face-network

Release 1.0

Ollie Ballinger

Jul 15, 2021

CONTENTS

Given a large volume of images of people, this tool uses artificial intelligence to generate an interactive social network graph based on co-appearance in images:

Each node is a person, and each edge linking two nodes represents an image in which the two people appear together. The network above can be clicked, dragged, and zoomed to reveal the latent social structure present in unstructured image data.

A folder of images can be turned into a co-appearance network in the following three steps:

CHAPTER

ONE

EXTRACTING FACES FROM IMAGES

This function extracts all faces from a directory of images using Dlib's face detector, and must be run prior to further analysis.

```
face_network.extract_faces(source_dir, age_gender=False)
```

1.1 Parameters

source_dir (*str*); The path to the image folder (note: this folder can contain sub-directories, so specify the highest level image directory).

age_gender (*bool, defaul=False*); Estimates apparent age and gender using a pretrained Convolutional Neural Network (ResNet-50). Results are stored in the columns "age" and "gender" in the resulting dataframe. Gender is predicted on a scale from 0 (male) to 1 (female).

1.2 Outputs

This function creates a new folder called "Face Network" in your image directory. When a face is identified, it is cropped and stored in a new folder "source_dir/Face Network/Faces/". Given "Image.jpg" containing two faces, this function will save two cropped faces: "face1_Image.jpg" and "face2_Image.jpg". Facial encodings (128-dimensional vectors used for clustering and matching similar faces) are stored in a file called "FaceDatabase.h5".

1.3 Performance

The facial extraction function is threaded, by default using n_cpus-1. Using seven cores on an M1 MacBook Pro, all 14,000 faces were extracted from the Labeled Faces in the Wild (LFW) database in 42 minutes (~6 faces per second). Facial encodings for the entire LFW database take up 15 MB (~1kb per face).

CHAPTER

TWO

CLUSTERING SIMILAR FACES

Once faces are extracted, similar faces are clustered together. This function uses a density-based clustering algorithm (DBSCAN) to identify clusters of similar faces in the list of facial encodings. Starting with loose clustering parameters, the function iteratively decreases the neighborhood distance parameter. In each iteration, facial similarity within clusters is evaluated. Dense clusters are extracted, and sparse clusters are assigned to be re-evaluated in the next iteration. When an iteration returns no new clusters, the function returns a dataframe containing facial encodings grouped into clusters based on similarity.

```
face_network.network(source_dir, algorithm='DBSCAN', iterations=1, initial_eps=0.45, max_
↪distance=45)
```

2.1 Parameters

source_dir (*str*) The path to the image folder

algorithm (*{'DBSCAN','OPTICS','AHC'}, default="DBSCAN"*) The algorithm used for clustering. Possible options are "DBSCAN", "OPTICS", and "AHC" (agglomerative hierarchical clustering).

iterations (*int, default=10*) The number of iterations that the function will perform. Each iteration restricts the clustering parameters.

max_distance (*float, default=50*) Sets the maximum euclidean distance between each face in a cluster and the core sample. This weeds out outliers and sparse clusters.

initial_eps (*float, default=0.45*) Determines Sets the Epsilon parameter for the DBSCAN algorithm.

mosaic (*bool, default=True*) Creates a mosaic of face tiles for each image.

2.2 Outputs

Rows in the FaceDatabase.h5 file now contain a unique numeric identifier, grouping similar faces into clusters. If the "mosaic" option is enabled, an image composed of all of the face tiles in a given cluster is created:

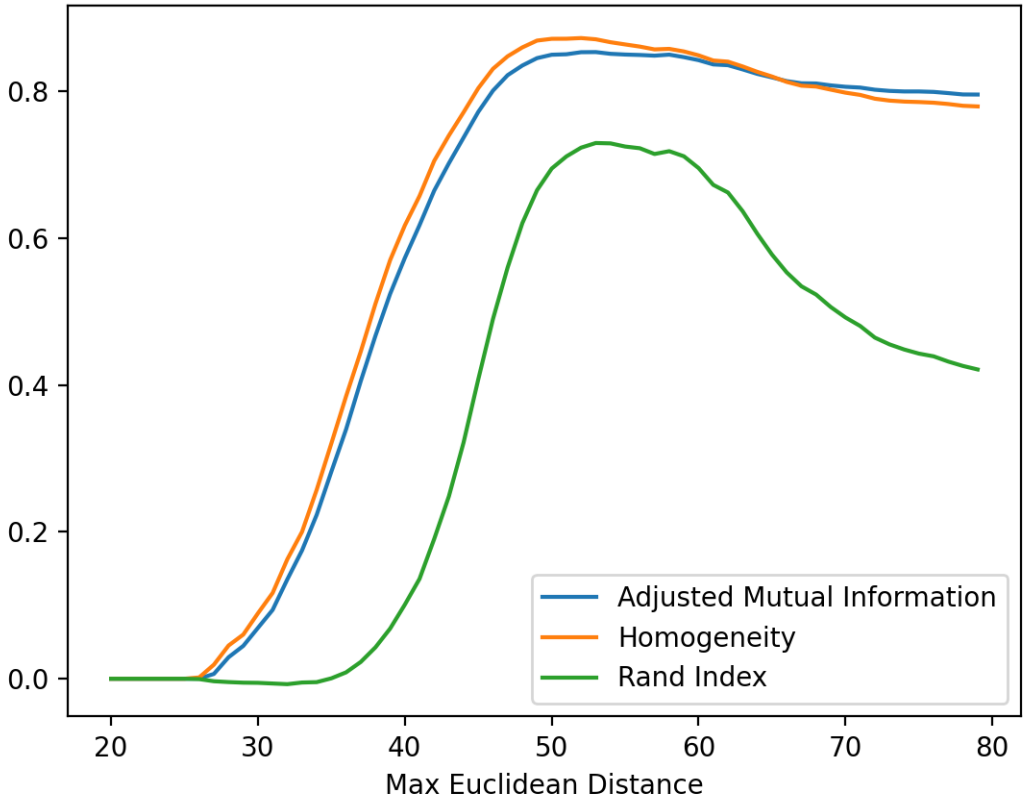
2.3 Performance

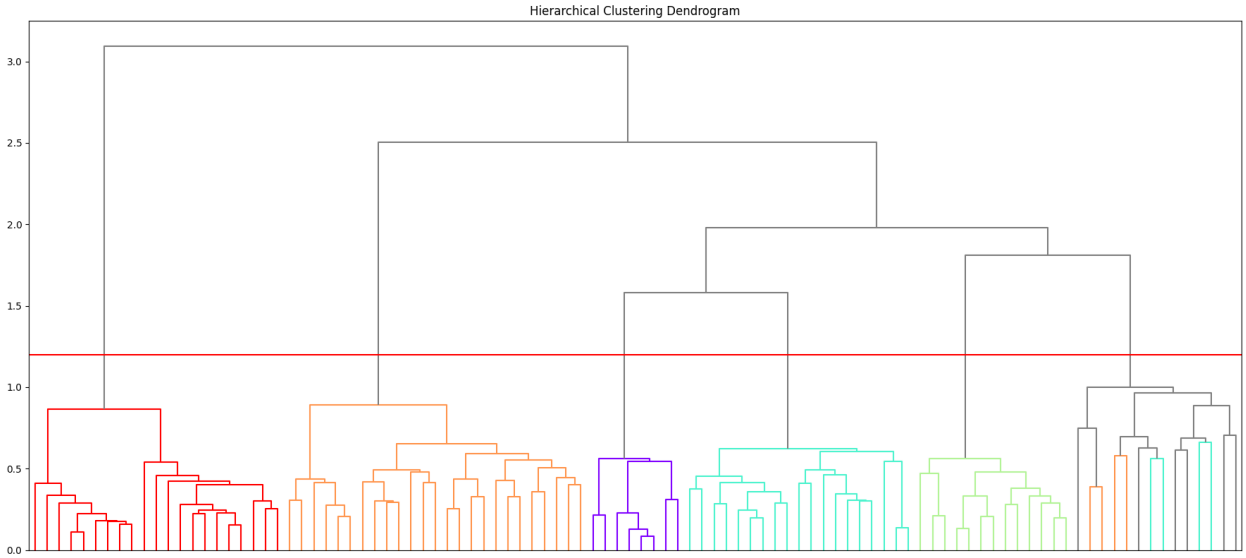
The graph below shows the effect of tuning the "max_distance" parameter on clustering accuracy in the LFW dataset, using the DBSCAN algorithm.

The optimal range seems to be between 50 and 60: if the cutoff is too low, false negatives increase as faces are not sorted into clusters. If the cutoff is too high, the number of false positives increases. Optimal parameters will vary based on your dataset. Further information on the clustering evaluation metrics used above can be found here.

A visual representation of the clustering process is shown in the dendrogram below, generated using Agglomerative Hierarchical Clustering (AHC). Each "leaf node" (the points where the dendrogram intersects the X axis) represents a face. The Y axis indicates the euclidean distance (similarity) between facial encodings. If the link between two leaf nodes is very low, the facial encodings are more similar. An interesting property of using AHC to cluster facial images is that the first branch of the dendrogram almost always separates men and women.

The example above uses 100 labeled faces, with colors denoting images of the same individual. We can see groups of faces that are all quite similar to eachother, but quite dissimilar from faces in the other groups. We could probably tell without relying on the colors that there are 5-6 distinct individuals in these 100 images. Indeed, the colors suggest there are 5 main individuals and one "bin" cluster on the far right composed mainly of unlabeled faces (the grey leaf nodes). There are some errant faces from two of the five individuals in this cluster, likely due to poor image quality, pose, or lighting. 94 out of the 100 faces above are labeled. Of these, 86 (91%) were correctly sorted into clusters representing distinct individuals using the red cutoff line.


0.8
0.6
0.4
0.2
0.0
20
30
40
50
60
70
80
Max Euclidean Distance
Adjusted Mutual Information
Homogeneity
Rand Index


Hierarchical Clustering Dendrogram
3.0
2.5
2.0
1.5
1.0
0.5
0.0

CHAPTER

THREE

CREATING A NETWORK

Having identified individuals across multiple pictures, this function generates a force directed graph based on co-appearance in images. Each individual is a node, and each co-appearance is an edge.

```
face_network.network(photo_dir, scale=10)
```

3.1 Parameters

source_dir (*str*) The path to the image folder

scale (*int, default=10*) Dictates the size of the nodes

3.2 Outputs

A file called "Image_Network.html" is created in "photo_directory/Face Network/Data/".The graph can be opened in a web browser and is fully interactive. Hovering over a node will display a tooltip showing the cluster's unique identifier. This corresponds to the filenames of the mosaics generated in the previous step.

The diagram below illustrates this process:

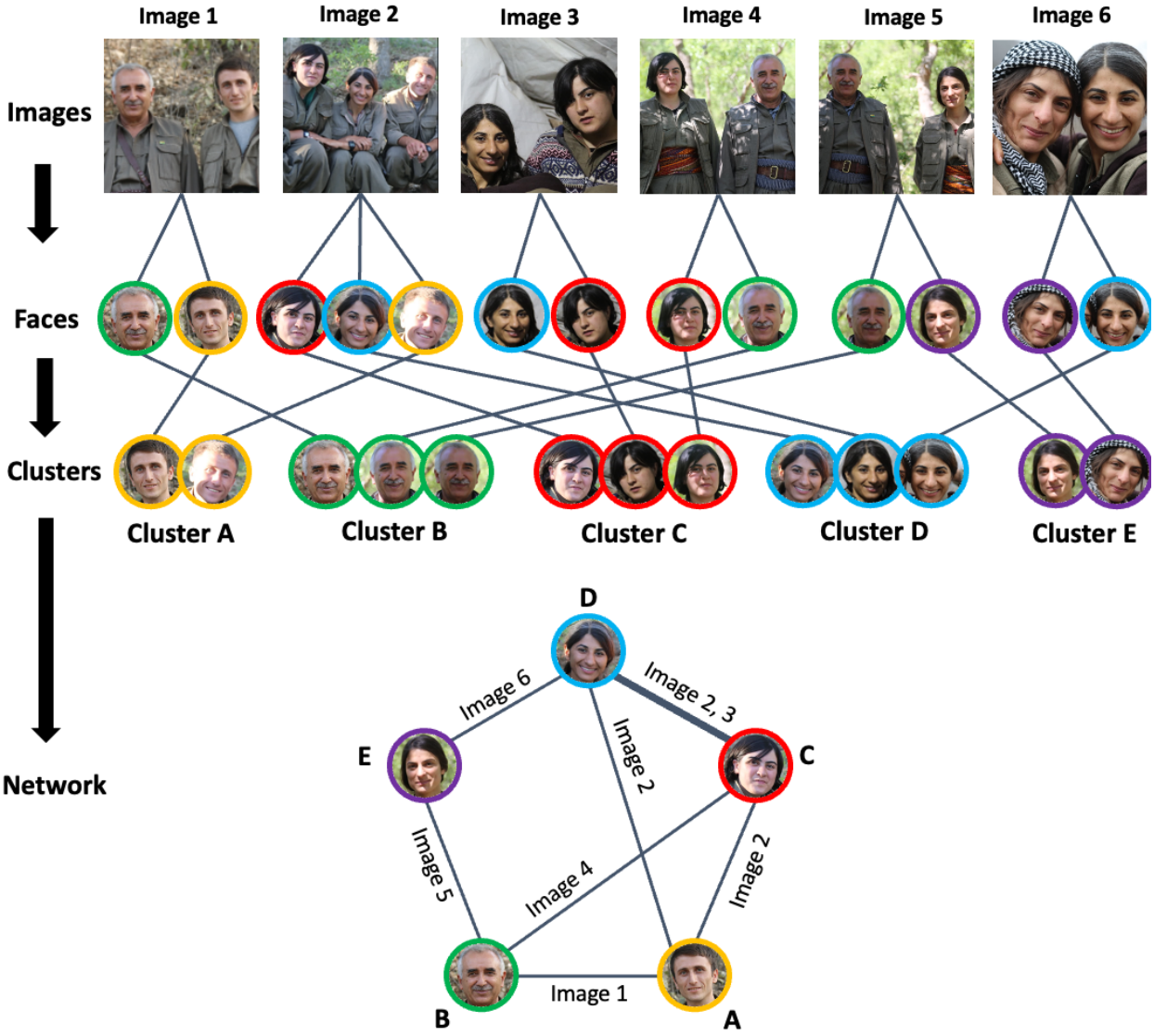


Image 1
Image 2
Image 3
Image 4
Image 5
Image 6
Images
Faces
Clusters
Cluster A
Cluster B
Cluster C
Cluster D
Cluster E
Network
D
E
C
B
A
Image 6
Image 2, 3
Image 2
Image 5
Image 4
Image 2
Image 1