CSE 6363 HW3

The given data is a set of pgm images of faces of 40 people provided by **AT&T Laboratories Cambridge** . There are 40 different classes(**s1 to s40**) and each class consists of 10 images each.

Initial Steps:

1. Read all the images and save in an image set **all\_Images.**
2. Vectorize all the images and save in **images\_vector**.
3. Create a matrix of all the vectorized images **images\_array**(400\*10304)

Steps for Task 1, 1.1, 1.2:

Since we are doing 5fold cross validation,

---Start of PCA for 1st fold-----

1. For 1st fold,

Consider 1.pgm and 2.pgm of each class as test data i.e. **test1**.

Consider rest images from each class as train data i.e. **train1.**

So we have total **80 test data** and **320 train data.**

1. To centerize the images, calculate mean of train data **u\_train\_1**. Then subtract mean from **train1** . The centerized images that we get is **Z\_train1.**
2. Similarly calculate mean of test data **u\_test\_1** , subtract from **test1**. The centerized images that we get is **Z\_test1.**
3. Compute scatter matrix **S1** using **S1=319\*cov(Z\_train\_1).**
4. Compute the eigen values and eigen vectors using **[V1,D1]=eig(S1). D1** has the eigen values and **V1** has the eigen vectors.
5. As per the plot of eigen values, the value of reduced dimension is taken as **80**.
6. Create matrix **E1** such that it has eigen vectors from **V1** corresponding to the top **80** eigen values from **D1.**
7. The train and test data with reduced dimensions is **train1\_red** and **test1\_red** respectively. These are calculated as **train1\_red= E1\* Z\_train1** and **test1\_red= E1\* Z\_test1.**
8. Create the label of the test data in **label1**.

--- End of PCA for 1st fold----

---Start of KNN for 1st fold-----

1. We need to calculate the Eucledian distance between test data and train data. Eucledian distance of each test data is computed with all 320 train data.
2. All the calculated Eucledian distances are stored in a matrix **dist1mat**.
3. Each test image has **320** distances calculated. Find the minimum distance among these 320 distances as it is **1NN.**
4. Find the label of the corresponding train image that produced minimum distance with test image. All such labels are stored in **label1pred**.
5. Count the number of matching labels in **label1** and **label1pred.** We get **count.**
6. The accuracy of 1st fold Knn we get is:

**acc1= count/80\*100**

**= 96.25 %**

---End of KNN for 1st fold-----

1. Perform similar steps as from **Step 1 – 15** for rest of the **4 folds.**

For 2nd fold,

Consider 3.pgm and 4.pgm of each class as test data i.e. **test2**.

Consider rest images from each class as train data i.e. **train2.**

**acc2 = 98.75%**

For 3rd fold,

Consider 5.pgm and 6.pgm of each class as test data i.e. **test3**.

Consider rest images from each class as train data i.e. **train3.**

**acc3 = 96.25 %**

For4th fold,

Consider 7.pgm and 8.pgm of each class as test data i.e. **test4**.

Consider rest images from each class as train data i.e. **train4.**

**acc4 = 98.75 %**

For5th fold,

Consider 9.pgm and 10.pgm of each class as test data i.e. **test5**.

Consider rest images from each class as train data i.e. **train5.**

**acc4 = 95%**

1. The overall Knn accuracy is calculated as

**knnaccu\_pca = (acc1+ acc2+ acc3+ acc4+ acc5)/5**

**= 97.5%**

Steps for Task 2:

1. All the images are read, vectorized and resized using function **imresize().**
2. The resized images i.e. **56\*46** are stored in matrix **images\_array\_resize.**
3. **PCA** is performed over these resized images. As per the plot, the value of reduced dimension is taken as **20**. along with 5 fold cross validation and Knn similar to **Steps 1-17.**

For 1st fold,

**acc1\_resize = 96.25%**

For 2nd fold,

**acc2\_resize = 95%**

For 3rd fold,

**acc3\_resize = 96.25%**

For 4th fold,

**acc4\_resize = 95%**

For 5th fold,

**acc5\_resize = 97.5%**

The overall Knn accuracy for resized images is :

**Knnaccu\_pca\_resize = (acc1\_resize + acc2\_resize + acc3\_resize + acc4\_resize + acc5\_resize)/5;**

**= 95.625%**

Comparison of results:

**knnaccu\_pca=97.5%**

**Knnaccu\_pca\_resize=95.625%**

Thus the accuracy decreases as we resize the image.

Steps for Task 3:

-------Start of LDA for 1st fold -----------

1. For 1st fold,

Consider 1.pgm and 2.pgm of each class as test data i.e. **test1\_lda**.

Consider rest images from each class as train data i.e. **train1\_lda.**

So we have total **80 test data** and **320 train data.**

1. Store labels of train images in **c1train\_lda** and labels of test images in **c1test\_lda.**
2. Form separate matrix of train data for each class i.e. **c1\_train1 , c2\_train1, .., c40\_train1.** Each of these matrices consists of the **8 train images.**
3. Calculate mean of all training images as **meantotal\_train1\_lda** and mean of all testing images as **meantotal\_test1\_lda.**
4. Calculate mean of each **c1\_train1 , c2\_train1, .., c40\_train1** and store in **mean1\_lda.**
5. Calculate the between scatter matrix **Sb1\_lda** and within scatter matrix as **Sw1\_lda.**

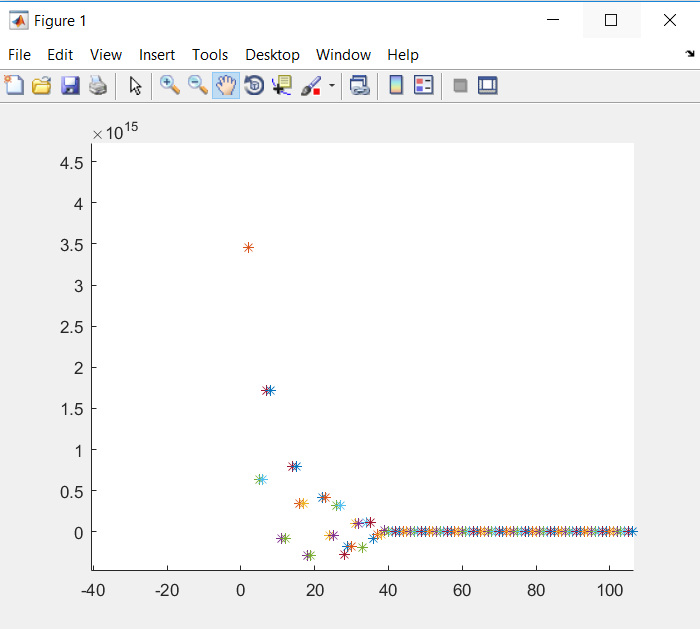
**Sw1\_lda=(double(train1\_lda)-mean1\_lda(c1train\_lda,:))'\*(double(train1\_lda)-mean1\_lda(c1train\_lda,:));**

**Sb1\_lda=(ones(40,1)\*meantotal\_train1\_lda-mean1\_lda)'\*(ones(40,1)\*meantotal\_train1\_lda-mean1\_lda);**

1. Calculate the eigen values **D1\_lda** and eigen vectors **V1\_lda** from **Sw1\_lda** and **Sb1\_lda.**

**[V1\_lda,D1\_lda]=eig(Sw1\_lda\Sb1\_lda);**

1. Sort the eigen vectors in descending order and plot the eigen values. The plot is as below. As from the plot, the value of reduced dimension is taken as **39**. Also we know that value of reduced dimension can be at most c-1 i.e. 40-1 in our case.



1. Then centerize the train and test data by subtracting **meantotal\_train1\_lda** from **train1\_lda** and subtracting **meantotal\_test1\_lda** from **test1\_lda**respectively.
2. The top 39 eigen vectors are considered and stored in **Etrain1\_lda.**
3. The train and test data with reduced dimensions is calculated as **Ytrain1\_red** and **Ytest1\_red** respectively as:

**Ytrain1\_red=Etrain1\_lda.'\*Ztrain1\_lda.';**

**Ytest1\_red=Etrain1\_lda.'\*Ztest1\_lda.';**

1. Further perform knn similar to **Steps 10-15** of Task1 ,1.1, 1.2.
2. **acc1\_lda = 6.25%**
3. For the remaining 4 folds, perform similar steps as above. i.e. **Step 1-13** of Task3

We get,

**acc2\_lda = 11.25%**

**acc3\_lda = 6.25%**

**acc4\_lda = 5%**

**acc5\_lda = 6.25%**

1. The accuracy of Knn using LDA is:

**Knnaccu\_lda = (acc1\_lda+ acc2\_lda+ acc3\_lda+ acc4\_lda+ acc5\_lda)/5**

**= 7%**

Steps for Task 4:

1. For 1st fold,

Consider 1.pgm and 2.pgm of each class as test data i.e. **test1**.

Consider rest images from each class as train data i.e. **train1.**

1. Perform steps similar to **Steps 2-8** of Task 1,1.1,1.2. As said in the question, the value of reduced dimension is taken as **320** i.e. the number of training data.
2. The reduced dimension images are further considered as input to LDA.
3. Perform steps similar to **Step2-13** of Task3 for LDA. In this case as per plot, the reduced dimension is considered as 39.
4. The calculated accuracies for 5 folds are:

**acc1\_pcalda = 6.25%**

**acc2\_pcalda = 11.25%**

**acc3\_pcalda = 6.25%**

**acc4\_pcalda = 5%**

**acc5\_pcalda = 6.25%**

1. Accuracy of Knn for PCA followed by LDA is:

**Knnacc\_pcalda** = **(acc1\_pcalda+ acc2\_pcalda+ acc3\_pcalda+ acc4\_pcalda+ acc5\_pcalda)/5**

**= 7%**

Steps for Task 5:

SVM kernel used: Linear kernel.

SVM strategy used: one v/s all

1. For 1st fold,

Consider 1.pgm and 2.pgm of each class as test data i.e. **test1\_svm**.

Consider rest images from each class as train data i.e. **train1\_svm.**

1. **z1\_svm**: If considered class 1 vs rest. First 8 elements of **z1\_svm** will be 1 and rest will be -1. Similarly the elements of **z1\_svm** will change considering each class as 1 vs rest.

**H1\_svm= (double(train1\_svm)\*double(train1\_svm')).\*(z1\_svm\*z1\_svm');**

**f1\_svm:** vertical vector of -1

**A1\_svm:** Identity matrix of -1

**a1\_svm:** vertical vector of 0

**B1\_svm:** The first row of B is **z**. Rest rows contain all 0

**b1\_svm:** vertical vector of 0

Using quadprog function, we find **alpha1\_svm**

**alpha1\_svm=quadprog(H1\_svm+eye(320)\*0.001,f1\_svm,A1\_svm,a1\_svm,B1\_svm,b1\_svm);**

**w1\_svm=(alpha1\_svm.\*z1\_svm)'\*double(train1\_svm);**

**w1all\_svm:** Stores all the w1\_svm values

**wnot1\_svm=(1/z1\_svm(i,1)) - (w1\_svm(1,:)\*double(train1\_svm(i,:))');**

**wnot1all\_svm:** Stores all the wnot1\_avm values.

1. For all the test images, **r** is calculated as **w\*test1\_svm’+wnot1\_svm**

The maximum value of **r** is found and using this we find the corresponding label of that test image. These labels are stored in **Index1\_svm**.

1. The labels in **test1label\_svm** and **Index1\_svm** are compared and the accuracy that we get is

**acc1\_svm= 86.25%**

1. Perform similar steps for rest 4 folds. Accuracy calculated is:

**acc2\_svm=91.25%**

**acc3\_svm=90%**

**acc4\_svm=82.5%**

**acc5\_svm=90%**

1. Overall SVM accuracy is

**svmacc=( acc1\_svm+ acc2\_svm+ acc3\_svm+ acc4\_svm+ acc5\_svm)/5**

**=88%**

Steps for Task 6:

1. For 1st fold,

Consider 1.pgm and 2.pgm of each class as test data i.e. **test1**.

Consider rest images from each class as train data i.e. **train1.**

1. Perform steps similar to **steps 2-8** of Task 1,1.1,1.2 to perform PCA and reduce dimension of images to 80.
2. These images with reduced dimension is considered as input of SVM. For SVM perform steps similar to **Step 2-4** of Task 5.
3. The accuracy of all 5 folds are:

**acc1\_svm\_red=92.5%**

**acc2\_svm\_red=85%**

**acc3\_svm\_red=88.75%**

**acc4\_svm\_red=82.5%**

**acc5\_svm\_red=80%**

1. Overall SVM accuracy with PCA is:

**svmacc\_pca = (acc1\_svm\_red+ acc2\_svm\_red+ acc3\_svm\_red+ acc4\_svm\_red+ acc5\_svm\_red)/5**

**= 87%**

As per above tasks, SVM is more sensitive to the high dimensionality of data.

**References:**

<http://www.bytefish.de/blog/pca_lda_with_gnu_octave/>

<http://www.csd.uwo.ca/~olga/Courses/CS434a_541a/Lecture7.pdf>

<http://www.csd.uwo.ca/~olga/Courses/CS434a_541a/Lecture11.pdf>

**http://www.csd.uwo.ca/~olga/Courses/CS434a\_541a/Lecture8.pdf**