HOMEWORK 2

Class: CS 5785
Authors: Anna Guidi, Samantha Yip

September 27th, 2017

A. <u>Purpose of the Study</u>

Problem 1 - Face Recognition: Implement the SVD for feature reduction, create a logistics regression prediction model based on rank-r approximation and recognize images of human faces using the prediction model.

Problem 2 - What's Cooking: Attempt to predict type of cuisine based on ingredients of sample dishes by testing and verifying the accuracy of three different prediction models (Gaussian, Bernoulli, and Logistic Regression).

B. <u>Procedure</u>

Problem 1 - Face Recognition

After importing and reshaping both the testing and training data, we print out the 10th image in grayscale (figure 1). Then, we compute the average face µ from the training set by taking the average of each column for every sample (*train_data.mean(axis=0)*). We display the average face (figure 2), and then compute mean subtraction by subtracting the average face vector from every row in the training data, which is equivalent to subtracting µ from every column in the set. To verify what effect this has on the training data, we print out the new 10th image (figure 3), and compare this one to the original image (figure 1). This step is repeated for the test set as well. Now that all of the samples in the training set have been **normalized**, moving forward we will use this new matrix for SVD and future computations.

We then perform Singular Value Decomposition (SVD) on the training set, which gets split up into vectors U, Σ, and V transpose. In order to achieve this, we use the np.linalg function, but also have to resort to np.diag in order to obtain Σ. V transpose has the same dimensions as the original training set, and we refer to each row in it as the *i*th eigenface, of which we display the first 10 (see figures 4 and 5).

Having U, Σ, and V transpose, we can now find the rank-r approximation of the original *training_data* matrix by computing the dot product of the first r elements of Σ, the first r columns of U, and the first r rows of V transpose, the result of which is stored as $\mathbf{X}_r$. These operations can be accomplished with a simple for loop which iterates over values 1 through 200 for r. For each iteration, we want to compute the rank-r approximation error, which is simply $||X - \mathbf{X}_r||$. The rank-r approximation error is then plotted as a function of r (figure 6).

For our last computation, we wish to generate an r -dimensional feature matrix (F and F_{test}, respectively) which we accomplish by calculating the dot product of the training data with the transpose of the first *r* rows of V transpose. We again use a *for* loop to compute this for r's 1 through 200, both for the train and the test matrices.

We then set r = 10, and compute two 10-dimensional feature matrices, F and $F_{test.}$ Using LogisticRegression() and OneVsRestClassifier(), we create a Logistic Regression prediction model with the "ovr" mode, based on F and the training labels. To check the accuracy score for our model, we use the sklearn.metrics.accuracy_score() function on F_{test} and testing labels. We then compute the accuracy score for each r from 1 to 200, and plot a line graph to display the set of classification accuracy on the test set (figure 7).

<u>Problem 2: What's Cooking</u>

We download the .JSON files containing the information and count the number of dishes, categories, and unique ingredients in the test data set (see results). The unique ingredients (stored in *ingredients_list*) are found by flattening the column containing all ingredients for each dish and then finding the set.

Each dish needs to be represented by a binary ingredient feature vector, with a 1 representing if an ingredient is present, and a 0 otherwise (for each of the 6714 ingredients). We do this by first initializing an n x p matrix full of zeros (where n is the number of sample dishes, and p being the number of ingredients), and making a dictionary from the ingredients_list. Then for each row (sample dish) in the training data, we go through each ingredient, and retrieve its position in the aforementioned dictionary, filling with a 1 in the newly initialized matrix for that particular column index (we have the row number from the parent loop), and a 0 otherwise. The same procedure is repeated for the test data.

Using Naïve Bayes, we perform 3 fold cross-validation on the training set and, trying both Gaussian distribution prior assumption and Bernoulli distribution prior assumption. We do this with only the first 6000 samples because that is when the accuracy starts to stabilize. The average accuracies are discussed in the results section, and variances in performance are discussed in the conclusion. We then use the Logistic Regression Model to also perform 3 fold cross-validation on the training set (the average classification accuracy of which can also be found in the results section).

Finally, we pick the classifier with the highest average accuracy (in this case Logistic Regression), and train it with all of the training data, and generate test labels on test set, which we submit to Kaggle.

C. Results

Problem 1: Face Recognition

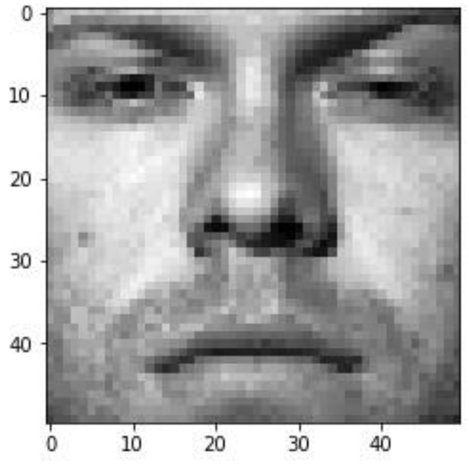

Figure 1: Sample image of training data

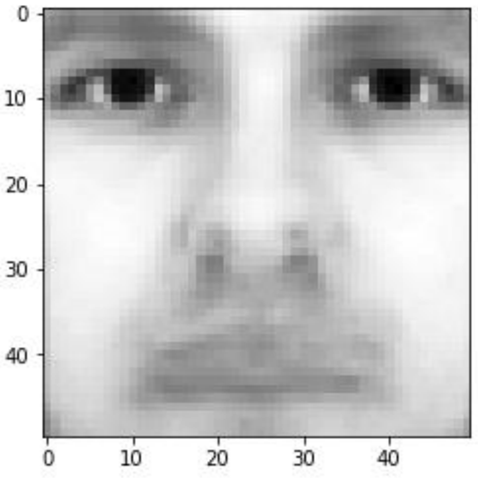

Figure 2: Average face of all training data

When compared to figure 1, the average face in figure 2 shows certain characteristics, such as blurriness in expected areas (nose, nostrils which are in a different position for each person and are difficult to centralize), and a lack of texture in certain areas (cheeks and eyes that usually have different tones of gray in them and tend towards white/black respectively turn almost completely white and black). **The average face can be treated as a uniform bias of all faces**.

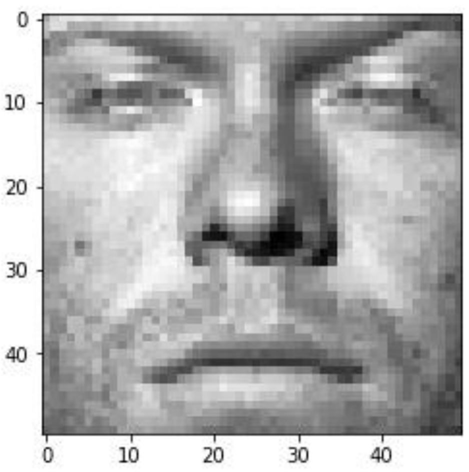

Figure 3: Figure 1 (original sample face) - Figure 2 (average face)

In figure 3, we can observe that this normalized image has a lighter tone and blurriness within certain areas, such as the eye and lips. Overall, the face in figure 3 still strongly resembles the sample face.

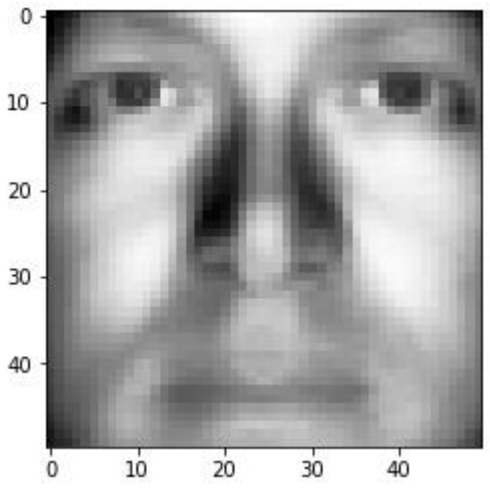

Figure 4: Example of Eigenface, r= 1

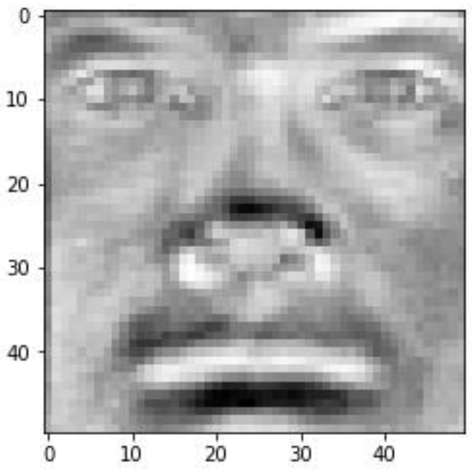

Figure 5: Example of Eigenface, r = 10

Figure 4 is an eigenface of one of the faces in the training data set, where r=1. It is a more efficient representation than the original image due to SVD, which results in a face that is lower resolution. It is worth noting that Figure 5, which is a higher-level Eigenface than Figure 1 has significantly more detail added to it. This makes sense because the lowest eigenface contains the basic information needed to identify a human face in general, whereas the detail in Figure 5 and higher order eigenfaces is combined and used to distinguish an individual.

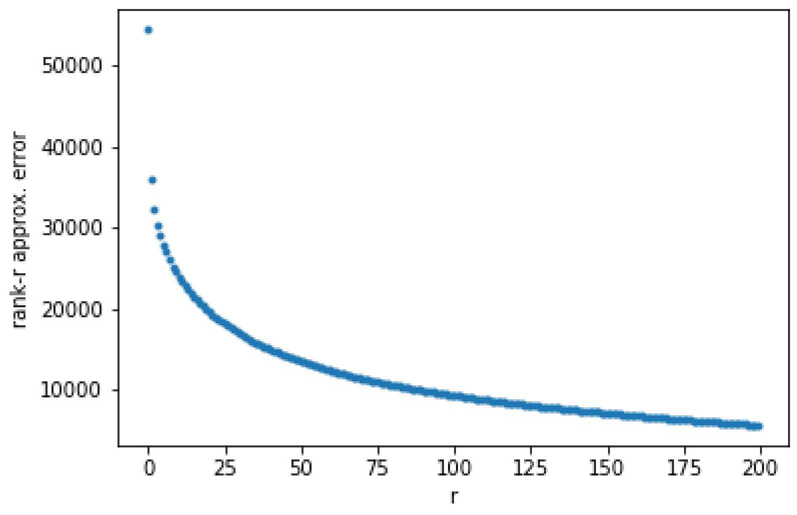

Figure 6: Rank-r approximation error when r = 1 to 200

As expected, the approximation error decreases significantly as r (and subsequently the amount of information) increases, but there is a diminishing return on the reduction of the error margin as r increases.

Report the classification accuracy on the test set (for 10th element of r)
0.79

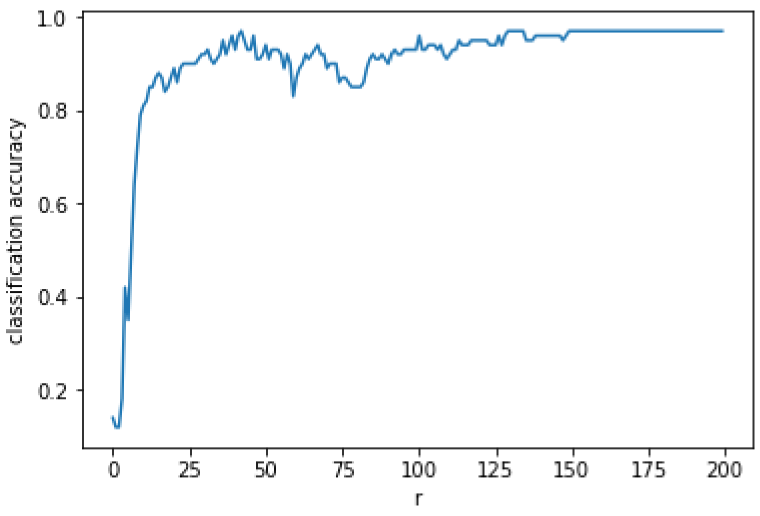

Figure 7: Classification accuracy when r = 1 to 200

We see a general trend towards improvement as we increase the value of r, however, the classification accuracy fluctuates after r = 50 and stabilizes after r = 125.

Problem 2: What's Cooking

How many samples (dishes) are there in the training set?
39774
How many categories (types of cuisine)?
20
How many unique ingredients are there?
6714

Report your average classification accuracy

- Gaussian: 0.54822
- Bernoulli: 0.71244
- Logistic: 0.8776

It follows that Bernoulli has a higher accuracy than Gaussian, but Logistic is higher than both of those models.

Report the final accuracy (of test labels submitted to Kaggle)
0.74718 using Logistic Regression

D. Conclusion

How well your solution works

Problem 1: Eigenface for face recognition
The solution is satisfactory, as the accuracy score is 0.79 when we did the rank-10 approximation. Based on figure 5, the approximation error decreases as the number of ranks (r) increases, which is what we expected. Since more features are taken into account in the prediction model as r goes up, there will be less error. On the other hand, based on figure 6, the classification accuracy generally goes up from r = 1 to 125, and plateaus afterwards. This shows us that as more features are taken into account in the model, the accuracy goes up. Therefore, the outcome of our prediction model is what we expected, where approximation error decreases and classification accuracy increases as r goes up.

Problem 2 - What's Cooking
The final accuracy of the prediction labels of the test data was ~0.75, lower than expected given the average classification accuracy of ~0.88 across the three folds of the testing data. However, given that no manipulation and data cleaning was performed this is not a terrible result.

Any insights you found

Problem 1: Eigenface for face recognition

We learned that SVD is a good tool for feature reduction. As shown in figure 6, the classification accuracy increases as r goes up from r = 1 to 125 and stabilizes afterwards. This tells us that if we want to be lean, an r value around 125 might be optimal for rank-r approximation. If resources are constrained, we do not need to necessarily consider the other 75 features into the prediction model, since the classification accuracy does not increase significantly thereafter. The lesser amount of features would reduce costs such as time and memory space. On the other hand, after r = 125, the rank-r approximation error decreases in a lower rate. However, it would be best if we consider all features within the prediction model for minimal error and maximum accuracy. Therefore, when using the SVD, it is important to balance between cost and accuracy given the amount of resources.

Problem 2: What's Cooking

For Gaussian prior and Bernoulli prior, which performs better in terms of cross-validation accuracy? Why?

Bernoulli is the better performer. Gaussian assumes that features follow a normal distribution, which is not necessarily the case with the ingredients in our samples. Bernoulli on the other hand is useful when feature vectors are binary (i.e. zeros and ones), which is the case in this exercise: either the ingredient is present, or it is not. Therefore the Bernoulli distribution is much more accurate, because it is the more appropriate distribution.

However, ultimately Logistic Regression still performed better than either Bayes' prior, because it takes into consideration the relationship between dependent binary variables (ideal for our model) and one or more nominal independent variable (in this case the type of cuisine).