BM20BTECH11001-Lab12

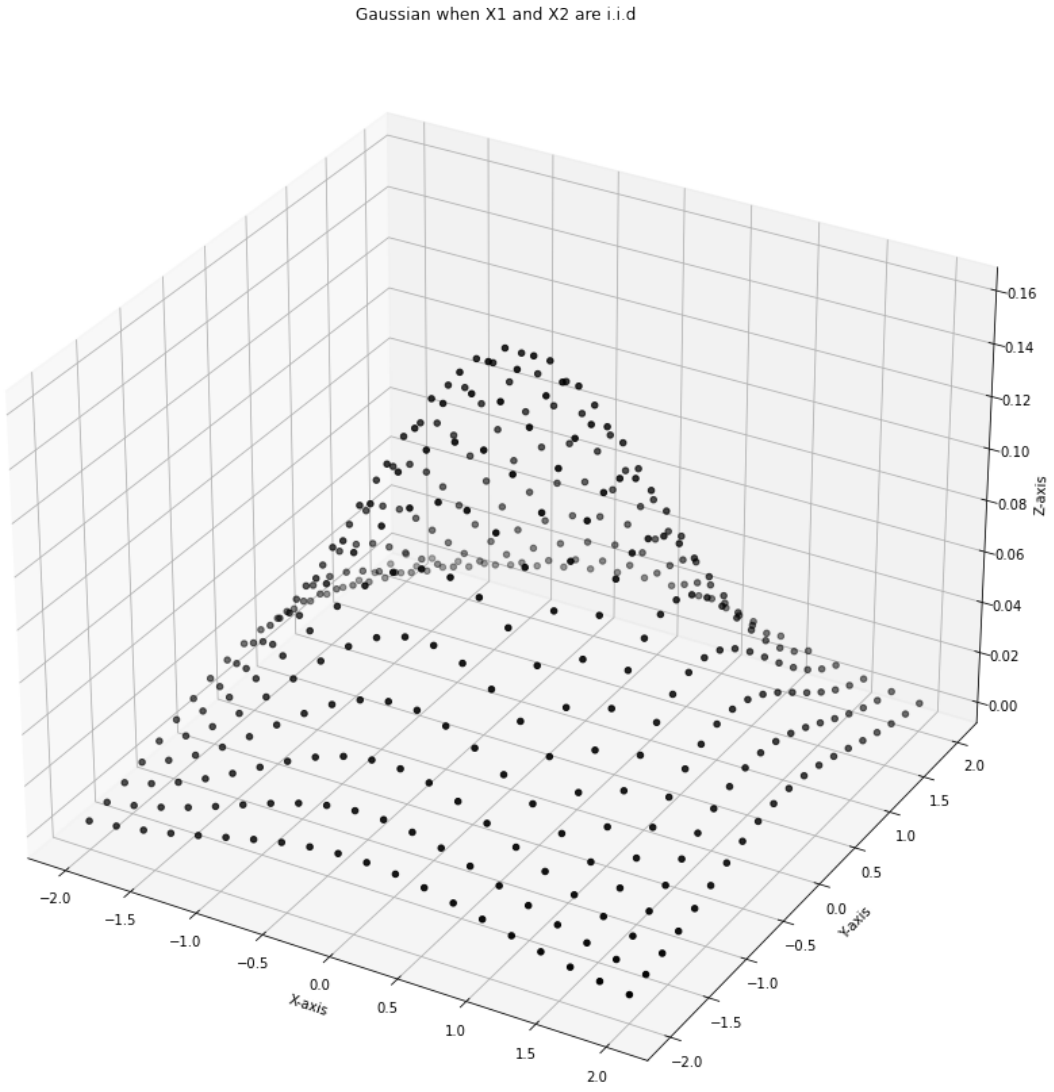
November 17, 2021

[1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
```

0.1 When X1 and X2 are independent identical distributions

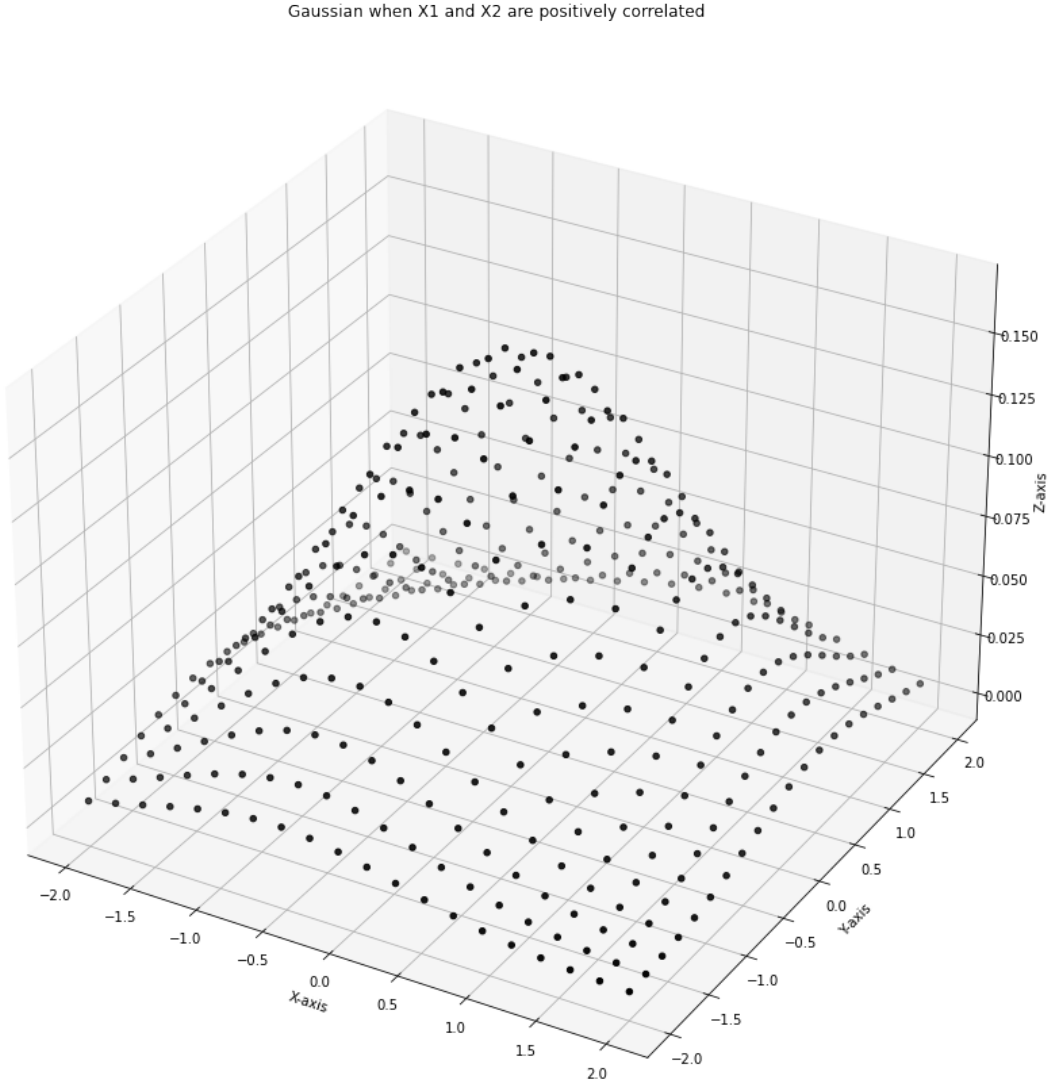
[2]:
```
#Covariance matrix
matrix = [[1, 0], [0, 1]]

mean = 0

#Y is column vector
Y = lambda x,y: np.transpose([[x-mean, y-mean]])

#Inverse of covariance matrix
Inv = np.linalg.inv(matrix)
f_z = []


x = np.linspace(-2, 2, 20)
y = np.linspace(-2, 2, 20)
x_1 = []
y_1 = []
for d in x:
    for e in y:
        x_1.append(d)
        y_1.append(e)
        t = np.dot(np.transpose(Y(d,e)), Inv)
        a = np.dot(t, Y(d,e))
        gauss = ((np.e)**(-a[0][0]/2))/(2*np.pi)
        f_z.append(gauss)
fig = plt.figure(figsize=(15,15))
ax = plt.axes(projection='3d')
ax.scatter(x_1, y_1, f_z, color='black')
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
```

```
ax.set_zlabel('Z-axis')
ax.set_title('Gaussian when X1 and X2 are i.i.d');
```



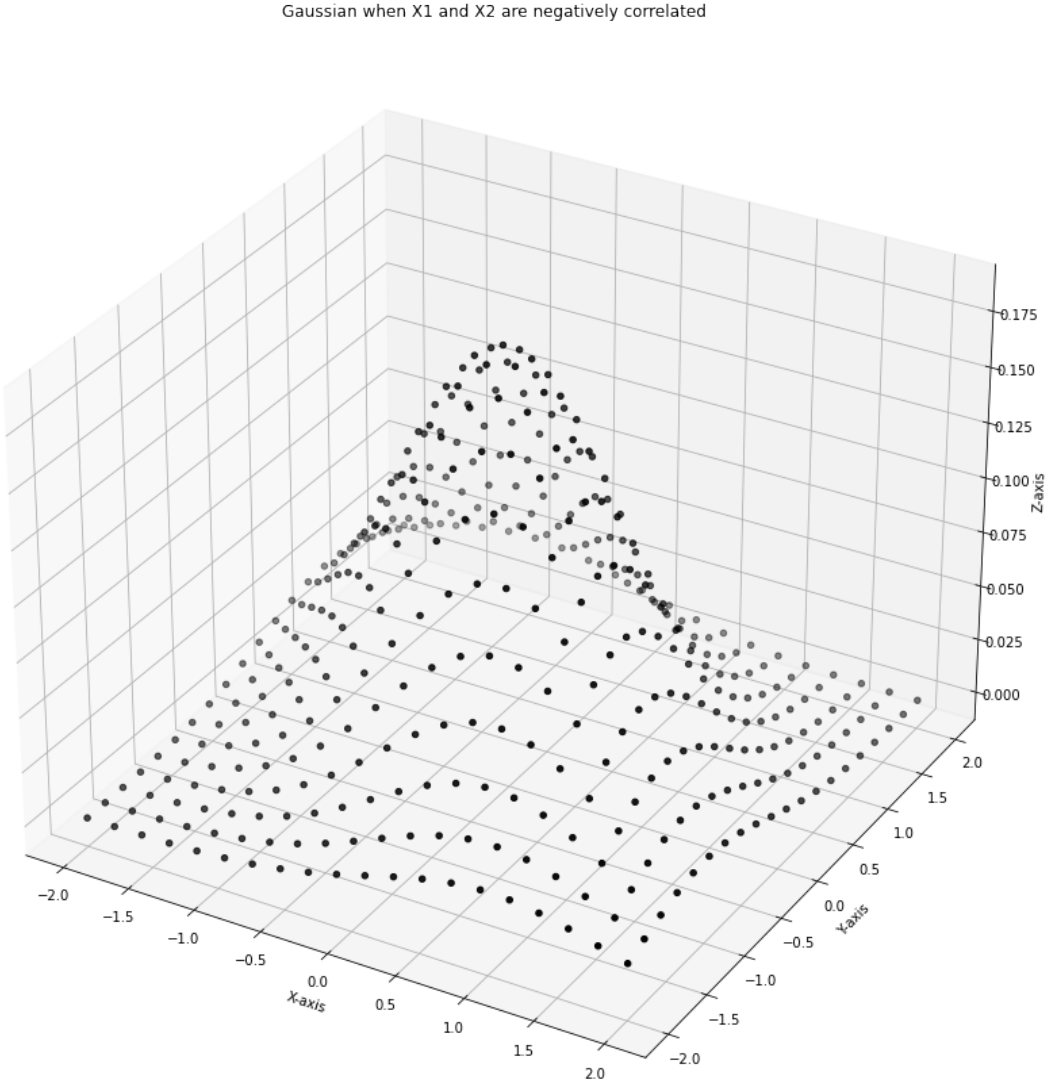
[3]:
```
#Covariance matrix
matrix = [[1, 0.3], [0.3, 1]]

mean = 0

#Y is column vector
Y = lambda x,y: np.transpose([[x-mean, y-mean]])

#Inverse of covariance matrix
```

```
Inv = np.linalg.inv(matrix)
f_z = []


x = np.linspace(-2, 2, 20)
y = np.linspace(-2, 2, 20)
x_1 = []
y_1 = []
for d in x:
    for e in y:
        x_1.append(d)
        y_1.append(e)
        t = np.dot(np.transpose(Y(d,e)), Inv)
        a = np.dot(t, Y(d,e))
        gauss = ((np.e)**(-a[0][0]/2))/(2*np.pi*(np.sqrt(0.91)))
        f_z.append(gauss)
fig = plt.figure(figsize=(15,15))
ax = plt.axes(projection='3d')
ax.scatter(x_1, y_1, f_z, color='black')
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')
ax.set_title('Gaussian when X1 and X2 are positively correlated');
```



[4]:
```
#Covariance matrix
matrix = [[1, -0.5], [-0.5, 1]]

mean = 0

#Y is column vector
Y = lambda x,y: np.transpose([[x-mean, y-mean]])

#Inverse of covariance matrix
Inv = np.linalg.inv(matrix)

f_z = []
```

```
x = np.linspace(-2, 2, 20)
y = np.linspace(-2, 2, 20)
x_1 = []
y_1 = []
for d in x:
    for e in y:
        x_1.append(d)
        y_1.append(e)
        t = np.dot(np.transpose(Y(d,e)), Inv)
        a = np.dot(t, Y(d,e))
        gauss = ((np.e)**(-a[0][0]/2))/(2*np.pi*(np.sqrt(0.75)))
        f_z.append(gauss)
fig = plt.figure(figsize=(15,15))
ax = plt.axes(projection='3d')
ax.scatter(x_1, y_1, f_z, color='black')
ax.set_xlabel('X-axis')
ax.set_ylabel('Y-axis')
ax.set_zlabel('Z-axis')
ax.set_title('Gaussian when X1 and X2 are negatively correlated');
```


Gaussian when X1 and X2 are negatively correlated
0.175
0.150
0.125
0.100
0.075
0.050
0.025
0.000
Z-axis
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0
X-axis
Y-axis