Project 1

CMSC828C/ENEE633 Statistical Pattern Recognition

November15, 2021

Instructors:
Behtash Babadi

Course:
Statistical Pattern Recognition, Fall 2021

Course code:
CMSC828C/ENEE633

Student:
Siddharth Telang

Contents

1 Introduction

- In this project we implement the below classifiers for face recognition
  - Baye's Classifier , k-NN Classifier , Kernel SVM , Boosted Linear SVM

  As the image data is of high dimension, we use the below dimensionality reduction methods
  - PCA , MDA

2 Data sets

We are provided with three data sets

- Facial expressions(neutral, expression and illumination variants) - 200 subjects with 3 images each
- Pose - 68 subjects with 13 images each of different poses
- Illumination - 68 subjects with 21 images under different illuminations

3 Subject Label Identification

In this task we classify the subject label from the given data sets using the below classifiers. We use PCA and MDA to reduce the dimension of the input data and check their performance on various data sets with different choices of dimensions.

3.1 Bayes' Classifier

- The data set is divided into training and testing data with a 2/3 and 1/3 ratio. Next, PCA is performed and the dimension are reduced so as to keep 95% of the data. The below figure shows a plot of the dimension and variance

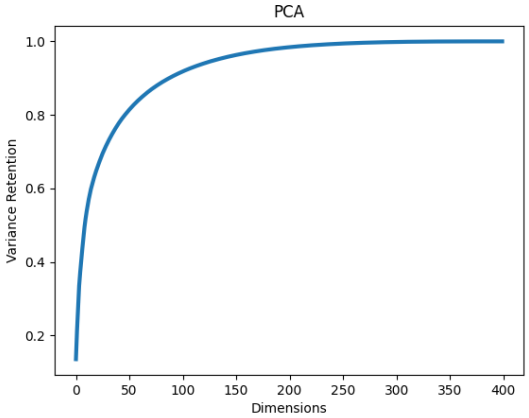

Figure 1: PCA

Following are the results on data set 1 (with PCA)

Training Data set	Testing Data set	Accuracy(PCA)
Expression, Illumination	Neutral	72%
Expression/Neutral, Illumination	Expression/Neutral	72%
Neutral, Illumination	Expression	67%
Neutral, Illumination/Expression	Illumination/Expression	67%

- It can be seen that randomly shuffling the data set results in different accuracy. This is due to the features in the data, if captured properly in the training, we get a better accuracy.
- We also see that when expression and illuminations are in the training and neutral in testing, we achieve max accuracy for raw data(72%).
- However, the results on pose and illumination data set were far better than the face data set.

Data set	Train vs Test Ratio	Accuracy(PCA)
Pose	9:3	60% - 97%
Illumination	13:8	70% - 100%

- As we shuffle the data, we get different accuracy which also reached 100% - no mistakes on the labels, the perfect classifier! If we keep the first few samples of each subject in test data and the rest in training data, we observe a boost in accuracy, which again highlights that it is feature dependent.
- Next, we check the observation on performing MDA. Below are the results

Data set	Train vs Test Ratio	Accuracy(MDA)
Face	2:1	98.5%
Pose	9:3	100%
Illumination	13:8	100%

- Performing MDA with 199 dimensions (number of classes-1), gave an accuracy of 100% even with the random shuffling of data. This highlights that MDA was able to separate the subjects very well.
- However, if we reduce the dimensions in MDA to a very low value, let's say 2 or 4, then the accuracy decreases. 13% when taken 2 dimensions, 46% with 4 dimensions, and 83% with 10 dimensions. With 20 dimensions we achieve 98% accuracy. Hence instead of working with 199 dimensions, we can also work with 20 and get the perfect result.

3.2 K-NN Classifier

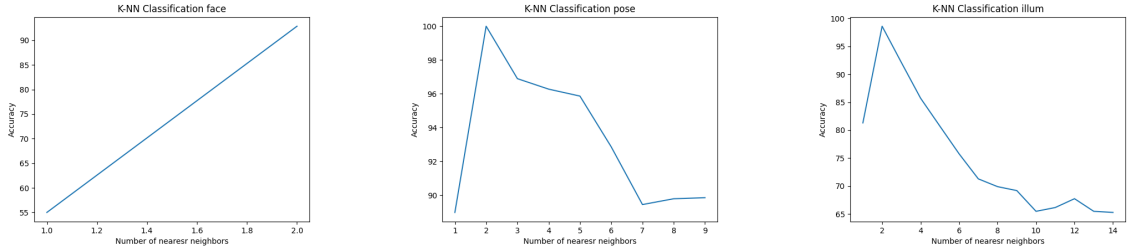

Figure 2: K-NN Subject Classification Accuracy PCA (ties skipped)

- Figure 2 shows the accuracy of K-NN with the number of nearest neighbors with PCA. The maximum accuracy was achieved with 2-NN 95% for all the three data sets.

- Note that we have achieved this accuracy only when we skip those samples for which there is a tie. Otherwise, the accuracy will decrease as we will have to choose randomly when there is a tie.

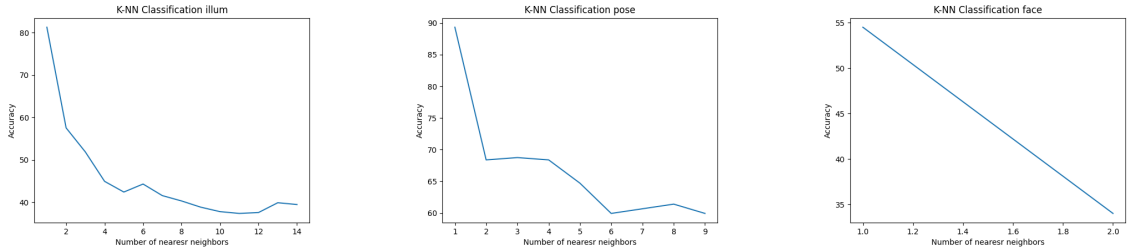

Figure 3: K-NN Subject Classification Accuracy PCA With no tie skipped

- Figure 3 shows what we just discussed. If we randomly assign a point to a class when there is a tie, we get a really bad classifier. We will see now how MDA deals with this issue.

- The results with MDA are shown in Figure 4

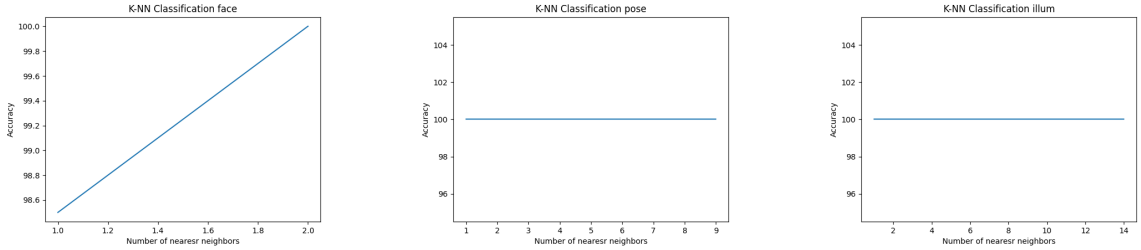

Figure 4: K-NN Subject Classification Accuracy MDA

- We can see that we got a 100% accuracy with MDA!

- Unlike PCA, in the MDA case we did not apply that rule to skip the sample if two classes have same votes. Hence, MDA was able to do a better job in the classification here also.

- On reducing the dimension from total classes - 1, our accuracy decreases, which is illustrated by the below table

MDA Dimensions	Accuracy of 1-NN	Accuracy of 2-NN
2	10.5%	33.33%
4	47.5%	73.8%
10	82%	93%
20	89%	99%

4 Neutral vs Facial Expression

- In this classification, we use the Data set 1 which includes 200 subjects, with either neutral or face expressions. Given any test image, out task is to identify neutral or expression face.
- We divide our data set into training data and testing data (3:4) and (1:4) ratio, which can be modified by simple command line arguments. In this task we also implement the SVM and AdaBoost classifiers.

4.1 Bayes' Classifier

- Below table summarizes the accuracy of Bayes' classifier with varying the training data size and with PCA and LDA.

Training size	PCA Accuracy	LDA Accuracy
10	74%	90%
50	85%	90%
100	87%	91%
150	88.5%	90%
200	89.5%	92%
300	87.5%	91%

- We see that as we increase the training size, PCA with ML gives better accuracy, while this is not the case for LDA.
- Even with just 10 data samples, LDA with ML was able to give 90% accuracy on the rest 390 test samples.

4.2 K-NN Classifier

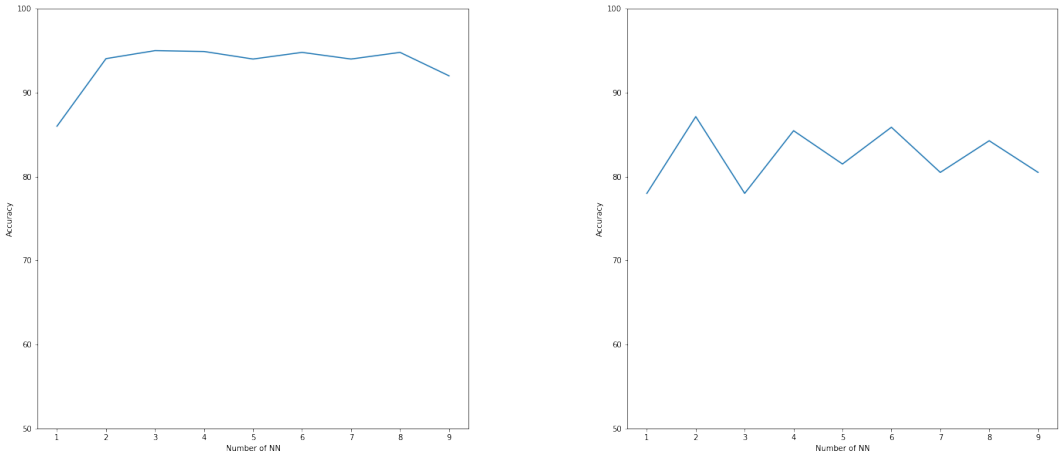

Figure 5: K-NN Pose Classification Accuracy LDA, PCA

- Figure 5 shows the accuracy with K-NN - LDA on the left and PCA on the right To summarize in a tabular form

Nearest Neighbors	LDA Accuracy	PCA Accuracy
1	86%	76%
2	94%	87%
3	93%	79%
4	95%	86%
5	93%	81%
6	94%	86%

- We see that LDA performs better than PCA here. Moreover, taking 2-NN gives us a good classification in both the scenarios, same as we saw in the K-NN subject classification.

4.3 Support Vector Machine (SVM) Classifier

- In this classification task, RBF, Polynomial and Linear kernels were used to classify the neutral or expression face.
- We use the library **cvxopt** solver to solve the dual optimization problem to find the best values of the Lagrangian multipliers.
- The kernels have a parameter - σ for the RBF kernel and r for the polynomial kernel. To determine these, we split our data into train and validation set and evaluate the parameters on the validation set to get the best value of our parameters.
- We also consider a soft margin in this case and test it's accuracy.

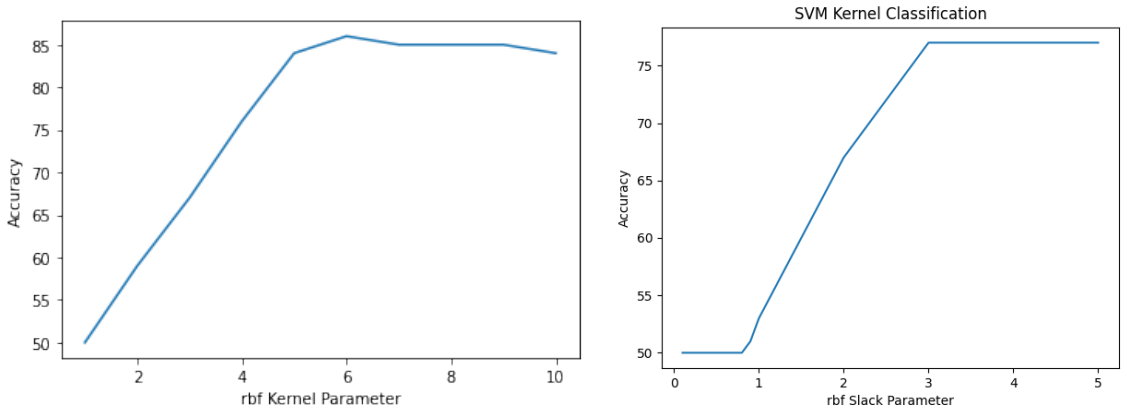

Figure 6: Accuracy vs RBF Kernel Parameter

- Figure 6 shows the accuracy vs RBF kernel parameter. We see that the kernel parameter $\sigma = 6$ gives the best accuracy of around 86%. The choice of the parameter is a crucial task and hence needs to be chosen wisely.
- We also test the values for C from 1 to 6 for $\sigma = 6$ - Figure 5 on the right side. We observed that with the for the soft margin of 3, we get the best accuracy. Moreover, these values are bound to change if we change the dimensions of our data.

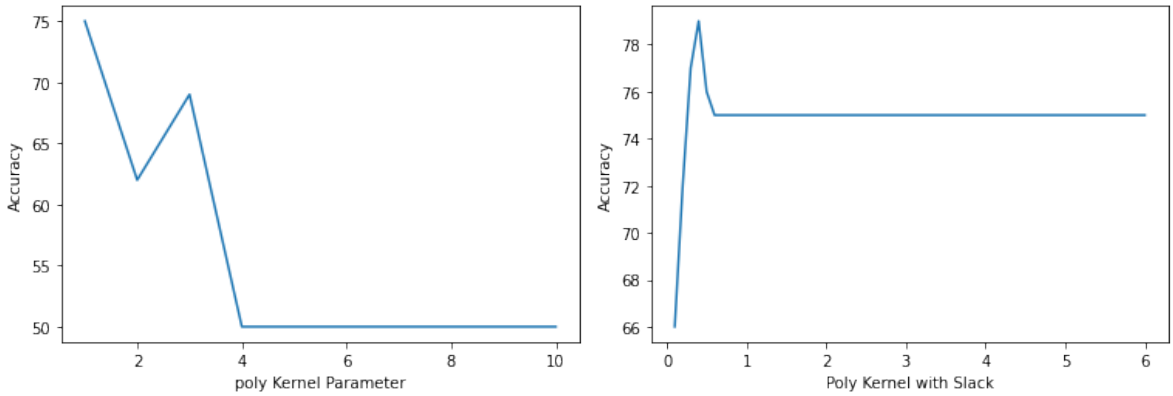

Figure 7: Accuracy vs Polynomial Kernel Parameter

- Figure 7 shows the accuracy of SVM with Polynomial kernel parameter. We observe that with the polynomial order = 1, we get the maximum accuracy of around 75%(left). Moreover, the value of C = 0.6 gives a good accuracy(right).
- With a linear kernel, the SVM gives an accuracy of around 80% using a soft margin when C = 0.6.

4.4 Boosted SVM - Linear Kernel

- As we saw in previous subsection, the accuracy of the Linear SVM classifier was maximum 80%, we will apply the AdaBoost algorithm to the linear kernel SVM classifier to boost it's accuracy on training data set and then check the errors made on the test data set.

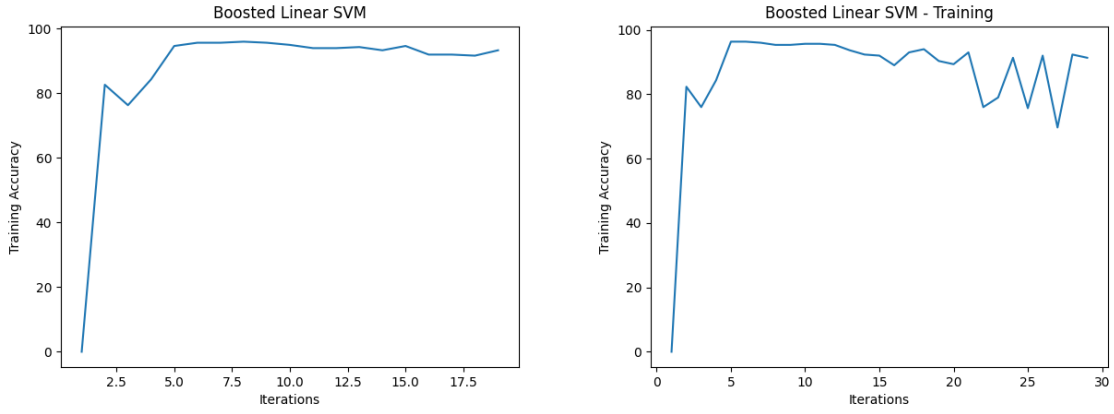

Figure 8: Accuracy on training data set vs the iterations

- Figure 8 shows the accuracy of the linear boosted SVM on the training data set over 20 and 30 iterations of the AdaBoost.
- We observe that we get almost 100% accuracy, that is zero training error.

- Figure 9 shows the accuracy of the linear boosted SVM on the testing data set after 20 and 30 iterations of the AdaBoost. Note that these are the unseen samples. The boosted classifier makes around 20% error on the unseen samples.

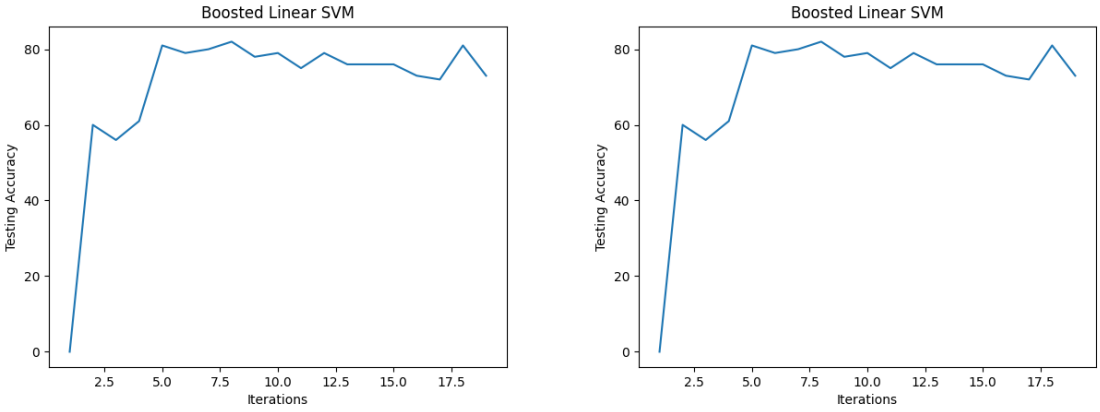

Figure 9: Accuracy on testing data set vs the iterations

- Importantly, we note that after boosting, we got almost zero training error, which is nice and just 20% error on the unseen samples. Hence, our AdaBoost algorithm was able to boost the SVM's accuracy on the training set.

5 Conclusion

- The accuracy of any classifier depends on the data provided to it. We saw that if we shuffle the data in the training and testing sets we obtain different results for some classifiers.
- Dimensionality reduction plays a critical part. It takes very high computational power if we process the raw data. This technique helps to bring the dimension down while keeping the desired features. We observed differences in the ac curacies with PCA, MDA, and LDA. Overall, MDA gave us better results than PCA, as it tries to separate the data.
- As we increase the number of samples/observations, the probability of making errors by the Bayes' classifier is reduced, and at some point it predicts the new test sample perfectly.
- K-NN gave 95% accuracy with 2-NN. However, if there is a tie, we have to randomly assign it to one of the classes, which will reduce the accuracy. In PCA case, we found that there are a lot of ties which decreased the accuracy. However in the MDA case, we do not skip such samples and still get 95% accuracy. This shows that MDA should be used with K-NN as it maximizes the distance between different classes and we thus poll the correct neighbors.
- SVM with a RBF and polynomial kernel were good classifiers with almost 86% accuracy, but required kernel parameter estimation on validation data set.
- AdaBoost boosted the linear kernel SVM classifiers training accuracy, with almost zero errors on training data set and 20% errors on test data set.

6 Challenges faced

- Singular matrices: The covariance matrices were found to be singular and hence non-invertible. Noise was added along the diagonal elements till the matrix determinant was greater than zero.
- Complex eigen values and vectors while doing MDA: In this case a proper threshold value of determinant was selected to avoid the complex values.
- k-NN with PCA gave really bad results (not the case with MDA). When checked, an observation was made where there were multiple ties and as the samples were randomly assigned to one of the class, the accuracy fell below 50%. In this case, we have skipped the sample, that is we do not make a decision. In future works, we intend to come up with an algorithm which can estimate the correct class and assign the sample properly if there is a tie.