MATH 728D: Machine Learning Lab #14: Facial Analysis

John Burkardt

January 11, 2019

Can a Computer Match a Person to a Photo?

Facial recognition is an instance of machine learning in which the data is snapshots of faces. Given an album of such data, the task is to accept a new image, and determine whose face it is. While we can't construct an entire such system in our lab work, we will look at some of the steps that make such a system possible.

1 Selecting and Aligning Images

To begin, we need to collect about 10 pictures of some person, in which the person's face is in roughly the same attitude, facing forward and well lit. The images can be of different types, such as JPEG, PNG, TIF or other common image formats, and may be in color or black-and-white (also called "gray" or "gray scale" images).

If you don't want to collect images yourself, you can start with images of Angela Merkel available in the file **http://people.math.sc.edu/burkardt/classes/ml_2019/angela.zip**.

If your first image is named `face1.jpg`, we can read and display it with MATLAB commands like

```
I1 = imread ( 'face1.jpg' );
imshow ( I1 );
```

To see all the images you collected in a single plot, try something like this:

```
subplot ( 2, 5, 1 );
imshow ( I1 );
subplot ( 2, 5, 2 );
imshow ( I2 );
...
subplot ( 2, 5, 10 );
imshow ( I10 );
```

In order to make a useful set of images, we need to trim or "crop" each image to cover roughly the same area of the face. The easiest goal is to crop the image so that it extends from chin to the top of the head, and from one ear to the other. This can be done with MATLAB's `imcrop()` command. After we create the cropped image J, we want to display it, and save it. When you issue the `imcrop(I)`, a plus sign appears on the image. Place the cursor at one corner of the cropping rectangle, hold down the mouse button while moving the cursor to the opposite corner of the cropping rectangle, and release the mouse button. This will create the cropped image J:

```
J1 = imcrop ( I1 );
imshow ( J1 );
imwrite ( J1, 'face1_head.jpg'. 'JPEG' );
```

Exercise 1:

1. Find about 10 images of a celebrity;
2. Save your images as files with a common filename and sequence number, such as “face1.jpg”, “face2.jpg”, and so on;
3. Use commands like `I1=imread('face1.jpg')` to read your images into memory as `I1, I2, ..., I10`;
4. Display your images together in a single plot.
5. Use commands like `J1 = imcrop(I1)` to create cropped images `J1, J2, ..., J10` in which the face appears in roughly the same position;
6. Using the `imwrite()` command, save your cropped images in files, with a common filename and sequence number, such as “face1_head.jpg”;
7. Display all your cropped images in a single plot.

2 Converting to Gray Scale Images of a Common Size

Your collection of images may include both color and black-and-white images, and the images also probably vary in the number of vertical and horizontal rows of pixels used. We will need to standardize the size and color of the images as the next part of our processing.

If J1 is an image, then the MATLAB command

```
GRAY = rgb2gray ( J1 );
```

creates a black-and-white version of the image. This command is legal even if the original image K1 is already black-and-white.

If ‘GRAY is an image, then the MATLAB command

```
K1 = imresize ( GRAY, [ M, N ] );
```

creates a new $M \times N$ image, using interpolation.

Exercise 2:

1. For each image J1, J2, ... in your collection:
2. Use `rgb2gray()` to convert to black and white:
3. Use `imresize()` to convert to a common size, such as [200,175]:
4. Use `imwrite()` to save a copy of each image, with a common filename and sequence number, such as “face1_gray.jpg”;

3 Computing an Average Face

You should now have a set of about 10 images, stored as files or perhaps still held in memory as `K1, K2, ..., K10`. We want to compute an average of these images, which we’ll call `L`. We hope that `L` is a good single representative of the information that is common to the image collection.

To do this, we have to:

- pack all 10 images into a single array `K`;
- make `K_numeric`, a numeric copy of `K`;
- compute `K_mean`, the numeric average of `K_numeric,` using the `mean()` function;
- convert `K_mean` to an image `L`;

Exercise 3:

1. Make sure your gray scale images are available as `K1,...,K10`; if not, read them back into memory with the `imread()` function;
2. Pack **all but the last image** into a 3D array using commands like:

   ```
   K(:,:,1) = K1;
   K(:,:,2) = K2;
   ...
   K(:,:,9) = K9;   <-  Stop one early, don't include your last image!
   ```

3. Make a numeric copy:

   ```
   K_numeric = double ( K );
   ```

4. Average the numeric array over the third dimension:

   ```
   K_mean = mean ( K_numeric, 3) ;
   ```

5. Convert the numeric average back to an image:

   ```
   L = uint8 ( K_mean );
   ```

6. Use the `imwrite()` command to save your average face to the file "face_average.jpg";

Use the `imshow()` command to display `L`. If your data was good, properly processed, and the averaging operation was done correctly, the image should resemble the individual whose faces were averaged.

4 "Recognizing" a Face

Now we are going to try to recognize a face. To do this, we need some test images. Let test image:

- T1 be K10, that is, the last image in your collection, which you did not include in the averaging operation;
- T2 be K1, the first image in your collection, which did contribute to the average;
- T3, an image of another celebrity, which you have cropped, grayed, and resized.

Now we will try to measure the distance between each of these test images and your average. Test images that are relatively close will then be assumed to be a match with your subject.
We can use a norm function to estimate the distance. But remember that MATLAB images are stored in a special format that does not allow numeric operations, so the comparison will require first making a numeric copy, using the `double()` function. Thus, the norm of the difference between test function T1 and the average L is the number

```
diff = norm ( double ( T1 ) - double ( L ) );
```

Exercise 4:

1. compute the differences between your averaged image L, and your test images T1, T2, and T3.
2. Did our simple averaging operation help to identify matching faces?