MP2: Principal Component Analysis (PCA), K-Nearest Neighbors (KNN)

Group: Joey Byrnes, Ojus Deshmukh, Akhil Alapaty

I. Introduction

The first feature vector being called on the images is the Raw Pixel vector. As we know, the raw image vector contains minimally processed data from the original images. We reshaped and resized the original image into different vectors in order to compute various sizes of raw pixel feature vectors. Next, we implemented the PCA vector through the eigenvalue decomposition algorithm given to us. The PCA extraction method is better than the Raw Pixel method because as the name suggests, it only measures the data in terms of the principal components, which are defined as those with the most variance. Each eigenvector has a respective eigenvalue, which is the number that contains how much variance exists in the data in that given direction. The principal component would be the eigenvector that has the highest corresponding eigenvalue. This was carried out by applying the Gram matrix transformation on our data.

The last feature we used was Random Projection feature vector, which reduces the size of the matrices, but still preserves the approximate distances between each of the samples in the dataset. Random Projection feature vectors are expected to be better representations than Raw high-dimensional Pixel feature vectors because they cut down the runtimes and make algorithms, such as the EM algorithm, run smoother. However, in the context of our data, we saw the Raw Pixel vector accuracies being higher when compared to the Random Projection vector. A major part of the implementation was to project the random vector onto the raw vector. Finally, the learning algorithm being used in this MP is the kNN classifier, which is used to compare the images after being processed by each of the above feature vectors. It incorporates a “leave one out” strategy, which refers to 1 image being part of the test dataset, while the other 79 images are part of the train dataset. This cycles through all 80 images until accuracies are computed for each of them using the nearest neighbor algorithm with values of 1 and 5 for k.

II. Methods

A. Gram Matrix

1. We first made subtracted the mean of the raw data from the raw data, to center all the values around 0.
2. We multiplied the resulting matrix with the transpose of itself to get the Gram matrix. This Gram matrix allows us to reduce the dimensions of our matrix from 6300 to 80 without losing a significant amount of information. This was done in line #7 of “pca.m”.
3. If A is a set of m vectors, then the gram matrix G can be computed through:

$$G = A^T A$$

B. Projection onto PCA eigenvectors

1. After calculating the gram matrix of the image data as described above, we found the top N principal components of the data. To do this, we calculated the minimum number of eigenvectors required for keeping 95% on the energy.
2. We found that we needed 68 eigenvectors to keep 95% of the energy. We selected the eigenvectors corresponding to the 68 largest eigenvalues, and projected our image data onto them. See lines 12-15 of pca.m
3. To project the data, we tried two methods. First we tried multiplying the eigenvectors by the image data directly as described in the walkthrough. However, since we used the gram matrix to reduce time we found that it was more accurate to project our data by multiplying the gram matrix by the selected eigenvectors and then multiplying by the square root of the eigenvalue matrix. See lines 30 - 34 of pca.m.

$$: \ U = Z * V * S^{-1}$$

C. Random Projection

1. Using the randn function in matlab, we generated a projection matrix of size 6300 by N, which was 68 for us.
2. Next, we projected each of the 6300 x 1 images onto the aforementioned Random Projection matrix, which gives a N x 1 feature vector. This was done in line 26-27 of “mp2.m”.
3. If R is the random eigenvectors generated, D is the raw data vector, and F is the feature vector, then the Random Projection can be computed through:

$$F = R^T D$$

III. Results

```
>> run('newimdata')
Feat = Raw, Input Dims = [90,70],  (row 1: kNN = 1) (row 2: kNN = 5)
A:          B:          C:          D:          Overall:
  100.0000    75.0000    80.0000   100.0000    88.7500
  100.0000    80.0000    60.0000    95.0000    83.7500

Feat = Raw, Input Dims = [45,35],  (row 1: kNN = 1) (row 2: kNN = 5)
A:          B:          C:          D:          Overall:
  100.0000    75.0000    85.0000   100.0000    90.0000
  100.0000    80.0000    70.0000    95.0000    86.2500

Feat = Raw, Input Dims = [22,17],  (row 1: kNN = 1) (row 2: kNN = 5)
A:          B:          C:          D:          Overall:
  100.0000    80.0000    85.0000   100.0000    91.2500
  100.0000    85.0000    65.0000    95.0000    86.2500

Feat = Raw, Input Dims = [ 9, 7],  (row 1: kNN = 1) (row 2: kNN = 5)
A:          B:          C:          D:          Overall:
  100.0000   100.0000    95.0000   100.0000    98.7500
  100.0000    90.0000    65.0000    95.0000    87.5000

Feat = PCA, Input Dims =  6300  ,  (row 1: kNN = 1) (row 2: kNN = 5)
A:          B:          C:          D:          Overall:
   95.0000    90.0000    85.0000   100.0000    92.5000
   95.0000    75.0000    45.0000   100.0000    78.7500

Feat = Rand, Input Dims = 6300  ,  (row 1: kNN = 1) (row 2: kNN = 5)
A:          B:          C:          D:          Overall:
   95.0000    80.0000    85.0000    85.0000    86.2500
   95.0000    70.0000    85.0000    70.0000    80.0000
```

IV. Discussion

The average accuracy for a 90x70 Raw Pixel vector when k = 1 is 88.75%. The average accuracy for a 90x70 Raw Pixel vector when k = 5 is 83.75%. In theory, we expect the average accuracy to be higher when k = 5 because k represents the number of nearest neighbors that we wish to take a vote from in order to determine the class of a given test image. On the other hand, when k = 1, the algorithm just determines the class of a test image by the class of the nearest neighbor for that image. However, since the trainset is only 79 images, this expectation isn't reflected in the data because it's more likely to pick 1 of the necessary 19 images rather than picking 5 of those 19 images. Regardless, we believe that if we had significantly more train images to work with, we would see that the average accuracy when k = 5 would be higher than the average accuracy when k = 1.