Capstone Project Report

Name: Sumit Patel

Course: AI and ML (Batch – AUG 2020)

Duration: 10 months

Handwritten Digit Recognition

Problem Statement:

Use MNIST dataset to create a classifier for all the 10 digits. First implement the classifier by squeezing the image into a vector and then using a MLP. Now, try the same task using a different machine learning classifier such as an SVM to check the gain in performance by using perceptrons as compared to conventional machine learning techniques.

Prerequisites

What things you need to install the software and how to install them:

Python 3.6 This setup requires that your machine has latest version of python. The following url https://www.python.org/downloads/ can be referred to download python. Once you have python downloaded and installed, you will need to setup PATH variables (if you want to run python program directly, detail instructions are below in how to run software section). To do that check this: https://www.pythoncentral.io/add-python-to-path-python-is-not-recognized-as-an-internal-or-externalcommand/. Setting up PATH variable is optional as you can also run program without it and more instruction are given below on this topic. Second and easier option is to download anaconda and use its anaconda prompt to run the commands.

To install anaconda check this url https://www.anaconda.com/download/ You will also need to download and install below 3 packages after you install either python or anaconda from the steps above Sklearn (scikit-learn) numpy scipy if you have chosen to install python 3.6 then run below commands in command prompt/terminal to install these packages:

pip install numpy

pip install tensorflow

pip install matplotlib

pip install sklearn

If you have chosen to install anaconda then run below commands in anaconda prompt to install these packages:

conda install -c anaconda numpy

conda install -c anaconda tensorflow

conda install -c anaconda matplotlib

conda install -c anaconda sklearn

Dataset used:

mnist = tf.keras.datasets.mnist.load_data()

Method used for detection

Apriori, Association_rules

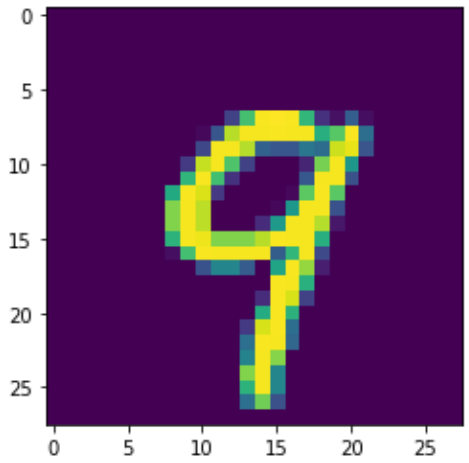
Importing the libraries and loading dataset.

```
# Import libraries
import matplotlib.pyplot as plt
import numpy as np
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.callbacks import ModelCheckpoint, TensorBoard
from tensorflow.keras.layers import Dense, Flatten, Softmax
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
```

Training Data:

```
IMGNO = 12
# Uncomment to see raw numerical values.
# print(x_test[IMGNO])
plt.imshow(x_test[IMGNO].reshape(HEIGHT, WIDTH));
print("The label for image number", IMGNO, "is", y_test[IMGNO])
```

```
The label for image number 12 is 9
```




```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

```
((60000, 28, 28), (10000, 28, 28), (60000,), (10000,))
```

```
x_train = x_train.reshape(len(x_train), 784)
x_test = x_test.reshape(len(x_test), 784)
```

```
model = MLPClassifier(hidden_layer_sizes = (512, 256, 128 ), batch_size = 128, verbose = True, early_stopping = True)
model.fit(x_train, y_train)
```

```
Iteration 1, loss = 1.35578617
Validation score: 0.916000
Iteration 2, loss = 0.25114493
Validation score: 0.939167
Iteration 3, loss = 0.13670175
Validation score: 0.955333
Iteration 4, loss = 0.10572811
Validation score: 0.956167
Iteration 5, loss = 0.09063741
Validation score: 0.957500
Iteration 6, loss = 0.08551972
Validation score: 0.957833
```

Test Results

```
y_pred = model.predict(x_test)
print(classification_report(y_pred, y_test))
```

```
              precision    recall  f1-score   support

           0       0.99      0.97      0.98       993
           1       0.99      0.98      0.99      1150
           2       0.97      0.98      0.97      1027
           3       0.98      0.97      0.98      1014
           4       0.97      0.98      0.97       969
           5       0.97      0.98      0.98       881
           6       0.99      0.99      0.99       958
           7       0.96      0.98      0.97      1007
           8       0.98      0.98      0.98       975
           9       0.97      0.96      0.97      1026

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```

Classification Using SVM

```
from sklearn import svm
```

```
clf = svm.SVC(decision_function_shape='ovo')
clf.fit(x_train, y_train)
```

```
SVC(decision_function_shape='ovo')
```

```
y_pred_svm = clf.predict(x_test)
print(classification_report(y_pred_svm, y_test))
```

```
              precision    recall  f1-score   support

           0       0.99      0.98      0.99       993
           1       0.99      0.99      0.99      1139
           2       0.97      0.98      0.98      1031
           3       0.99      0.97      0.98      1021
           4       0.98      0.98      0.98       978
           5       0.98      0.99      0.98       883
           6       0.99      0.99      0.99       958
           7       0.97      0.98      0.97      1021
           8       0.98      0.97      0.97       978
           9       0.96      0.97      0.97       998

    accuracy                           0.98     10000
   macro avg       0.98      0.98      0.98     10000
weighted avg       0.98      0.98      0.98     10000
```