SEP786 AI and Machine Learning Fundamentals

Course Project - Comparing Feature Extraction Approaches

Name: Ken Tsoi

MacID: tsoip

Student No: 400284550

Data Used:

Dataset used is from the UCI Machine Learning Repository, the “Diabetic Retinopathy Debrecen Data Set” (https://archive.ics.uci.edu/ml/datasets/Diabetic+Retinopathy+Debrecen+Data+Set). The dataset contains features extracted from the Messidor image set to predict whether an image contains signs of diabetic retinopathy or not. All features represent either a detected lesion, a descriptive feature of an anatomical part or an image-level descriptor. In the dataset there are totally 1151 data samples, each contains 19 attributes. The following is the information of the attributes:

0) The binary result of quality assessment. 0 = bad quality 1 = sufficient quality.

1) The binary result of pre-screening, where 1 indicates severe retinal abnormality and 0 its lack.

2-7) The results of MA detection. Each feature value stand for the number of MAs found at the confidence levels alpha = 0.5, . . . , 1, respectively.

8-15) contain the same information as 2-7) for exudates. However, as exudates are represented by a set of points rather than the number of pixels constructing the lesions, these features are normalized by dividing the number of lesions with the diameter of the ROI to compensate different image sizes.

16) The Euclidean distance of the center of the macula and the center of the optic disc to provide important information regarding the patient’s condition. This feature is also normalized with the diameter of the ROI.

17) The diameter of the optic disc.

18) The binary result of the AM/FM-based classification.

19) Class label. 1 = contains signs of DR (Accumulative label for the Messidor classes 1, 2, 3), 0 = no signs of DR.

Experiment Conducted:

Abbreviations:

LDA: Linear Discriminant Analysis

GNB: Gaussian Naïve Bayes

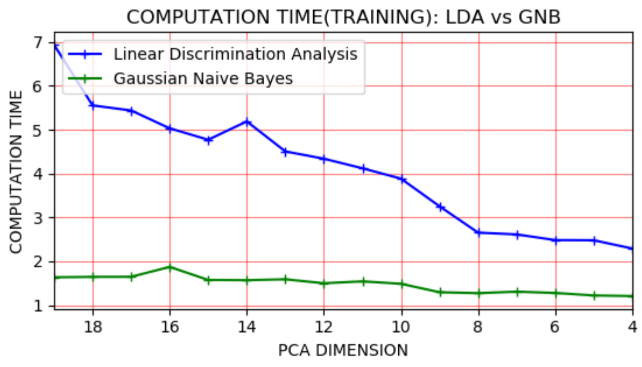
PCA: Principle Component Analysis

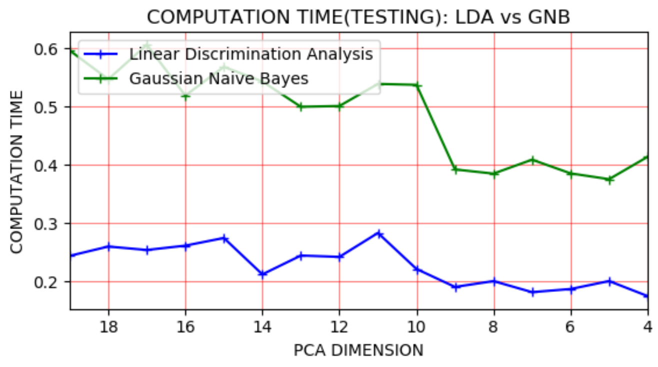
BWS: Feature selection using Backward Search

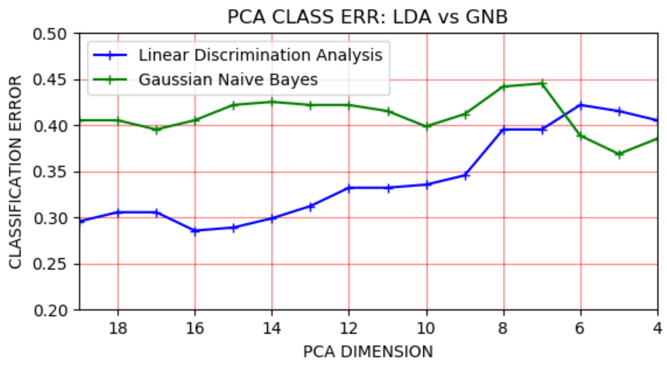
The following are the steps done in the python program:

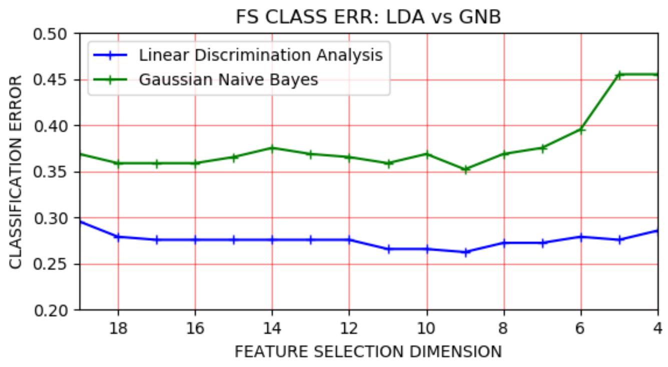
- The data were first normalized to the range of (0, 1), and then separated to training set (850 samples) and testing set (301 samples). The last attribute is used as classification result.
- As the training and testing time is too short to measure, I used a for-loop to magnify the time for 1000 times, and measure the duration.
- For full dimension data, the following are calculated:
    - PCA Eigenvalue decomposition
    - Transformed training data after PCA
    - Calculate the training and testing time for LDA (both magnified for 1000 times)
    - Calculate PCA-LDA classification error and confusion matrix
    - Calculate BSW-LDA classification error and confusion matrix
    - Calculate the training and testing time for GNB (both magnified for 1000 times)
    - Calculate PCA-GNB classification error and confusion matrix
    - Calculate BSW-GNB classification error and confusion matrix
- Reduce the dimension of PCA from 19 to 4 by removing eigenvector entries with the smallest eigenvalue one by one, calculate the following items in each loop:
    - Calculate the training and testing time for LDA (both magnified for 1000 times)
    - Calculate PCA-LDA classification error and confusion matrix
    - Calculate the training and testing time for GNB (both magnified for 1000 times)
    - Calculate PCA-GNB classification error and confusion matrix
- Plot the graph of “Computation time(Training): LDA vs GNB”
- Plot the graph of “Computation time(Testing): LDA vs GNB”
- Plot the graph of “PCA Classification Error: LDA vs GNB”
- Reduce the dimension of training data from 19 to 4, by removing the attribute contribute less to the classification task each time. One loop was used for LDA and another for GNB. In each loop, the following is calculated:
    - By removing the attribute one by one, calculate the corresponding classification error of the corresponding remaining vectors
    - Find the smallest classification error and remove that corresponding attribute
    - Store the smallest classification error
- Plot the graph of “Feature Selection (Backward Search) Classification Error: LDA vs GNB”
-

Plots of the Python Code









Confusion Matrix:

Confusion matrix: PCA - LDA (at dimension 16)		
	No signs of DR	Contains signs of DR
Classified as DR	6.98%	32.56%
Classified as no DR	38.87%	21.60%

Confusion matrix: PCA - GNB (at dimension 5)		
	No signs of DR	Contains signs of DR
Classified as DR	9.30%	26.58%
Classified as no DR	36.55%	27.58%

Confusion matrix: BSW - LDA (at dimension 9)		
	No signs of DR	Contains signs of DR
Classified as DR	5.65%	33.56%
Classified as no DR	40.2%	20.6%

Confusion matrix: BWS - GNB (at dimension 9)		
	No signs of DR	Contains signs of DR
Classified as DR	11.30%	30.23%
Classified as no DR	34.55%	23.92%

Discussion:

<u>Feature Extraction for Classification task: (Principle Component Analysis(PCA) vs Feature Selection (Backward Search) (BWS))</u>

Like what was mentioned in class, PCA does not really do a good job in feature selection for classification task. For PCA, in the case of LDA, most of the time, decrease in number of feature means increase in classification error. In some case, the increase can be quite significant (16->12, 9->8, 7->6). In the case of GNB, there are some significant increase (10->9,8) and decrease (7->6,5), but overall, we do not see any relation between what is considered the “principle components” of data and the classification result. Most of the times, getting rid of those components which is considered not useful can be harmful to the Classification task.

For Feature Selection, in the case of LDA, most of the time when the features considered insignificant get removed, the classification result is improved a little bit. But after the number of features decreases to 9, further reducing the number of feature starts to increase the classification error. In the case of GNB, the result fluctuates a bit, and it also reaches the lowest point when the number of features reaches 9, and after that it starts to increase significantly.

So we can conclude that for this dataset, in the case of LDA, feature selection has done a better job than PCA. In the case of GNB, feature selection is also slightly better than PCA in terms of the classification error.

Classification algorithms: (Linear Discrimination Analysis (LDA) vs Gaussian Naïve Bayes(GNB))

From the plot, we can see that for this dataset, LDA has done a much better job than GNB, either in the PCA case or in the Backward Search Feature selection case.

Computational time, Training and Testing

In the experiment, since the training and testing time of both algorithm (LDA and GNB) are very short, in order to get a more accurate time, I have tried to do the process for 1000 times, and plot the training time and testing time of both algorithms.

From the plot we can see that, In the case of training, for LDA, when the dimensions of the feature decreases, the computational time also decreases accordingly. For GNB, the computational time also decreases, but it is not as significantly as the LDA case.

In the case of testing, seems the computational time also decrease when feature dimension decreases for both LDA and GNB. GNB takes longer time than LDA.

The reason of decrease in computational time is the number of calculation decreases when the feature dimension decreases.

Confusion matrix

The lowest classification error we got in the experiment with the dataset is using LDA + backward search feature selection at dimension 9. From the confusion matrix, both TP, TN reach highest point (33.56%, 40.2%), and FP, FN reach lowest point (5.65%, 20.6%).

Confusion matrix: BSW - LDA (at dimension 9)		
	No signs of DR	Contains signs of DR
Classified as DR	5.65%	33.56%
Classified as no DR	40.2%	20.6%

One problem with the classifier on the dataset is that the FN percentage is quite high, meaning that a high percentage of patients containing signs of DR may be classified as no DR using the classifier.