2η ΣΕΙΡΑ ΑΣΚΗΣΕΩΝ

Κουτρούδης Ιωάννης 3258

Γεωργιάδης Γεώργιος 3199

Αρχικά φορτώσαμε τα δεδομένα, χρησιμοποιώντας την εντολή tf.keras.datasets.fashion_mnist και τα αποθηκεύσαμε στα train_images, ο οποίος αναπαριστά την κάθε εικόνα σε pixels και έχει μέγεθος (60000,28,28) και train_labels, ο οποίος αναπαριστά την κατηγορία στην οποία ανήκει η κάθε εικόνα

C: > Users > giorgos > Desktop > 10oExamhno > Μηχανική μάθηση > εργασια2 > ask2.py > countMostNumber

```python
import tensorflow as tf
import numpy as np
import matplotlib.pyplot as plt
import random
import math
from sklearn import metrics

fashion_mnist = tf.keras.datasets.fashion_mnist

(train_images, train_labels), (test_images, test_labels) = fashion_mnist.load_data()
```

K-means clustering

Αρχικά κάναμε reshape τις εικόνες από 28x28 σε 784x1 ,ώστε να μπορέσουμε να δουλέψουμε πιο εύκολα και διαιρέσαμε την τιμή του κάθε pixel με το 255.Έτσι πλέον όλα τα στοιχεία του κάθε πίνακα έχουν τιμές από 0 έως 1.

Στην συνέχεια αρχικοποιήσαμε 10 ‘κέντρα’ σε μορφή πινάκων (784x1) , δηλαδή όσες και οι κατηγορίες που θέλουμε να χωριστούν τα δεδομένα , και τους γεμίσαμε με τυχαίους αριθμούς από 0 έως 1.

```python
def initialize_centers():
    global train_images
    #arxikopoioume ta 10 kentra mas,osa dhladh kai oi kathgories pou 8eloume na xwristoun ta dedomena mas
    centers = []
    for i in range(0,10):
        centers.append([0]*784)

    #vazoume tuxaious ari8mous gia ka8e ena apo ta 10 kentra mas.
    for i in range(0,10):
        for j in range(0,784):
            centers[i][j] = random.randint(0, 255)/255.0


    #allazoume to sxhma tou ka8e pinaka apo 28x28 se 784x1
    train_images = train_images.reshape(len(train_images),-1)
    #print(train_images.shape)
    return centers
```

Έτσι ξεκινήσαμε την διαδικασία του clastering χρησιμοποιώντας την ευκλείδια απόσταση(βάση του τύπου sqrt(sum(center-eikona)), πιο συγκεκριμένα υπολογίσαμε την ευκλείδια απόσταση τις κάθε εικόνας από το κάθε κέντρο(π.χ. για την πρώτη εικόνα βρήκαμε την ευκλείδια απόστασή της και από τα 10 κέντρα).

```
50  def euclidian_distance(a,d,centers,arrayPic):
51      bestDic = 1000000
52      for i in range(0,10):
53          center = centers[i]
54          dist = np.sqrt(np.sum(np.square(center-arrayPic)))
55          if dist < bestDic:
56              bestDic = dist
57              bestCenter = i
58
59      centers= changeWeight(a,d,centers,arrayPic,bestCenter)
60      return centers , bestCenter , bestDic
61
```

Κρατήσαμε το κέντρο το οποίο είχε την μιρκότερη ευκλείδια απόσταση από την εικόνα και αλλάξαμε τα τις τιμές του βάση του τύπου(center + a*(eikona-center), όπου a ο ρυθμός εκπαίδευσης).

```
34
35  def changeWeight(a,d,centers,arrayPic,bestCenter):
36      start = bestCenter - d
37      end = bestCenter + d
38
39      if start < 0:
40          start = 0
41      if end > 9 :
42          end = 9
43      for j in range(start,end+1):
44          center = centers[j]
45          for i in range(0,784):
46              center[i] = center[i]+ a*(arrayPic[i]-center[i])
47          centers[j] = center
48      return centers
49
```

Όμως με αυτόν τον τρόπο στο τέλος μας ταξινομούσε όλες τις εικόνες στην ίδια ομάδα, γιαυτό κάθε φορά που βρίσκαμε την μικρότερη ευκλείδια απόσταση , εκτός από τα βάρη του συγκεκριμένου ‘κέντρου’ αλλάζαμε και τα βάρη κάποιον κοντινών κέντρων βάση του αριθμού dstart(πχ αν το dstart =3 και το κέντρο με την μικρότερη ευκλείδια απόσταση ήταν το 5, τότε αλλάζαμε τα βάρη των κέντρων από 5-3=2 μέχρι 5+3=8).Όσο προχωρούσαν οι επαναλήψεις τις διαδικασίας τόσο μειώναμε και των αριθμό αυτό(dstart),έτσι στο τέλος ο αριθμός αυτός είχε τον αριθμό 0 και

αλλάζαμε μόνο το κέντρο με την μικρότερη ευκλείδια απόσταση,ώστε να είναι πιο ακριβής οι υπολογισμοί της κατηγορίας στην οποία ανήκει η κάθε εικόνα.

Τέλος αντιστοιχίσαμε το κάθε κέντρο με μία κατηγορία ,ανάλογα με την πλειοψηφούσα πραγματική κατηγορία μεταξύ των εικόνων τις ομάδας και υπολογίσαμε πόσες από τις εικόνες ταξινομήθηκαν σωστά(**Purity**) και το F1 score.

```
63  def countMostNumber(array):
64      global trueCat
65      BestNum = 0
66      BestCat = -1
67      for i in range(0,10):
68          count = 0
69          for j in range(0,len(array)):
70              if(array[j] == i):
71                  count = count +1
72          if count>BestNum:
73              BestNum = count
74              BestCat = i
75      trueCat = trueCat + BestNum
76      #print("to pososto twn swsta taxinomhmenwn gia th kathgoria", BestCat," einai: ",(BestNum/len(array))*100
77      return BestCat
78
79  def findTpFpFn(predCat,originalCat):
80      catArr = []
81      for i in range(0,len(predCat)):
82          catArr.append(originalCat[i])
83      fmeasure = metrics.f1_score(catArr,predCat,average="weighted")
84      print("f1 score: ",fmeasure)
85
```

Main

```
def main():
    global train_images
    astart = 0.5
    dstart = 5
    trainImagesLen = 100
    category = [0]*trainImagesLen
    train_images = train_images/255.0
    centers = initialize_centers()
    totalRepeat = 10
    for x in range(0,totalRepeat):
        print("Repeat number: ",x)
        a = astart*(1-x/totalRepeat)
        d = int(dstart*(1-x/totalRepeat))
        #print("Eimaste sto vhma: ",x)
        for i in range(0,trainImagesLen):
            arrayRet = euclidian_distance(a,d,centers, train_images[i])
            centers = arrayRet[0]
            category[i] = arrayRet[1]
            #print("best Center: ",arrayRet[1],"Dic: ",arrayRet[2],"right category: ",train_labels[i])
    for j in range(0,10):
        count = 0
        helpArray = []
        for k in range(0,trainImagesLen):
            if(category[k]==j):
                helpArray.append(train_labels[k])
                count = count + 1
        num = countMostNumber(helpArray)
        for l in range(0,len(category)):
            if category[l] == j:
                category[l] = num

    findTpFpFn(category, train_labels)

    print("To sunoliko Purity einai: ",(trueCat/trainImagesLen)*100)
```

Γενικά τρέξαμε όλα τα παραδείγματά μας χρησιμοποιώντας μόνο **1000 εικόνες** γιατί με 60000 εικόνες ήταν πολύ χρονοβόρο.

Έτσι παρατηρήσαμε ότι όσο αυξάναμε των αριθμό των επαναλήψεων, τόσο αυξανόταν και το Purity.

Αποτελέσματα

Για 1 επανάληψη:

```
Gia  1000  eikones kai  1  epanalhpshs
f1 score:  0.11446292665843251
To sunoliko Purity einai:  27.500000000000004 %
PS C:\Users\giorgos>
```

Για 3 επαναλήψεις:

```
Gia  1000  eikones kai  3  epanalhpshs
f1 score:  0.15465033747321946
To sunoliko Purity einai:  55.00000000000001 %
PS C:\Users\giorgos>
```

Για 10 επαναλήψεις:

```
Gia  1000  eikones kai  10  epanalhpshs
f1 score:  0.3390613359719051
To sunoliko Purity einai:  59.8 %
PS C:\Users\giorgos>
```

Για 20 επαναλήψεις:

```
Gia  1000  eikones kai  20  epanalhpshs
f1 score:  0.2989291050111993
To sunoliko Purity einai:  72.89999999999999 %
PS C:\Users\giorgos>
```