INDIAN INSTITUTE OF TECHNOLOGY GOA

CS331

[Machine Learning]

Lab Assignment-2

Submitted By Group :-

OmNiSiYaKrMa

I. Illustrative exercises

1) **(K-means) Download the old faithful dataset. It is instructive to understand the origin of this dataset. Implement K-means (you cannot use the inbuilt library) and recreate plots similar to Figure 9.1 and Figure 9.2 from Bishop's book.**

   The old faithful dataset was taken from source (Old Faithful | Kaggle)
   By observing the plots of data points, one can clearly see that there will be two clusters. So, the K-means algorithm was implemented with k = 2 clusters. The corresponding plots have been drawn at certain iterations in which 2 clusters are shown with different colors along with their current centroids and dividing line(perpendicular bisector of centroids). The loss function is also plotted along with iterations.

2) **(K-means for Image segmentation / Image compression) Pick a suitable image and apply the clustering technique of the previous step to generate figures similar to Figure 9.3 from Bishop's book. Read about Image segmentation and Image compression problems.**

   The *"kids image"* was taken for this.
   Pixels of image were our data points - 3 - dimensional vectors with #RGB as its component values.
   The data was scaled to 0 to 1 and the k-means algorithm was implemented to cluster points into clusters of 2, 5 and 10 and depending upon the centroid of clusters compressed images have been shown.

3) **(Soft-assignment intuition) The objective of this exercise is to understand the multivariate gaussian distribution.**

   a) **Refer to figure 2.23 in Bishop and generate the plots similar to that. As a self study, explore how covariance matrix changes the contours.**

      Three different multivariate(2-D) gaussians are taken with positive correlations, zero and negative correlations respectively among variables.
      Corresponding independent contours are plotted in the first graph.
      In second, mixture of these 3 multivariate gaussians is plotted as a contour plot.
      And the surface plot of mixture of gaussians is plotted in third.

   b) **Generate 500 points from previous step clearing indicating the × responsible cluster. Thereafter, generate figure 9.5 from Bishop. 4. (GMM EM)**

      500 points were taken from the distribution using 2 ways:

      1) Using the joint distributions(probability of selection of each gaussian×point_probability given that gaussian). These are colored according to their gaussians.
      2) Using marginal distribution(using mixture of gaussian as pdf of x). There are 2 plots of this - one showing points and in another color intensity indicating the selected point's probability of being selected)

4) **Generate Figure 9.8 using the Old Faithful.**

   Dataset from part 1 was again classified into two clusters using a mixture of Gaussians approaches. And corresponding contour circles were plotted for a few iterations.

5) **(Unsupervised labeling) This exercise concerns the classdemo.py file shared with you. We saw that the EM algorithm learnt the hidden parameters fairly well. This question asks you to classify every point to each coin. Record the error rate of this "classifier".**

   On running the classifier, we get the mean error as 0.4388 %.

II. Exploratory exercises

1) **(Auto labeling with EM) In practice, availability of labeled datasets is difficult. One approach is to cluster the dataset suitably and then retrospectively assign labels to each cluster. Using the approach described in Figure 9.10 of Bishop, cluster the MNIST dataset into 10 clusters using a mixture of Bernoulli distributions and then examine the average cluster. Thereafter, each cluster is labeled based on the average cluster. Record, how many points were misclassified based on this approach.**

   The classifier was implemented on a new MNIST dataset(handwritten digits for 2,3,4) using the Mixture *of Bernoulli RVs approach* and we obtained accuracy and error. We implemented it using a matrix-based algorithm because it was efficient and able to return smaller numbers.

   The clusters were examined on the basis of what true labels the points in that cluster have and then labeled using the most occurring label in the given cluster.
   The accuracy of the classifier was 88.90 %.

2) **See this lecture (upto slide 14) https://developers.google.com/machine-learning/crashcourse/classification/video-lecture. Design a bayes classifier (optimal) when p0 = 0.95. Also class conditional densities for class 0 are normal mean 0 and variance 1. For class 1, class conditional density is mean 1 and variance 1.**
    a) **FP for bayes classifier.**
    b) **TP for bayes classifier.**
    c) **Accuracy for bayes classifier.**

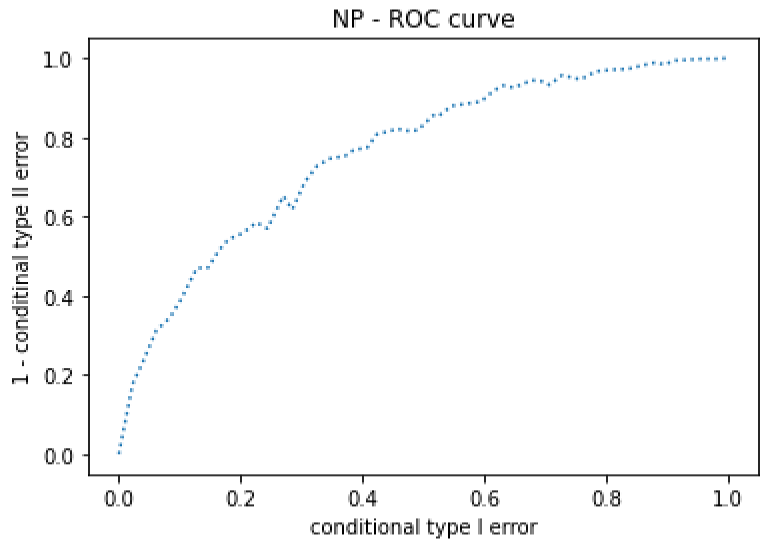
The Data was generated from given Distribution and then Bayes Classifier was implemented on it:
After that we get, in 10^4 samples,(C1 is positive class)
FP = 495
TP = 4
Accuracy = 0.95
This is quite expected because as you can see the priors are very skewed for the two classes and so, there is a high chance that points from C1(prior 0.05) would be misclassified in C0(prior 0.95).
As Bayes is a minimal risk classifier in order to minimize risk in such a skewed prior case, it will most likely classify each point in class C0.
So, the accuracy will be increased (which is 0.95 here, almost close to prior of C0).

d) **For the same setup implement Neyman Pearson Classifier.**

Neyman Pearson Classifier is a classifier which is used to bound one of the errors(mostly type 1 error) and simultaneously, minimize another error.
This was implemented for data generated from above distribution for bounding error corresponding to class c0 and we found that lowering the bound was increasing the type 2 error.

e) **Generate the ROC curve for various thresholds for the family of classifiers which classify based on likelihood ratio and a threshold as done in Neyman Pearson classifier.**

The corresponding ROC curve was plotted by setting various $\alpha's$ (bounds on type 1 error) and calculating conditional errors each time.



f) **Implement max-min classifier.**

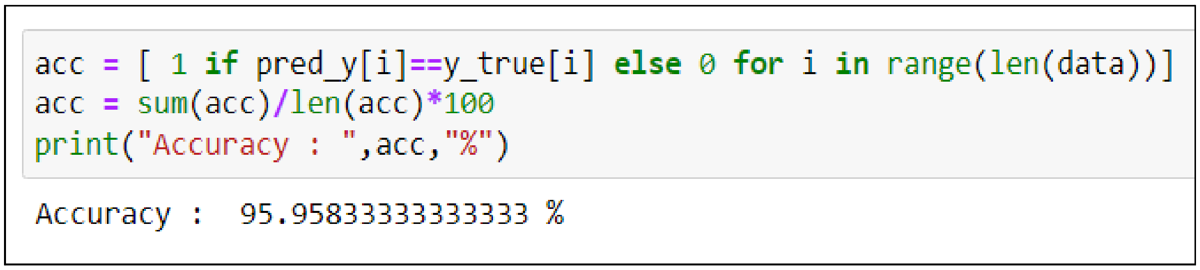
Minmax classifier is a classifier in which the risk of the classifier is made independent of priors, that's why it is used in cases where priors are unknown.

The corresponding Tau(threshold) was approximated using graphical plotting and np.cdf.

III. Optional exercises for extra credit

1) Implement Bayes Classifier for the 'MNIST Dataset' by modeling class conditional densities as

a) Multivariate Gaussian distribution.



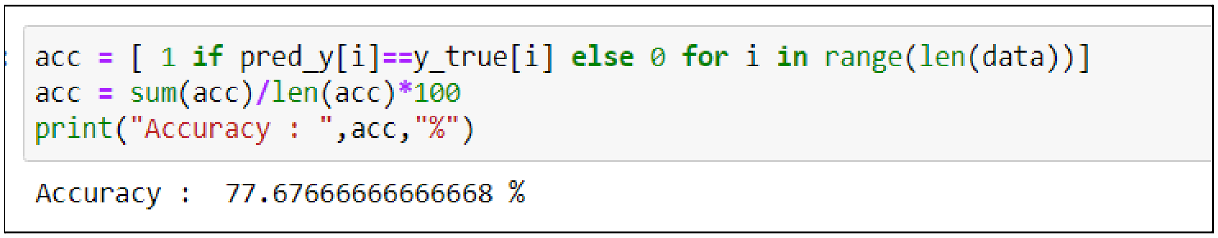
```
acc = [ 1 if pred_y[i]==y_true[i] else 0 for i in range(len(data))]
acc = sum(acc)/len(acc)*100
print("Accuracy : ",acc,"%")
```

```
Accuracy :  95.95833333333333 %
```

Assuming that all the 784 pixels in each picture of MNIST dataset are following Multivariate Gaussian distribution as class conditionals, we applied the Multivariate Gaussian model and estimated parameters as MLEs.
To avoid the corresponding Covariance matrices from being singular, we regularize them by adding $\sigma^2 I$ in each of them.

Then we predicted labels for MNIST data and checked for accuracy which we got as 95.958 %.

b) Multivariate Exponential distribution.



```
acc = [ 1 if pred_y[i]==y_true[i] else 0 for i in range(len(data))]
acc = sum(acc)/len(acc)*100
print("Accuracy : ",acc,"%")
```

```
Accuracy :  77.67666666666668 %
```

Assuming that all the 784 pixels in each picture of MNIST dataset are independent and are following exponential distributions as class conditionals, we applied the Multivariate Exponential model and estimated parameters as MLEs.

To avoid the corresponding mean from being 0(because, we want to take the inverse of it), we modify the dataset by adding 1 in each of the pixels.

Then we predicted labels for MNIST data and checked for accuracy which we got as 77.677 %.