Assignment 4 and 5 Report
Machine Learning (CS-596)

Name: Dhaval Harish Sharma

Red ID: 824654344

Assignment 4 FNN:

I implemented 3 models with different hyper parameters in this part.

Model 1:

2 Dense layers, 512 – output node

Relu activation function

Softmax output function

Adam optimizer

```
Model: "sequential_3"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_7 (Dense)              (None, 512)               401920
_________________________________________________________________
dense_8 (Dense)              (None, 10)                5130
=================================================================
Total params: 407,050
Trainable params: 407,050
Non-trainable params: 0
```

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/5
WARNING:tensorflow:Entity <function
Function._initialize_uninitialized_variables.<locals>.initialize_variables at
0x00000163AF276DC8> could not be transformed and will be executed as-is. Please report
this to the AutoGraph team. When filing the bug, set the verbosity to 10 (on Linux,
`export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause:
WARNING: Entity <function
Function._initialize_uninitialized_variables.<locals>.initialize_variables at
0x00000163AF276DC8> could not be transformed and will be executed as-is. Please report
this to the AutoGraph team. When filing the bug, set the verbosity to 10 (on Linux,
`export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause:
50000/50000 [==============================] - 11s 223us/sample - loss: 0.2159 -
accuracy: 0.9366 - val_loss: 0.1154 - val_accuracy: 0.9675
Epoch 2/5
50000/50000 [==============================] - 10s 199us/sample - loss: 0.0882 -
accuracy: 0.9728 - val_loss: 0.0898 - val_accuracy: 0.9726
Epoch 3/5
50000/50000 [==============================] - 9s 187us/sample - loss: 0.0558 -
accuracy: 0.9829 - val_loss: 0.0762 - val_accuracy: 0.9789
Epoch 4/5
50000/50000 [==============================] - 9s 183us/sample - loss: 0.0391 -
accuracy: 0.9880 - val_loss: 0.0766 - val_accuracy: 0.9794
Epoch 5/5
50000/50000 [==============================] - 10s 203us/sample - loss: 0.0290 -
accuracy: 0.9905 - val_loss: 0.0668 - val_accuracy: 0.9803
```

Model 2:

3 Dense layers

Relu activation function

Sigmoid output function

RMSprop optimizer with 0.001 learning rate

```
Model: "sequential_4"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_9 (Dense)              (None, 512)               401920
_________________________________________________________________
dense_10 (Dense)             (None, 64)                32832
_________________________________________________________________
dense_11 (Dense)             (None, 10)                650
=================================================================
Total params: 435,402
Trainable params: 435,402
Non-trainable params: 0
```

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/5
WARNING:tensorflow:Entity <function
Function._initialize_uninitialized_variables.<locals>.initialize_variables at
0x00000163B0F9E8B8> could not be transformed and will be executed as-is. Please report
this to the AutoGraph team. When filing the bug, set the verbosity to 10 (on Linux,
`export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause:
WARNING: Entity <function
Function._initialize_uninitialized_variables.<locals>.initialize_variables at
0x00000163B0F9E8B8> could not be transformed and will be executed as-is. Please report
this to the AutoGraph team. When filing the bug, set the verbosity to 10 (on Linux,
`export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause:
50000/50000 [==============================] - 17s 337us/sample - loss: 0.2435 -
accuracy: 0.9280 - val_loss: 0.1174 - val_accuracy: 0.9663
Epoch 2/5
50000/50000 [==============================] - 15s 303us/sample - loss: 0.1045 -
accuracy: 0.9699 - val_loss: 0.1044 - val_accuracy: 0.9716
Epoch 3/5
50000/50000 [==============================] - 14s 283us/sample - loss: 0.0737 -
accuracy: 0.9790 - val_loss: 0.0978 - val_accuracy: 0.9742
Epoch 4/5
50000/50000 [==============================] - 15s 301us/sample - loss: 0.0584 -
accuracy: 0.9841 - val_loss: 0.1094 - val_accuracy: 0.9745
Epoch 5/5
50000/50000 [==============================] - 13s 260us/sample - loss: 0.0456 -
accuracy: 0.9872 - val_loss: 0.0900 - val_accuracy: 0.9786
```

Model 3:

2 Dense layers, 128 – output node

Relu activation function

Linear output function

RMSprop optimizer with 0.01 learning rate

```
Model: "sequential_5"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_12 (Dense)             (None, 128)               100480
_________________________________________________________________
dense_13 (Dense)             (None, 10)                1290
=================================================================
Total params: 101,770
Trainable params: 101,770
Non-trainable params: 0
```

```
Train on 50000 samples, validate on 10000 samples
Epoch 1/5
WARNING:tensorflow:Entity <function
Function._initialize_uninitialized_variables.<locals>.initialize_variables at
0x00000163A4835D38> could not be transformed and will be executed as-is. Please report
this to the AutoGraph team. When filing the bug, set the verbosity to 10 (on Linux,
`export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause:
WARNING: Entity <function
Function._initialize_uninitialized_variables.<locals>.initialize_variables at
0x00000163A4835D38> could not be transformed and will be executed as-is. Please report
this to the AutoGraph team. When filing the bug, set the verbosity to 10 (on Linux,
`export AUTOGRAPH_VERBOSITY=10`) and attach the full output. Cause:
50000/50000 [==============================] - 7s 148us/sample - loss: 2.3125 -
accuracy: 0.1304 - val_loss: 2.3026 - val_accuracy: 0.1275
Epoch 2/5
50000/50000 [==============================] - 6s 126us/sample - loss: 2.3026 -
accuracy: 0.1247 - val_loss: 2.3026 - val_accuracy: 0.1275
Epoch 3/5
50000/50000 [==============================] - 6s 125us/sample - loss: 2.3026 -
accuracy: 0.1247 - val_loss: 2.3026 - val_accuracy: 0.1275
Epoch 4/5
50000/50000 [==============================] - 7s 135us/sample - loss: 2.3026 -
accuracy: 0.1247 - val_loss: 2.3026 - val_accuracy: 0.1275
Epoch 5/5
50000/50000 [==============================] - 6s 126us/sample - loss: 2.3026 -
accuracy: 0.1247 - val_loss: 2.3026 - val_accuracy: 0.1275
```

Observation:

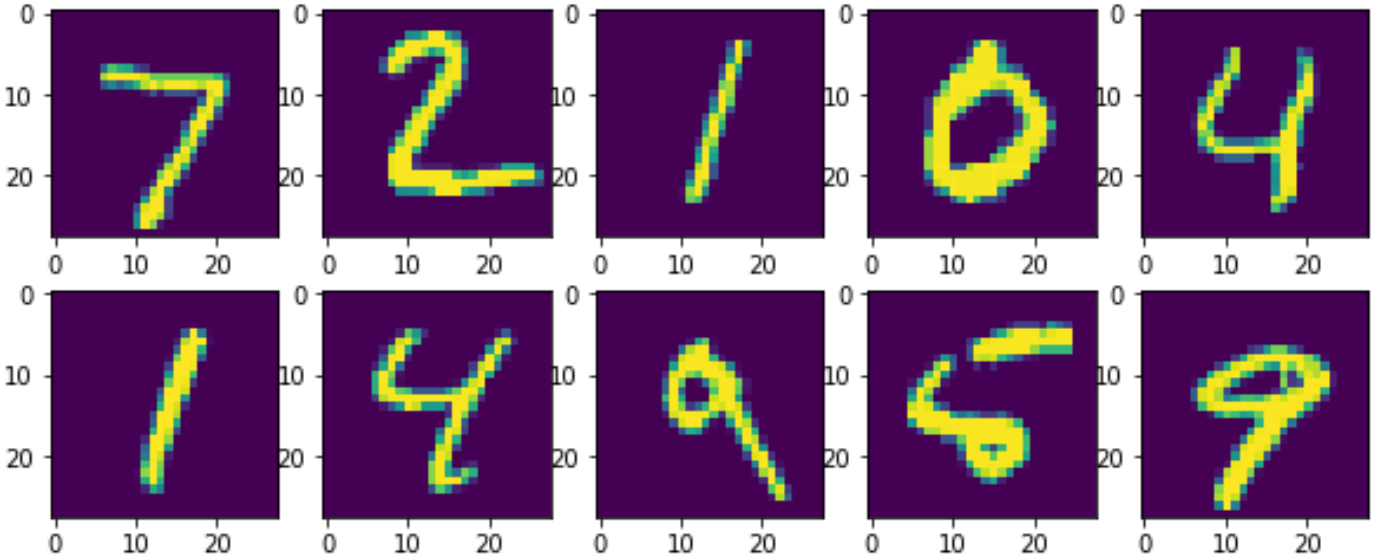
I trained the 3 models and tested those on the validation data. The performance of the 3rd model is the worst. Performance of 1st and 2nd model is almost similar. But 2nd model is a little bit better than the 1st model. I tested 2nd model on the testing data. After that I generated confusion matrix, accuracy, recall per class and precision per class.

```
Confusion Matrix :
 [[ 967    1    2    2    3    0    3    0    1    1]
 [   0 1129    2    2    0    0    2    0    0    0]
 [   2    1 1016    3    1    0    1    4    3    1]
 [   1    0    3  993    0    6    0    4    2    1]
 [   0    1    4    0  965    0    3    2    0    7]
 [   2    0    0    7    1  875    3    0    2    2]
 [   3    3    0    1    1    5  944    0    1    0]
 [   0    6   11    2    1    1    0 1000    3    4]
 [   0    2    6   10    7    7    3    3  933    3]
 [   0    4    0    6   15    7    1    8    0  968]]


Accuracy :  0.979
```

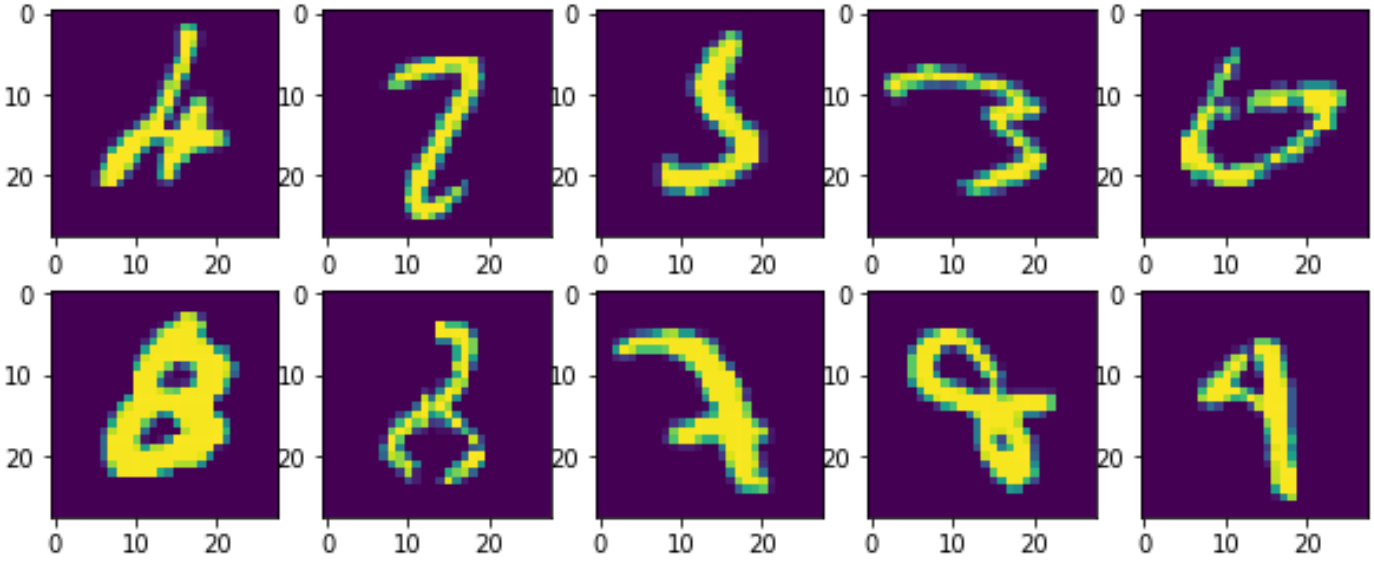
```
Class  0  Recall :  0.986734693877551
Class  1  Recall :  0.9947136563876652
Class  2  Recall :  0.9844961240310077
Class  3  Recall :  0.9831683168316832
Class  4  Recall :  0.9826883910386965
Class  5  Recall :  0.9809417040358744
Class  6  Recall :  0.9853862212943633
Class  7  Recall :  0.9727626459143969
Class  8  Recall :  0.9579055441478439
Class  9  Recall :  0.9593657086223984



Class  0  Precision :  0.9917948717948718
Class  1  Precision :  0.984306887532694
Class  2  Precision :  0.9731800766283525
Class  3  Precision :  0.9678362573099415
Class  4  Precision :  0.9708249496981891
Class  5  Precision :  0.9711431742508324
Class  6  Precision :  0.9833333333333333
Class  7  Precision :  0.9794319294809011
Class  8  Precision :  0.9873015873015873
Class  9  Precision :  0.9807497467071935
```

Next thing, I have done is the plotting of classified and misclassified images.

```
Classified Images
True Label      Predicted Label
7                7
2                2
1                1
0                0
4                4
1                1
4                4
9                9
5                5
9                9
```



```
Misclassified Images
True Label      Predicted Label
4                2
2                7
5                3
3                7
6                0
8                2
8                2
7                2
8                4
4                9
```



Few Reasons of Misclassifications:

- Blur images
- Resemblance with other digits

Assignment 5 SVM:

Problem Statement:

In this assignment, I tried to build a classifier that can identify the gender of a crab from its physical measurements. For every crab, there are six physical features: species, front-allip, rear- width, length, width and depth. We need to train a binary SVM model from a set of training samples and apply this model over the testing samples to evaluate its performance.

The dataset is provided in the file 'crab.csv', which 200 samples with gender labels (1: male, -1: female).

Steps:

Step-1: First step is to load the data from 'crab.csv' to get feature matrix X and label vector Y.

Step-2: Then I divide the training set into two EVEN subsets: one for training the model, and another for validation.

Step-3: Then, I tried out various values of C hyper-parameter keeping another parameter as fixed and keeping kernel as "Linear". So, with this, I got the best C value.

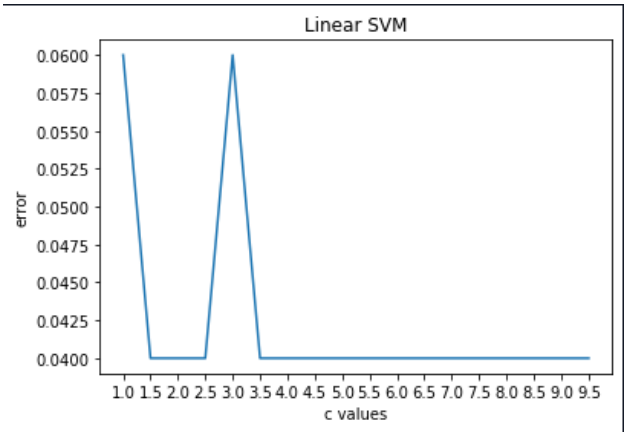
Step-4: Then, with keeping the C value as best C value, I tried 3 different kernels "Linear", "Polynomial" and "RBF".

So, after this step, I got the best kernel value as well as best C value. Taking this combination into consideration, I trained the model and test it on test data.

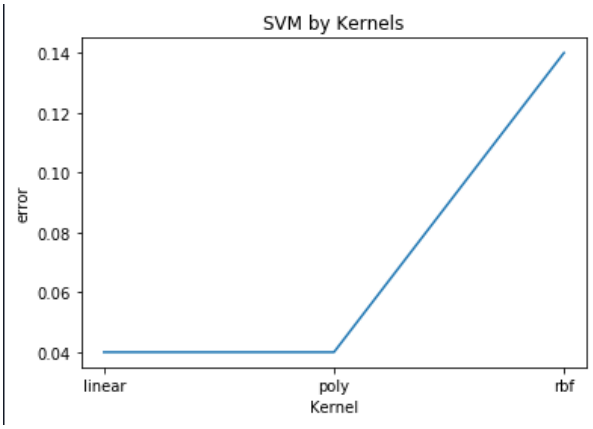
Step-5: At last, I evaluated my model's performance using confusion matrix and other metrics, including accuracy, precision and recall rates. I have also included 5 success examples and 5 failure examples in this report.

3.1 Plot the validation errors using different values of C (with other model parameters fixed). Use at least 3 different values for C.

1) c_range = np.arange(1,10,0.5)



3.2 Plot the validation errors while using linear, RBF kernel, or Polynomial kernel (with other hyper-parameters fixed);



Metrics:

```
BEST c value: 1.5
BEST kernel value: linear
Confusion Matrix:
[[43  4]
 [ 2 51]]
Average Accuracy: 0.94
Per-Class Precision: [0.95555556 0.92727273]
Per-Class Recall: [0.91489362 0.96226415]
```

Step 6:

Code was written to derive failure and true cases from the testing samples.
Failure occurs when
Case 1: Male is identified as female that is 1 is predicted as -1.
Case 2: Female is identified as male that is -1 is predicted as 1.
In all other cases true value predicted

Output:

```
--------Success Examples--------
                      Crab_features  Ground Truth  Prediction
0  [1.0, 12.6, 12.2, 26.1, 31.6, 11.2]          1.0         1.0
1    [1.0, 10.8, 9.5, 22.5, 26.3, 9.1]          1.0         1.0
2  [1.0, 11.6, 11.0, 24.6, 28.5, 10.4]          1.0         1.0
3  [0.0, 15.6, 13.5, 31.2, 35.1, 14.1]          1.0         1.0
4  [0.0, 17.5, 12.7, 34.6, 38.4, 16.1]         -1.0        -1.0
--------Failure Examples--------
                      Crab_features  Ground Truth  Prediction
0     [1.0, 9.5, 8.2, 19.6, 22.4, 7.8]          1.0        -1.0
1  [0.0, 15.0, 12.3, 30.1, 33.3, 14.0]          1.0        -1.0
2  [1.0, 11.8, 10.5, 25.2, 29.3, 10.3]         -1.0         1.0
3     [1.0, 9.1, 8.2, 19.2, 22.2, 7.7]          1.0        -1.0
4  [0.0, 12.5, 10.0, 24.1, 27.0, 10.9]          1.0        -1.0
```