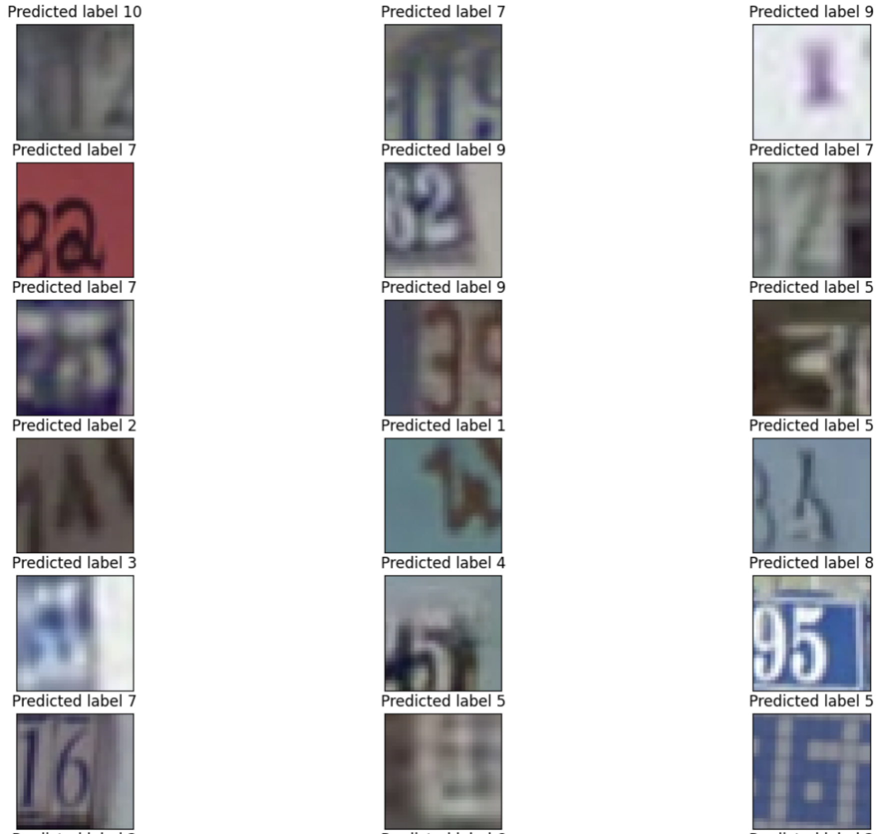


Since validation loss was not a parameter I have plotted validation score for the activation functions respectively.

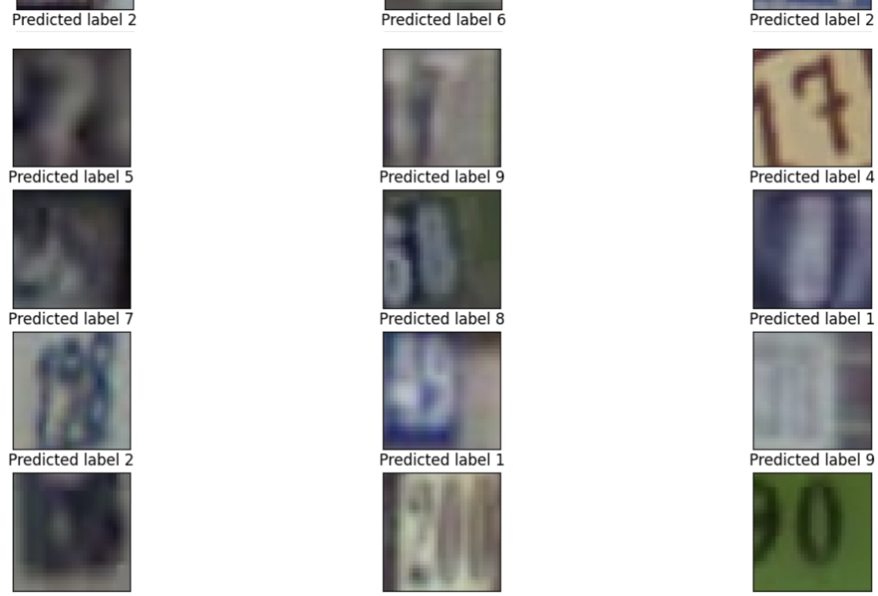
7. Testing Accuracy with ReLu activation function 83.6343581966083,
Testng Accuracy with tanh activation function 81.58003584723562,
Testing Accuracy with logistic activation function 82.28319316145043,
Testing Accuracy with identity activation function 27.037088101475254

   The model was able to learn well with ‘ReLu’, ‘tanh’ and ‘logistic’ function but it was unable to learn well with the identity function. I think that the reason behind this is a non-linear decision boundary of the dataset which could not be achieved through identity function.

8. Yes my model is able to achieve a decent accuracy as the best test accuracy is 83.63 percent. I was able to reach this through grid search.



9.



The images for the test labels are shown in sequence starting from 1 to 10. There are multiple reasons for the mis-classification of the images' poor quality, incorrectly labeled images for a 20 in the set of 10 or multiple digits in the image such as 198 in case of 9 which might be confusing the model.