STOCHASTIC MODEL: NAÏVE BAYES CLASSIFIER

SANYAT HOQUE

UNIVERSITY OF MANITOBA

TOPICS *OF* DISCUSSION

Table of Contents

Lists of Figures

I. INTRODUCTION TO MODEL BASED CLASSIFICATION

Most Classification techniques fall under 2 categories:

- Generative Models
- Discriminative Model

A. Generative Model:

Generative Model is a model that learns from joint distribution probability of a set of random observable data variables $p(x, y)$, where x and y are observations and labels. Main advantage of Generative model is that it can either learn directly from a set of observed variables $p(x|y)$, where x is observed data and y is its labeled class, or from Joint distribution $p(x, y)$. From Joint distributions $p(x, y)$, it can be transformed to maximum likelihood estimate or conditional distribution $p(y|x)$ by means of Bayes Rule that states: $p(x, y) = p(x|y)p(y)$; $where, p(y)$ is belief/prior probability, $p(x|y)$ is conditional probability, and $\{p(x|y) * p(y)\}$ is the Bayes rule.

Advantage of a Generative Model is that for every classification of maximum a posterior, there is also a misclassification error, where probability of error for any classification (or action). This is in contrast to a deterministic classifier where there is absence of error uncertainty and gives a binary outcome of either 0 or 1.

i. Examples of Generative Algorithm:

Some examples of generative model includes Gaussian mixture model, Hidden Markov model, Naive Bayes, Restricted Boltzmann machine, Generative adversarial networks. Generative model is a probabilistic model can be used to generate probability distribution values of any variable in the model via probabilistic inference, whereas a discriminative model can sample observed pre labeled variables.

ii. Probabilistic model

Generative model can also be referred as a probabilistic model that gives a distribution of possible outcomes (i.e. it describes all outcomes and gives some measure of how likely or unlikely each set of events are to occur), eg. via $Joint\ Probabilty,\ p(x, y) = \{p(x|y) * p(y)\}$.

B. Discriminative Model:

A Discriminative Model learns directly from conditional distribution $p(y|x)$, ie. probability of classifying class y given feature variable x which is labeled directly to the class.

i. Examples of Discriminative Algorithm:

Some examples of discriminative modeled algorithm are Logistic Regression, Neural Network, Support Vector Machine. These models class y directly to feature x. In order to find any optimal boundary, gradient descent technique is used via back propagation to find the optimal convex solution or decision boundary in $Conditional\ Probability,$

$$p(y|x) = \frac{\{p(x|y) * p(y)\}}{p((x)},\ where\ p(x) is\ Marginal\ Distribution\ p(x) = \sum\{y * p(x, y)\} \quad (1.1)$$

ii. Deterministic model

Discriminative model gives rise to deterministic model which yields a single solution that describes outcome of events given appropriate inputs, eg. Support Vector Machines and Logistic regression provides a global minima/unique answer since those are convex models (consists of a global minima).

iii. Joint Distributions

Joint Distribution over a set of random variables : T,W specifies a probability for each set of Events.

$$For\ Example, P(T,W)\ T: Temperature\ W: Weather$$

Table 1 Joint Distribution of T and W

T	W	Pr
h1	W1	0.4
h1	W2	0.1
c2	W1	0.2
c2	W2	0.3

iv. Marginal Probability

Marginal Distributions are subtables which eliminate other variables. Marginalization also means summing out of other variable while keeping one variable. In other words, combine collapsed rows by adding them together.

$$For\ Example, P(T,W)\ T: Temperature\ W: Weather$$
$$P(T) = \sum \{W * p(T,W)\}$$

Table 2 Marginal Distribution of random variable T

T	W	Pr		Pr
h1	W1	0.4	h1	0.5
h1	W2	0.1	h2	0.5
c2	W1	0.2		
c2	W2	0.3		

II. Probabilistic Inference: Conditional Probability:

This is the process of computing a desired probability distribution from other known or observed probability distribution, eg. joint probability distribution to conditional probability distribution.

Simple relationship between Joint Probability Distribution and Marginal Probability Distribution is called Conditional probability. This inference of probability distributions is called Probabilistic Inference.

$$P(x|y) : inference\ of\ variable\ x\ based\ on\ observed\ event\ y;\ \ p(x,y) = \{p(x|y) * p(y)\};$$
$$Or,\ \ p(x|y) = \frac{p(x,y)}{p(y)} \qquad (1.2)$$

- *Bayes Decision Rule*

Relating Conditional Probability Distribution, $p(x|y)$, Joint Probability Distribution, $p(x,y)$ and Marginal Probability Distribution, $p(x)$;

$$we\ get, p(y|x) = \frac{p(x|y) * p(y)}{p(x)}, where\ p(y|x), y\ is\ querry\ variable; \qquad (1.3)$$
$$p(x) is\ marginalization\ of\ y\ in\ Joint\ probability\ in\ p(x,y);$$

III. Bayes Net: Conditional Independence

Bayes Net represents a graphical model which is an acyclic graph that enables to express conditional Independence.

In the Figure, Bayes' Net or the acyclic graph, there are no arrows between Node U and Node T. This states that Node U and Node T does not influence each other. So, Node U and T are independent of each other given Event R (Node R) already occurred.

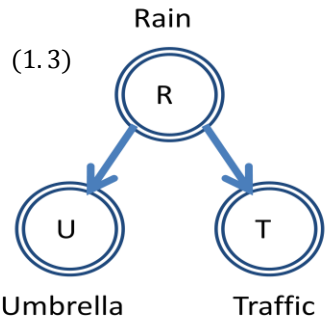

Fig. 1 Graphical Model of Conditional Independence

$Node\ R\ influences\ Node\ U\ and\ Node\ T,$ ie. (U ⊥ T) |R, $where$ ⊥ means orthogonal

From the above graphical model, it can be seen that Conditional independence in a probability distribution helps to simplify Bayes Model. In other words, if $x1, x2, x3$ are not conditionally independent,

$P(x1, x2, x3) = p(x1)p(x2|x1)p(x3|x1, x2)$, *makes a very lengthy joint probability distribution Table.*

A. Conditional Independence: Chain Rule

Two variables x, y are independent to each other $iff\ for\ all\ values\ of\ x, y : p(x, y) = p(x) * p(y)$;
$where\ p(x), p(y) are\ both\ Marginal\ Distribution\ of\ the\ Joint\ Probability\ Distributions\ p(x, y)$

- *Proof: Chain Rule for Conditional Independence*

$For\ all\ values\ of\ x, y: p(x, y) = p(x|y) * p(y)$

$$Or, p(x|y) = \frac{\{p(x, y)\}}{p(y)} \quad (1.4)$$

$$Or, p(x|y) = \frac{\{p(x) * p(y)\}}{p(y)}\ ;\ if\ \{x, y\} are\ independent \quad (1.5)$$

$Therefore, p(x|y) = p(x)$, $where\ p(x) is\ Marginal\ distribution\ and$
$p(x, y)\ is\ Joint\ probability\ distribution$

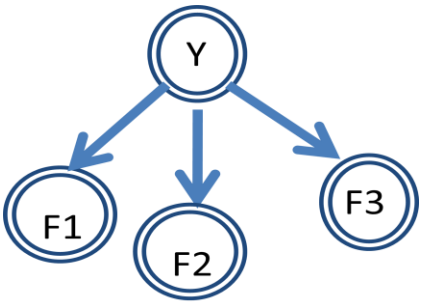

Fig. 2 Graphical Model of Bayes' Net

i. Example: Chain Rule via Conditional Independence

$Let\ Events\ be$: $Rain\ R$; $Umbrella\ U$; $Traffic\ T$;
Without assumption of Conditional Independence between these Events,
$P(R, T, U) = P(R) * P(T|R) * P(U|R, T)$; (1.6)

With assumption of Conditional Independence between these Events, $P(T \perp U \mid R)$, states $Event\ R$ makes $Event\ T$ and $Event\ U$ are conditionally independent

$Therefore, P(R, T, U) = P(R) * P(T|R) * P(U|R)$; (1.7)
$where\ P(R) * P(T|R) * P(U|R)\ simplifies\ model$;

In other words, *Condition of Independence* prevents use of 3 or more Joint Distribution Tables.

$If\ 3\ Features\ F1, F2\ and\ F3\ are\ conditionally\ independent\ to\ each\ other,$
$P(Y1|F1, F2, F3)$

$$= \frac{P(Y1) * P(F1|Y1) * P(F2|Y1) * P(F3|Y1)}{\{P(Y1) * P(F1|Y1) * P(F2|Y1) * P(F3|Y1) + p(Y2)\} + \{P(Y2) * P(F1|Y2) * P(F2|Y2) * P(F3|Y2)} \quad (1.8)$$

ii. Example of Computing Bayes Decision via Chain Rule

$Let\ No.of\ classes = \boldsymbol{\omega 1}, \boldsymbol{\omega 2}, \boldsymbol{\omega 3}$;
$Prior\ (Belief)\ Probability = p(\boldsymbol{\omega 1}), p(\boldsymbol{\omega 2}), p(\boldsymbol{\omega 3})$;
$Calculate\ Maximum\ Likelihood\ Estimate\ via\ Parameter\ Est. = p(\boldsymbol{\omega 1}|x), p(\boldsymbol{\omega 2}|x), p(\boldsymbol{\omega 3}|x)$;
$where\ x\ is\ obervation\ variable\ features\ that\ are\ pre\ labeled\ as\ class\ \boldsymbol{\omega 1}, \boldsymbol{\omega 2}, or\ \boldsymbol{\omega 3}$

$Taking\ query\ variable\ p(\boldsymbol{\omega 1}|x), p(\boldsymbol{\omega 2}|x), p(\boldsymbol{\omega 3}|x)\ to\ obtain\ which\ class\ an\ observation\ variable$
$p(x|\boldsymbol{\omega 1}), p(x|\boldsymbol{\omega 3})\ or\ p(x|\boldsymbol{\omega 2})\ belongs\ to$:
After calculating p(**ω**1|x) and p(**ω2**|x)
if p(**ω**1|x) > p(**ω2**|x) and p(**ω**3|x) then decision is in favor of **ω1** ;
Or, if p(**ω**2|x) > p(**ω1**|x) and p(**ω**3|x) then decision is in favor of **ω2** ;
Or, if p(**ω**3|x) > p(**ω1**|x) and p(**ω**2|x) then decision is in favor of **ω3** ;
In Naïve Bayes operator (kernel) operator, probability with covariance matrix is computed for non parametric density estimators having no fixed structures unlike $Gaussian\ density\ distribution, N(\mu, \sigma^2)$.
See Appendix A1.1 to see the implementation of Naïve Bayes Classifier in Matlab and plot its 3-d surface.

Algorithm 1.1

Algorithm 2.1 Test Observation Vector in Naïve Bayes Algorithm and Plot its 3-D Surface for Each Class

1. Construct Naïve Bayes Algorithm
   (a) Load `data_bayes3` in M-file.
   (b) Plot labels, markers, color data points by finding position of Meshgrid for each classes in 2 column vectors of X_1 and X_2 faetures consisting of 90 samples each for each.
   (c) Find Position of data points in mesh grid and label each class (eg. 1,2 or 3) with 1 and others with 0 and color them with 3 different colors and plot data.
   (d) Calculate Mean for each Classes.
   (e) Calculate Covariance Matrix for each classes.
   (f) Use Bayes Assumption, convert Covariance Elements into Null.
   (g) Compute Maximum Likelihood Estimate for all 3 Classes.
   (h) Use reshape Matlab function to convert an array of 1D to 2D meshgrid, c is no. of classes
   (i) Use reshape Matlab function to make 3 separate arrays to construct contours over 2 feature meshgrid

2. Test Observation Vector for Classification

For equal Prior Belief Probability, for ex,
   (a) Labelled Data for Class 1 = 30; Class 2 = 30; Class 3 = 30.
   (b) So, Belief Probability for each Class (Uniform Priors) = 30/90; (Since,Total Data for each Feature Xn = 90) = 1/3 ;

For equal Priors, Maximum Aposterior gives same values as Maximum Likelihood Estimate
   Therefore, aposterior = [3*1]; *row of 3 values for 3 classes*
   From 3 aposterior values, only *Index* of Maximum a Posterior is taken.

3. Plot 3-D Surface for each Classes
   (a) Use Function to calculate *pdf* for each points in Meshgrid,
   probability_density_function = density_norm([X1(:),X2(:)], class_mean(c,:), feature_covar(:,:,c));
   (b) Use Reshape Function to converts an array of 1D to 2D/3D meshgrid, c is no. of classes
   (c) Use reshape to make 3 separate arrays to construct contours over 2 feature mesh grid

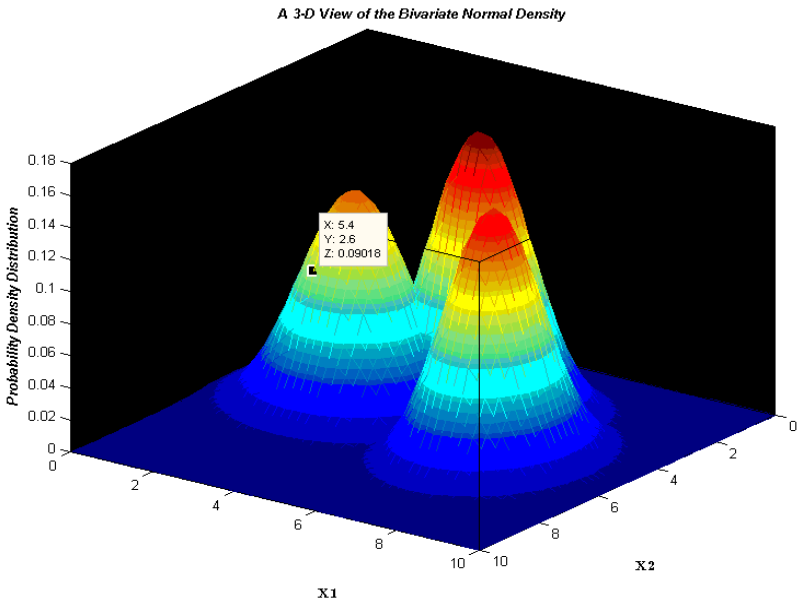

Fig. 3 Probability Density Distribution for 3 Classes

II. PARAMETER ESTIMATION

A. Covariance

Variance or Covariance is the second moment of the probability density of a feature vector distributed about its mean position in a feature space. Therefore, second moment about the mean and *second central moment or variance is defined as*:

$$Covariance, \quad \sigma^2 = E\left[(x-\mu)^2\right] = \int_{-\infty}^{+\infty} (x-\mu)^2 p(x)\, dx \tag{2.1}$$

The Covariance Matrix for $X_1, X_2 \ldots X_N$ features for discrete feature space variable vectors is defined as:

$$\sum_i = \begin{bmatrix} \sigma_{11}^2 & & & & & & \phi \\ & \sigma_{22}^2 & & & & & \\ & & \sigma_{33}^2 & & & & \\ & & & \sigma_{44}^2 & & & \\ & & & & \sigma_{55}^2 & & \\ \phi & & & & & \sigma_{66}^2 & \\ & & & & & & \ddots \\ & & & & & & \sigma_{nn}^2 \end{bmatrix}$$

where $\sum_i$ is covariance matrix for each class *i* and non diagonal elements are ϕ or null for independent feature variables in light of Bayes' assumption.

- $Covariance\ for\ each\ variable\ X_N is\ defined\ as, \sigma_{ij} = E\left[(x_{Ni}-\mu_N)\left(x_{Nj}-\mu_N\right)^t\right]$; (2.2.1)

where x_{Ni}, x_{Nj} *are* i^{th} *and* j^{th} *component of feature variables* X_N

- $For\ diagonal\ components\ i = j, \sigma_{ii} = E\left[(x_i-\mu_i)^2\right] = \sigma_i^2$; (2.2.2)

where σ_i^2 *describes variance of individual features variables* X_N

- $For\ Non-diagonal\ components\ where\ i \neq j,$

σ_{ij} *describes covariance between individual features between* $X_1, X_2 \ldots . X_N$;

- $For\ continuous\ variables\ X_N,\ covariance, \sigma^2 = \int_{-\infty}^{+\infty} (x-\mu)(x-\mu)^t p(x)\, dx$ (2.2.3)

i. Probability density function

Probability density function of an observation feature vector is defined as:

$$p(x) = \frac{1}{\sqrt{(2\pi)^d |\sum|}} \exp\left[-\frac{1}{2}(x-\mu)^t \sum^{-1} (x-\mu_i)\right] ; \tag{2.2.4}$$

where x *is number of dimensions,* d *is* $d-dimensions$

- *Maximum Likelihood Estimate for Bivariate Case*

For bivariate independent feature vectors, $\sum = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12} \\ \sigma_{21} & \sigma_{22}^2 \end{bmatrix}$, $\sigma_{12}, \sigma_{21} = 0$ (2.3.1)

For bivariate maximum likelihood estimate, $where\ Features\ X = \begin{bmatrix} X_1 \\ X_2 \end{bmatrix}$ and its $class\ mean, \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$ (2.3.2)

$and\ covariance\ matrix, \sum = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12} \\ \sigma_{21} & \sigma_{22}^2 \end{bmatrix}$ (2.3.3)

$$Finding\ determinant, |\sum| = \begin{vmatrix} \sigma_{11}^2 & \sigma_{12} \\ \sigma_{21} & \sigma_{22}^2 \end{vmatrix} = E(X_1X_2) - E(X_1) * E(X_2) \tag{2.3.4}$$

$= \sigma_{11}^2 * \sigma_{22}^2 - \sigma_{12} * \sigma_{21}$; *if* X_1 *and* X_2 *are independant* $\sigma_{12}, \sigma_{21} = 0$

Probability density function for bivariate features, $p(x) = \frac{1}{\sqrt{(2\pi)^d|\sum|}}\exp[-\frac{1}{2}\{\frac{(x_{1i}-\mu_1)^2}{\sigma_1^2} + \frac{(x_{2i}-\mu_2)^2}{\sigma_2^2}\}]$ (2.3.5)

See Appendix A2.1 to see the implementation of expression of Maximum Likelihood Estimate for Bivariate features for all classes by calculating mean for each class and covariance for each class.

Algorithm 2.1

Implement Probability density function for Bivariate Features

1. Estimate Mean and Covariance matrix from the data
2. Fit class-conditional Gaussians for a particular class; (for c = 3 % index of class Data)
3. Finds position of each classes on Mesh grid
4. Finds position of Mesh grid for each classes from 2 columns of X_N out of 90 elements in each column.
   (a) Find Mean for each Class position of Mesh grid
   $class\ mean, \mu = \begin{bmatrix}\mu_1\\ \mu_2\end{bmatrix}$
   $covariance\ matrix, \quad \sum = \begin{bmatrix}\sigma_{11}^2 & \sigma_{12}\\ \sigma_{21} & \sigma_{22}^2\end{bmatrix}$
   (b) Find Covariance Matrix, (2x2), Maximum Likelihood Estimate for each class c position of Mesh grid
   Use Matlab function, "cov"
   feature_covar(1:2,1:2,c) = cov(X(position,1:2),1); { Normalized by N (not N-1) }

5. Use Function of Parameter Estimation Method to calculate Maximum Likelihood Est. with parameters mu and covariance

```
function probability_density = density_norm(X,mu,cov_mat);
[n,d] = size(X);
probability_density =
1/((2*pi)^(d/2)*sqrt(det(cov_mat)))*exp((-0.5)*diag((X-ones(n,1)*mu)*inv(cov_mat)*(X-
ones(n,1)*mu)'));   end%%
```

Probability density function for bivariate features, $p(x) = \frac{1}{\sqrt{(2\pi)^d|\sum\ |}}\exp[-\frac{1}{2}\{\frac{(x_{1i}-\mu_1)^2}{\sigma_1^2} + \frac{(x_{2i}-\mu_2)^2}{\sigma_2^2}\}]$

6. Use Matlab '*Reshape*' function
(a) Converts an array of 1D to 2D meshgrid c is no. of classes
(b) Makes 3 separate arrays to construct contours of 2 features mesh grid.

- *For Bayes' feature independent assumption,*

Since, $\sigma_{11}^2 * \sigma_{22}^2 - \sigma_{12} * \sigma_{21}$; $if\ X_1\ and\ X_2\ are\ independant\ \ \sigma_{12}, \sigma_{21} = 0$
See Appendix A2.2 to see the implementation of Matlab Codes to implement Bayes assumption.

Algorithm 2.2

Implement Naïve Bayes assumption

1. Compute Covariance Matrix for each Class,
2. Convert Non-diagonal Elements in Matrix, $\sigma_{12}, \sigma_{21} = 0$

$Determinant, |\sum| = \sigma_{11}^2 * \sigma_{22}^2$; and
$For\ independant\ assumption, f(x_1, x_2) = f(x_1) * f(x_2),$
$Maximum\ Likelihood\ Estimate\ for\ independent\ bivariate\ case,$

$$p(\omega_i|x) = \frac{1}{\sqrt{(2\pi)^d\sigma_{11}^2 * \sigma_{22}^2}}\exp[-\frac{1}{2}\{\frac{(x_{1i}-\mu_1)^2}{\sigma_{11}^2} + \frac{(x_{2i}-\mu_2)^2}{\sigma_{22}^2}\}];\ assuming\ dimension\ d = 2 \quad (2.4.1)$$

Using Product Rule or Chain Rule,
$p(\omega_i|x) = \prod_{j=1}^{2} p(x_j|\omega_i)$

$$p(\omega_i|x) = \frac{1}{\sqrt{(2\pi)^1\sigma_{11}^2}}\exp[-\frac{1}{2}\{\frac{(x_{1i}-\mu_1)^2}{\sigma_{11}^2}\}] * \frac{1}{\sqrt{(2\pi)^1\sigma_{22}^2}}\exp[-\frac{1}{2}\{\frac{(x_{2i}-\mu_2)^2}{\sigma_{22}^2}\}]; \quad (2.4.2)$$

See Appendix A2.3 to see the implementation of above Chain Rule for Naïve Bayes assumption.

Algorithm 2.3

Implement Chain Rule Function via Naïve Bayes Assumption by M-File

1. Calculate Class Mean and
2. Calculate Variance for Each Class
3. Calculate Maximum Likelihood Estimate using mean and covariance parameters.

(a) Use `function probability_density to calculate MLE`

(b)
```
function probability_density = density_norm(X,mu,cov_mat);
probability_density = 1/((2*pi)^(d/2)*sqrt(det(cov_mat)))*exp((-0.5)*diag((X-
ones(n,1)*mu)*inv(cov_mat)*(X-ones(n,1)*mu)'));
```

4. Calculate Probability Distribution function for each Class $P(\omega_1)$ and $P(\omega_2)$

Using Product Rule or Chain Rule,

$p(\omega_i|x) = \prod_{j=1}^{2} p(x_j|\omega_i)$

$$p(\omega_i|x) = \frac{1}{\sqrt{(2\pi)^1\sigma_{11}^2}}\exp[-\frac{1}{2}\{\frac{(x_{1i}-\mu_1)^2}{\sigma_{11}^2}\}] * \frac{1}{\sqrt{(2\pi)^1\sigma_{22}^2}}\exp[-\frac{1}{2}\{\frac{(x_{2i}-\mu_2)^2}{\sigma_{22}^2}\}];$$

5. To calculate MLE for each point in Mesh grid, $x = P(\omega_1) * P(\omega_2)$;

See Appendix2.4 to see the m-file of plotting the contour of Gaussian Distribution of Bivariate Case with Bayes Assumption.

Algorithm 2.4

Plot Contours of MLE for single Class Bivariate Case

1. Find Position of Each Classes in mesh grid X_1 and X_2
2. Use "contour" to Plot 2-D contour of a single Bivariate Feature Case.
   `"contour(X1,X2,probability_density(:,:,i));"`

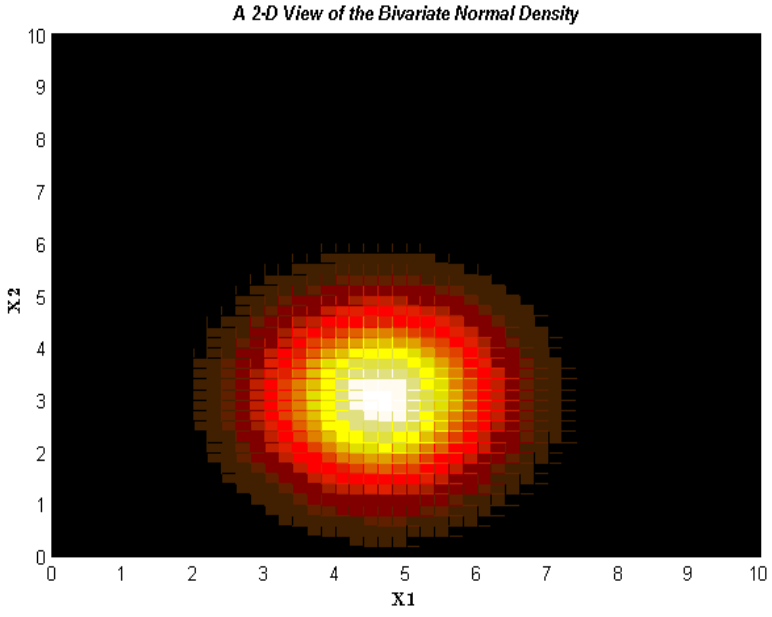

Fig. 4 Feature Independent of Bivariate Gaussian Distribution.

1.1 Matlab Code

Plot Predictive Contours for a single class Bivariate Features with Independent Features

```
%% Plot the predictive contours for class W_1
%normalize the probabilities first
variable_7 = repmat(sum(probability_density,3),[1,1,3]);
probability_density1 = probability_density./repmat(sum(probability_density,3),[1,1,3]);
for i = 3
```

```
    figure(i+2);%figures 3 to 5
    hold off
    for c = 1:3
        position = find(y==labels(c)) ;
        variable_8 = X(position,1);% finds position of Meshgrid's X1 for each classes
        variable_9 = X(position,2);% finds position of Meshgrid's X1 for each classes
plot(X(position,1),X(position,2),markers{c},'markersize',5,'linewidth',1,'markerfacecolor',color{
c});hold on;
end
    contour(X1,X2,probability_density1(:,:,i));
    info = sprintf('Probability contours for class %g',i);
    set(gca,'Title',text('String',info,'FontAngle', 'italic', 'FontWeight',
'bold'),'xlabel',text('String', '$\mathbf{X_1}$', 'Interpreter', 'latex'),'ylabel',text('String',
'$\mathbf{X_2}$', 'Interpreter', 'latex'));end%%
```

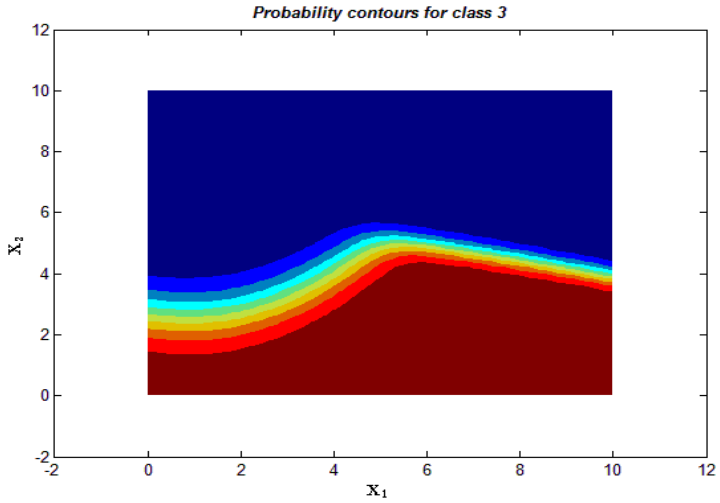

Fig. 5 Predictive Contour of Independent Bivariate Feature of Single Class.

1.2 Matlab Code *Plot Probability Density Function in 3-D Surface for a single class Bivariate Features*

```
%% plot the surface in 3D in Meshgrid
for c = 3  %Index of class
 % Function that calculates probability_density for each points in Meshgrid
    probability_density_function = density_norm([X1(:),X2(:)], class_mean(c,:),
feature_covar(:,:,c));
% reshape function converts an array of 1D to 2D/3D meshgrid, c is no. of classes
% reshape makes 3 separate arrays to construct contours over 2 feature meshgrid
    probability_density(:,:,c) = reshape(probability_density_function,size(X1));
    mesh(X1, X2, probability_density(:,:,c));
end
% Add title and axis labels
set(gca,'Title',text('String','A 3-D View of the Bivariate Normal Density',...
'FontAngle', 'Italic', 'FontWeight', 'bold'),...
'xlabel',text('String', '$\mathbf{X}$', 'Interpreter', 'latex'),...
'ylabel',text('String', '$\mathbf{Y}$', 'Interpreter', 'latex'),...
'zlabel',text('String', 'density', 'FontAngle', 'Italic', 'FontWeight', 'bold'))
view(-10, 50)
colormap('hot')%%
```

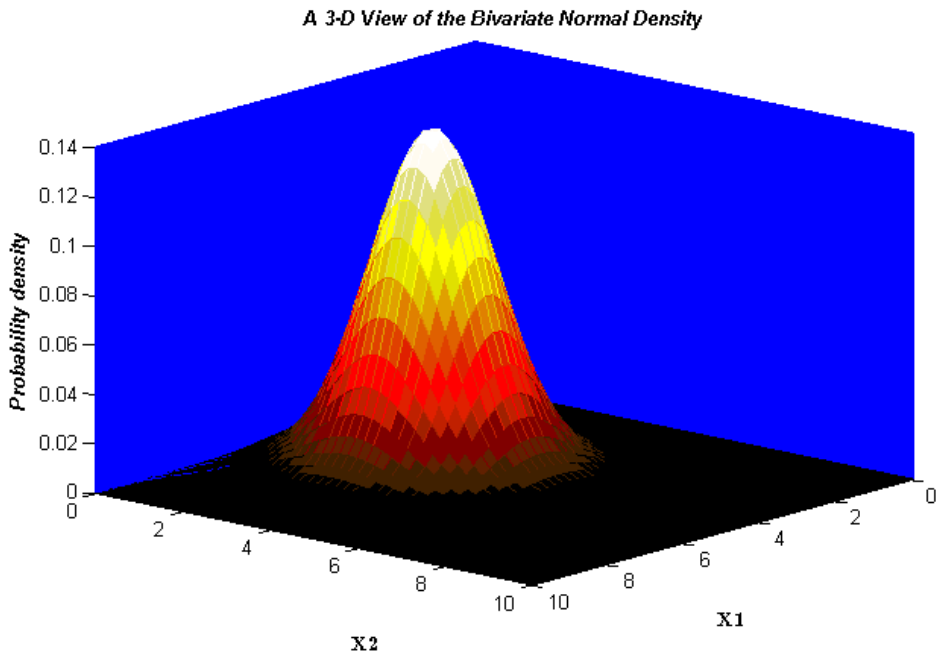

Fig. 6 View of 3-D Feature Independent of Bivariate Gaussian Distribution.

- *For Non- Bayes' assumption,*

$$Determinant, |\sum| = \sigma_{11}^2 * \sigma_{22}^2 - \sigma_{12} * \sigma_{21} \; ; \; where \; \sigma_{12}, \sigma_{21} \neq 0 \tag{2.5.1}$$

$$Finding \; inverse \; of \; Covarience, \sum^{-1} = \frac{\text{Adjoint/Transpose}(\sum)}{\text{Determinant}(\sum)} \tag{2.5.2}$$

$$\sum^{-1} = \frac{1}{\sigma_1^2 * \sigma_2^2 - \sigma_{12} * \sigma_{21}} \begin{bmatrix} \sigma_{22}^2 & -\sigma_{21} \\ -\sigma_{12} & \sigma_{11}^2 \end{bmatrix} \tag{2.5.3}$$

For Maximum Likelihood Estimate between features X_1, X_2

For Dependant Bivariate Features X_1, X_2, Using Product Rule is not Valid, $f(x_1, x_2) \neq f(x_1) * f(x_2)$

$p(\omega_i|x) \neq \prod_{j=1}^{2} p(x_j|\omega_i)$

$$p(\omega_i|x) \neq \frac{1}{\sqrt{(2\pi)^1 \sigma_{11}^2}} \exp[-\frac{1}{2}\{\frac{(x_{1i} - \mu_1)^2}{\sigma_{11}^2}\}] * \frac{1}{\sqrt{(2\pi)^1 \sigma_{22}^2}} \exp[-\frac{1}{2}\{\frac{(x_{2i} - \mu_2)^2}{\sigma_{22}^2}\}] ;$$

Therefore,

$$p(\omega_i|x) = \frac{1}{\sqrt{(2\pi)^d \sigma_{11}^2 * \sigma_{22}^2}} \exp[-\frac{1}{2}(x-\mu)^t \sum^{-1} (x-\mu_i)]; \; assuming \; dimension \; d = 2 \tag{2.5.4}$$

$$p(\omega_i|x) = \frac{1}{\sqrt{(2\pi)^d |\sum|}} \exp[-\frac{1}{2}(x-\mu)^t \sum^{-1} (x-\mu_i)]; \; assuming \; dimension \; d = 2 \tag{2.5.5}$$

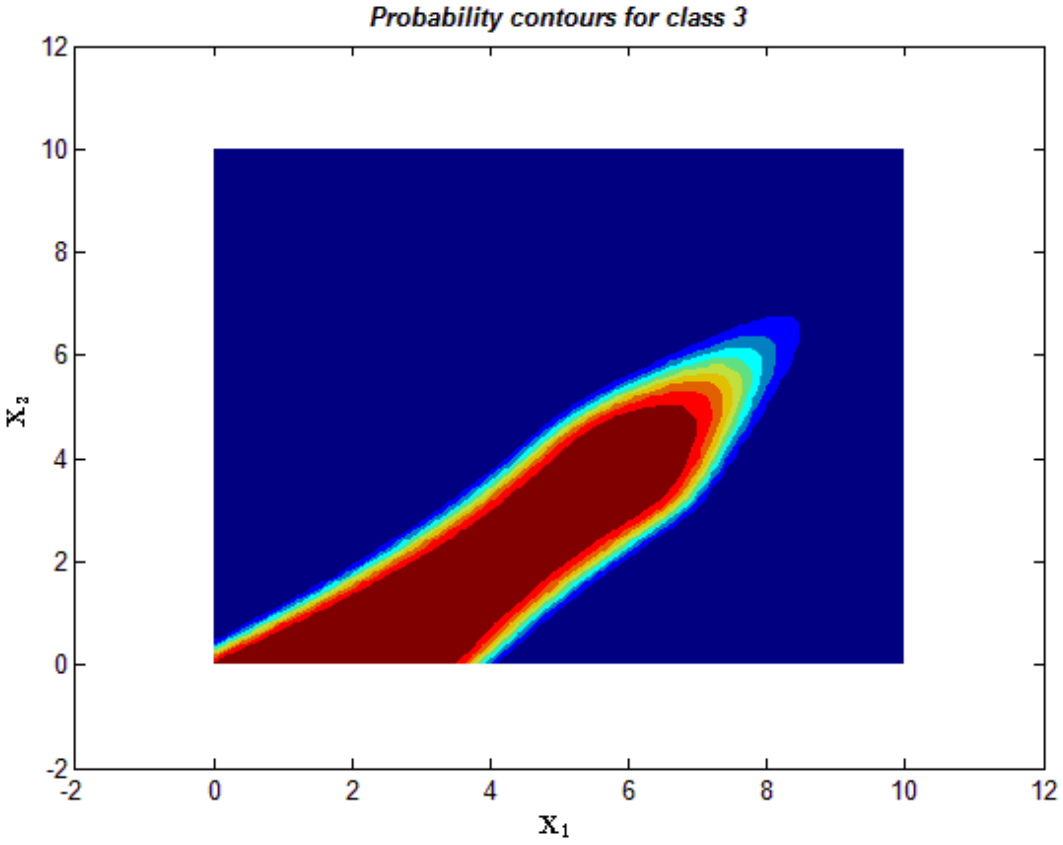

Fig. 7 Non-Indepenent Contour of Class 1 of Bivariate Gaussian Distribution.

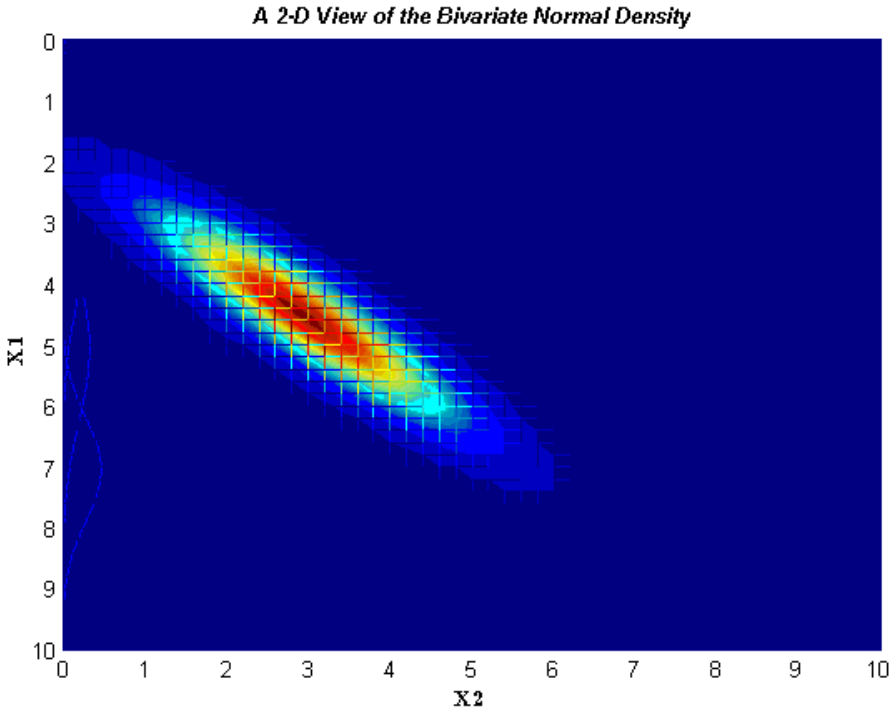

Fig. 8 Non-Indepenent Feature Distribution of Bivariate Gaussian Distribution.

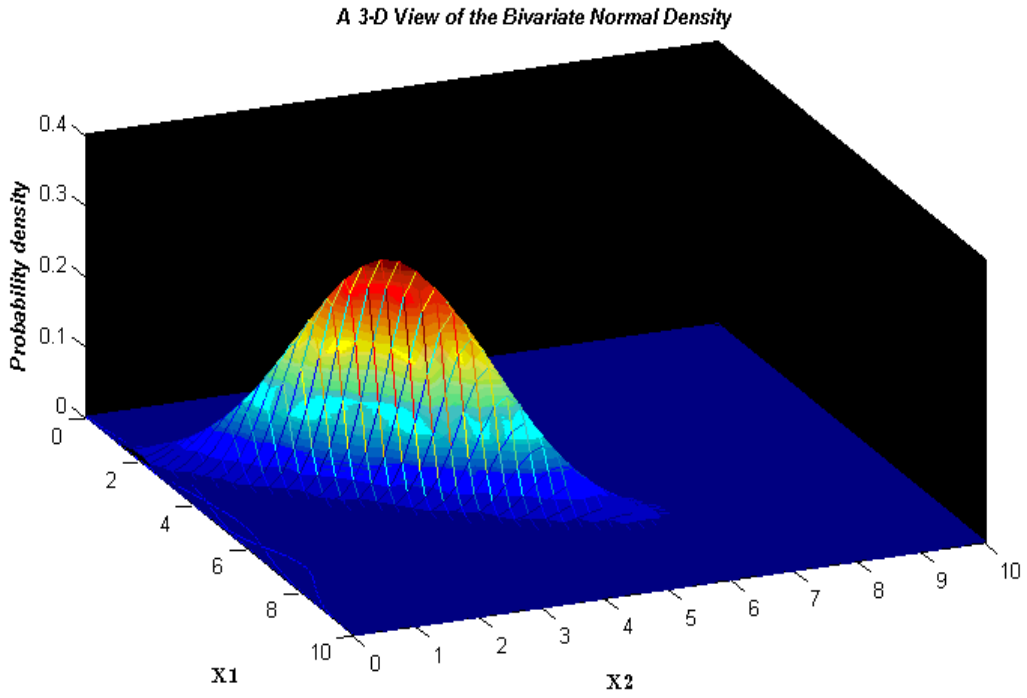

Fig. 9 View of 3-D Feature non-independent of Bivariate Gaussian Distribution.

III. DISCRIMINANT FUNCTION $g_i(x)$

For Mahalanobi's distance between mean feature variables for each individual points in a meshgrid of *N* dimensions, $\gamma^2 = (x_{Ni} - \mu_N)^t \sum^{-1} (x_{Nj} - \mu_N)^1$, where i and j are feature space points over a mesh grid and μ_N is mean of each features.

Discriminant function helps to classify feature space by creating a common decision boundary,

$discriminant\ function,\ g_i(x) = p(x|\omega_i) * p(\omega_i)$; taking ln on both sides of the equation gives,

$$g_i(x) = \ln p(x|\omega_i) + \ln p(\omega_i) \ ; \text{where } i \text{ stands for classes } \omega_i \quad (3.1.1)$$

$$where, p(x|\omega_i) = \frac{1}{\sqrt{(2\pi)^d |\sum_i|}} \exp[-\frac{1}{2}(x-\mu)^t \sum_i{}^{-1} (x-\mu_i)] \quad (3.1.2)$$

Rewriting Discriminant fn and elaborating the term $\ln p(x|\omega_i)$ gives,

$$\ln p(x|\omega_i) = -\frac{1}{2}\{(x-\mu_i)^t \sum_i{}^{-1} (x-\mu_i)\} - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln|\sum_i| \ ; \quad (3.1.3)$$

$$rewriting\ discriminant\ function, \quad g_i(x) = \ln p(x|\omega_i) + \ln p(\omega_i) \ ; \ gives \quad (3.1.4)$$

$$g_i(x) = -\frac{1}{2}\{(x-\mu_i)^t \sum_i{}^{-1} (x-\mu_i)\} - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln|\sum_i| + \ln p(\omega_i) \quad (3.1.5)$$

$where, \ln p(\omega_i)$ becomes redundant if Prior Probabilities are same $p(\omega_i)$,

B. Quadratic Decision Boundary

Since the term $\ln p(x|\omega_i) = -\frac{1}{2}\{(x-\mu_i)^t \sum_i{}^{-1} (x-\mu_i)\} - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln|\sum_i|$, is a quadratic decision equation, Bayes Discriminant function becomes a quadratic boundary function.

Case 1 : $\sum_i = \sigma^2 I$ *, considering Bayes' assumption*

For the case where all diagonal elements $\sigma_{11}^2 = \sigma_{22}^2 = \sigma_{33}^2$ are equal to each other and taking Bayes' assumption that features X_N are independent to each other and its corresponding non diagonal elements are zero.

$$\text{Covariance Matrix}, \sum_i = \sigma^2 \mathrm{I}, \quad where\ \mathrm{I}\ is\ the\ Identity\ Matrix \quad (3.2.1)$$

$$\text{Covariance Matrix}, \sum_i = \begin{bmatrix} \sigma_{11}^2 & \phi & \phi \\ \phi & \sigma_{22}^2 & \phi \\ \phi & \phi & \sigma_{33}^2 \end{bmatrix}, where\ \sigma_{11}^2 = \sigma_{22}^2 = \sigma_{33}^2 \quad (3.2.2)$$

Φ null represents non diagonal covariance values that states that an arbitrary N features X_N are independent to each other, ie. $\sigma_{ij} = 0\ \forall\ values$.

1.3 Matlab Code — *Implement Bayes' Assumption for Bivariate features for Three Classes*

```
%% With assumption: 2 Features have same variance for each class
for c = 1:3
    feature_covar(1,2,c) = 0;
    feature_covar(2,1,c) = 0;
    variable1 = feature_covar(1,1,c) ;
    feature_covar(2,2,c) = variable1 ;
end%%
```

$$Finding\ Determinant, |\sum_i| = \sigma^{2d}, where\ d\ is\ no.of\ dimensions \quad (3.2.3)$$

$$Finding\ Inverse, \sum_i^{-1} = \frac{1}{\sigma^2} * \mathrm{I} \quad (3.2.4)$$

Therefore $Discriminant\ function\ becomes, g_i(x) = -\frac{||x-\mu_i||^2}{2\sigma^2} + \ln p(\omega_i), where\ i\ no.of\ classes$ (3.2.5)

For an arbitrary point x in a feature space to be classed as ω_i , $g_i(x) - g_j(x) - g_k(x) > 0$

Since, $||x_{ij} - \mu_i||^2$ states the squared distance between ij^{th} point of x and its mean μ_i. The term $-||x_{ij} - \mu_i||^2$ is maximum if the term $||x_{ij} - \mu_i||$ is minimum. Therefore, $||x_{ij} - \mu_i||^2$ computes distance from its class mean at all points of x over a mesh grid. Points lying in the nearest class ω_N that are close to its class mean μ_N is classified to that particular class ω_N.

1.4 Matlab Code — *Plot Gaussian Distribution with Mean and Variance for 3 Classes*

```
%% plot Contours of MLE iso-probability contour ellipses for each class
figure
for i=3   %%Index of class
    for c = 1:3
        position = find(y==labels(c));
plot(X(position,1),X(position,2),markers{c},'markersize',5,'linewidth',1,'markerfacecolor',color{
c});  hold on
    end
% X1 and X2 are features, probability_density is 3rd variable that describes elevation of graph
    contour(X1,X2,probability_density(:,:,i));
    set(gca,'Title',text('String','Data with Class-conditional Iso-probability
Lines','FontAngle', 'italic', 'FontWeight', 'bold'),'xlabel',text('String', '$\mathbf{X_1}$',
'Interpreter', 'latex'),'ylabel',text('String', '$\mathbf{X_2}$', 'Interpreter', 'latex'))
    legend('class 1','class 2','class 3');end
```

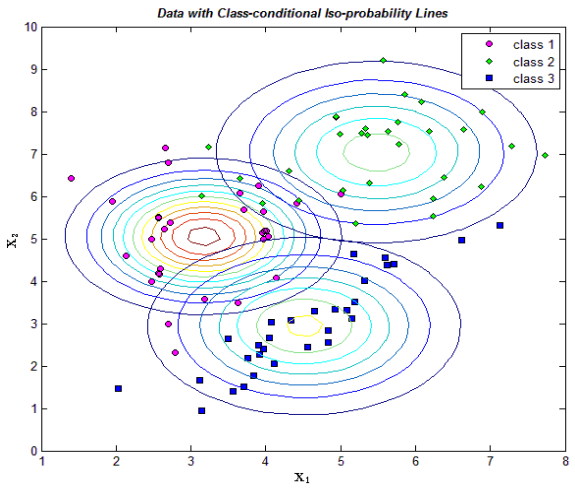

Fig. 10 Gaussian Data Distribution of 3 classes

1.5 Matlab Code *Plot 3-D surface with Bayesian Assumption with equal Variance for Three Classes*

```
%% plot the surface in 3D in Meshgrid
for c = 1:3
 % Function that calculates probability_density for each points in Meshgrid
    probability_density_function = density_norm([X1(:),X2(:)], class_mean(c,:),
feature_covar(:,:,c));
% reshape function converts an array of 1D to 2D/3D meshgrid, c is no. of classes
% reshape makes 3 separate arrays to construct contours over 2 feature meshgrid
    probability_density(:,:,c) = reshape(probability_density_function,size(X1));
    mesh(X1, X2, probability_density(:,:,c));
end
% Add title and axis labels
set(gca,'Title',text('String','A 3-D View of the Bivariate Normal Density','FontAngle', 'Italic',
'FontWeight', 'bold'),'xlabel',text('String', '$\mathbf{X}$', 'Interpreter', 'latex'),...
'ylabel',text('String', '$\mathbf{Y}$', 'Interpreter', 'latex'),...
'zlabel',text('String', 'density', 'FontAngle', 'Italic', 'FontWeight', 'bold'))view(-10,
50);colormap('hot')
```

For all features, $\sigma_{ii} = \sigma$,

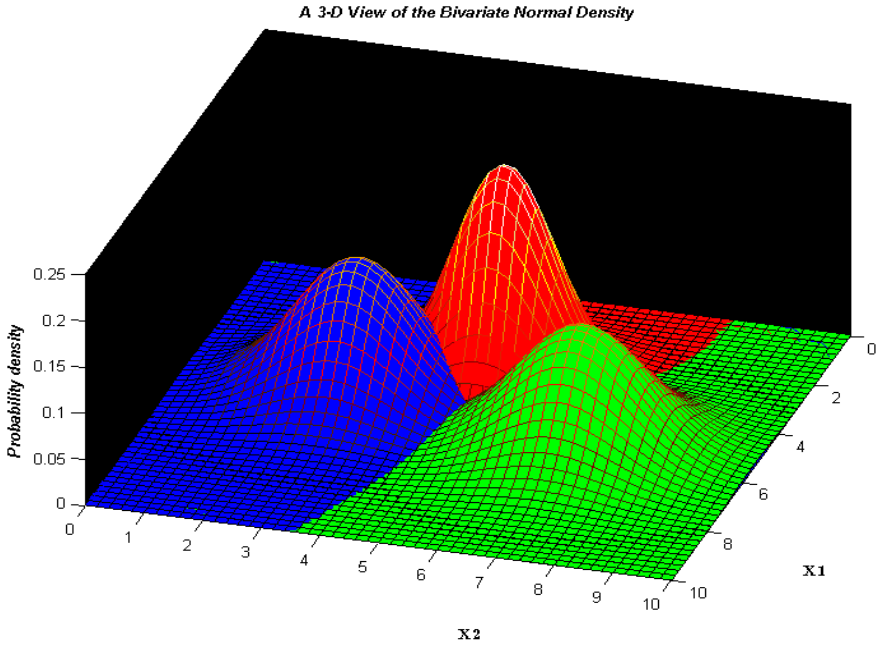

Fig. 11 3-D view of Bivariate Independent Feature with equal variance

Decision surface between classes are orthogonal to each other. So, orthogonal distance bisector between its mean μ_i, μ_j, μ_k becomes a minimum distance classifier.

Case 2 : $\sum_i = \sum * \boldsymbol{I}$, *considering Bayes' assumption*

Using Covariance with arbitrary diagonal elements and non diagonal elements $= \phi$, *we get*

$$Determinant\ function, g_i(x) = -\frac{1}{2}\{(x-\mu_i)^t \sum_i{}^{-1}(x-\mu_i)\}, \quad (3.3.1)$$

$$where, -\frac{1}{2}\{(x-\mu_i)^t \sum_i{}^{-1}(x-\mu_i)\},\ \text{represents squared Mohalanobi's distance.} \quad (3.3.2)$$

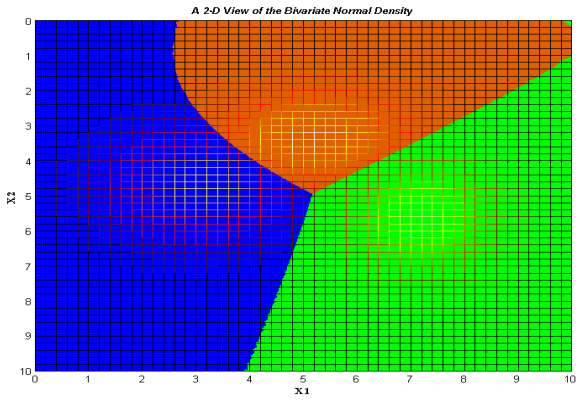

Fig. 12 Contour of Bivariate Independent Feature with Arbitrary variance

$$Since\ in\ the\ eqn, g_i(x) = -\frac{1}{2}\{(x-\mu_i)^t \sum_i{}^{-1}(x-\mu_i)\} - \frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln|\sum_i| + \ln p(\omega_i), \quad (3.3.3)$$

The terms, $-\frac{d}{2}\ln(2\pi) - \frac{1}{2}\ln|\sum_i| + \ln p(\omega_i)$ becomes redundant in the discriminant function for equal priors $\ln p(\omega_i)$. Also $\frac{1}{2}\ln|\sum_i|$is constant for all classes since $determinant, |\sum_i| = \sigma_{ii}^2\ \sigma_{ii}^2\ \sigma_{ii}^2$

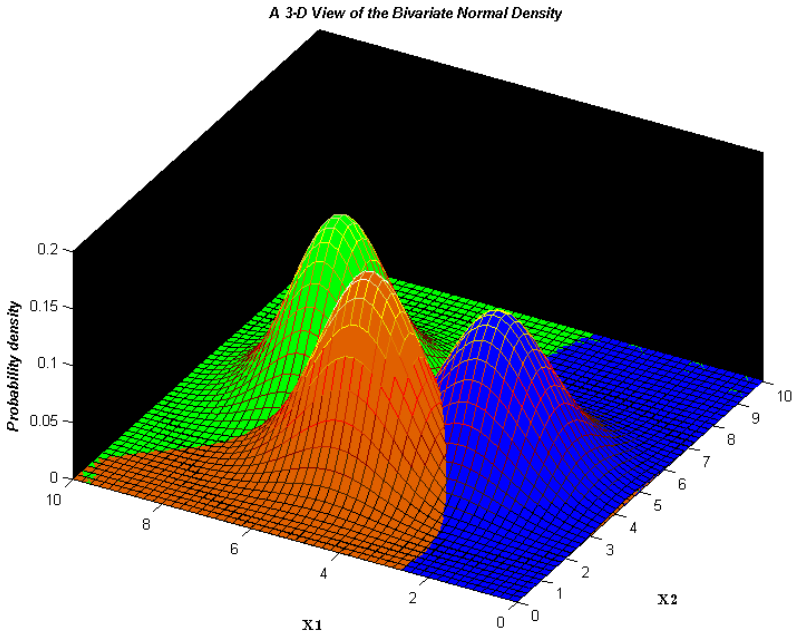

Fig. 13 3-D of Bivariate Independent Feature with Arbitrary variance

Since above $discriminant\ fn,\ g_i(x)$ is mohalanobi's distance, above decision surdace represents a hyperplane minimum distance classifier.

Case 3 : $\sum_i = \sum$ considering Non-Bayes' assumption

$$\sum_i = \begin{bmatrix} \sigma_{11}^2 & \cdots & \sigma_{in}^1 \\ \vdots & \ddots & \vdots \\ \sigma_{nj}^1 & \cdots & \sigma_{nn}^2 \end{bmatrix}, where\ non\ diagonal\ elements\ \neq\ \phi \quad (3.4.1)$$

Above Covariance for each class i represents non independent features X_N.

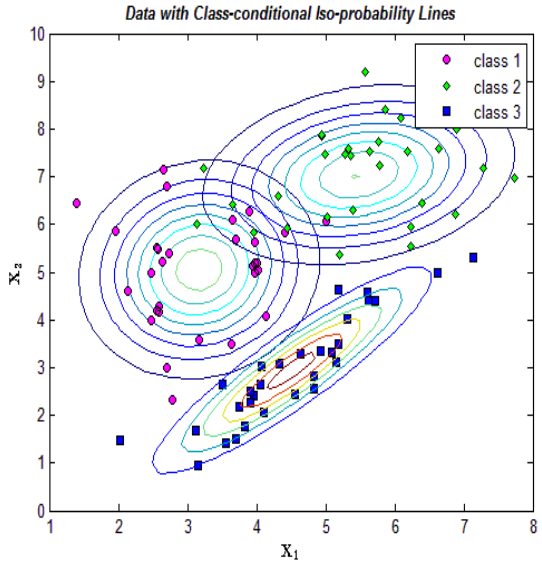

Fig. 14 Feature Distribution with Arbitrary variance

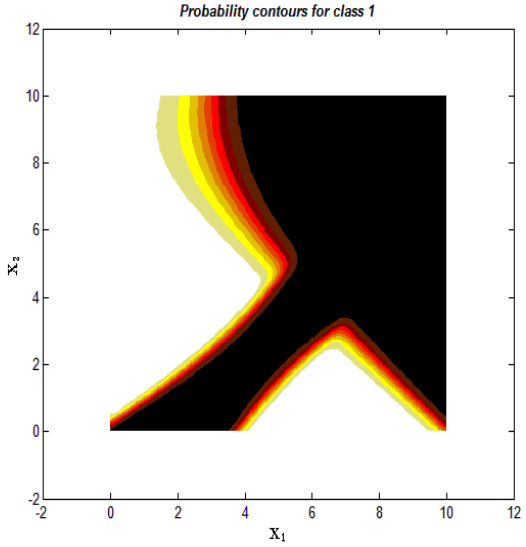

Fig. 15 Contour of Bivariate Feature Dependent Feature with Arbitrary variance

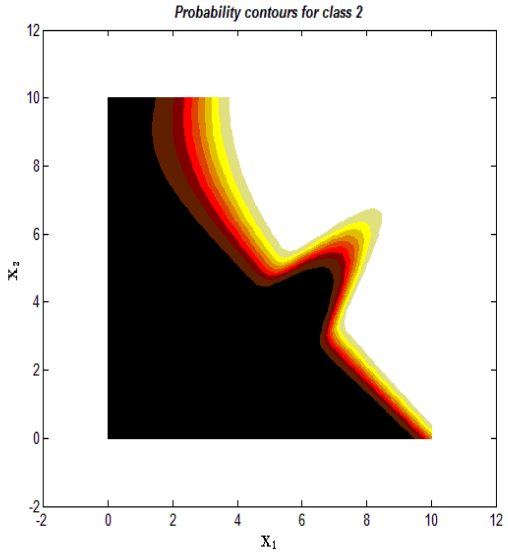

Fig. 16 Contour of Bivariate Feature Dependent Feature with Arbitrary variance

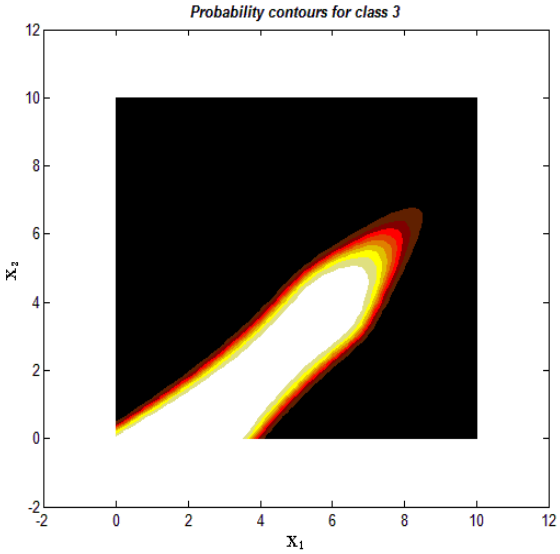

Fig. 17 Contour of Bivariate Feature Dependent Feature with Arbitrary variance

$Discriminant\ fn, g_i(x)$ represents a quadratic decision boundary and becomes a hyper quadratic surface for dimension = 2.

$$g_i(x) = x_{ij}^t \sum_i^{-1} x_{ij} + [\sum_i^{-1}\mu_i]^t\ x - \frac{1}{2}\mu_i^t\ \sum_i^{-1}\ \mu_i - \frac{1}{2}\ \ln|\sum_i| + \ln\ p(\omega_i) \qquad (3.4.2)$$

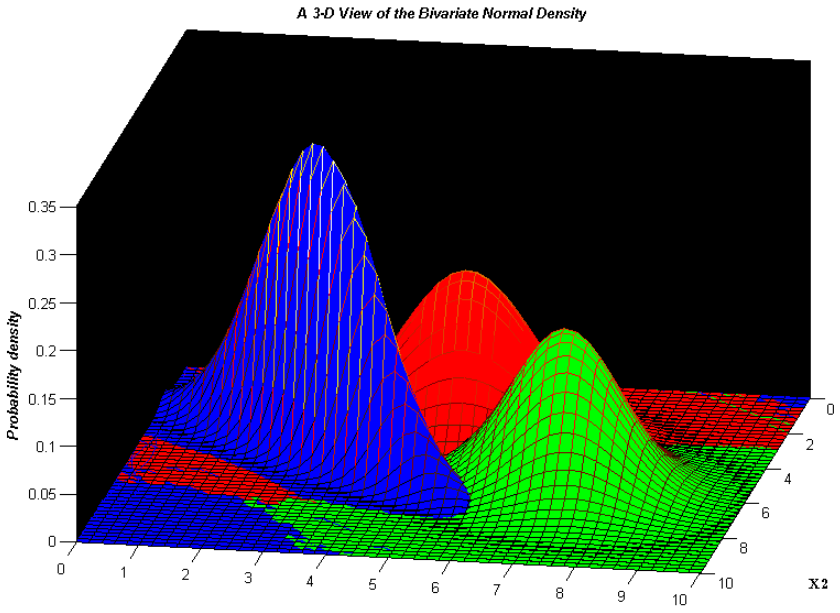

Fig. 17 3-D of Bivariate Non-Independent Feature with Arbitrary variance

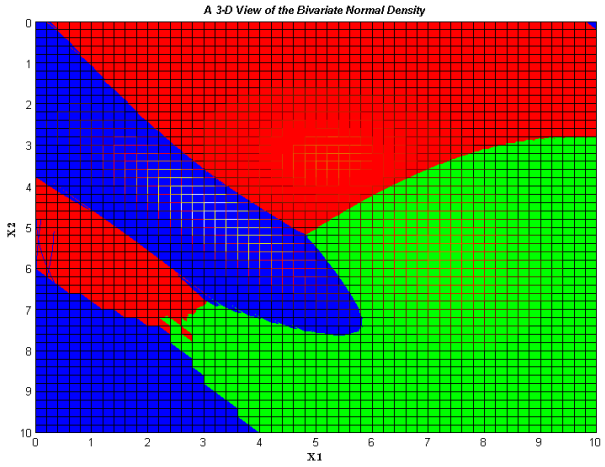

Fig. 18 Contour of Bivariate Non-Independent Feature with Arbitrary variance

IV. PROBABILITY OF ERROR

Main advantage of a stochastic Naïve Bayes classifier is the probability of error for any classification (or action) in contrast to a deterministic classifier where there is absence of uncertainty and gives a binary outcome having a unique global binary minimum.

To calculate optimal Bayes Decision boundary and to reduce probability of error for an observation vector *x*, Bayes classifier classifies *x* in feature space as class ω1 and vice versa for an observation vector with likelihood P(ω1|x) greater than P(ω2|x).

So, Probability of error *for all likelihood estimates of x*,

$$P(error|x) = \begin{cases} P(\omega_1|x) \text{ if we decide } \omega_2; \text{ or} \\ P(\omega_2|x) \text{ if we decide } \omega_1; \end{cases}$$

- *Example of Minimizing Probability of Error via Proper Classification of Bivariate Case with two clasess*

Our Goal is to minimize, P(*error* | *x*) *where,* P(*error* | *x*) = min {$P(\boldsymbol{\omega_1}|x)$ or $P(\boldsymbol{\omega_2}|x)$}
For proper classification,
If Bayes Decision is made in Favor of Class 1 or ω_1, *then* Probability of Error is P($\omega_2|x$) ;
If Bayes Decision is made in Favor of Class 2 or ω_2, *then* Probability of Error is P($\omega_1|x$) ;

For every test observation vector, P(*error*|*x*) is optimized and is calculated by taking integral,
P(*error*) = $\int$ P(*error*,*x*) *dx* = $\int$ P(*error*|*x*)*P(*x*)*dx* ;
Or, P(*error*|*x*) = min [P($\omega_1|x$), P($\omega_2|x$)] ;
Decision is made in favor of class ω1 if P($\omega_1|x$) > P($\omega_2|x$) ; otherwise decide ω2 to minimize P(*error*).

- *For special cases,*

(a) If prior probabilities, $P(\boldsymbol{\omega_1}) = P(\boldsymbol{\omega_2})$, decision for classes is based on likelihood estimates p(x|ω1), p(x|ω2).
(b) If likelihood estimate probabilities are $p(x|\omega_1) = p(x|\omega_2)$, decision for classes is based on prior probabilities $p(\omega_1)$, $p(\omega_2)$.

Both of these factors are used in making a Bayes decision to achieve the minimum probability of error.

i. Loss Function for Multi Category Case

Loss function is defined by actions α_1, ..., α_c for each classes ω_1, ..., ω_c which is used to make a decision for classification from likelihood estimates. Loss function $\lambda(\alpha_i|\omega_j)$, where $i \neq j$, analyses minimization of overall errors that states loss incurred for taking action α_i when actual class is ω_j. Costs for each action λ varies for each dataset of random variables that impacts decision for classification.

- Example of Bayes Risk

if $P(\omega_i|x) > P(\omega_j|x)$, probability of observed variable being in actual state is ω_i,
and its corresponding expected loss for taking action α_i, *Risk function*, $R(\alpha_i|x) = \lambda(\alpha_i|\omega_j)P(\omega_j|x)$;
For every observation vector *x* in a search space, overall risk for every action α(x) R function associated with action α_i is:
$R = R(\alpha(\mathbf{x})|\mathbf{x})p(\mathbf{x})\, d\boldsymbol{x}$, *where* dx is d-dimensional, integrated through entire feature space.
Every α(x) is selected to minimize $R(\alpha_i(x))$ for every *x* test vector or likelihood estimate,
$R(\alpha i|x) = \lambda(\alpha i|\omega j)P(\omega j|x)$; *where* $i \neq j$ (4.1.1)
Corresponding Overall risk function in a multi-category classed dataset is called Bayes risk.

For simplicity, let $\lambda_{ij} = \lambda(\alpha_i|\omega_j)$ is the loss for error classification of ω_i where actual class is ω_j. Therefore conditional risk for two category classification is:

- $R(\alpha_1|x) = \lambda_{11}P(\omega_1|x) + \lambda 12P(\omega_2|x)$; *and* (4.1.2)
- $R(\alpha_2|x) = \lambda_{21}P(\omega_1|x) + \lambda_{22}P(\omega_2|x)$; *classify* ω_1 *if* $R(\alpha_1|x) < R(\alpha_2|x)$ *or vice versa* (4.1.3)

Assuming $(\lambda_{21} - \lambda_{11})$ and $(\lambda_{12} - \lambda_{22}) > 0$
In terms of posterior probabilities, $(\lambda_{21} - \lambda_{11})*P(\omega_1|x) > (\lambda_{12} - \lambda_{22})*P(\omega_2|x)$ *classify* ω_1 *or vice versa*

Rewriting above eq^n *in terms of posterior probabilities by the prior probabilities and the conditional densities,*
we get, if $(\lambda_{21} - \lambda_{11})*P(x|\omega_1)*P(\omega_1) > (\lambda_{12} - \lambda_{22})*P(x|\omega_2)*P(\omega_2)$, *classify as* ω_1 *or vice versa*
where $P(\omega_1)$, $P(\omega_2)$ is *Belief Probability* ; and
λij *is independent of likelihood observation variable x*

ii. Minimum-Error-Rate Classification : Error Probabilities Integrals
For further simplification, loss function is defined by zero-one loss function, ie. *either* $\lambda(\alpha_i|\omega_i) = 0$; *or* $\lambda(\alpha_i|\omega_j) = 1$;

This assumption is based on unit loss where all errors are equally costly. So, total risk for all feature space related to loss function with unit loss is the average probability of error. So, for minimization of probability of error right classification is necessary.

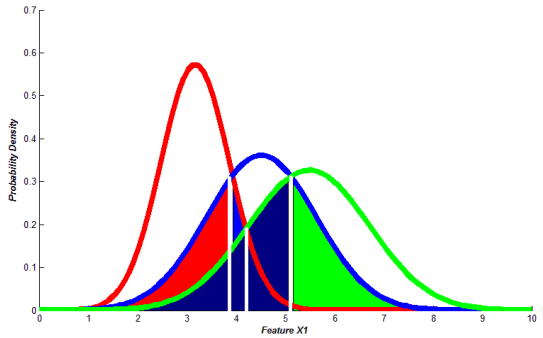

Fig. 19 Pdf for Feature X_1 for all 3 Classes.

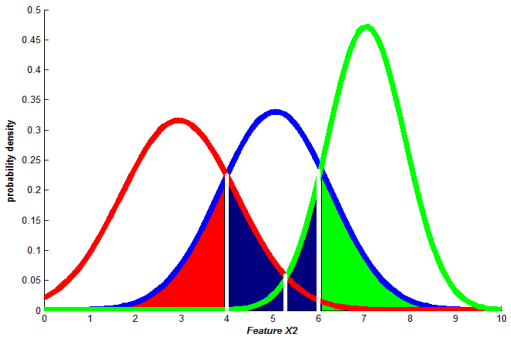

Fig. 20 Pdf for Feature X_2 for all 3 Classes.

In the above diagrams for bivariate features X_1, X_2, if λ_{ij} is not equal to null, then threshold of decision boundaries X_a and X_b changes other than the ones shown above that is exactly between the Gaussian mean of each classes.

Area under probability distribution function for class ω_1 (red) represents probability of error for classifying an observed variable *x as* ω_2 or ω_3 where actual class is ω_1; other areas represents the converse. Decision boundary is at equal posterior probabilities marked by white. Each shaded area separated by white marker is the loss incurred while misclassifying an observed variable. Total shaded area of all 3 classes is Bayes error rate.

Algorithm 4.1

Plot Features X1, X2, X3 and find the Shaded Areas of all three classes ω1, ω2 and ω3

To Calculate the Probability of Error,

1. Calculate the Area of Shaded Region,
2. For each Misclassification Error, Calculate Area of Minimum a Posterior.
3. For Total Bayes Risk,
   (a) Range of Area under Blue curved Gaussian Distribution
   (b) Range of Area under Red curved Gaussian Distribution
   (c) Range of Area under Green curved Gaussian Distribution
   (d) Range of Area under Red curved Gaussian Distribution

4. Range of Area of Gaussian Distribution of Feature X_1,

```
Range = (X>4.0 & X<8); %Range of Area under Blue curved Gaussian Distribution
Range = (X22>0 & X22<4.0); %Range of Area under Red curved Gaussian Distribution
Range = (X22>6.0599989999 & X22<9); %Range of Area under Blue curved Gaussian Distr
Range = (X22>1 & X22<6.0599989999); %Range of Area under Blue curved Gaussian Distr
Range = (X22>6.0599989999 & X22<10);  %Range of Area under Green curved Gaussian Distr
```

5. Range of Area of Gaussian Distribution of Feature X_2,

```
Range = (X11>3.93939385 & X11<6); %Range of Area under Blue curved Gaussian Distr
Range = (X11>1 & X11<3.93939385); %Range of Area under Red curved Gaussian Distr
Range = (X11>5.149 & X11<9); %Range of Area under Green curved Gaussian Distr
Range = (X11>1 & X11<5.149); %Range of Area under Red curved Gaussian Distr
```

iii. Minimax Criterion

Minimax criterion is the minimization of maximum overall risk function :

$$R = \int_{\mathcal{R}_1} [\lambda_{11} P(\omega_1)\ p(\mathbf{x}|\omega_1) + \lambda_{12} P(\omega_2)\ p(\mathbf{x}|\omega_2)]\ d\mathbf{x} + \int_{\mathcal{R}_2} [\lambda_{21} P(\omega_1)\ p(\mathbf{x}|\omega_1) + \lambda_{22} P(\omega_2)\ p(\mathbf{x}|\omega_2)]\ d\mathbf{x}. \quad (4.2.1)$$

For minimax solution, risk in independent of priors. Therefore rewriting in terms of substitution, minimax risk becomes independent of Prior probability,

$$R_{mm} = \lambda_{22} + (\lambda_{12} - \lambda_{22}) \int_{\mathcal{R}_1} p(\mathbf{x}|\omega_2)\, d\mathbf{x}$$
$$= \lambda_{11} + (\lambda_{21} - \lambda_{11}) \int_{\mathcal{R}_2} p(\mathbf{x}|\omega_1)\, d\mathbf{x}. \quad (4.2.2)$$

iv. Game Theory

Minimax criterion is mostly used in game theory where agents take actions maximally damaging to other agents. Agents use actions or classifies where costs/errors are minimized to maximize gain.

v. Discriminant functions for Multi-Category Case

Discriminant functions $g_i(x)$, $i = 1, ...,c$. The classifier is said to classifies feature/observation vector x to class ω_i if $g_i(x) > g_j(x)\ \forall\ j \neq i$. Bayes classifier classifies using c discriminant functions and selects the category corresponding to the largest discriminant.

For c number of classes, the discriminant functions (g_1, g_2, g_3) of a Bayes classifier classifies a test vector x to ω_1
if $g_1 > g_2 > g_3$, $g(x) \equiv g_1(x) - g_2(x) - g_3(x)$, (4.2.3)
Minimum-error-rate discriminant function is defined as $g(\boldsymbol{x}) = P(\omega_1|\boldsymbol{x}) - P(\omega_2|\boldsymbol{x}) - P(\omega_3|\boldsymbol{x})$ (4.2.4)

IV. Risk Minimization via Optimal Action

From concept of Probability of Error, we can introduce a Loss Function
Let classes be: $\boldsymbol{\omega_1, \omega_2, \omega_3}$
Actions: α_1, α_2, α_3

A. Example : Risk Minimization

$\boldsymbol{\lambda}_1(\alpha_1|\ \boldsymbol{\omega}_1)$ is the loss incurred when taking action α_1 in favor of class $\boldsymbol{\omega}_1$, and vice versa

- For above mentioned classes, if an Action is selected it can be computed by function,

$\boldsymbol{\lambda}_1(\alpha_1|\ x) = \boldsymbol{\lambda}_{11}(\alpha_1|\ \boldsymbol{\omega}_1)\ P(\boldsymbol{\omega}_1|x) + \boldsymbol{\lambda}_{12}(\alpha_1|\ \boldsymbol{\omega_2})\ P(\boldsymbol{\omega_2}|x) + \boldsymbol{\lambda}_{13}(\alpha_1|\ \boldsymbol{\omega_3})\ P(\boldsymbol{\omega_3}|x)$ (4.2.5)
$\boldsymbol{\lambda}_2(\alpha_2|\ x) = \boldsymbol{\lambda}_{21}(\alpha_2|\ \boldsymbol{\omega}_1)\ P(\boldsymbol{\omega}_1|x) + \boldsymbol{\lambda}_{22}(\alpha_2|\ \boldsymbol{\omega_2})\ P(\boldsymbol{\omega_2}|x) + \boldsymbol{\lambda}_{23}(\alpha_2|\ \boldsymbol{\omega_3})\ P(\boldsymbol{\omega_3}|x)$ (4.2.6)
$\boldsymbol{\lambda}_3(\alpha_3|\ x) = \boldsymbol{\lambda}_{31}(\alpha_3|\ \boldsymbol{\omega}_1)\ P(\boldsymbol{\omega}_1|x) + \boldsymbol{\lambda}_{32}(\alpha_3|\ \boldsymbol{\omega_2})\ P(\boldsymbol{\omega_2}|x) + \boldsymbol{\lambda}_{33}(\alpha_3|\ \boldsymbol{\omega_3})\ P(\boldsymbol{\omega_3}|x)$ (4.2.7)

Here, for unit Loss function, λ_{11}, λ_{22}, λ_{33} is correct action for correct class where λ_{11}, λ_{22}, $\lambda_{33} \approx 0$
λ_{12}, λ_{13}, λ_{21}, λ_{23}, λ_{31}, λ_{32} is action for wrong class, and so: λ_{12}, λ_{13}, λ_{21}, λ_{23}, λ_{31}, $\lambda_{32} \approx 1$

- *Since Main Objective : Minimize Total Risk Function via taking Optimal Action*

So, if Action, α_1 is selected then action is chosen in favor of class $\boldsymbol{\omega}_1$ is chosen, or
if Action, α_2 is selected then action is chosen in favor of class $\boldsymbol{\omega}_2$ to minimize total risk, or
if Action, α_3 is selected then action is chosen in favor of class $\boldsymbol{\omega}_3$ to minimize total risk,

Therefore, if taking Action α1 makes Risk function: $\boldsymbol{\lambda}_1(\alpha_1|\ x) > \boldsymbol{\lambda}_2(\alpha_2|\ x) > \boldsymbol{\lambda}_3(\alpha_3|\ x)$, *then* Action α_1 is necessary to minimize Total Risk.

Table 3 Bayes Risk and its corresponding Actions

	ω1	**ω2**	**ω3**
α1	λ11 = 0	λ12 = 1	λ13 = 1
α2	λ21 = 1	λ22 = 0	λ23 = 1
α3	λ31 = 1	λ32 = 1	λ33 = 0

V. Conclusion

One of the major traits of Naïve Bayes classifier is its probability of misclassification, whereas, support vector machines or Neural Network can give only a binary outcome that typically depends on threshold of a sinusoid function.

Naïve Bayes classifier model is computationally faster than other learning techniques which require greater computational power. Naïve Bayes classifier uses chain rule to reduce dimensionality based on assumption that features $X_1, X_2 \ldots X_N$ are independent to each other with arbitrary or equal variance for each features. While this assumption helps in reducing curse of dimensionality, real life data with independent features are very rare and do not exist very often. Kernel density estimation computes maximum likelihood estimate without making any assumptions of independence of features. In other words, it is a non parametric method to calculate probability density estimation of a random variable.

VI. Appendix

A1.1

Algorithm 1.1

Test Observation Vector in Naïve Bayes Algorithm and Plot its 3-D Surface

```
%%MATLAB Code to compute Maximum a Posteriori:
load data_bayes3
%% Plot the data
labels = [1;2;3];
markers = {'ko','kd','ks'};        %appearance of a point on the plot
color = {'magenta','green','blue'};
figure;hold off
for c = 1:3
    position = find(y==labels(c));%finds position of each classes on Meshgrid
plot(X(position,1),X(position,2),markers{c},'markersize',5,'linewidth',1,'markerfacecolor',color{
c});hold on ; end
set(gca,'Title',text('String','Data, 3 Distinct Classes','FontAngle', 'italic', 'FontWeight',
'bold'),'xlabel',text('String', '$\mathbf{X_1}$', 'Interpreter', 'latex'),'ylabel',text('String',
'$\mathbf{X_2}$', 'Interpreter', 'latex'))
legend('class 1','class 2','class 3')
%% Fit class-conditional Gaussians for each class
%for each class, estimate the mean and covariance matrix from the data
for c = 1:3
    position = find(y==labels(c));%finds position of each classes on Meshgrid
    variable_1=y==labels(c);   %labels each class (eg. 1,2 or 3) with 1 and others with 0
    variable_2=(X(position,1:2)); %finds position of Meshgrid for each classes  %2 columns in X
out of 90*2
    variable_3=mean(X(position,1:2));  %calculates Mean position of Meshgrid for each classes
    class_mean(c,1:2) = mean(X(position,1:2)) ;  % (2x1)
% Covariance Matrix, (2x2), Maximum Likelihood Estimate for each class c
    feature_covar(1:2,1:2,c) = cov(X(position,1:2),1); %Normalized by N (not N-1)
    variable_4=cov(X(position,1:2),1);end
% With Naive Bayes assumption: 2 Features are independent to each other for each class
for c = 1:3
    feature_covar(1,2,c) = 0;
    feature_covar(2,1,c) = 0; end
%% Compute the Probability density for each class
[X1,X2] = meshgrid(0:0.2:10,0:0.2:10);
for c = 1:3
    variable_5 = class_mean(c,:);  %Mean for each classes
    variable_6 = feature_covar(:,:,c);   %Calculates Covariance for each features of classes
% Function that calculates probability_density for each points in Meshgrid
    probability_density_function = density_norm([X1(:),X2(:)], class_mean(c,:),
feature_covar(:,:,c));
% reshape function converts an array of 1D to 2D meshgrid, c is no. of classes
% reshape makes 3 separate arrays to construct contours over 2 feature meshgrid
    probability_density(:,:,c) = reshape(probability_density_function,size(X1));
```

```
end

%% TESTING OBSERVATION VECTOR
x = [5.4 2.6];   %Test Observation Vector %class_mean(class,:);%feature_covar(:,:,class);
%% Using Parameter Estimation using Mean and Covariance of a Feature Distribution
%if Features X1, X2 are independent, Covariance with non zero non-diagonal elements are used.
% if Features X1, X2 are dependant, Variance with null components of non-diagonal elements are
used.
for class = 1:3
aposterior(class) = density_norm(x, class_mean(class,:), feature_covar(:,:,class))end
% Weak Prior Belief probability (i.e., uniform prior),
MAP gives the same result as MLE.
%Labelled Data for Class 1 = 30; Class 2 = 30; Class 3 = 30;
%Belief Probability for each Class (Uniform Priors) = 30/90;
(Since,Total Data X = 90) = 1/3 ;
aposterior = ((1/3)*aposterior) %maximum posterior probability
maximum_aposterior = max(aposterior) %maximum posterior probability
find(maximum_aposterior == aposterior)  %index class of maximum a posterior

%% PLOT 3-D SURFACE IN MESHGRID FOR EACH CLASS USING GAUSSIAN DISTRBUTION
for c = 1:3
 % Function that calculates probability_density for each points in Meshgrid
    probability_density_function = density_norm([X1(:),X2(:)], class_mean(c,:),
feature_covar(:,:,c));
% reshape function converts an array of 1D to 2D/3D meshgrid, c is no. of classes
% reshape makes 3 separate arrays to construct contours over 2 feature meshgrid
    probability_density(:,:,c) = reshape(probability_density_function,size(X1));
    mesh(X1, X2, probability_density(:,:,c));
end
% Add title and axis labels
set(gca,'Title',text('String','A 3-D View of the Bivariate Normal Density','FontAngle', 'Italic',
'FontWeight', 'bold'),'xlabel',text('String', '$\mathbf{X}$', 'Interpreter', 'latex'),...
'ylabel',text('String', '$\mathbf{Y}$', 'Interpreter', 'latex'),...
'zlabel',text('String', 'density', 'FontAngle', 'Italic', 'FontWeight', 'bold'))view(-10, 50)
colormap('hot'); end
```

A2.1

Algorithm 2.1

Implement Probability density function for Bivariate Features

```
%% Estimate MEAN and COVARIANCE matrix from the data
%% Fit class-conditional Gaussians for class 3,
for c = 3   % index of class Data
position = find(y==labels(c));   %finds position of each classes on Mesh grid
variable_1=y==labels(c);   %labels each class (eg. 1,2 or 3) with 1 and others with 0
variable_2=(X(position,1:2)); %finds position of Mesh grid for each classes
%2 columns in X out of 90*2
variable_3=mean(X(position,1:2));  %calculates Mean position of Mesh grid for each classes
```

$$class\ mean, \mu = \begin{bmatrix} \mu_1 \\ \mu_2 \end{bmatrix}$$

```
class_mean(c,1:2) = mean(X(position,1:2)) ;  % (2x1)
% Covariance Matrix, (2x2), Maximum Likelihood Estimate for each class c
feature_covar(1:2,1:2,c) = cov(X(position,1:2),1); %Normalized by N (not N-1)
variable_4=cov(X(position,1:2),1);
```

$$covariance\ matrix, \qquad \Sigma = \begin{bmatrix} \sigma_{11}^2 & \sigma_{12} \\ \sigma_{21} & \sigma_{22}^2 \end{bmatrix}$$

```
%% Use above function to compute Probability densitiy for each class
%% Compute Class Mean and Inverse Covariance for each classes
[X1,X2] = meshgrid(0:0.2:10,0:0.2:10);
for c = 3   % index of class Data
variable_5 = class_mean(c,:);  %Mean for each classes
variable_6 = feature_covar(:,:,c); % Covariance for each features of classes
% Function that calculates probability_density for each points in Meshgrid
probability_density_function = density_norm([X1(:),X2(:)], class_mean(c,:), feature_covar(:,:,c));
%% Implements Normal probability density with parameters mu and covariance
%X - points at which probability density is evaluated

function probability_density = density_norm(X,mu,cov_mat);
[n,d] = size(X);
probability_density = 1/((2*pi)^(d/2)*sqrt(det(cov_mat)))*exp((-0.5)*diag((X-ones(n,1)*mu)*inv(cov_mat)*(X-
ones(n,1)*mu)'));end%%
```

Probability density function for bivariate features, $p(x) = \frac{1}{\sqrt{(2\pi)^d|\Sigma|}}\exp[-\frac{1}{2}\{\frac{(x_{1i}-\mu_1)^2}{\sigma_1^2} + \frac{(x_{2i}-\mu_2)^2}{\sigma_2^2}\}]$

```
% reshape function converts an array of 1D to 2D meshgrid c is no. of classes
```

```
% reshape makes 3 separate arrays to construct contours of 2features meshgrid
probability_density(:,:,c) = reshape(probability_density_function,size(X1));end%%
```

A2.2

Algorithm 2.2

Implement Naïve Bayes assumption

```
%% With Naive Bayes assumption: 2 Features are independent to each other for each class
for c = 3
    feature_covar(1,2,c) = 0;
    feature_covar(2,1,c) = 0;end%%
```

A2.3

Algorithm 2.3

Implement Chain Rule Function via Naïve Bayes Assumption by M-File

```
%% Implements Normal probability density with parameters mu and covariance
%X - points at which probability density is evaluated
function probability_density = density_norm(X,mu,cov_mat);
[n,d] = size(X);
probability_density = 1/((2*pi)^(d/2)*sqrt(det(cov_mat)))*exp((-0.5)*diag((X-ones(n,1)*mu)*inv(cov_mat)*(X-
ones(n,1)*mu)'));
%% Training pdf for each feature X1,X2;using: Using CHAIN RULE with Bayes Assumption is taken
% CHAIN RULE using Bayesian Conditional Independence
% Using Bayesian Conditional Independence and finding X using MLE gives same value if X1, X2 are independent
```

Using Product Rule or Chain Rule,

$p(\omega_i|x) = \prod_{j=1}^{2} p(x_j|\omega_i)$

$$p(\omega_i|x) = \frac{1}{\sqrt{(2\pi)^1\sigma_{11}^2}}\exp[-\frac{1}{2}\{\frac{(x_{1i}-\mu_1)^2}{\sigma_{11}^2}\}] * \frac{1}{\sqrt{(2\pi)^1\sigma_{22}^2}}\exp[-\frac{1}{2}\{\frac{(x_{2i}-\mu_2)^2}{\sigma_{22}^2}\}];$$

```
for c = 3
    for feature = 1:2
    test101 = class_mean(c,feature);
    test102 = feature_covar(feature,feature,c);
    maximum_likelihood1(c,feature) = density_norm(x(feature), class_mean(c,feature),
feature_covar(feature,feature,c));end
    %Implements Product Rule via Chain Rule
    maximum_ likelihood(c) = maximum_ likelihood1(c,1)*maximum_ likelihood1(c,2); end%%
```

A2.4

Algorithm 2.4

Plot Contours of MLE for single Class

```
%% plot Contours of MLE p(W_n|X1,X2) iso-probability contour ellipses for each class
figure
for i=3   %%Index of class
    for c = 1:3
   position = find(y==labels(c));
plot(X(position,1),X(position,2),markers{c},'markersize',5,'linewidth',1,'markerfacecolor',color{
c});hold on;end
%X1 is feat 1 X2 is feat 2,probability_density is 3rd variable that describes elevation of graph
    contour(X1,X2,probability_density(:,:,i));
    set(gca,'Title',text('String','Data with Class-conditional Iso-probability
Lines','FontAngle', 'italic', 'FontWeight', 'bold'),'xlabel',text('String', '$\mathbf{X_1}$',
'Interpreter', 'latex'),'ylabel',text('String', '$\mathbf{X_2}$', 'Interpreter', 'latex'))
    legend('class 1','class 2','class 3')
end%%
```

A4.1

Algorithm 4.1

Plot Features X1, X2, X3 and find the Shaded Areas of all three classes ω1, ω2 and ω3

```
%% Feature X1 : 1D Gaussian Normal Distribution
format long
X11 = linspace(min(X1(1,1:end)), max(X1(1,1:end))); % Feature X1
norm11 = normpdf(X11,class_mean(1,1),feature_covar(1,1,1));
norm21 = normpdf(X11,class_mean(2,1),feature_covar(1,1,2));
norm31 = normpdf(X11,class_mean(3,1),feature_covar(1,1,3));
figure ;hold on;
plot(X11,norm11,'r')
plot(X11,norm21,'b')
plot(X11,norm31,'g')
xs=X11(X11>3.93939385 & X11<6); %Range of Area under Blue curved Gaussian Distribution
xs1=X11(X11>1 & X11<3.93939385); %Range of Area under Red curved Gaussian Distribution
xs2=X11(X11>5.149 & X11<9); %Range of Area under Green curved Gaussian Distribution
```

```
xs3=X11(X11>1 & X11<5.149); %Range of Area under Red curved Gaussian Distribution
figure;hold on;
x_total = X11(X11>0 & X11<10);
Area11 = area(xs,normpdf(xs,class_mean(1,1),feature_covar(1,1,1))) ;
Area21 = area(xs1,normpdf(xs1,class_mean(3,1),feature_covar(1,1,3))) ;
Area31 = area(xs2,normpdf(xs2,class_mean(3,1),feature_covar(1,1,3))) ;
Area41 = area(xs3,normpdf(xs3,class_mean(2,1),feature_covar(1,1,2))) ;
plot(X11,normpdf(X11,class_mean(1,1),feature_covar(1,1,1)),'r')
plot(X11,normpdf(X11,class_mean(3,1),feature_covar(1,1,3)),'b')
plot(X11,normpdf(X11,class_mean(2,1),feature_covar(1,1,2)),'g')
%% Feature X2 : 1D Gaussian Normal Distribution for X2
X22 = linspace(min(X2(1:end,1)), max(X2(1:end,1))); % Feature X2
norm12 = normpdf(X22,class_mean(1,2),feature_covar(2,2,1));
norm22 = normpdf(X22,class_mean(2,2),feature_covar(2,2,2));
norm32 = normpdf(X22,class_mean(3,2),feature_covar(2,2,3));
figure ;hold on;
plot(X22,norm12,'b')
plot(X22,norm22,'g')
plot(X22,norm32,'r')
xr1=X22(X22>4.0 & X22<8); %Range of Area under Blue curved Gaussian Distribution
xr=X22(X22>0 & X22<4.0); %Range of Area under Red curved Gaussian Distribution
xr2=X22(X22>6.0599989999 & X22<9); %Range of Area under Blue curved Gaussian Distr
xr3=X22(X22>1 & X22<6.0599989999); %Range of Area under Blue curved Gaussian Distr
xr4=X22(X22>6.0599989999 & X22<10);  %Range of Area under Green curved Gaussian Distr
figure;hold on;
Area12 = area(xr,normpdf(xr,class_mean(1,2),feature_covar(2,2,1))) ;
Area22 = area(xr1,normpdf(xr1,class_mean(3,2),feature_covar(2,2,3))) ;
Area32 = area(xr2,normpdf(xr2,class_mean(3,2),feature_covar(2,2,3))) ;
Area42 = area(xr3,normpdf(xr3,class_mean(2,2),feature_covar(2,2,2))) ;
Area52 = area(xr4,normpdf(xr4,class_mean(1,2),feature_covar(2,2,1))) ;
plot(X22,normpdf(X22,class_mean(1,2),feature_covar(2,2,1)),'b')
plot(X22,normpdf(X22,class_mean(3,2),feature_covar(2,2,3)),'r')
plot(X22,normpdf(X22,class_mean(2,2),feature_covar(2,2,2)),'g')
```

V. REFERENCES

[1] "Pattern Classification by Richard O. Duda, David G. Stork, Peter E.Hart .pdf." .