CS 5785: APPLIED MACHINE LEARNING

HOMEWORK 2

September 27, 2017

Sarah Le Cam - sdl83
Yunie Mao - ym224

Cornell Tech

Contents

EIGENFACE FOR FACE RECOGNITION

Summary

We were given a set of black and white pictures of faces with training and testing data files containing the links to those images and corresponding labels identifying the individuals. Using this data, we calculated the mean image and subtracted it from each of the images in our training set. We then performed a Singular-Value Decomposition to find the set of Eigenfaces. Using our Eigenfaces, we computed the Eigenfeatures and the ranked r-dimensional feature vectors for both the training images and test images. Finally, we fitted the Eigenfeatures and labels of our training set to a logistic regression using *scikit-learn*'s logistic regression model. We used this model to find predicted labels for our test data using the Eigenfeatures of the test images and calculated the mean accuracy on the given test data and labels. To visualize the fit of our model, we plotted our classification's accuracy for the first 200 dimensions in the face space of our test data.

Data

We were provided with a set of 640 pictures total of 10 distinct subjects and two files - a testing and a training text file - matching each image to its respective label. The training set contained the image links and label pairings for 540 images and the testing set contained 100. Each image is 50 x 50 pixels black and white photograph. In order to use this data for model fitting, we converted the images into grayscale and stored the pixel data in a matrix.

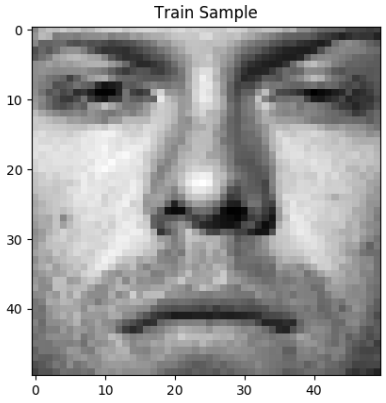
Procedure & Insights

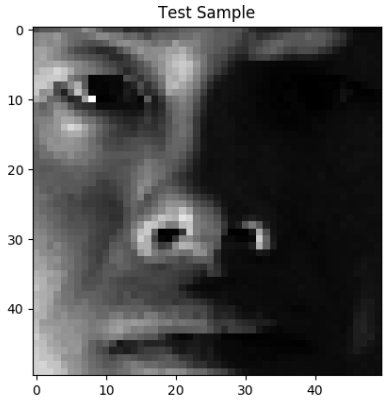
Question 1 (a)

We downloaded and unzipped the faces data file. We then used Anaconda Navigator's Spyder IDE to create a Python project and included our images folder (*faces/images*) and our training and testing data files (*faces/train.txt* & *faces/test.txt*). We generated a Python file (*eigenFaces.py*) and imported the relevant external libraries (NumPy, SciPy, MatPlotLib and sklearn).

Question 1 (b)

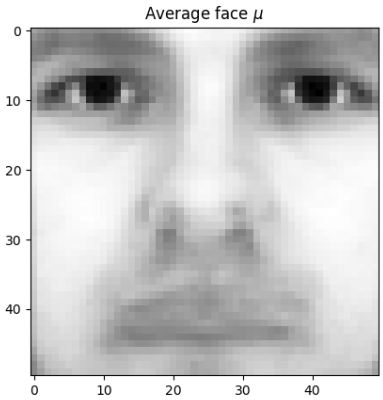
We retrieved each image link from the training and testing datasets using the *split()* function. We then computed the images' greyscale pixel information using MatPlotLib's *imread()* and stored the pixel configurations of the training and testing images in two matrices of size, respectively, 540 x 2500 and 100 x 2500. We then displayed a sample image (the 10^{th} in each dataset) using the pixel information stored in each of these matrices using *imshow()*. We saved these the sample image for the training set as *training_image.png* and that for the testing set as *test_sample.png*. For both the training and testing datasets, we also extracted the labels from the text files into 2 flat arrays of size 540 (training labels) and 100 (testing labels) using the *split()* function.





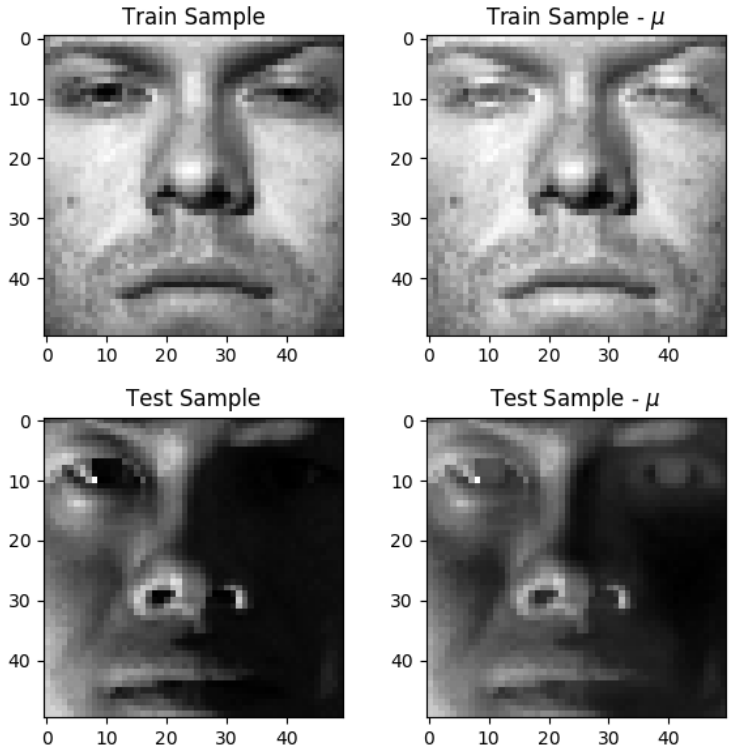
Question 1 (c)

Using the NumPy's *mean()* function along the vertical axis of the training dataset pixel matrix, we found the average face μ and displayed it using the MatPlotLib *imshow()* function. We save the image as *average_image.png*.



Question 1 (d)

We then subtracted our average face μ's pixel values from those of every image in the training and testing matrices to form new adjusted matrices. These new matrices indicate distance from the mean, allowing us to centralize our data. Again, we displayed a sample (the 10^{th} in each matrix) from each of the new adjusted datasets (see *original_and_adjusted_images.png*).



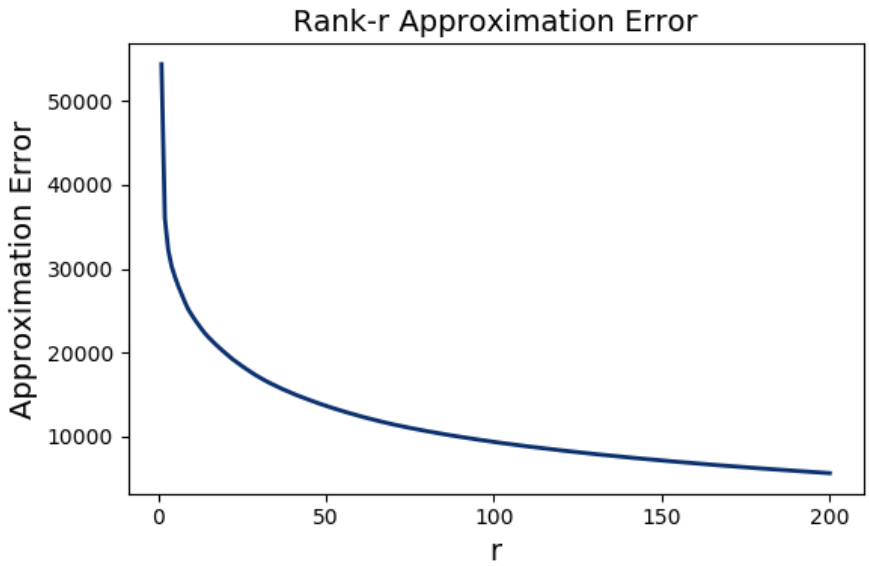
Question 1 (e)

We performed a Singular Value Decomposition (SVD): $X = U\Sigma V^T$ where X is the matrix representation of the adjusted training set. Using NumPy's *linalg.svd()* function, we computed U (the left-singular vector matrix), Σ (the covariance matrix), and V^T (the transpose of the left-singular vectors of X). We then displayed the top ten Eigenfaces as images in grayscale using *imshow()* (see *first_ten_eigenfaces.png*).

[TODO: Fix Plot]

Question 1 (f)

We generated a helper function to compute the rank-r approximation of our adjusted training data by taking the first r columns of U, the first r elements of Σ and the first r rows of V^T. We then computed the low-rank approximation error of our adjusted training data to the rank-r approximation for $r = 1, 2, ..., 200$ and plotted the results as a function of the value of r (see *low_rank_approximation_err.png*). As the plot shows, as r increases the approximation error decreases exponentially. A value of only 200 for r corresponds to a relatively low approximation error.



Question 1 (g)

[TODO]

Question 1 (h)

[TODO]

Conclusion

[TODO]

WHAT'S COOKING?

Summary

[TODO]

Data

[TODO]

Procedure & Insights

Question 2 (a)

[TODO]

Question 2 (b)

[TODO]

Question 2 (c)

[TODO]

Question 2 (d)

[TODO]

Question 2 (e)

[TODO]

Question 2 (f)

[TODO]

Question 2 (g)

[TODO]

Conclusion

[TODO]

WRITTEN EXERCISES

Question 1

[TODO]

Question 2

2. a.

[TODO]

2. b.

[TODO]

2. c.

[TODO]

2. d.

[TODO]

2. e.

[TODO]

Question 3

3. a.

[TODO]

3. b.

[TODO]

3. c.

[TODO]

3. d.

[TODO]

3. e.

[TODO]

SOURCES & EXTERNAL LIBRARIES

Stéfan van der Walt, S. Chris Colbert and Gaël Varoquaux. *The NumPy Array: A Structure for Efficient Numerical Computation*, Computing in Science & Engineering, 13, 22-30 (2011), DOI:10.1109/MCSE.2011.37

John D. Hunter. *Matplotlib: A 2D Graphics Environment*, Computing in Science & Engineering, 9, 90-95 (2007), DOI:10.1109/MCSE.2007.55

Jones E, Oliphant E, Peterson P, et al. *SciPy: Open Source Scientific Tools for Python*, 2001-, `http://www.scipy.org/`

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, Édouard Duchesnay. *Scikit-learn: Machine Learning in Python*, Journal of Machine Learning Research, 12, 2825-2830 (2011)

Wes McKinney. *Data Structures for Statistical Computing in Python*, Proceedings of the 9th Python in Science Conference, 51-56 (2010)