MACHINE LEARNING ASSIGNMENT

Dr. Arun Raman

By-
Sneha Subramaniam
ENG20AM0054
Khushi Kumari
ENG21AM3017

Problem statement: Build a k nearest neighbor classifier in Python to classify the MNIST digit data. This is a multi-class classification problem with labels from 0 to 9.

1. The value of k
   USNs - ENG20AM0054 & ENG21AM3017 maximum of the last two digits of both USNs - max{17, 54} = 54
   k value for our group = 54

2. Number of data points
   n value for our group: (54+17)*100+1000 = 8100

3. CODE:

   PART1: KNN using functions

```
# import the necessary packages
from __future__ import print_function
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report
from sklearn import datasets
from skimage import exposure
import numpy as np
import imutils
import cv2
import sklearn
from sklearn.metrics import confusion_matrix

from sklearn.model_selection import train_test_split
(trainData, testData, trainLabels, testLabels) =
train_test_split(np.array(mnist.data),mnist.target, test_size=0.2)
```

```
mnist = datasets.load_digits()
trainData = trainData[:8100]
knn = KNeighborsClassifier(n_neighbors=54)
knn.fit(trainData,trainLabels)
y_pred = knn.predict(testData)
print(y_pred)
n_cross_val = 54
cm_prev = np.zeros((10,10))
for i in range(n_cross_val):
  cm_now = cm_prev + confusion_matrix(testLabels, y_pred)
  cm_prev = cm_now

print(cm_now/n_cross_val)
```

OUTPUT:

```
[[35.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0. 34.  2.  1.  0.  0.  0.  0.  0.  1.]
 [ 0.  1. 31.  0.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0. 39.  0.  1.  0.  1.  0.  0.]
 [ 0.  1.  0.  0. 42.  0.  0.  1.  0.  0.]
 [ 0.  0.  0.  0.  0. 29.  1.  0.  0.  3.]
 [ 1.  0.  0.  0.  0.  0. 27.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0. 37.  0.  0.]
 [ 0.  4.  0.  0.  0.  0.  0.  0. 29.  0.]
 [ 1.  3.  0.  0.  0.  1.  0.  1.  0. 33.]]
```

Link for google collab code:
https://colab.research.google.com/drive/1D7_gAnilLO0JJcLve4G8OMLAuk87kY-j?usp=sharing

PART 2: KNN from scratch

```
#from keras.datasets import mnist
from matplotlib import pyplot
from sklearn import neighbors, datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix
import numpy as np
import operator
from operator import itemgetter
from sklearn.datasets import load_digits

#euclidean distance formula
def euc_dist(x1, x2):
    return np.sqrt(np.sum((x1-x2)**2))

#knn function
class KNN:
    def __init__(self, K=54):
        self.K = K
    def fit(self, x_train, y_train):
        self.X_train = x_train
        self.Y_train = y_train

#predict function
def predict(self, X_test):
    predictions = []
    for i in range(len(X_test)):
        dist = np.array([euc_dist(X_test[i], x_t) for x_t in
        self.X_train])
        dist_sorted = dist.argsort()[:self.K]
        neigh_count = {}
        for idx in dist_sorted:
            if self.Y_train[idx] in neigh_count:
                neigh_count[self.Y_train[idx]] += 1
            else:
                neigh_count[self.Y_train[idx]] = 1
        sorted_neigh_count = sorted(neigh_count.items(),
        key=operator.itemgetter(1), reverse=True)
        predictions.append(sorted_neigh_count[0][0])
    return predictions
```

```
mnist = load_digits()
#(X_train, y_train), (X_test, y_test) = mnist.load_data()
X_train = X_train[:8100]
X = mnist.data
y = mnist.target
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=123)
#confusion matrix generation
k_neighbors = 54
k_cross_val = 54

cm_prev = np.zeros((10,10))
for i in range(k_cross_val):
  cm_now = cm_prev + confusion_matrix(y_test, y_pred)
  cm_prev = cm_now

print(cm_now/k_cross_val)
```

Output:

```
[[33.  0.  0.  0.  0.  0.  0.  0.  0.  0.]
 [ 0. 27.  4.  0.  0.  0.  0.  0.  0.  0.]
 [ 1.  0. 32.  0.  0.  0.  0.  1.  1.  0.]
 [ 0.  0.  0. 29.  0.  0.  0.  0.  0.  0.]
 [ 0.  0.  0.  0. 45.  0.  0.  0.  1.  0.]
 [ 0.  0.  0.  0.  1. 32.  0.  0.  0.  2.]
 [ 0.  0.  0.  0.  0.  0. 43.  0.  0.  0.]
 [ 0.  0.  0.  0.  0.  0.  0. 42.  0.  0.]
 [ 0.  5.  1.  0.  0.  1.  1.  0. 25.  0.]
 [ 0.  2.  0.  0.  0.  0.  0.  1.  1. 29.]]
```

Link for google collab code:
https://colab.research.google.com/drive/1dClmODCSuZXykf8TzPrErOsVSfIZVmqh?usp=sharing

CITATIONS:

1. **Links used from the document provided by sir.**
   a) **https://www.askpython.com/python/examples/load-and-plot-mnist-dataset-in-python**

b) https://scikit-learn.org/stable/modules/neighbors.html
c) https://scikit-learn.org/stable/auto_examples/neighbors/plot_classification.html#sphx-glr-auto-examples-neighbors-plot-classification-py

2. Code reference for building KNN from scratch.

https://medium.com/analytics-vidhya/a-beginners-guide-to-knn-and-mnist-handwritten-digits-recognition-using-knn-from-scratch-df6fb982748a

3. Confusion matrix reference
https://stackoverflow.com/questions/60748497/how-to-include-a-confusion-matrix-for-a-knn-in-python

4. Help from Friends:
a) Raghav Nanjappan STUDENT of G section
b) Adarsh Pryan STUDENT of G section

5.KNN using library function references
a) https://customers.pyimagesearch.com/lesson-sample-k-nearest-neighbor-classification/
b) https://www.geeksforgeeks.org/k-nearest-neighbor-algorithm-in-python/

CONCLUSION:

Through this project/assignment we learned a lot of things which include-

a) How and when to use keras and tensorflow.
b) Many python functions and how to implement them in machine learning algorithms.
c) How to work with huge data, ways and methods to load it.
d) How to use library functions to work on the data.
e) How to build a classifier such as KNN from scratch.
f) How and when confusion matrix can be used.

THANKING YOU-

Khushi and Sneha (G section)