MP2:Face Recognition

Group:Siheng Pan, Yangge Li

1. Introduction

In this mp, we are using raw pixels, PCA and random projections as features to solve the problem. In our case, the PCA is more useful than raw pixel data because in PCA analysis we are able to get similar result as the raw pixel and the numbers of data in PCA is much smaller than raw pixel.

In this MP, we are using the knn classifier to classify the images into four distinct group. The algorithm learns

2. Methods

For the raw pixels, we just reshape each image to a column vector and return the vector as the feature of each image.

The algorithm we use for raw pixels is illustrated as followed:

function k-NEARESTNEIGHBORS(test sample $\boldsymbol{x}$, Training Set $\mathcal{T} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N}$, dissimilarity measure between feature vectors $\Delta(\cdot,\cdot)$)
  for i=1,...,N **do**
    Calculate $\delta_i = \Delta(\boldsymbol{x}, \boldsymbol{x_i})$
  end for
  Find indices $\{i_1, \ldots, i_k\}$ corresponding to k smallest values of δ_i (break ties arbitrarily).
  return most frequently occurring element in $\{y_{i_1}, \ldots, y_{i_k}\}$ (break ties arbitrarily).
end function

(Reference from ECE398 course note, *Fundamentals of Machine Learning*, J. G. Ligo and V. V. Veeravalli, February 8, 2017)

Then after predicting the face each image belongs to, we calculate the accuracy of recognizing each person's face.

For PCA analysis, we first subtract the mean with respect to each pixel from the raw pixel data X and get matrix Z.

Secondly, we Compute the scatter matrix S by

$$S = (1/(M-1)) * (Z*Z')$$

(Reference from ECE417 MP2 Walkthrough)

Then, we Compute the eigenvalue of the scatter matrix S using matlab function eig

Then, we sort the eigenvalues from high to low and find the top N principal components that keep K% of the total energy.

After that, we project the data on selected eigenvectors and get the PCA feature.

At the end, we reconstruct the data and using kNN algorithm for k =1 and k =5 again to do classifying. As well, we calculate out the accuracy of recognizing each person's face.

For random projection, we first use matlab function randn to generate a projection matrix of size 6300*N, where N is the number of principal components we get from the PCA part. Similarly, we reconstruct the data with the projected data and test the accuracy.
Then we project the raw pixels to the random subspace and get feature vectors for each of the image.

3. Results

```
>> run('./newimdata/')
========== extracting features ==========

========== calculating knn-1 ==========

Feat = raw, Input Dims = [90 70], kNN = 1
A: 100%  B: 75%  C: 80%  D: 100%  Overall: 88.75%

Feat = raw, Input Dims = [45 35], kNN = 1
A: 100%  B: 75%  C: 85%  D: 100%  Overall: 90%

Feat = raw, Input Dims = [22 17], kNN = 1
A: 100%  B: 80%  C: 85%  D: 100%  Overall: 91.25%

Feat = raw, Input Dims = [9 7], kNN = 1
A: 100%  B: 100%  C: 95%  D: 100%  Overall: 98.75%

Feat = 95% PCA, Input Dims = [70 90], kNN = 1
A: 100%  B: 75%  C: 80%  D: 100%  Overall: 88.75%

Feat = ramdom, Input Dims = [70 90], kNN = 1
A: 100%  B: 75%  C: 60%  D: 85%  Overall: 80%

========== calculating knn-5 ==========

Feat = raw, Input Dims = [90 70], kNN = 5
A: 100%  B: 80%  C: 65%  D: 95%  Overall: 85%

Feat = raw, Input Dims = [45 35], kNN = 5
A: 100%  B: 80%  C: 70%  D: 90%  Overall: 85%

Feat = raw, Input Dims = [22 17], kNN = 5
A: 100%  B: 85%  C: 60%  D: 90%  Overall: 83.75%

Feat = raw, Input Dims = [9 7], kNN = 5
```

A: 100% B: 90% C: 60% D: 95% Overall: 86.25%

Feat = 95% PCA, Input Dims = [70 90], kNN = 5
A: 100% B: 70% C: 60% D: 95% Overall: 81.25%

Feat = ramdom, Input Dims = [70 90], kNN = 5
A: 100% B: 60% C: 45% D: 85% Overall: 72.5%

4. Discussion

From the results above, as dimension of images decreases, the accuracy of kNN algorithm usually increases. What is interesting in the result is that, against what people may think that as the number of neighbors increases, the accuracy should increases, it turns out the accuracy decreases with k selected to be higher in all dimensions. PCA happens to have lowest accuracy except for random generation method where the accuracy entirely relies on luck. Does it mean PCA cannot outperform kNN? We need more labs to discover.

5. Extra credit

For testing PCA with different amount of energy, we are getting different result.

The PCA with energy 90% show same result for knn-1 but the accuracy for knn-5 is lower than that of PCA with 95% energy.

The PCA with energy 98% still show same result for knn-1 but the accuracy for knn-5 is slightly higher than that of PCA with 95% energy.

========== pca with energy 90% ==========

Feat = 90% PCA, Input Dims = [70 90], kNN = 1
A: 100% B: 75% C: 80% D: 100% Overall: 88.75%

Feat = 90% PCA, Input Dims = [70 90], kNN = 5
A: 100% B: 55% C: 50% D: 100% Overall: 76.25%

========== pca with energy 98% ==========

Feat = 98% PCA, Input Dims = [70 90], kNN = 1
A: 100% B: 75% C: 80% D: 100% Overall: 88.75%

Feat = 98% PCA, Input Dims = [70 90], kNN = 5
A: 100% B: 75% C: 60% D: 95% Overall: 82.5%

For the 10 more random projections, the accuracy various a lot. The average accuracy for knn-1 is 84.5%, which is lower than the accuracy for both PCA and raw pixels. The average

accuracy for knn-5 is 74.625%, which is lower than the accuracy for both PCA and raw pixels.

========== 10 more random projections ==========

Feat = ramdom, Input Dims = [70 90], kNN = 1
A: 95% B: 90% C: 75% D: 95% Overall: 88.75%

Feat = ramdom, Input Dims = [70 90], kNN = 5
A: 90% B: 75% C: 55% D: 75% Overall: 73.75%

Feat = ramdom, Input Dims = [70 90], kNN = 1
A: 90% B: 65% C: 70% D: 85% Overall: 77.5%

Feat = ramdom, Input Dims = [70 90], kNN = 5
A: 100% B: 60% C: 50% D: 70% Overall: 70%

Feat = ramdom, Input Dims = [70 90], kNN = 1
A: 95% B: 85% C: 75% D: 95% Overall: 87.5%

Feat = ramdom, Input Dims = [70 90], kNN = 5
A: 100% B: 80% C: 45% D: 90% Overall: 78.75%

Feat = ramdom, Input Dims = [70 90], kNN = 1
A: 100% B: 70% C: 80% D: 90% Overall: 85%

Feat = ramdom, Input Dims = [70 90], kNN = 5
A: 95% B: 75% C: 70% D: 75% Overall: 78.75%

Feat = ramdom, Input Dims = [70 90], kNN = 1
A: 100% B: 60% C: 85% D: 90% Overall: 83.75%

Feat = ramdom, Input Dims = [70 90], kNN = 5
A: 95% B: 65% C: 35% D: 85% Overall: 70%

Feat = ramdom, Input Dims = [70 90], kNN = 1
A: 100% B: 70% C: 85% D: 95% Overall: 87.5%

Feat = ramdom, Input Dims = [70 90], kNN = 5
A: 95% B: 85% C: 70% D: 85% Overall: 83.75%

Feat = ramdom, Input Dims = [70 90], kNN = 1
A: 90% B: 70% C: 75% D: 80% Overall: 78.75%

Feat = ramdom, Input Dims = [70 90], kNN = 5
A: 100% B: 65% C: 60% D: 65% Overall: 72.5%

Feat = ramdom, Input Dims = [70 90], kNN = 1
A: 90% B: 80% C: 95% D: 95% Overall: 90%

Feat = ramdom, Input Dims = [70 90], kNN = 5
A: 90% B: 65% C: 60% D: 80% Overall: 73.75%

Feat = ramdom, Input Dims = [70 90], kNN = 1
A: 100% B: 70% C: 90% D: 75% Overall: 83.75%

Feat = ramdom, Input Dims = [70 90], kNN = 5
A: 100% B: 75% C: 55% D: 60% Overall: 72.5%

Feat = ramdom, Input Dims = [70 90], kNN = 1
A: 100% B: 80% C: 75% D: 75% Overall: 82.5%

Feat = ramdom, Input Dims = [70 90], kNN = 5
A: 100% B: 75% C: 50% D: 65% Overall: 72.5%

average accurcy:
knn-1 A: 96% B: 74% C: 80.5% D: 87.5% Overall: 84.5%
knn-5 A: 96.5% B: 72% C: 55% D: 75% Overall: 74.625%