Design, Implementation and Analysis of a Bayes Classifier Using Discriminant Functions

Haroon Khan*
Department of Mechatronics Engineering, Air University, Islamabad, Pakistan
Email: *haroon.khan@mail.au.edu.pk

Abstract—The Naive Bayes Classifier technique is based on the so-called Bayesian theorem and is particularly suited when the dimensionality of the inputs is high. Despite its simplicity, Naive Bayes can often outperform more sophisticated classification methods. Navies Bayes classifier is designed in this assignment using concept of Gaussian distribution function. Different cases has been stuided in this assignment where variance and co-variance with in and between the class exists. MATLAB code for Naives Bayes classifier is designed for data of two class with 2 features. Effect of standerd deviation is also studied with in this assignment.

Keywords—Naive Bayes Classifier, Bayes Discriminator, Gaussain Distribution funtion.

I. Design of Naive Bayes Classifier

MATLAB code of Naive Bayes classifier is described in this section. The matlab code is divied into 23 different steps. Code is also divided into three cases. Each case study the relation of variance and co-variance of data with in and between the classes [1]. In Step 1 number of class, feature, samples and means of each class and feature are defined. For study of case I,variance for both classes and feature is same. No co-variance exist with in and between the classes. The step 2 and 3 random data is generated with predefined mean and variance as mentioned in step 1 for both the classes of 200 samples and two features. Since data is generated randomly for reproducibility of same data set "rng default" command is used. The data generated is then stored in 'class1' and 'class2' variables. In Step 4 histogram of both the classes are plotted on same plot using hist3 $(X,'Nbins',nbins)$ command. Hist3 generated histogram of bivariate data where 'X' is input class and 'nbins' are the number of bins. Gaussian distribution are ploted for both the data using Multivariate normal probability density function y = mvnpdf $(X,MU,SIGMA)$. 'MU' and 'SIGMA' are corresponding class means and variance matrix. For three dimensional grid 'meshgrid' and plot 'surf' command is used. In step 5 and 6 first 1:160 colunms of randomly generated data in step 2 of each class is stored as training data of each class and labeled as 'class1Tr' and 'class2Tr' Similarly, remaining 40 columns i-e from 161:200 for each class is stored as testing data. In step 7 random noise is added into the both training data of both the class with $mean = 0$ and $std = 0.5$ as shown.

```
class1Tr_noise = class1Tr + 0.5*randn(size(class1Tr)) + 0
```

In step 8 'mean' for noisy data of both classes are calculated using 'Mean' commond and save in labels as 'mui' and 'muj'. Since in first case we consider that the features are statistically independent, and when each feature has the same variance . In this case the co-variance matrix is diagonal being merely ^{2} times the identity matrix I. Mean of diagonal entries of co-variance matrix is taken is $sigma_n$ for our first case as shown.

```
sigma_case1 = mean(diag(sigma_n))
```

The decision boundary of the classifier lies where the probability of Gaussian distribution function equals to each other. To implements this equation (54) [1] is implemented. $W, X_0 and X$ can be calculated using following commands.

```
W=(mui-muj)
X_0 =
    .5*(mui+muj)-(sigma_case1^2/norm(W))*log(pwi/pwj)*W
X=pinv(W_t).*W_t*X_0
```

where 'pinv' = pseudo inverse, 'norm' = Euclidean norm. X_0 give us the point on the decisions boundary which is orthogonal to means of the class and for case where aprior probability of both the classes are same it lies at the center on the line joining the means of the two classes. $X = [xy]$ give us any point that lies on the decision boundary by finding the two point we can plot a line. For give range of x we get y and then the value are plotted as shown in following code.

```
x = linspace(-1,15,100)
y
    =((X_0(2,1)-X(2,1))/(X_0(1,1)-X(1,1)))*(x-X(1,1))+X(2,1)
plot(x,y)
```

Scatter plot is used to plot all the noisy data points of both classes. The same code is repeated to draw the decision boundary of case-I by changing the standard deviation to 1 and 1.5. In Step 12 new co-variance matrix is defined. In this case the co-variance matrices of both the class are same. Same procedure as discussed is followed for case-II also. In the case since co-variance exists the equation of calculating X and and X_0 differs as shown

```
X_0 = .5*(mui+muj)-
    (log(pwi/pwj).*(mui-muj).*pinv((mui-muj)'.*W.*(mui-muj)));
X=pinv(W_t).*W_t*X_0
```

In Step 18 for case III different co-variance matrices are defined. Equation 64 [1] is equated for both the classes to find the decision boundary. Difference in weight matrix $W[i]$,wi and $w[io]$ is labeled as An, Bn and Cn as shown

```
An=Ai-Aj;
Bn=Bi-Bj;
Cn=Ci-Cj;
```

The value of this matix are found using equation 65,66 and 67 [1]. The equation of decision boundry is solved manualy and label as 'eq'. as shown

```
eq=An(1,1)*x1^2+An(1,2)*x1*x2+An(2,1)*x1*x2+An(2,2)*x2^2+Bn(1,1)*x1+Bn(2,1)*x2+Cn
```

After then using 'solve()' matlab function the 'eq' is equated to zero and solve for x1 as shown

```
x1=solve(eq==0,x1)
```

'matlabFunction' is used to the converts the symbolic expression or function f to a MATLAB® function with handle to which it is equated. In our case its 'Fx11'. Since 'eq' was quadratic and it return to value of 'x1' so to plot this function 'fplot' is used. 'fplot' is used to plot matlab function as shown

```
Fx11=matlabFunction(x1(1));
Fx12=matlabFunction(x1(2));
fplot(Fx11)
title('Case III - Decision Boundry (STD=1)')
xlabel('Feature1'); ylabel('Feature 2');
hold on
fplot(Fx12)
```

The same procedure is repeated by changing the standard deviation to 1 and 1.5.

II. Analysis of Results

In figure 1 it can be seen that the bins of both the class have peaks at the mean value of the their feature. The same can be verified by figure 2 that the center of Gaussian distribution are at the center of the class mean. Both the plots can be easily verified visually that the data generated is actual of same actual mean which is expected.

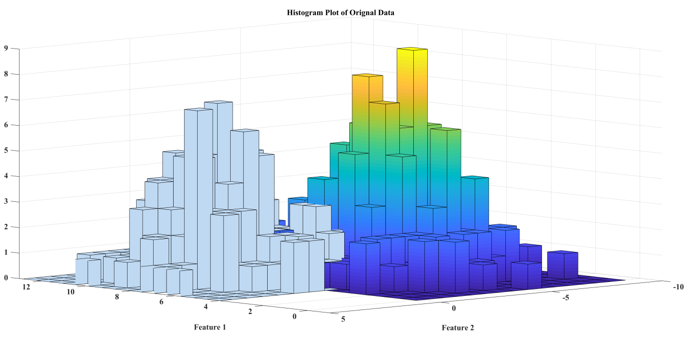

Fig. 1: Histogram Plot of Original Data

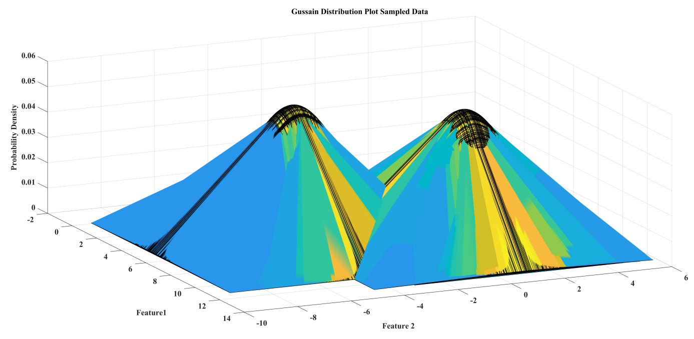

Fig. 2: Gaussian Distribution Plot of Original Data

Decision boundary has been plotted in different for different case as discussed in previous section. In Figure 3 shows the decision boundary for case-I between bi-variate data of two classes with standard deviation of 0.5 of each class. Both the classes are well separated expects few points. As we increase the spread (standard deviation) of the data from 0.5 to 1 and 1.5 the classifier become worse in classifying the data. The deterioration of the results can easily be seen in 4 and 5.

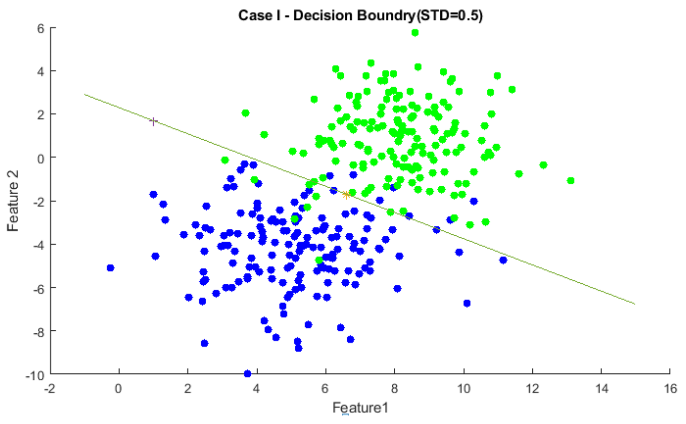

Fig. 3: Case I: Decision Boundary Plot with STD=0.5

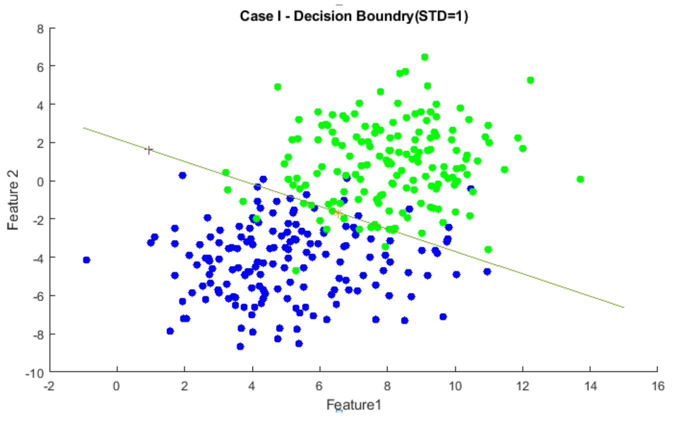

Fig. 4: Case I: Decision Boundary Plot with STD=1

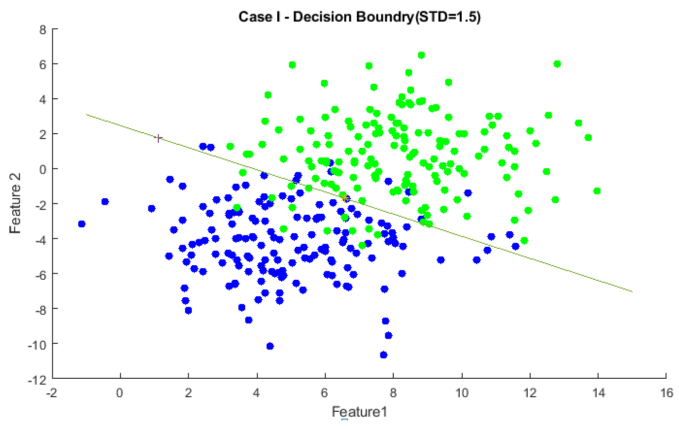

Fig. 5: Case I: Decision Boundary Plot with STD=1.5

The same pattern is repeated for case II also as we increase the value of standard deviation the classifier results deteriorates as shown in figure 6,7 and 8.

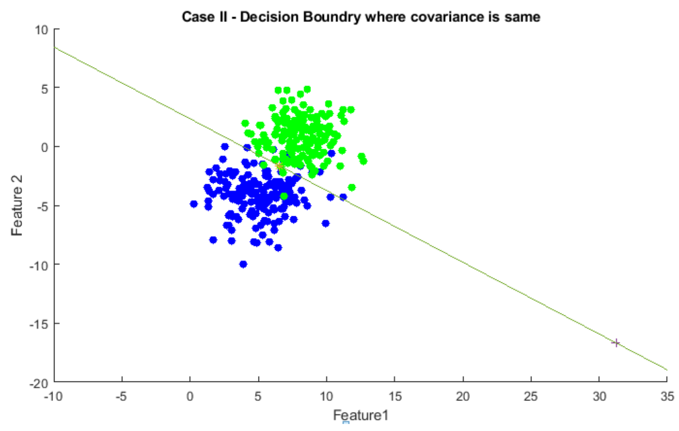

Fig. 6: Case II: Decision Boundary Plot with STD=0.5

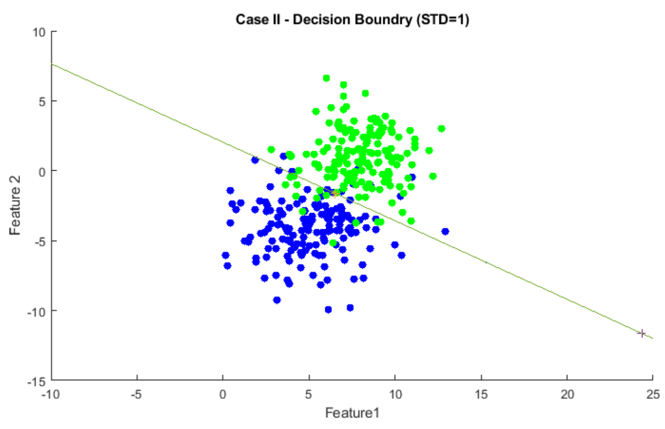

Fig. 7: Case II: Decision Boundary Plot with STD=1

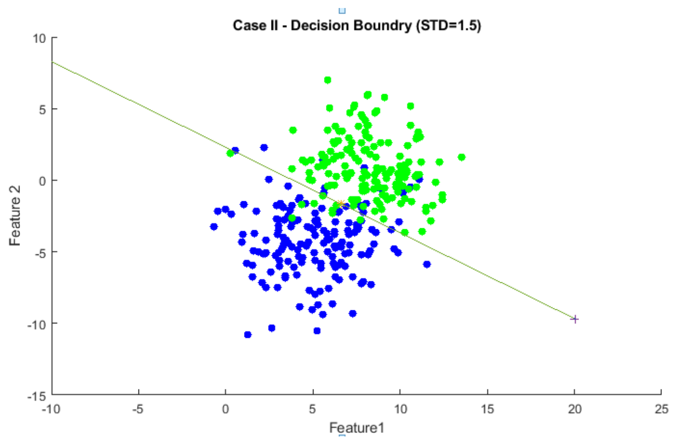

Fig. 8: Case II: Decision Boundary Plot with STD=1.5

For figure of case-III shown in figure 9,?? and ?? the decision boundary is no more linear one. In 9 we can see that where co-variance matrices for each class are different the classifier doesnot gave us good boundary. With the increase in standard deviation the results become more worse.

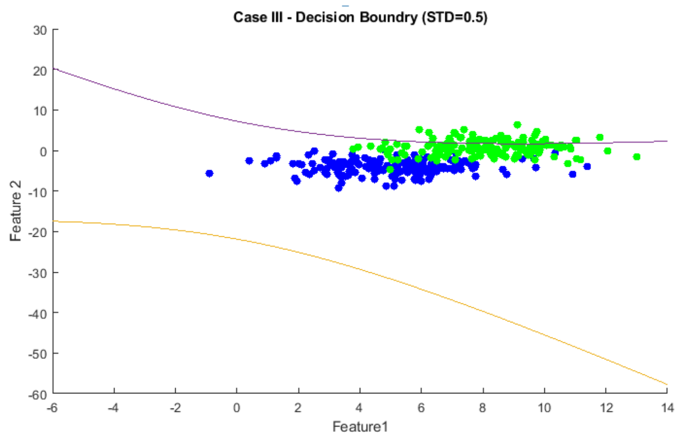

Fig. 9: Case III: Decision Boundary Plot with STD=0.5

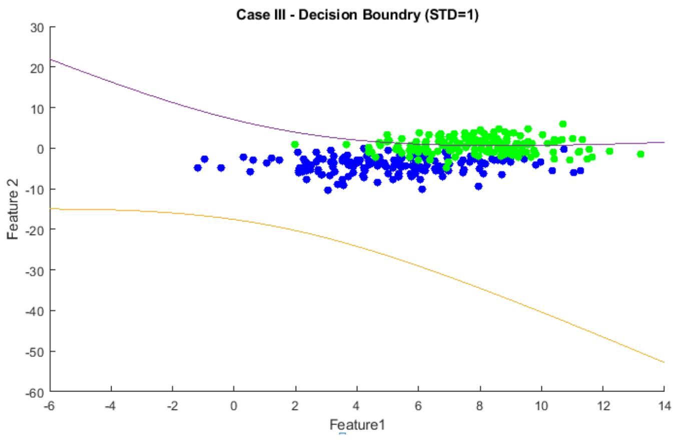

Fig. 10: Case III: Decision Boundary Plot with STD=1

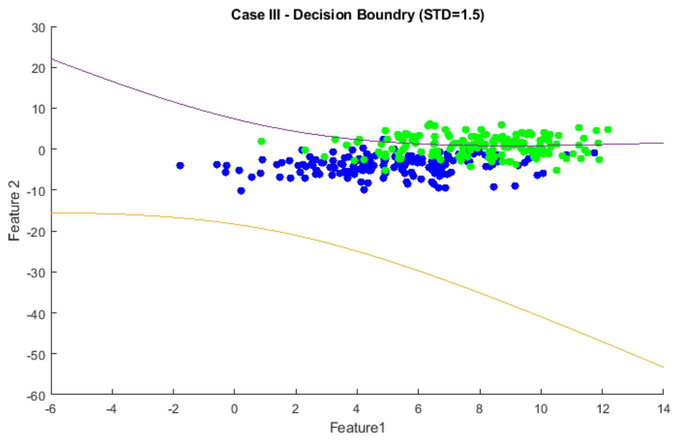

Fig. 11: Case III: Decision Boundary Plot with STD=1.5

III. Accuracy of the classifier

Accuracy of the can be found by putting all the test data in to classifier and then calculate posterior for each point in the data set if the posterior. The test point will be assigned the class for which the value of posterior is greater. Formula for posterior probability: $Posteriorprobability = LikelihoodxPriorProbability$

IV. Conclusion

Naive Bayes classifier is a straightforward and powerful algorithm for the classification task. Even if we are working on a data set with millions of records with some attributes.The classifier studied her for two bivariate two class problem. Its easily classify the data. The classifier perform good for separating data having same co-variance matrices. Its is studied that increasing standard deviation of class data increase its spread and difficult to discriminate between the data hence overall accuracy decreases. It doesn't accurately classify classes having different co-variance matrices.

References

[1] R. O. Duda, P. E. Hart, and D. G. Stork, Pattern classification. John Wiley & Sons, 2012.