CSC321: Assignment #1,
Face Recognition and Gender Classification with k-NN

Due on Wednesday, February 3, 2016

Jurgen Aliaj

February 4, 2016

Part 1

Dataset description

The dataset consists of approximately 900 images of 3 actors and 3 actresses (150 images of each). In part 6, the dataset is expanded to include other actors and actresses. Images of each actor/actress come in a variety of different angles. Some of the images contain bruised faces of actors, as shown in the third example of Figure 1. This could pose challenges in the task of face recognition, as one would expect it more difficult to recognize such faces given that the majority of the training data contains faces which are not bruised. In addition, the dataset contains annotations for a bounding box that surrounds the face in each image. Some of the bounding boxes are inaccurate. Take, for instance, the first example of Figure 2, in which the jaw and left cheek of Daniel Radcliffe are not visible. The fact that the annotations are not perfect will pose an extra challenge in recognizing faces. In general, however, most of the annotations are fairly accurate. In fact, the first example of Figure 2 was the worst example that could be found. Examples 2 and 3 of Figure 2, for example, can be reasonably aligned with each other (the alignment is shown in example 4 of Figure 2).

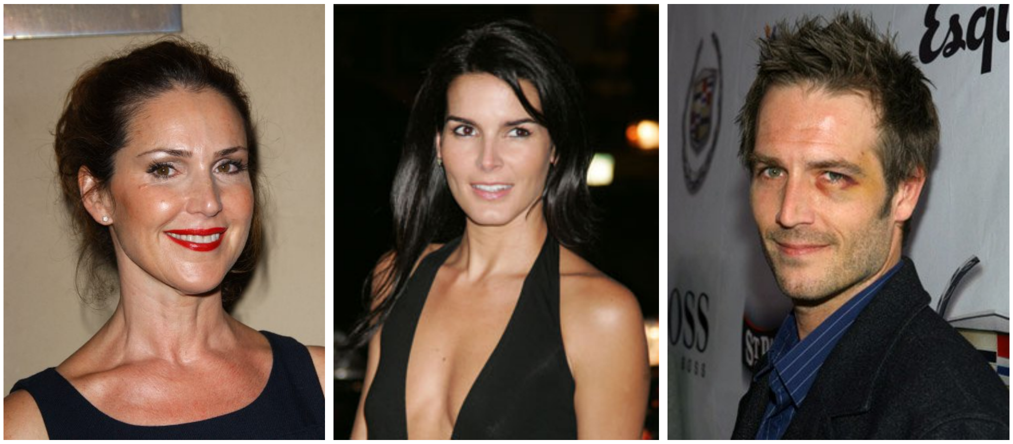

Figure 1: A selection of uncropped photos from the dataset.

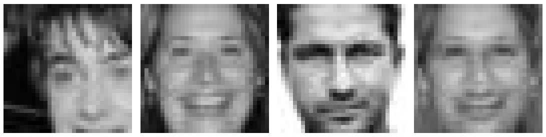

Figure 2: A selection of cropped, 32x32 grayscale photos from the dataset, the final example is an overlap of examples 2 and 3.

Part 2

Seperating the dataset

To seperate the data into training, testing, and validation sets, I used a bash script under `tools/make_split.sh` which creates 3 new directories (training, testing, and validation) and copies the first 100 images of each actor/actress into the training directory, the next 10 into the testing directory, and the next 10 into the validation directory. Although this is not the best method given that it does not impose any randomness, there are some advantages. Namely, it is clean and simple–which is why I decided to use this method.

Part 3

Face recognition using k-nearest neighbours

For each image, we find the k nearest neighbours in the training set, and assign to it a prediction which corresponds to the majority of the labels in the set of k nearest neighbours. We define those images which are nearest as the ones with the minimal euclidean distance between flattened grayscale image arrays. k is empirically chosen as 1, which gives the best performance on the validation set with an accuracy of approximately 78%. The performance on the test set for $k = 1$ then gives an accuracy of 65%.

Failure cases

Error #1:
original image: `vartan_male_108.jpg`
predicted label: butler

Nearest Neighbours:
first: `butler_male_2.jpg`
second: `harmon_female_32.jpg`
third: `harmon_female_71.jpg`
fourth: `butler_male_77.jpg`
fifth: `harmon_female_18.jpg`

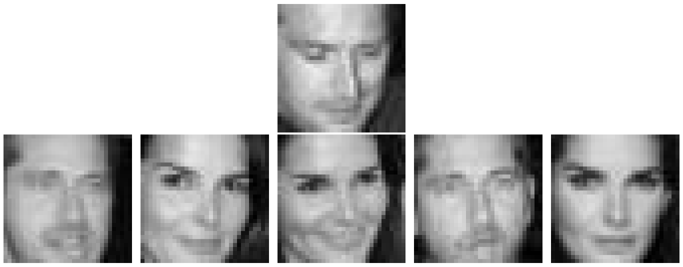

Figure 3: The photo on top is the failure case, and the five photos below are the nearest neighbours in order from left to right.

Error #2:
original image: `radcliffe_male_100.jpg`
predicted label: vartan

Nearest Neighbours:
first: `vartan_male_98.jpg`
second: `radcliffe_male_67.jpg`
third: `butler_male_60.jpg`
fourth: `radcliffe_male_16.jpg`
fifth: `radcliffe_male_59.jpg`

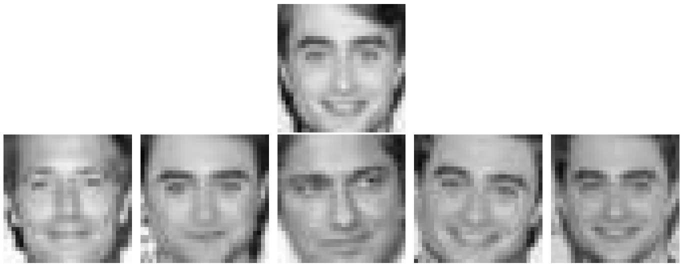

Figure 4: The photo on top is the failure case, and the five photos below are the nearest neighbours in order from left to right.

Error #3:
original image: `gilpin_female_105.jpg`
predicted label: harmon

Nearest Neighbours:
first: `harmon_female_40.jpg`
second: `bracco_female_64.jpg`
third: `bracco_female_34.jpg`
fourth: `gilpin_female_23.jpg`
fifth: `gilpin_female_17.jpg`

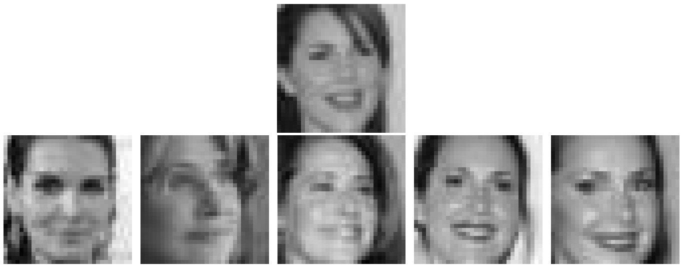

Figure 5: The photo on top is the failure case, and the five photos below are the nearest neighbours in order from left to right.

Error #4:
original image: `butler_male_106.jpeg`
predicted label: radcliffe

Nearest Neighbours:
first: `radcliffe_male_72.jpg`
second: `butler_male_69.jpg`
third: `gilpin_female_4.jpg`
fourth: `butler_male_40.jpg`
fifth: `radcliffe_male_93.jpg`

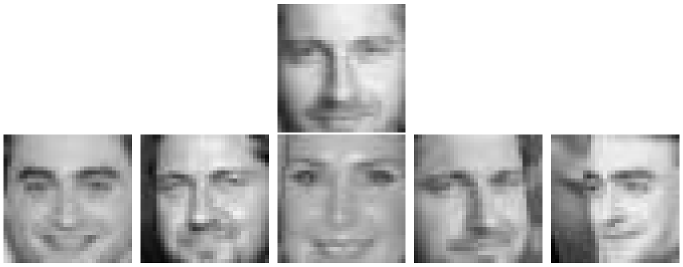

Figure 6: The photo on top is the failure case, and the five photos below are the nearest neighbours in order from left to right.

Error #5:
original image: `butler_male_102.jpg`
predicted label: vartan

Nearest Neighbours:
first: `vartan_male_63.jpg`
second: `vartan_male_96.jpg`
third: `vartan_male_59.jpeg`
fourth: `butler_male_88.jpg`
fifth: `vartan_male_11.jpg`

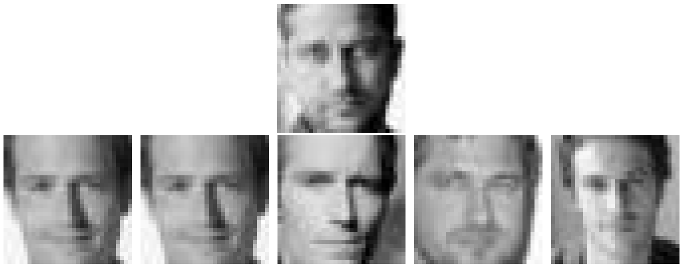

Figure 7: The photo on top is the failure case, and the five photos below are the nearest neighbours in order from left to right.

Part 4

Performance results

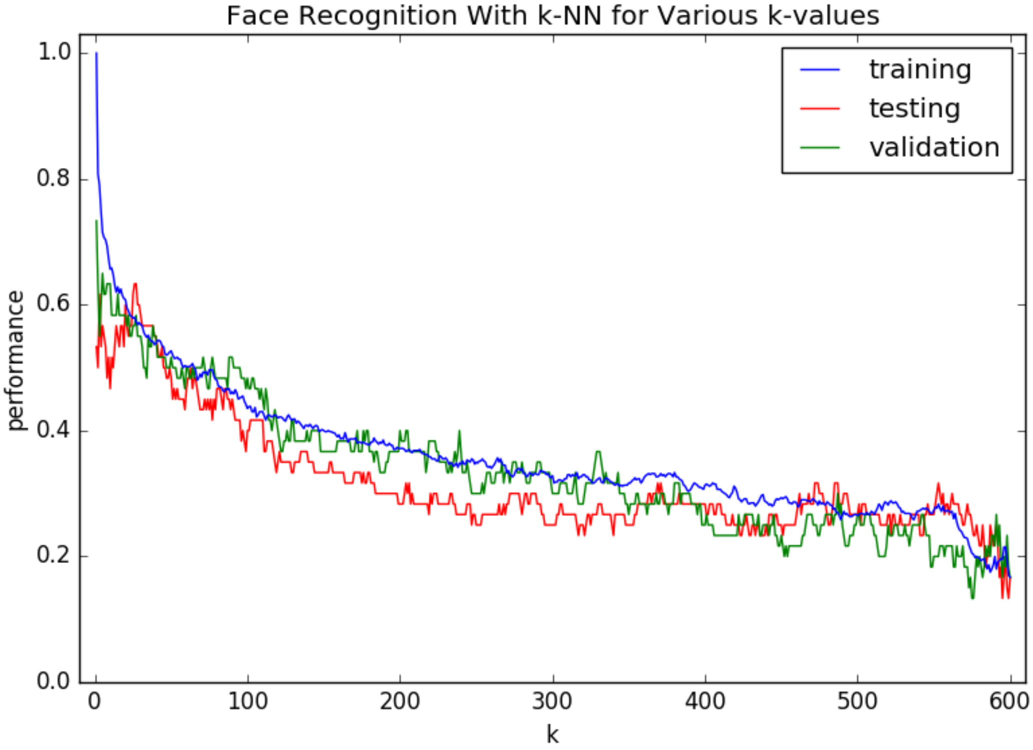

Figure 8: Performance on training, validation, and testing sets for given values of k. Performance is measured as the ratio of correct classifications over total classifications to be made.

For very small values of k, performance on the training, validation, and testing set is as high as possible. In particular, when $k = 1$, performance on the training set is 1 as expected (the nearest neighbour of an image in the training set is itself). The fact that validation results and testing results are also maximal (or very nearly maximal) at $k = 1$ could very well be a coincidence or it could be a result from the structure of the problem. There is no reason why this should happen in general (otherwise we could choose $k = 1$ all the time and there would be no need to use a validation set to optimize the k value). Conversely when k is very high, the performance for training, validation, and testing all drop to 1/6. The reasoning for this is as follows. The majority label of the k-nearest neighbours in this case is simply the majority label of the whole training set, which will be a constant for each test case. But since there are an equal number of actors/actresses in the training, validation, and testing sets, precisely 1/6 of the images will have the predicted label in each data set. This does not necessarily mean that there should be a minimum at $k = 600$, though, as there is no reason in principle why the performance can not do worse than random chance. As it turns out, an accuracy of 13% is obtained at $k = 599$. In general though, random guessing is a reasonable approximation to the worst case performance, since we are essentially taking random guesses as k becomes very large. Thus, we see that there is a maximum near $k = 1$ and a minimum near $k = 600$. This explains why the curve tends to decrease as k increases.

Part 5

Gender classification using k-nearest neighbours

The method for gender classification is precisely the same as the method described in part 3, except instead of labels for faces we simply use labels which specify whether the person is male or female. The k which gives the best performance on the validation set in this case is $k = 1$, with an accuracy of 93%. Here is a table which gives the performance for some different k-values.

k	performance
1	0.93
2	0.92
3	0.87
4	0.92
5	0.87
6	0.9
7	0.88
8	0.88
9	0.87
10	0.85

For values of $k > 10$ the performance slowly starts to drop. When the k value of 1 is used for the testing set, an accuracy of 95% is obtained. The fact that the testing set does better than the validation set in this case is most likely a coincidence, as this should not happen in general (in fact the opposite should happen).

Part 6

Gender classification with test cases outside of the training set

This is precisely the same procedure described in part 5 except the validation and testing sets contain actors/actresses not included in **act**. Obtaining this data was done with script `tools/get_others.py` (which downloads 10 of each actor/actress for a total of 220 images) and creating the validation and test sets was done with `tools/make_other_split.sh`. The k which gives the best performance on the validation set in this case is $k = 4$, with an accuracy of 77%. Here is a table which gives the performance for some different k-values.

k	performance
1	0.65
2	0.69
3	0.75
4	0.77
5	0.72
6	0.74
7	0.74
8	0.75
9	0.72
10	0.71

When the k value of 4 is used for the testing set, an accuracy of 75% is obtained. This performance is 20% worse than the performance obtained in part 5 due to the fact that in part 5, it is very likely that the best approximation to a person's face can be found from another photo of the same person's face, which would likely yield a correct result when classifying gender. In part 6, no such approximation is possible since the person being classified is not part of the training set.