1. Order of faces using ISOMAP (50 points)

The objective of this question is to reproduce the ISOMAP algorithm results that we have seen discussed in lecture as an excercise. The file isomap.mat (or isomap.dat) contains 698 images, corresponding to different poses of the same face. Each image is given as a 64×64 luminosity map, hence represented as a vector in $\mathbb{R}^{4096}$. This vector is stored as a row in the file. [This is one of the datasets used in the original paper for ISOMAP, J.B. Tenenbaum, V. de Silva, and J.C. Langford, Science 290 (2000) 2319-2323.]

(a) (20 points) Choose the Euclidean distance between images (i.e., in this case a distance in $\mathbb{R}^{4096}$). Construct a similarity graph with vertices corresponding to the images, and tune the threshold ϵ so that each node has at least 100 neighbors. Visualize the similarity graph (e.g., plot the adjacency matrix, or visualize the graph and illustrate a few images corresponds to nodes at different parts of the graph; you can be a bit creative here).

Answer:

Step 1: Find threshold epsilon using distance matrix from the function from sklearn package.

```
for i in range(99,102):
    distance_matrix = kneighbors_graph(data, n_neighbors = i, mode='distance', p=2, include_self=False)
    print("When n_neighbors is", i, ", largest distance is", np.amax(distance_matrix.toarray()))
```

```
When n_neighbors is 99 , largest distance is 22.370116979246244
When n_neighbors is 100 , largest distance is 22.388619203990135
When n_neighbors is 101 , largest distance is 22.414735265879724
```

So threshold epsilon should be smaller than 22.414735265879724 and equal or larger than 22.388619203990135.

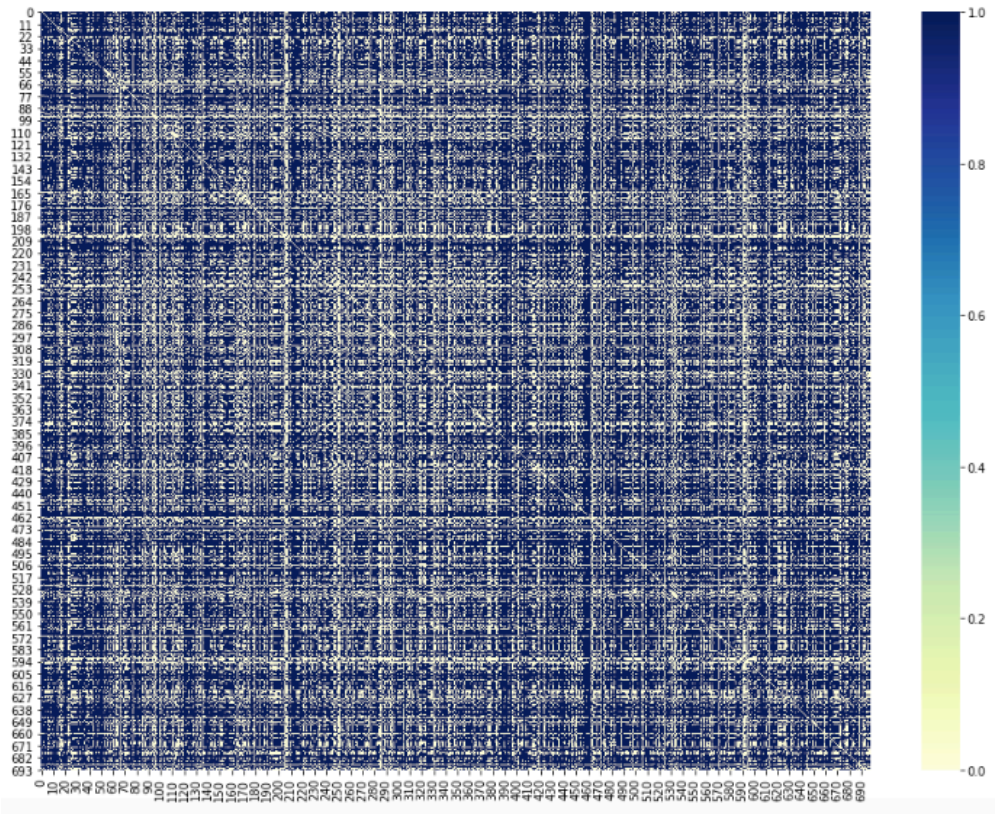
Step 2: Build adjacency matrix from a distance matrix.

There are 698 images. Setting the n_neighbors to 697, neighbors.kneighbors_graph function will give the distance matrix that contains all pairwise Euclidean distance.

Set epsilon as 22.41. When a distance is over 22.41, the distance will be set as 0 meaning no neighbors between image i and j.

Step 3: Visualize the 0/1 version of adjacency matrix.

Build the 0/1 version the adjacency matrix by setting distance over 0 as 1. In this 0/1 version the adjacency matrix, 0 means the non-existence of neighbor edges and 1 means the existence of neighbor edges.



Step 4: A sanity check to confirm that there are at least 100 neighbors (ones) in each row.

```
Adjacency_matrix_0_1_version:

array([[0., 1., 1., ..., 1., 0., 1.],
       [1., 0., 1., ..., 1., 1., 1.],
       [1., 1., 0., ..., 0., 0., 1.],
       ...,
       [1., 1., 0., ..., 0., 0., 1.],
       [0., 1., 0., ..., 0., 0., 1.],
       [1., 1., 1., ..., 1., 1., 0.]])

Sum ones in each row of adjacency_matrix_0_1_version, the smallet sum is:
100.0
```

(b) (20 points) Implement the ISOMAP algorithm and apply it to this graph to obtain a $d = 2$-dimensional embedding. Present a plot of this embedding. Find three points that are "close" to each other in the embedding space, and show what they look like. Do you see any visual similarity among them?

Answer:

Step 1: To compute the shortest path, set zeros in adjacent matrix to a large number (10000 in this case).

Step 2: Follow Floyd's algorithm, get the shortest path matrix D.
Do a sanity check, the longest distance in D is 49, much smaller than 10000.

Step 3: Follow ISOMAP algorithm, obtain matrix H and C.

Step 3: use a centering matrix $H = I - \frac{1}{m}11^{\top}$ to get

$$C = -\frac{1}{2m}H(D)^2H$$

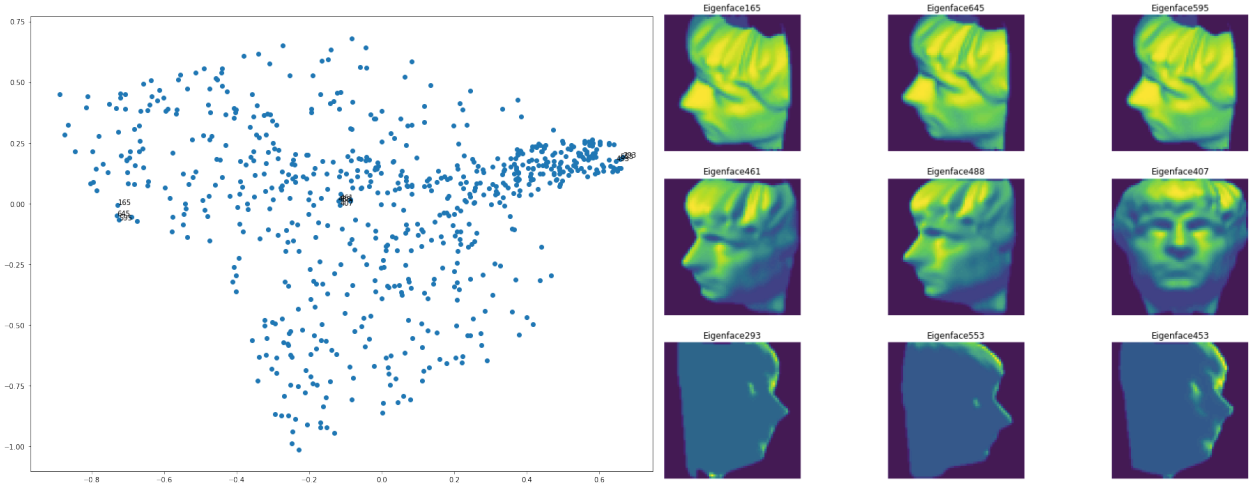
Step 4: Follow ISOMAP algorithm, obtain the biggest two eigenvalues of C, and corresponding eigenvectors.

Dimension 1 is eigenvector_1 times square root of eigenvalue_1. Dimension 2 is eigenvector_2 times square root of eigenvalue_2.

Step 4: compute leading eigenvectors $w^1, w^2, \ldots$ and eigenvalues $\lambda_1, \lambda_2 \ldots$. of C

$$Z^T = (w^1, w^2 \ldots)\begin{pmatrix} \lambda_1^{1/2} & & \\ & \lambda_2^{1/2} & \\ & & \ddots \end{pmatrix}$$

Step 5: Present the embedding plot and pictures. (k = 100)



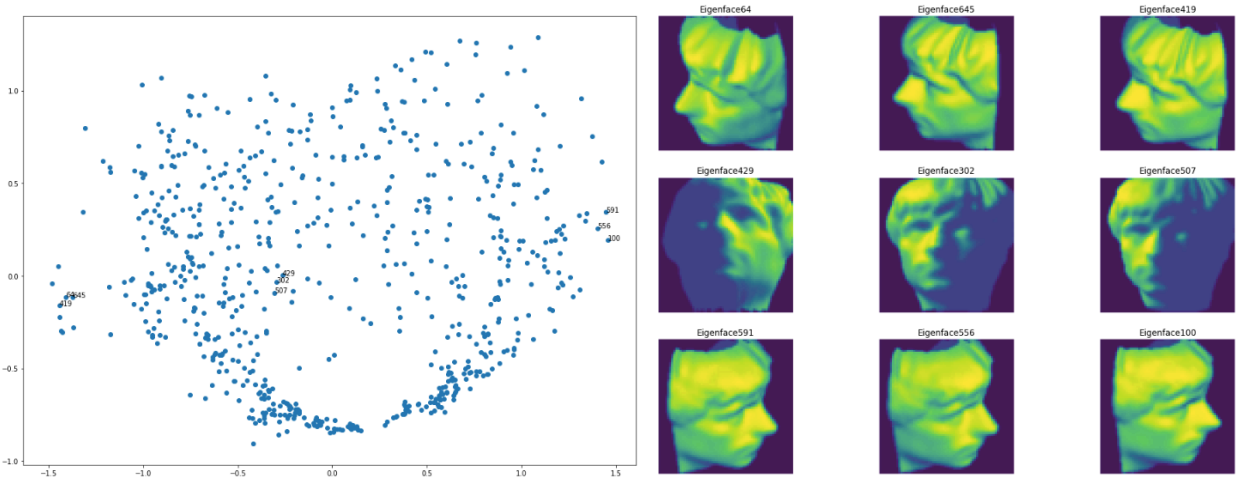
Picture: In the left picture the left three points are eigenface 165, 645, 595. In the left picture the middle three points are eigenface 461, 488, 407. In the left picture the right three points are eigenface 293, 553, 453.

All three sets of pictures showed similarities. Eigenface 165, 645, 595 are all facing left down. Eigenface 461, 488, 407 are all facing left (407 is more like an outlier). Eigenface 293, 553, 453 are all facing right.

The ISOMAP algorithm showed the ability of nonlinear dimensionality reduction. However, eigenface 407 is more like an outlier, and eigenface 461, 488 are not facing center. So I decide to change K from 100 to 6 to see any improvement.

Step 6: Present the embedding plot and pictures. (k = 6)

According to the plot, when the number of neighbors is 6, ISOMAP algorithm showed a better nonlinear dimensionality reduction ability to group similar faces together.



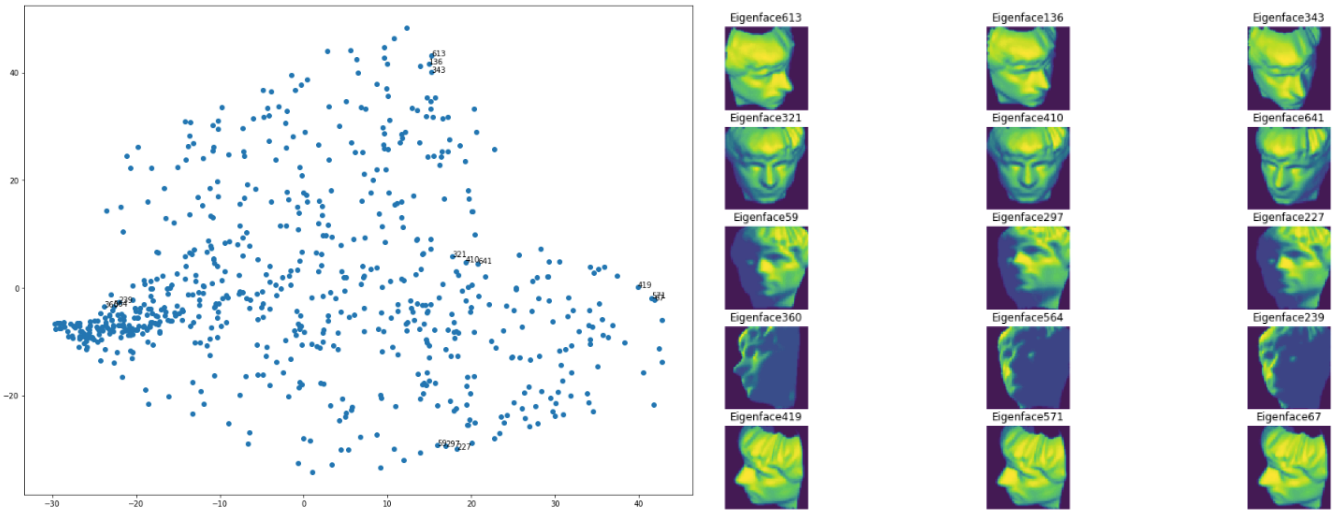
Picture: In the left picture the left three points are eigenface 64, 645, 419. In the left picture the middle three points are eigenface 429, 302, 507. In the left picture the right three points are eigenface 591, 556, 100.

(c) (10 points) Now choose ℓ_1 distance (or Manhattan distance) between images (recall the definition from "Clustering" lecture)). Repeat the steps above. Again construct a similarity graph with vertices corresponding to the images, and tune the threshold ϵ so that each node has at least 100 neighbors. Implement the ISOMAP algorithm and apply it to this graph to obtain a $d = 2$-dimensional embedding. Present a plot of this embedding. Do you see any difference by choosing a different similarity measure?

Answer:

Follow the same procedure (tune the threshold, k = 100, implement the ISOMAP algorithm), the embedding and pictures plots are as follows.

From the plot, using Manhattan distance, ISOMAP algorithm also showed a similar nonlinear dimensionality reduction ability to group similar faces together. The major difference is the face facing direction layout. Using Euclidean distance, face facing direction is outwards, more clear. Using Manhattan distance face facing direction not that clear.



Picture: In the left picture the top three points are eigenface 613, 136, 343. In the left picture the middle three points are eigenface 321, 410, 641. In the left picture the bottom three points are eigenface 59, 297, 227. In the left picture the left three points are eigenface 360, 564, 239. In the left picture the right three points are eigenface 419, 571, 67.

2. Density estimation: Psychological experiments. (50 points)

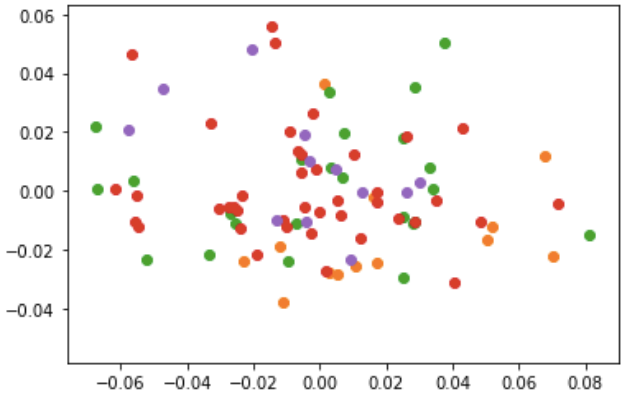
The data set n90pol.csv contains information on 90 university students who participated in a psychological experiment designed to look for relationships between the size of different regions of the brain and political views. The variables amygdala and acc indicate the volume of two particular brain regions known to be involved in emotions and decision-making, the amygdala and the anterior cingulate cortex; more exactly, these are residuals from the predicted volume, after adjusting for height, sex, and similar body-type variables. The variable orientation gives the students' locations on a five-point scale from 1 (very conservative) to 5 (very liberal).

(a) (20 points) Form 2-dimensional histogram for the pairs of variables (amygdala, acc). Decide on a suitable number of bins so you can see the shape of the distribution clearly.

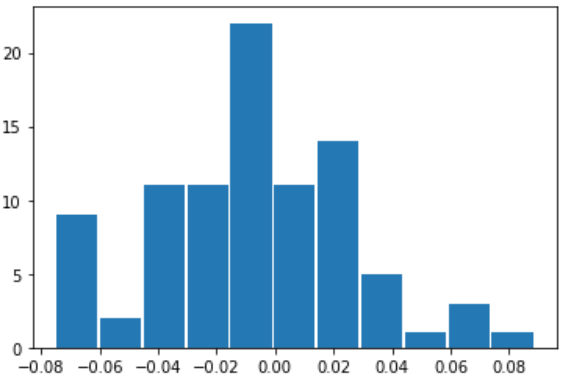
Answer:

Step 1: No data normalization/standardization is done because data has already been adjusted.

Step 2: Get a scatter plot of these 90 data points. (X-axis is amygdala variable and Y-axis is acc variable). Different color means different political orientation (range from 2 to 5)



Step 3: Get a 1-D histogram of data with X-axis as amygdala variable, and Y-axis as counts. The histogram plot confirms the scatter plot that more data points are in the amygdala range from -0.02 to 0.02.



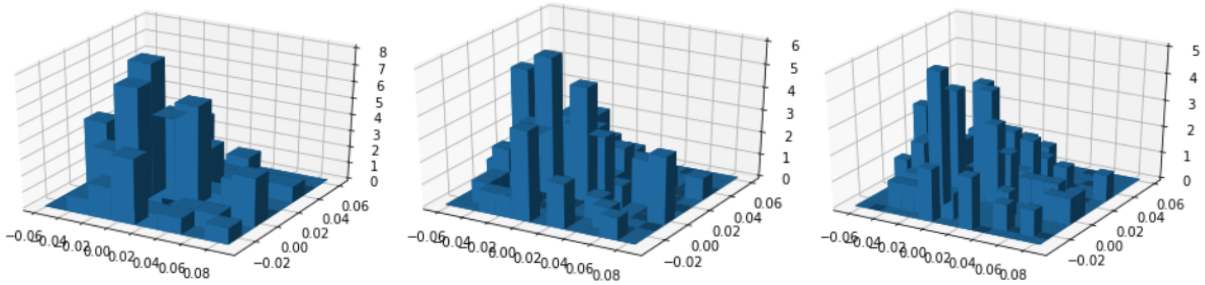
Step 4: Calculate bandwidth (h) using Silverman's rule of thumb as follows (m = 90 here). Then get the number of bin by dividing variable range with bandwidth. (The number of bin = range/bandwidth)

- Silverman's rule of thumb: If using the Gaussian kernel, a good choice for is

$$h \approx 1.06\,\hat{\sigma}\,m^{-1/5}$$

where is the standard deviation of the samples

Amygdala variable gives a bin number of 10.6, and acc variable gives a bin number of 10.7.
So bin number is chosen as 11. I also tried bin number as 8 and 14. According to the output, 11, 14 are both suitable, 8 is too small.



Picture: From left to right, number of bin is 8, 11 and 14 respectively.

(b) (20 points) Now implement kernel-density-estimation (KDE) to estimate the 2-dimensional with a two-dimensional density function of (**amygdala**, **acc**). Use a simple multi-dimensional Gaussian kernel, for

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^2,$$

where x_1 and x_2 are the two dimensions respectively

$$K(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{(x_1^2 + x_2^2)}{2}}.$$

Recall in this case, the kernel density estimator (KDE) for a density is given by

$$p(x) = \frac{1}{m} \sum_{i=1}^{m} \frac{1}{h} K\left(\frac{x^i - x}{h}\right),$$

where x^i are two-dimensional vectors, $h > 0$ is the kernel bandwidth. Set an appropriate h so you can see the shape of the distribution clearly. Plot of contour plot (like the ones in slides) for your estimated density.

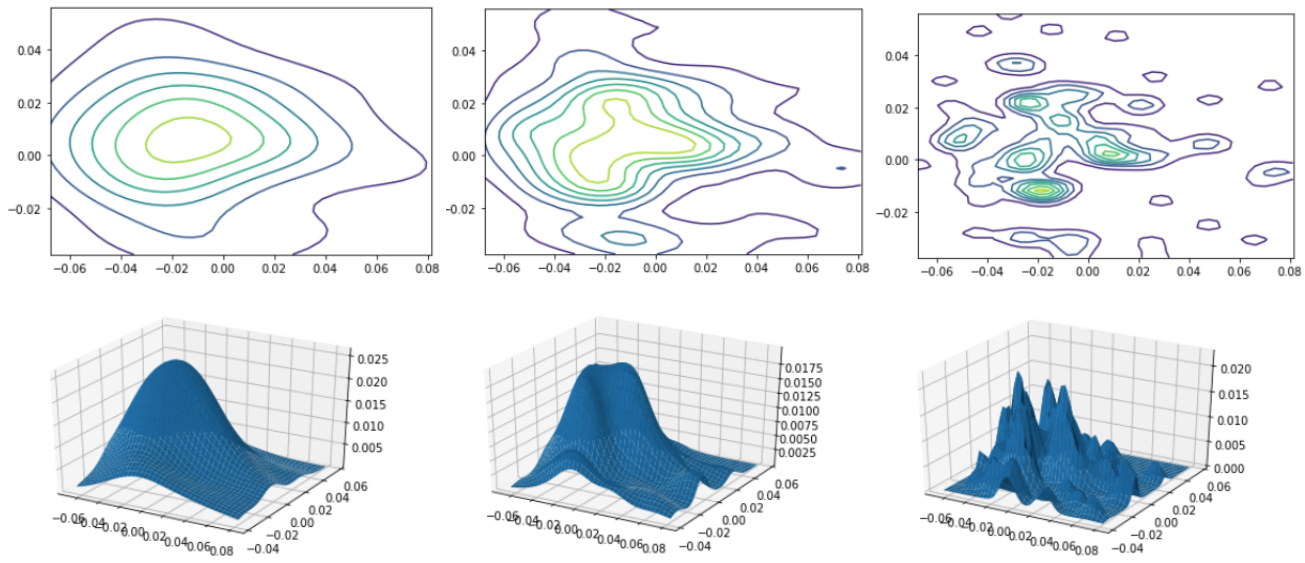
Answer:

Step 1: Build Gaussian kernel function and KDE function by my own. Then compute density p(x) for all positions on the mesh grid. All p(x) values are rescaled to make the sum of probability closer to 1. Please see details in the Jupiter Notebook.

Step 2: Calculate bandwidth (h) using Silverman's rule of thumb.

Amygdala variable gives a bandwidth of 0.014 (h1) and acc variable gives a bandwidth of 0.009 (h2). So bandwidth of h1, h2 and 0.004 are tested.

Step 3: Compare 2D and 3D contour plots. Bandwidth of 0.009 (h2) gives the best contour that are not too smooth or too rough.

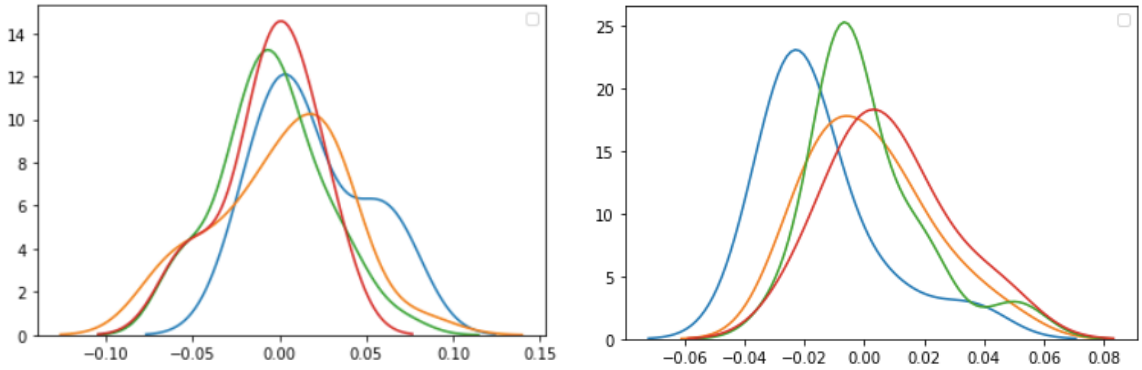


Picture: From left to right, bandwidth is 0.014 (h1), 0.009 (h2) and 0.004 respectively.

(c) (10 points) Plot the condition distribution of the volume of the **amygdala** as a function of political **orientation**: $p(\textsf{amygdala}|\textsf{orientation} = a)$, $a = 1, \ldots, 5$. Do the same for the volume of the **acc**. Plot $p(\textsf{acc}|\textsf{orientation} = a)$, $a = 1, \ldots, 5$. You may either use histogram or KDE to achieve the goal.

Answer:

Kdeplot function from seaborn is used to plot KDE.



Picture on the left is the distribution of amygdala. Picture on the right is the distribution of acc.
Color legends: Blue: Orientation 2, orange: orientation 3, green: orientation 4, red: orientation 5.