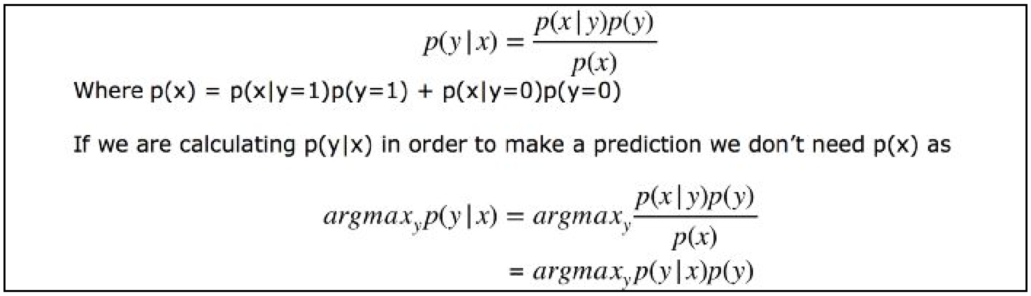
Report on Performance of GDA

IIT2018179 - Mohammed Aadil

1. Introduction:

Algorithms that model p(y|x) directly from the training set are called **discriminative algorithms**. In GDA we try to model p(x|y) and p(y), it's called **Generative Learning Algorithms**. Once we learn the model p(y) and p(x|y) using training set, we use Bayes Rule to derive the p(y|x) as



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

Where p(x) = p(x|y=1)p(y=1) + p(x|y=0)p(y=0)

If we are calculating p(y|x) in order to make a prediction we don't need p(x) as

$$\begin{aligned} argmax_y p(y\,|\,x) &= argmax_y \frac{p(x\,|\,y)p(y)}{p(x)} \\ &= argmax_y p(y\,|\,x)p(y) \end{aligned}$$

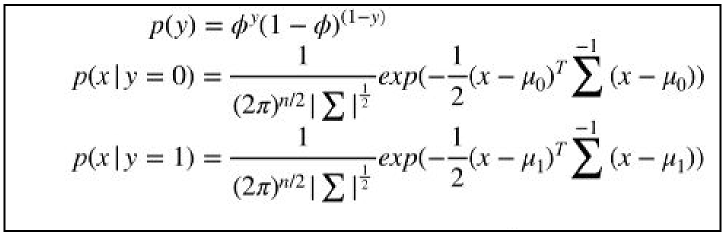
[2]

2. Gaussian Discriminant Analysis model:

When we have a classification problem in which the input features are continuous random variable, we can use GDA, we assume p(x|y) is distributed according to a multivariate normal distribution and p(y) is distributed according to Bernoulli.

P(y) ~ Bernoulli(φ)
P(x|y = 0) ~ N (μ0, Σ)
P(x|y = 1) ~ N (μ1, Σ)

Therefore the distribution is as follows.



$$p(y) = \phi^y(1-\phi)^{(1-y)}$$

$$p(x\,|\,y=0) = \frac{1}{(2\pi)^{n/2}|\sum|^{\frac{1}{2}}} exp(-\frac{1}{2}(x-\mu_0)^T \sum^{-1} (x-\mu_0))$$

$$p(x\,|\,y=1) = \frac{1}{(2\pi)^{n/2}|\sum|^{\frac{1}{2}}} exp(-\frac{1}{2}(x-\mu_1)^T \sum^{-1} (x-\mu_1))$$

[2]

3. Parameters of GDA:

There are Quite a few parameters in this algorithm, φ, Σ, μ0 and μ1.

$$\phi = \frac{1}{m}\sum_{i=1}^{m} 1\{y^{(i)} = 1\}$$

$$\mu_0 = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)} = 0\}}$$

$$\mu_1 = \frac{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}x^{(i)}}{\sum_{i=1}^{m} 1\{y^{(i)} = 1\}}$$

$$\sum = \frac{1}{m}\sum_{i=1}^{m} (x^{(i)} - \mu_k)(x^{(i)} - \mu_k)^T \text{ where } k = 1\{y^{(i)} = 1\}$$

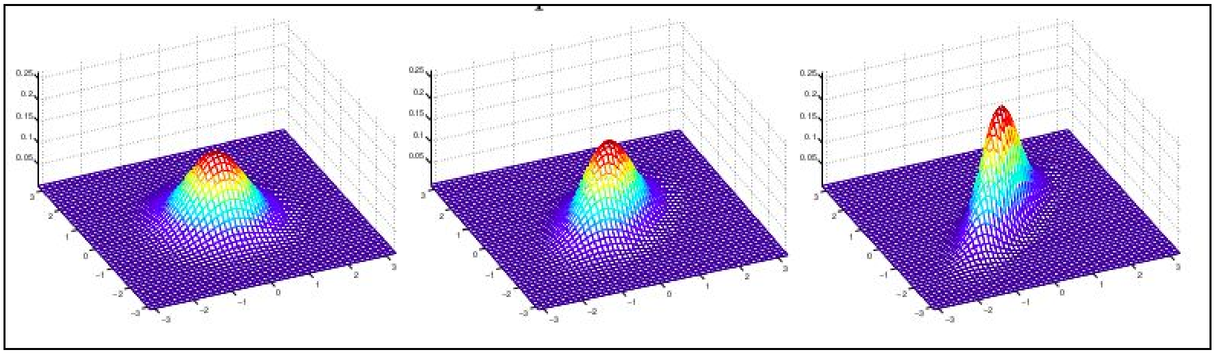
[2]

Here 1{} is the indicator function.
Notice that we have 2 Sigma values but we have considered only one of them, this is common as in practice 1 sigma value is enough.

4. Effect of Parameters on the Region:

If we change the value of sigma we see the following change.

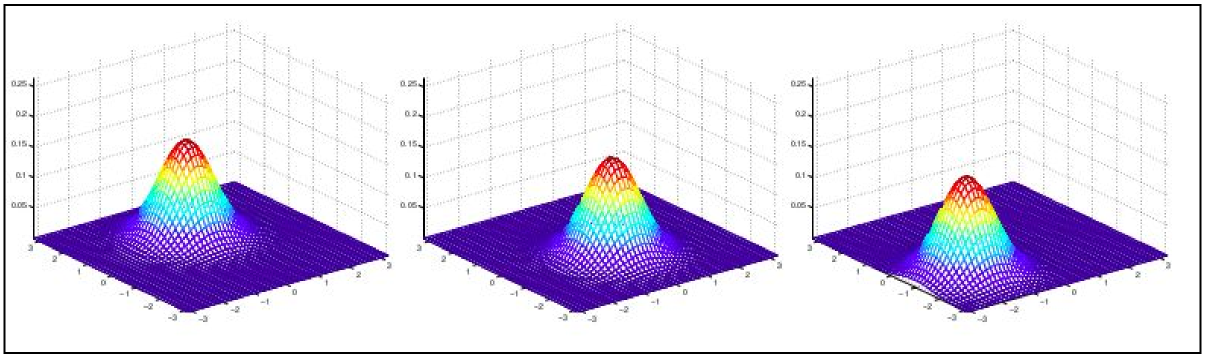
$$\Sigma = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & 0.5 \\ 0.5 & 1 \end{bmatrix}; \quad \Sigma = \begin{bmatrix} 1 & 0.8 \\ 0.8 & 1 \end{bmatrix}.$$



[3]

Similarly if we play around with the μ (mean) values,

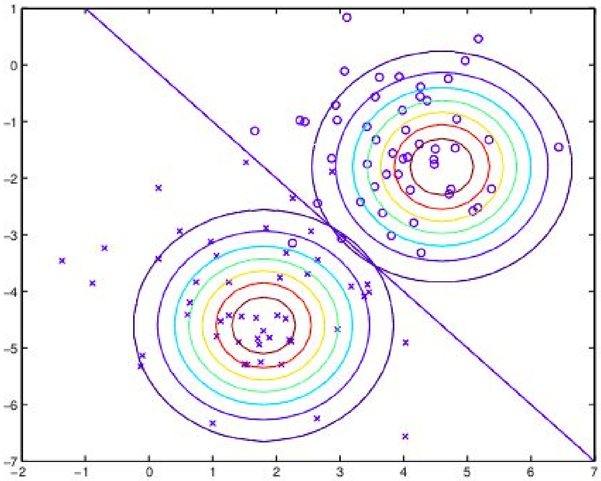
$$\mu = \begin{bmatrix} 1 \\ 0 \end{bmatrix}; \;\; \mu = \begin{bmatrix} \text{-}0.5 \\ 0 \end{bmatrix}; \;\; \mu = \begin{bmatrix} \text{-}1 \\ \text{-}1.5 \end{bmatrix}.$$



[3]

5. Working of the Model:

Shown in the figure are the training set, along with the contours of the two Gaussian distributions that have been fit to the data in each of the two classes.

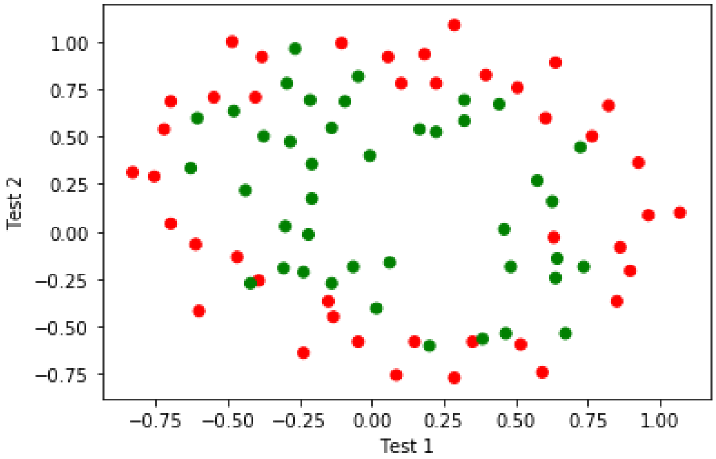


[3]

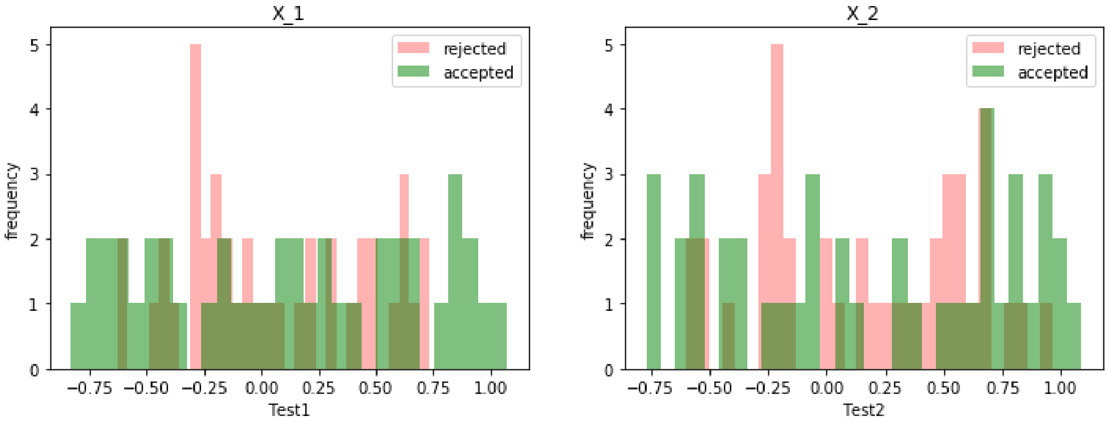
We see that the two Gaussians have contours that are the same shape and orientation, since they share a covariance matrix Σ, but they have different means μ0 and μ1 . The straight line giving the decision boundary at which p(y=1|x) = 0.5. On one side of the boundary, we'll predict y = 1 to be the most likely outcome, and on the other side, we'll predict y = 0.

6. GDA on Microchip Data without Transformation:

As we can see the data is distributed like so, Obviously GDA will not perform perfectly but just for argument sakes let's give it a go.



The Histogram of the features shows clearly that the data is not Gaussian.



Once we train the model using the data as it is we see a very disappointing result, but nothing surprising.

Accuracy = 48.57 %

From the scatter plot it's clear that the Algorithm is not able to create contours that fit the data properly. Lets try another approach.

7. Box - Muller Transformation:

It’s a transformation which transforms from a two-dimensional continuous uniform distribution to a two-dimensional normal distribution. If x_1 and x_2 are uniformly and independently distributed between 0 and 1, then z_0 and z_1 as defined below have a normal distribution with mean $\mu = 0$ and variance $\sigma^2 = 1$.

$$Z_0 = R\cos(\Theta) = \sqrt{-2\ln U_1}\cos(2\pi U_2)$$

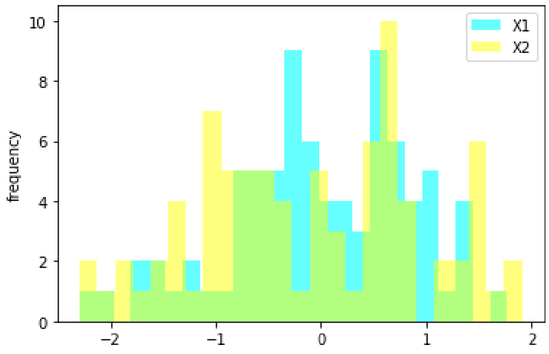
$$Z_1 = R\sin(\Theta) = \sqrt{-2\ln U_1}\sin(2\pi U_2).$$ [1]

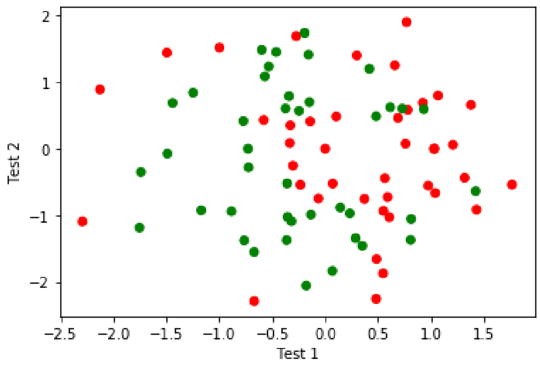
8. GDA using Box-Muller Transformation:

Before we can use the Box-Muller transformation on the data we need to scale it between 0 to 1. To do this I used a simple min-max normalization.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

After this we can use Box-Muller transformation, the result of the transformation on the scaled data is as follows:



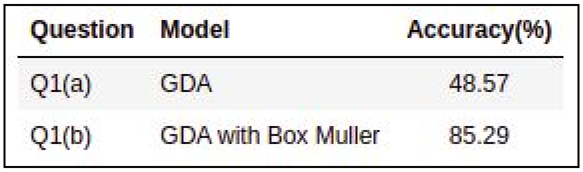


Now we can clearly see the two classes. And our GDA algorithm will be able to classify the data easily and more accurately. After training the data we see that the accuracy has almost doubled.

Accuracy = 85.289 %

9. Results and Conclusion:

The results speak for themselves, the shear domination that is caused by just using the correct algorithm along with the right transformation.



Question	Model	Accuracy(%)
Q1(a)	GDA	48.57
Q1(b)	GDA with Box Muller	85.29

Anyone can build a model that uses a lot of data and then comes up with a very high accuracy rate. But if you have a very limited amount of data then the one that knows how to utilize the way the data is distributed will be able to build a far superior model than the novice.

This is that kind of Algorithm that you don't see used that often, I personally prefer Logistic regression. As this has a really high error rate if used incorrectly.

10. References:

[1] Box-Muller

[2] GDA

[3] Andrew-Ng-GDA

[4] GDA-Code-help