Deep Learning (CSCE-598)
Assignment: Rotate a Bivariate Gaussian

Razin Farhan Hussain
C00303945

September 2019

1 Step: 1

The plotGauss.py code plot a contour map of a Gaussian distribution with covariance matrix.

$$\begin{vmatrix} 1 & 0 \\ 0 & 6 \end{vmatrix}$$

The plot of the contour is given in below figure.

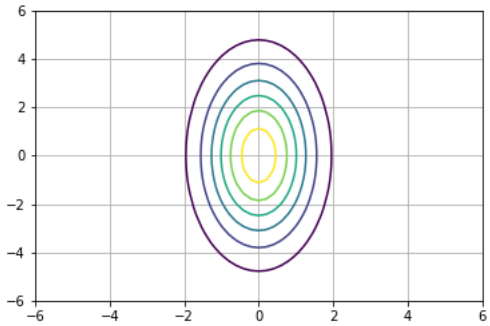

Figure 1: Contour plot for unrotated Gaussian distribution.

2 Step: 2

The matrix is diagonal that is why the contours are aligned with the axes in Figure 1. The figure is elliptical and the long axis is defined by the second eigenvalue in that is oriented in y axis. The contours in the figure indicate regions of constant probability.

3 Step: 3

We have to replace the covariance matrix so that the long axis of the ellipse is oriented 30 degrees to the x-axis. Using the eigendecomposition formula, we can get the rotated covariance matrix. At first we have to choose Λ with 30 degree rotation matrix and proper eigenvecs. This is actually matrix multiplication of 30 degree rotation matrix and original eigenvecs as we don't want to change the shape of the ellipse. Then we compute the inverse of V to use in eigendecomposition formula. Finally, we choose the diagonal Λ matrix as original covariance matrix.

$$\Sigma_{rotated} = V\Lambda V^{-1}$$

For finding the $\Sigma_{rotated}$, I used the below code snippet.

```
# covar matrix must be positive definite, or
# symmetric with an inverse for Gaussian distribution
covarMat = np.array([[1.0, 0.0],[0.0, 6.0]]) # 2x2 array
EigVecMat = np.array([[1.0, 0.0],[0.0, 1.0]])
#matrix multiplication with eigendecomposition.
#Step1: 30 degree rotation matrix multiplication with original covar matrix (eigen matrix) to get V
#rotatMat = np.array([[0.87,-0.5],[0.5,0.87]])
angle = -30
angle = math.radians(angle)

rotatMat = np.array([[math.cos(angle),-math.sin(angle)],[math.sin(angle),math.cos(angle)]])
vMat = np.matmul(rotatMat,covarMat)
#Step2: Computing  V inverse
vInv = np.linalg.inv(vMat)
#Step3: multiply V with covar matrix with same eigen values as we don't want to change the shape
temMat = np.matmul(vMat,covarMat)
#Step4: multiply resultant matrix with V inverse matrix to get final cover Matrix
finalMat = np.matmul(temMat,vInv)
```

Figure 2: Final rotated matrix calculation with eigendecomposition

4 Step: 4 & 5

I used the mean $\mu = [12]^T$ which position the center of the distribution at coordinates (1,2). Figure 4 represents the non-zero mean distribution where center is located at (1,2) position. Figure 3 represents the code snippet for changing the mean to (1,2).

5 Step: 6 & 7

In this step, I have generated 100 data points with $\mu = [12]^T$ and covariance $\Sigma_{rotated}$. I used the function $np.random.multivariate_normal$ to generate the

```
muMat = np.array([[1.0], [2.0]])

def gaussValue(x) : # x is 2-element column vec
    return np.exp(-0.5*np.dot(np.dot((x-muMat).T,
                                     covarInv),
                              (x-muMat)))
```

Figure 3: Code snippet for changing the mean to (1,2))

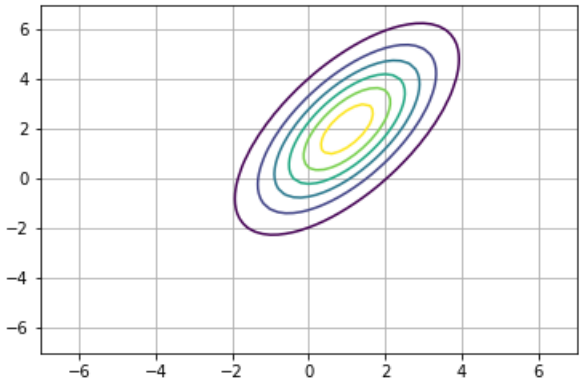

Figure 4: Rotated ellipse with center at position (1,2)

data points. Then I plot the data points and re-plot the contour map from previous section to check the consistency of data points with the distribution that they were sampled from. Figure 5 demonstrate the result of plotting the data points which represents the consistency of data points with the distribution.

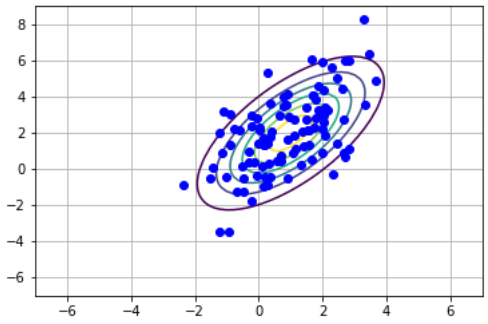

Figure 5: Data points plotted from the rotated distribution