2η Σειρά ασκήσεων - Report

Κωνσταντίνος Κυριάκου 3015
Ομάδα 93

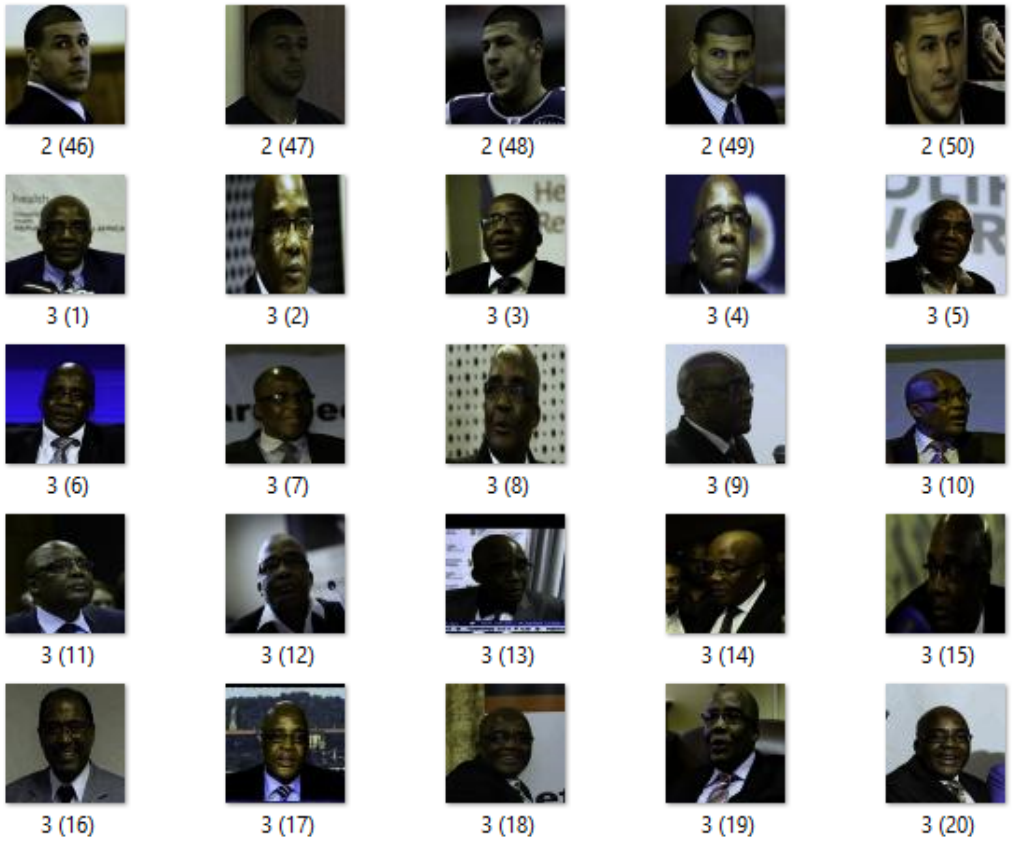
Γενικές σημειώσεις υλοποίησης.

Η υλοποίηση πραγματοποιήθηκε εξ' ολοκλήρου σε μορφή notebook του Kaggle και το αρχείο κώδικα είναι τύπου ".ipynb". Χρησιμοποιήθηκαν οι βιβλιοθήκες της Sklearn για τον αλγόριθμο KMeans, τον Agglomerative Hierarchical Clustering και τον υπολογισμό των Purity και F-Score.

Δεδομένα

Για τα δεδομένα του προγράμματος επιλέχθηκαν τυχαία 10 πρόσωπα από το train_data της εκφώνησης όπου κρατήσαμε 50 φωτογραφίες από το καθένα. Οι φωτογραφίες μετονομάστηκαν και μετακινήθηκαν σε ένα κοινό directory ωστέ τελικά τα δεδομένα εισόδου να έχουν την παρακάτω μορφή:

A



Προεπεξεργασία

Αρχικά, για κάθε εικόνα αποθηκεύουμε σε ποιό πρόσωπο αντιστοιχεί (ytrue).

Έπειτα, αφού έχουμε να κάνουμε με έγχρωμες εικόνες για κάθε εικόνα θα έχουμε αρχικά τρεις πίνακες 64x64 (στην πράξη πίνακας 64x64x3). Για να κάνουμε μετατροπή σε μονοχρωματική εικόνα χρησιμοποιούμε τη γραμμική σχέση της εκφώνησης και υπολογίζουμε για κάθε πίξελ στην κάθε εικόνα, εισάγοντας το αποτέλεσμα σε καινούργιο πίνακα. Έτσι, καταλήγουμε με έναν πίνακα 1x4096, που αντιπροσωπεύει την εικόνα.

Τέλος, εισάγοντας τον κάθε 1x4096 πίνακα σε έναν συνολικό, καταλήγουμε με έναν πίνακα με όλες τις 500 εικόνες στην είσοδο, διάστασης 500x4096.

```
image_list = list()
faces = list()
# Load images from path
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        image = np.asarray(Image.open(os.path.join(dirname, filename)))/255

        # **********************
        # Store each image's face (ytrue)
        face = int(filename[0:2])
        faces.append(face-1)
        # **********************

        grayscale_image = list()
        # For every pixel in 64x64 image
        for i in range(64):
            for j in range(64):
                # Extract RGB values
                rgb = image[i][j]
                # Convert and append to create 1x4096 image
                grayscale = 0.299*rgb[0]+0.587*rgb[1]+0.114*rgb[2]
                grayscale_image.append(grayscale)
        # Append to list of images 500x4096
        image_list.append(grayscale_image)

images = np.array(image_list)

images.shape
```

```
[27…  (500, 4096)
```

Μείωση διάστασης

Για την μέθοδο PCA χρησιμοποιήθηκε η βιβλιοθήκη της Sklearn ως εξής:

[29]:
```
# Dimensionallity Reduction with PCA

X = images.tolist()
pca = decomposition.PCA(n_components=M)
pca.fit(X)
X = pca.transform(X)

print(X.shape)
```

```
(500, 100)
```

Παρακάτω φαίνεται η αρχιτεκτονική του δικτύου Autoencoder ωστόσο, λόγω προγραμματιστικού σφάλματος δεν λειτουργεί.

[38]:
```
# Dimensionallity Reduction with Autoencoder (not working atm)

X=images.tolist()
d = 64*64
autoencoder = MLPRegressor(hidden_layer_sizes = (d, d/4, M, d/4, d),
                   activation = 'tanh',
                   solver = 'adam',
                   learning_rate_init = 0.0001,
                   max_iter = 20,
                   tol = 0.0000001,
                   verbose = True)
autoencoder.fit(X,X)
```

Ομαδοποίηση

Χρησιμοποιώντας τον αλγόριθμο KMeans της βιβλιοθήκης Sklearn παίρνουμε τα εξής αποτελέσματα.

```
# KMeans Algorithm (euclidean)

#from collections import Counter

model = KMeans(n_clusters= k, random_state=10)
kmeans = model.fit_predict(X)
centroids = model.cluster_centers_
labels = model.labels_

#print(Counter(labels).keys())
#print(Counter(labels).values())
pd.DataFrame(labels)
```

	0
0	1
1	5
2	1
3	4
4	9
...	...
495	5
496	2
497	6
498	5
499	9

500 rows × 1 columns

Ομοίως χρησιμοποιώντας τον αλγόριθμο Agglomerative Clustering της Sklearn παίρνουμε:

[33]:
```
# Agglomerative Hierarchical Clustering

model_AC = AgglomerativeClustering(n_clusters = k)
agglomerative_clustering = model_AC.fit_predict(X)
labels_AC = model_AC.labels_

pd.DataFrame(labels_AC)
```

[33…

	0
0	0
1	3
2	5
3	7
4	5
…	…
495	0
496	9
497	0
498	3
499	2

500 rows × 1 columns

Αποτελέσματα

Έπειτα από την εκτέλεση των παραπάνω αλγορίθμων υπολογίζουμε τις τιμές των Purity και F-Score με τη βοήθεια των παρακάτω συναρτήσεων:

```
def purity_score(y_true, y_pred):
    contingency_matrix = metrics.cluster.contingency_matrix(y_true, y_pred)
    return np.sum(np.amax(contingency_matrix, axis=0)) / np.sum(contingency_matrix)

def f_score(y_true, y_pred):
    return metrics.f1_score(y_true, y_pred, average= "macro")
```

Και τελικά παίρνουμε:

Method	Dimension of Data (M)	Purity	F-Measure
K-means (Euclidean)	100	0.222	0.09732
	50	0.254	0.08523
	25	0.228	0.09066
K-means (Cosine)	100	-	-
	50	-	-
	25	-	-
Agglomerative Hierarchical Clustering	100	0.194	0.111823
	50	0.204	0.107152
	25	0.228	0.09554

Όλα τα F-Scores έχουν στρογγυλοποιηθεί στα 5 δεκαδικά ψηφία.

Προβληματισμοί

Οι τιμές των metrics που υπολογίζουμε φαίνεται να είναι πολύ μικρότερες απ' όσο αναμέναμε. Αυτό μπορεί να οφείλεται είτε σε προγραμματιστικό/λογικό σφάλμα κατά την υλοποίηση, είτε στις ίδιες τις φωτογραφίες που χρησιμοποιήθηκαν.