AFRICAN INSTITUTE FOR MATHEMATICAL SCIENCES

(AIMS-GH, ACCRA)

Name: Ijuptil Joseph Kwajighu Assignment Number: 1

Course: Machine Learning and Pattern Recognition Date: May 1, 2021

Exercise 1

(a) Nelder-Mead Simplex: Using the difference of 0.3 for each value in x_0 such that the new x_0 is $[1.0, 0.4, 0.5, 1.6, 0.9]$. It is observed that the number of iterations increase from 339 to 445 and the function evaluation increases from 57 to 730. This shows higher values of x_0 implies less iteration and less function evaluation, while lower values of x_0 implies more iteration and higher function evaluation. The advantage of Nelder-Mead technique is that is simpler to implement.Additionally, it requires more time as well as function calls.

(b) Broyden-Fletcher-Goldfarb-Shanno algorithm(BGFS):
In Broyden-Fletcher-Goldfarb-Shanno algorithm, the solution converges fast than the Nelder-Mead simplest algorithm. Using the same x_0 as in the case of Nelder-Mead Simplex scheme, the number of iterations and the function evaluation reduces drastically to 25 and 30 respectively.

The method uses the gradient(in our case gradient evaluation is 30) of the objective function to compute the solution. The gradient is specified in the minimize function through the jac parameter. In addition, the BFGS method requires fewer function calls than the simplex algorithm.

In conclusion, the Nelder-Mead simplest algorithm is the easier way to minimize a function. But it requires more iterations and more function calls. This is because it does not use gradient evaluations. While the BGFS technique takes smaller time and less function evaluation because of the gradient evaluation.

Exercise 2

The prior probabilities are: $P_1 = 0.4$, $P_1 = 0.3$ and $P_1 = 0.3$ with mean $\mu_1 = 4$, $\mu_2 = 6$ and $\mu_3 = 8$. The figures below depict the class conditional probability, evidence, prior probabilities and posteriors of the 3 classes.

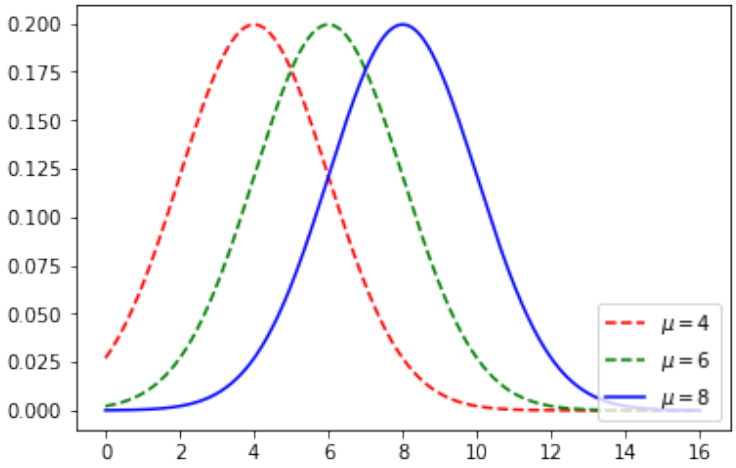

Figure 1: Class Conditional Densities

The Figure 1 depicts the class conditional densities for different mean with uniform standard deviation. The curves shift to the right with increase in the mean value. The probability for each curve (at 20% each) is as a result of the uniform standard deviation.

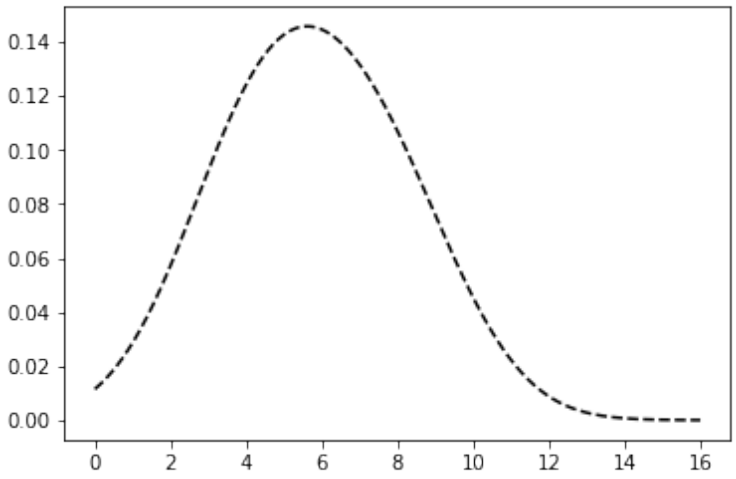

Figure 2: Evidence

Figure 2 shows the graph of the normalization factor for the mean of 4. The normalization factor is usually a constant. The evidences for the 3 classes are all constant.

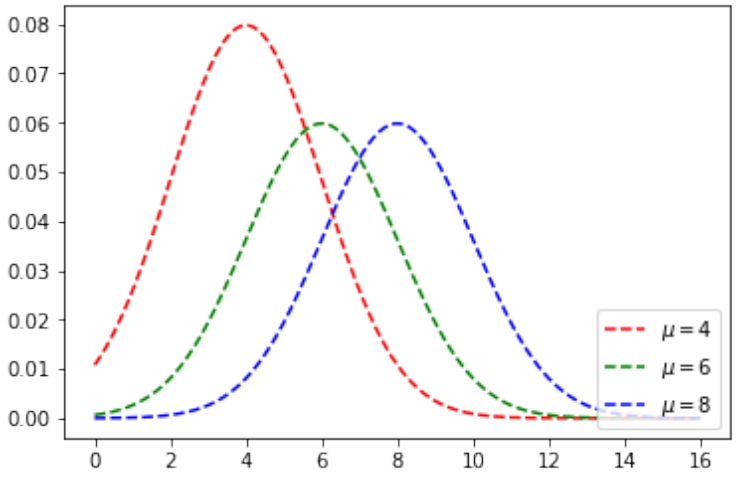

Figure 3: Prior Distributions

The Figure 3 depicts the prior probabilities of the 3 classes. The peak value for the red curve occur around the value of 0.08, which is the best guess prior probability. The fact that the green and blue with peak around 0.06 spread out and has a smaller peak less than the red curve means that prior probabilities expressed by the 2 curves are "less certain" about the true value than the curve with the highest peak. The shift in the curve is as a result of the variation in the mean values of the prior probabilities. Also, highly informative priors also lead to a smaller variance of the posterior distribution

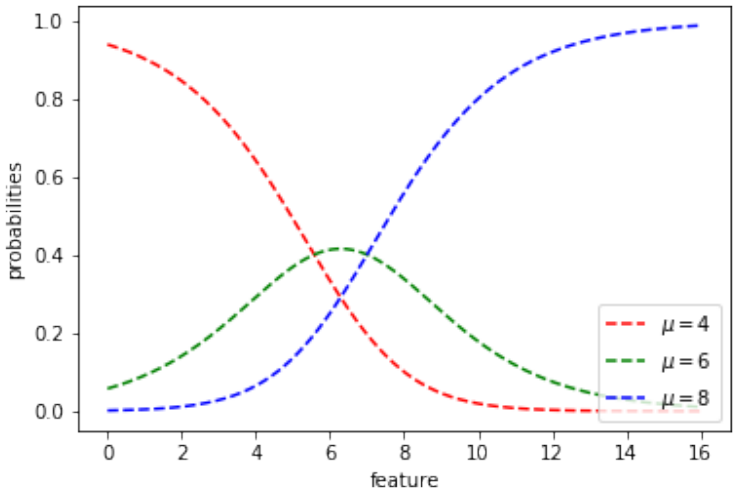

Figure 4: Posterior Distributions

The Figure 4 depicts the posterior distributions for the 3 different mean values. The blue curve correspond to the mean value of 8, while the red curve corresponds to the mean value of 4. This shows that the posterior distribution is highly driven by data. In essence, high data means high posterior. Moreover, a lot of data will dominate the posterior distribution. In practice, we can obtain precise estimate of the posterior distribution with small samples size with more uniform priors.

Exercise 3

The images were changed to Albert and Universe respectively. I also changed the filter **EDGES** to **EDGE ENHANCE.** This filter work by increasing the contrast of the pixels around the specific edges. The edges appear more visible after applying the filter. The code can be seen below:

```
from PIL import Image
from PIL import ImageFilter

def filter(image_file):
im = Image.open(image_file)
im.show()
im = im.filter(ImageFilter.EDGE_ENHANCE)
im.save(image_file + "edges.jpg")

# Opens the images
im1 = "Albert.jpeg"
im2 = "Universe.jpeg"
filter(im1)
filter(im2)
```

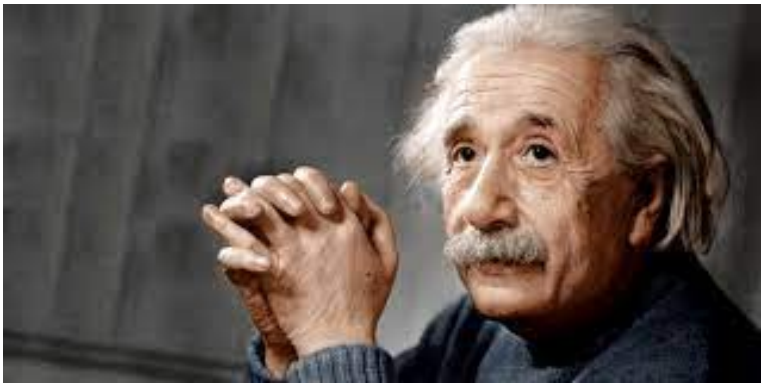

Figure 5: Normal Image

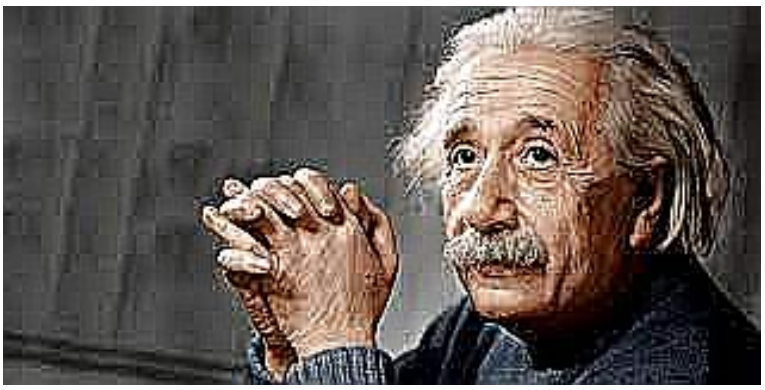

Figure 6: Filtered Image

Exercise 4

Consider a new mean $(\mu_1,\ \mu_2) = (-1, 1)$ with a new covariance matrix $[[1, 0], [0, 200]]$ with sample size of 1000. In this case, the covariance matrix is symmetric and positive semi-definite with the variances at the main diagonal as expected. But when the covariance matrix is changed to $[[1, 0], [2, 200]]$, python throws an error message. This is because the covariance matrix is non-positive semi-definite. Non-symmetric would mean that one or more of the eigenvalues are negatives. It would also mean that there are more dimensions than the data points.