# Sol_sheet03

November 14, 2022

#### 0.0.1 Submitted by: Simran Joharle, Sachin Gupta and Bhavesh Rajpoot

## 1 Sheet 3

### 1.1 1. Rand Index and Variation of Information

[ ]:
```
from IPython.display import Image, display


im = [Image(filename=f'images/Rand_Index_Sheet{i}.jpg') for i in range(1,5)]
display(*im)
```

RAND Index:

Set of $n$ objects $S = \{O_1, \ldots, O_n\}$

Two different partitions of $S$ – $U$ and $V$

$U = \{u_1, \ldots, u_R\}$ $\quad V = \{v_1, \ldots, v_C\}$

$\bigcup_{i=1}^{R} u_i = S = \bigcup_{j=1}^{C} v_j$ and $u_i \cap u_{i'} = \emptyset = v_j \cap v_{j'}$ for $1 \le i \ne i' \le R$ and $1 \le j \ne j' \le C$

$U$ = external idea $\quad V$ = clustering result

$a$ = # pairs of objects that are placed in the same cluster in $U$ and $V$.

$b$ = # pairs of objects that are in the same cluster in $U$ but not in the same cluster in $V$.

$c$ = # pairs of objects in the same cluster in $V$ but not in the same cluster in $U$.

$d$ = # no. of pairs in different clusters in both partitions

$$\text{Rand index} = \frac{a+d}{a+b+c+d}$$

$$= 0 < \text{Rand index} < 1$$

$= 1$ when two partitions agree perfectly.

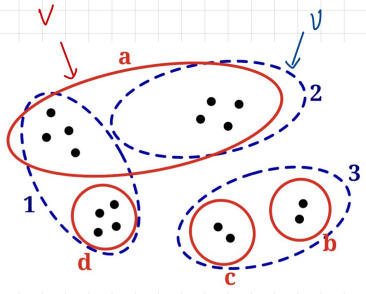
| *Class \ Cluster* | $v_1$ | $v_2$ | $\ldots$ | $v_C$ | *Sums* |
|---|---|---|---|---|---|
| $u_1$ | $n_{11}$ | $n_{12}$ | $\ldots$ | $n_{1C}$ | $n_{1.}$ |
| $u_2$ | $n_{21}$ | $n_{22}$ | $\ldots$ | $n_{2C}$ | $n_{2.}$ |
| $\vdots$ | $\vdots$ | $\vdots$ | | $\vdots$ | $\vdots$ |
| $u_R$ | $n_{R1}$ | $n_{R2}$ | $\ldots$ | $n_{RC}$ | $n_{R.}$ |
| *Sums* | $n_{.1}$ | $n_{.2}$ | $\ldots$ | $n_{.C}$ | $n_{..} = n$ |

Table 1: Notation for the contingency table for comparing two partitions.

1) Rand Index and Variation of Clustering.

a) Contingency table

| U \ V | a | b | c | d | Sums |
|---|---|---|---|---|---|
| 1 | 4 | 0 | 0 | 4 | 8 |
| 2 | 4 | 0 | 0 | 0 | 4 |
| 3 | 0 | 2 | 2 | 0 | 4 |
| Sums | 8 | 2 | 2 | 4 | 16 |




b) Rand index

#pairs in same cluster in both partitions $= \sum_{ij} \binom{n_{ij}}{2} = \binom{4}{2} + \binom{2}{2} + \binom{4}{2}$ –(i)

$= 13$

# pairs in same cluster in $U$ but not in same cluster in $V$ =

$$\sum_{i} \binom{n_i}{2} - \sum_{ij} \binom{n_{ij}}{2} = \binom{8}{2} + \binom{4}{2} + \binom{4}{2} - 13 = \quad \text{–(ii)}$$

$$= 28 + 6 + 28 - 13 = 49$$

# pairs of objects in same cluster in V but not in U $= \sum_j \binom{n_j}{2} - \sum_{ij} \binom{n_{ij}}{2}$ — (iii)

$= \binom{8}{2} + \binom{2}{2} + \binom{2}{2} + \binom{4}{2} - 7$

$= 28 + 1 + 1 + 6 - 7 = 29$

# pairs not in same cluster in U and not in same cluster in V $= \binom{n}{2} - (i + ii + iii)$

$= \binom{16}{2} - (13 + 49 + 29)$

$= 120 - 91 = 29$

$\therefore$ Rand index $= \frac{13 + 29}{120} = \frac{42}{120} = \frac{7}{20} = 0.35$

c) Entropy of a clustering $= -\sum_x P(x) \log_2(P(x))$

or $-\sum_{C \in \mathbb{C}} P(w_C) \log_2 P(w_C)$

where C: is a classification is the set $\mathbb{C}$ of all classifications.

$P(w_C)$ is probability of a data point being classified as C in clustering W.

$\therefore$ Entropy $= -\sum \frac{|w_C|}{n_w} \log_2 \frac{|w_C|}{n_w}$

$|w_C|$ is the count of points classified as C in w

$n_w$ is the count of points in cluster w.

$\therefore$ For clustering U:

Entropy $= -\frac{8}{16}\log_2\frac{8}{16} - \frac{4}{16}\log_2\frac{4}{16} - \frac{4}{16}\log_2\frac{4}{16}$

$= -(0.5 \times -1) - (0.25 \times -2) - (0.25 \times -2)$

$= 0.5 + 0.5 + 0.5$

$= 1.5$

For clustering V:

Entropy $= -\frac{8}{16}\log_2\frac{8}{16} - \frac{4}{16}\log_2\frac{4}{16} - \frac{2}{16}\log_2\frac{2}{16} - \frac{2}{16}\log_2\frac{2}{16}$

$= -(0.5 \times -1) - (0.25 \times -2) - (0.125 \times -3) - (0.125 \times -3)$

$= 0.5 + 0.5 + 0.375 + 0.375$

$= 1.75$

d) We know Entropy $= \sum_{x \in A} P(x) \log_2 \frac{1}{P(x)}$

Entropy is maximize if $P(x)$ is uniform

$H(X) \leq \log_2 |A|$ and with equality if and only if $p_i = \frac{1}{|A|}$

Here $|A|$ is no. of elements in A.

i.e we will have maximum entropy if we perform clustering such that each class will have only one data point, thus we will have 16 classes each containing a point.

P.T.O

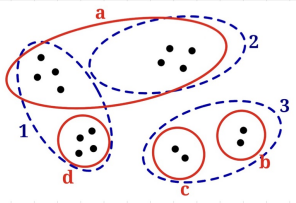
We will have minimum entropy when all the datapoints will be considered as one whole cluster

i.e $p_i = \frac{|A|}{|A|}$

The entropy will be 0 in this case.

e)

| P(x,y) | a | b | c | d | Sums |
|---|---|---|---|---|---|
| 1 | 1/4 | 0 | 0 | 1/4 | 1/2 |
| 2 | 1/4 | 0 | 0 | 0 | 1/4 |
| 3 | 0 | 1/8 | 1/8 | 0 | 1/4 |
| Sums | 1/2 | 1/8 | 1/8 | 1/4 | 1 |




$$P(x,y) = \frac{|X \cap Y|}{N}$$

Joint entropy : $H(X,Y) = -\sum_{x,y \in A,B} P(x,y) \log_2 P(x,y)$

$$= -3 \times \frac{1}{4} \log_2 \frac{1}{4} - 2 \times \frac{1}{8} \log_2 \frac{1}{8}$$

$$= 3 \times \frac{1}{4} \times 2 + 2 \times \frac{1}{8} \times 3$$

$$= \frac{6}{4} + \frac{3}{4} = \frac{9}{6} = 1.5$$

f) If X, Y are independent $P(X,Y) = P(X)P(Y)$

$$P_c(Z) = P_A(X) P_B(Y)$$

where X and Y are partitioning of the variables Z belonging to C. Then the total entropy is

$$\text{Entropy} = -\int P_c(Z) \ln(P_c(Z)) \, dZ$$

$$= -\int P_A(X) P_B(Y) \ln(P_c(X,Y)) \, dX \, dY$$

$$= -\int P_A(X) P_B(Y) \ln(P_A(X) P_B(Y)) \, dX \, dY$$

$$= -\int P_A(X) P_B(Y) [\ln(P_A(X)) + \ln(P_B(Y))] \, dX \, dY$$

$$= -\int P_A(X) P_B(Y) \ln[P_A(X)] \, dX \, dY - \int P_A(X) P_B(Y) \ln[P_B(Y)] \, dX \, dY$$

$$\because \int P_A(X) \, dX = 1 \quad \text{and} \quad \int P_B(Y) \, dY = 1$$

$$= -\int P_A(X) \ln[P_A(X)] \, dX - \int P_B(Y) \ln[P_B(Y)] \, dY$$

$$= \text{Entropy}_A + \text{Entropy}_B$$

$$\therefore H(X,Y) = H(X) + H(Y)$$

$$\because 1.5 \neq 1.5 + 1.79$$

Clusters $C_1$ and $C_2$ are not independent

g) Mutual information between Y and X

$$I(Y;X) = H(Y) - H(Y|X)$$

$$H(Y|X) = H(X,Y) - H(X)$$

$$= 1.5 - 1.5 = 0$$

$$\therefore I(Y;X) = 1.75 - 0 = 1.75$$

| P(x,y) | a | b | c | d | Sums |
|---|---|---|---|---|---|
| 1 | 1/4 | 0 | 0 | 1/4 | 1/2 |
| 2 | 1/4 | 0 | 0 | 0 | 1/4 |
| 3 | 0 | 1/8 | 1/8 | 0 | 1/4 |
| Sums | 1/2 | 1/8 | 1/8 | 1/4 | 1 |

h) Suppose we have two partitions X and Y

$$n = \sum_i |X_i| = \sum_j |Y_j| = |A|$$

$$p_i = |X_i|/n \qquad q_j = |Y_j|/n$$

$$r_{ij} = |X_i \cap Y_j|/n$$

$$VI(X;Y) = -\sum_{i,j} r_{ij} \left[\log_2(r_{ij}/p_i) + \log_2(r_{ij}/q_j)\right]$$

$$= -\frac{1}{4}\left[\log_2\left(\frac{1}{4}\times 2\right) + \log_2\left(\frac{1}{4}\times 2\right)\right] - \frac{1}{4}\left[\log_2\left(\frac{1}{4}\times 2\right) + \log_2\left(\frac{1}{4}\times 4\right)\right]$$

$$- \frac{1}{4}\left[\log_2\left(\frac{1}{4}\times 4\right) + \log_2\left(\frac{1}{4}\times 2\right)\right] - \frac{1}{4}\left[\log_2\left(\frac{1}{8}\times 4\right) + \log_2\left(\frac{1}{8}\times 8\right)\right]$$

$$- \frac{1}{4}\left[\log_2\left(\frac{1}{8}\times 4\right) + \log_2\left(\frac{1}{8}\times 8\right)\right]$$

$$= -\frac{1}{4}\left[\log_2\left(\frac{1}{2}\right) + \log_2\left(\frac{1}{2}\right)\right] - \frac{1}{4}\left[\log_2\left(\frac{1}{2}\right) + \log_2(1)\right]$$

$$- \frac{1}{4}\left[\log_2(1) + \log_2\left(\frac{1}{2}\right)\right] - \frac{2}{4}\left[\log_2\left(\frac{1}{2}\right) + \log_2(1)\right]$$

$$= -\frac{1}{4}[-1-1] - \frac{1}{4}[-1] - \frac{1}{4}[-1] - \frac{1}{2}[-1]$$

$$= \frac{1}{2} + \frac{1}{4} + \frac{1}{4} + \frac{1}{2} = \frac{3}{2} = 1.5$$

i) We know that $MI = H(Y) - H(Y|X)$

If X is known, in order to maximize MI we must maximize $H(Y)$. The clustering for which $H(Y)$ is maximum will include N clusters for N data points, each cluster containing 1 data point.

Also, VOI (unlike MI) obeys triangle inequality and thus is preferred over MI.

## 1.2 2 Similarity Measures on Jet Data Clusterings

### 1.2.1 (a) For each of the clusterings saved in the rows of `dijet_clusters.npy`, create a contingency table comparing it to the partition corresponding to the ground-truth labels. Visualize them using Hinton plots (you can use the provided method or create your own) and interpret what you see.

load the ground truth labels and k-means partitions

[ ]:
```
import numpy as np
from matplotlib import pyplot as plt

from sklearn.metrics.cluster import contingency_matrix
```

[ ]:
```
# the dijet labels, corresponding to bottom, charm, light quarks
labels = np.load('data/dijet_labels.npy')

# 5 rows, each corresponding to a k-means clustering with [2, 3, 5, 10, 20]␣
 ↪clusters respectively
partitions = np.load('data/dijet_clusters.npy')
ks = [2, 3, 5, 10, 20]

print(labels.shape, partitions.shape)
```

```
(2233,) (5, 2233)
```

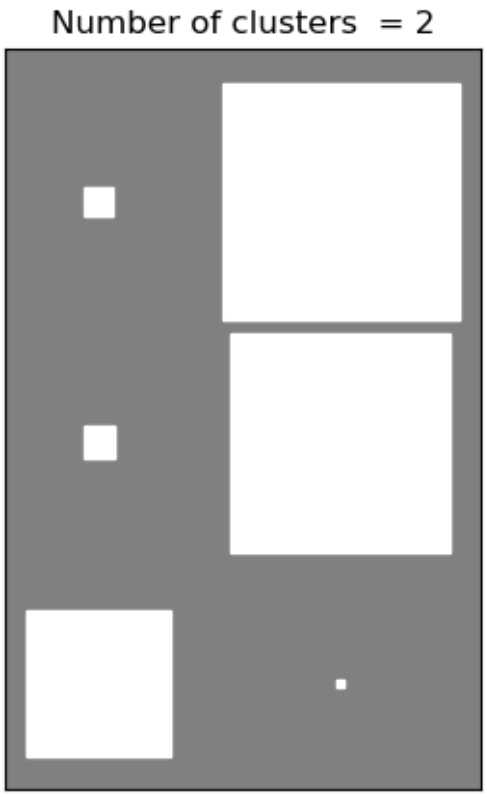
define function to create a hinton plot from a contingency matrix

[ ]:
```
def hinton_plot(matrix, max_weight=None, ax=None):
    """
    Draw Hinton diagram for visualizing a weight matrix.
    From https://matplotlib.org/3.1.1/gallery/specialty_plots/hinton_demo.html
    """
    ax = ax if ax is not None else plt.gca()

    if not max_weight:
        max_weight = 2 ** np.ceil(np.log(np.abs(matrix).max()) / np.log(2))

    ax.patch.set_facecolor('gray')
    ax.set_aspect('equal', 'box')
    ax.xaxis.set_major_locator(plt.NullLocator())
    ax.yaxis.set_major_locator(plt.NullLocator())

    for (x, y), w in np.ndenumerate(matrix):
        color = 'white' if w > 0 else 'black'
        size = np.sqrt(np.abs(w) / max_weight)
        rect = plt.Rectangle([x - size / 2, y - size / 2], size, size,
```

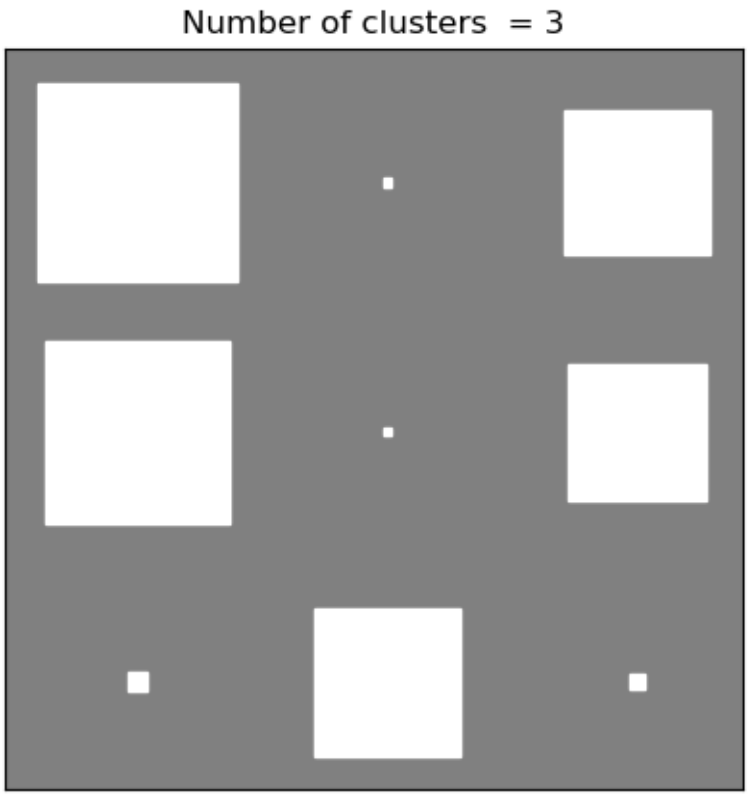
```
                                facecolor=color, edgecolor=color)
        ax.add_patch(rect)

    ax.autoscale_view()
    ax.invert_yaxis()
```

[ ]:
```
# TODO: compute the contingency matrices between labels and k-means partitions
#       (either implement your own or find the function in sklearn.metrics.
 ↪cluster)


# TODO: visualize the contingency matrices via hinton plots and
#       label the plots by the number of clusters used in k-means


cont_matrix = {2:np.array([]),3:np.array([]),5:np.array([]),10:np.array([]),20:
 ↪np.array([])}   ##this array will store contigency matrix for all clusters

for i in range(len(ks)):  ##looping over all the clusters
    cont_matrix[ks[i]] = contingency_matrix(labels.T,partitions[i,:].T)  ## ␣
 ↪calculating contigency matrix
    print(cont_matrix[ks[i]])
    fig, axs = plt.subplots(1, 1, figsize=(8, 5))   ##inittializing plot
    axs.set_title('Number of clusters  = '+str(ks[i]))

    hinton_plot(cont_matrix[ks[i]].T,max_weight=None,ax = axs)  ##passing the␣
 ↪values
    plt.show()
```

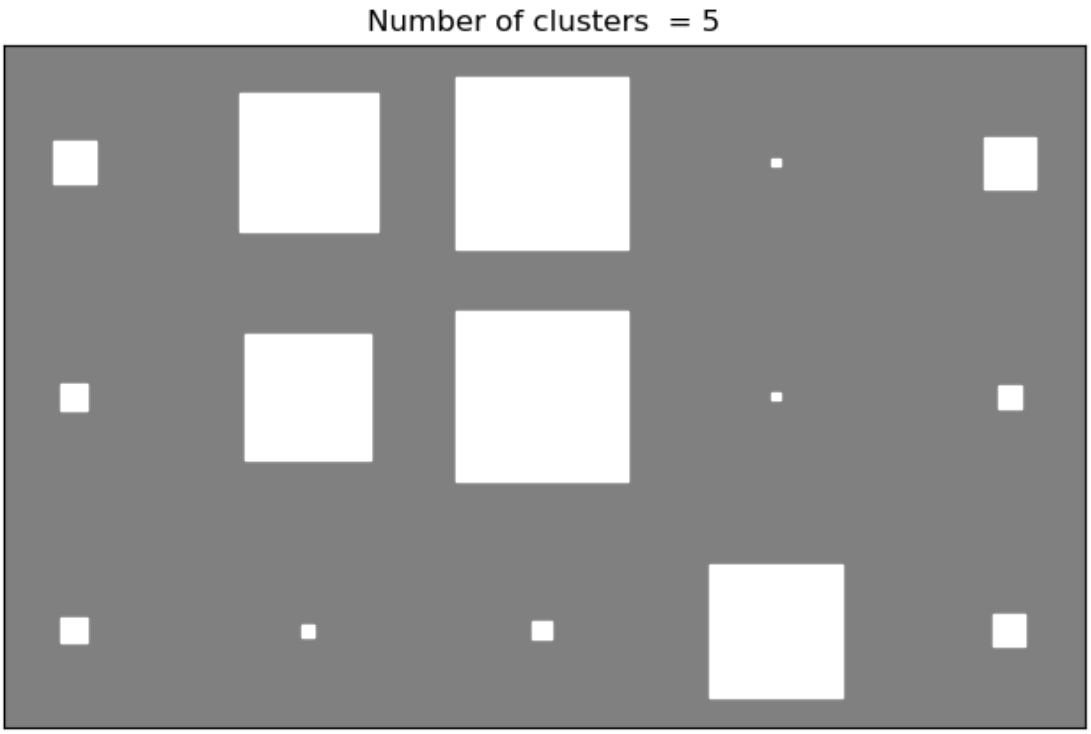
```
[[ 15 984]
 [ 17 847]
 [369   1]]
```



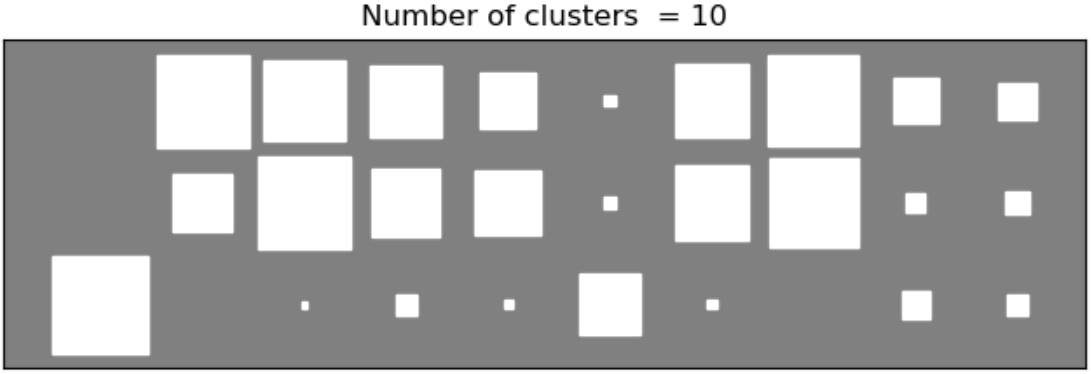

```
[[653   1 345]
 [552   1 311]
 [  6 360   4]]
```




```
[[ 34 357 556   1  51]
 [ 14 294 545   1  10]
 [ 12   3   6 330  19]]
```

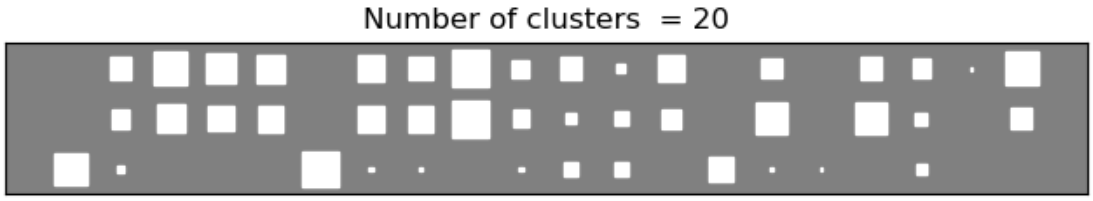



```
[[  0 210 160 125  79   3 133 204  51  34]
 [  0  85 204 113 108   4 135 191  10  14]
 [231   0   1  10   2  93   2   0  19  12]]
```




```
[[  0  48 109  85  82   0  73  56 140  32  51   7  77   0  42   0  49  34
    1 113]
 [  0  36  79  64  69   0  65  65 133  30  10  19  39   0  98   0  98  14
    0  45]
```

```
[108   5   0   0   0 131   2   2   0   2  19  22   0  65   2   1   0  11
   0   0]]
```




### 1.2.2 Interpretation -

- For the first two plots we see that there is exactly one cluster which has more common points of **truth-label 2** and other clusters has common points with **truth label 0 and 1** in almost equal amounts.
- For the last three plots (ks =5, 10,20) we see some clusters like above explanation but there are other clusters which has mixed points of all labels with them.

### 1.2.3 (b) Compute the Rand score, the adjusted Rand score, and the variation of information of the clusterings comparing them to the ground-truth clustering. Which clustering would you deem the best, and why?

[ ]:
```
from sklearn.metrics import rand_score, adjusted_rand_score, mutual_info_score,␣
 ↪v_measure_score
iter=  0
for k, partition in zip(ks, partitions):

    Rand_score = rand_score(labels,partition)
    Adjusted_rand_score =  adjusted_rand_score(labels, partition)
    Mutual_info_score = mutual_info_score(labels,partition)
    V_measure_score = v_measure_score(labels,partition)

    print(f'{Rand_score = },{Adjusted_rand_score = },{Mutual_info_score =␣
 ↪},{V_measure_score = }')


# TODO: compute and print the different scores as requested in the excercise
```

```
Rand_score = 0.6481299568062638,Adjusted_rand_score =
0.36077853232930374,Mutual_info_score = 0.39534332849247383,V_measure_score =
0.5285855979823327
Rand_score = 0.6465657689239447,Adjusted_rand_score =
0.259432761895148,Mutual_info_score = 0.41567365518450516,V_measure_score =
```

```
0.41316208346649186
Rand_score = 0.6413194394284495,Adjusted_rand_score =
0.2276969117496487,Mutual_info_score = 0.39303732230202754,V_measure_score =
0.35211448592312977
Rand_score = 0.6368315283776907,Adjusted_rand_score =
0.10880207235236414,Mutual_info_score = 0.3936409218177637,V_measure_score =
0.24650777719877184
Rand_score = 0.6310518982932776,Adjusted_rand_score =
0.06135047674304189,Mutual_info_score = 0.40449850012337163,V_measure_score =
0.211081072203569
```

#### 1.2.4 Which Clustering to choose ?

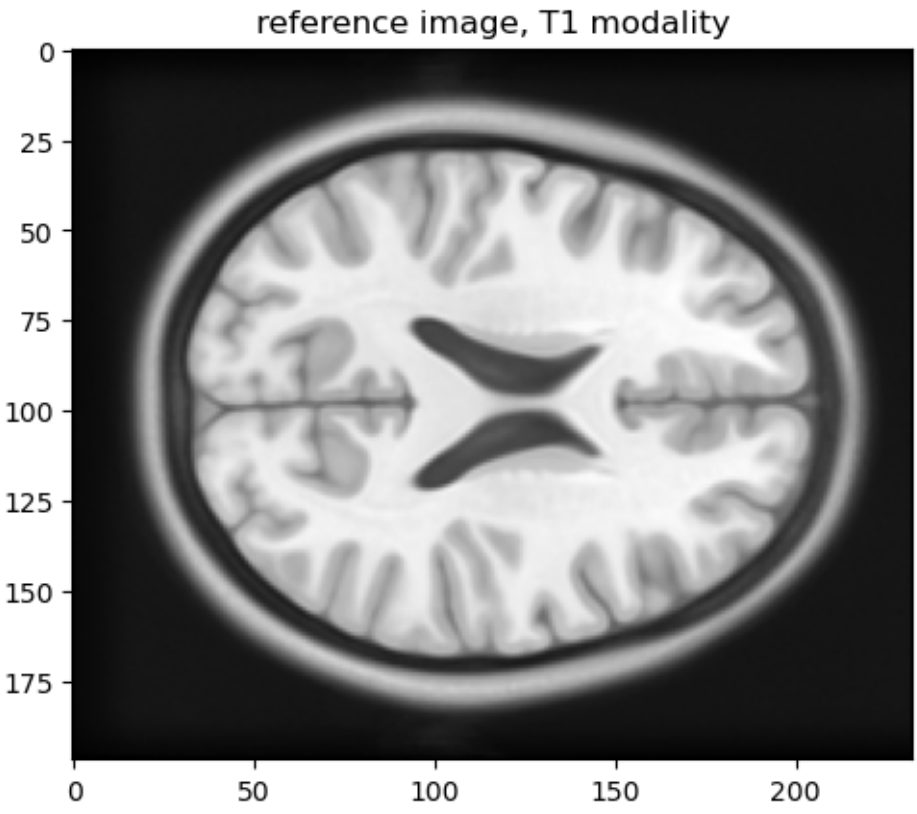
We see that first clustering outperforms from all other 4 in all scores. But Second Clustering( ks = 3) gives more mutual info than the first. However, when it comes to choosing the optimal clustering we should not only considers scores but some other performances as well. Since we see that all clustering algorithms have clusters which have mixed number of first two quarks and some clusters only contain the last quarks.

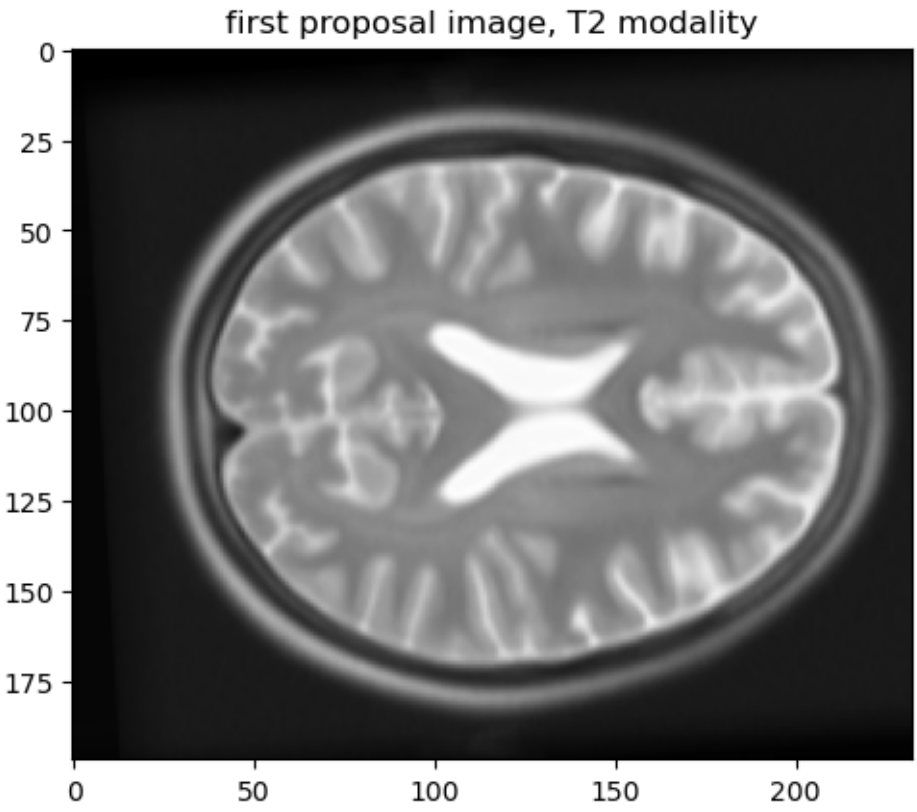
If we now look at our first two clustering on the basis of such approach we see that the In the **2nd clustering algorithm(ks = 3)** second cluster(column two) agrees 360 times with the third quark and only *1* times each with first two quarks. Similarly In the **1st clustering algorithm(ks = 2)** first cluster (first column) agrees 369 times but contains (15+17 = 32) first two quarks.

Hence when it comes to experimental analysis perspectives we will opt the second clustering with ks =3 :)

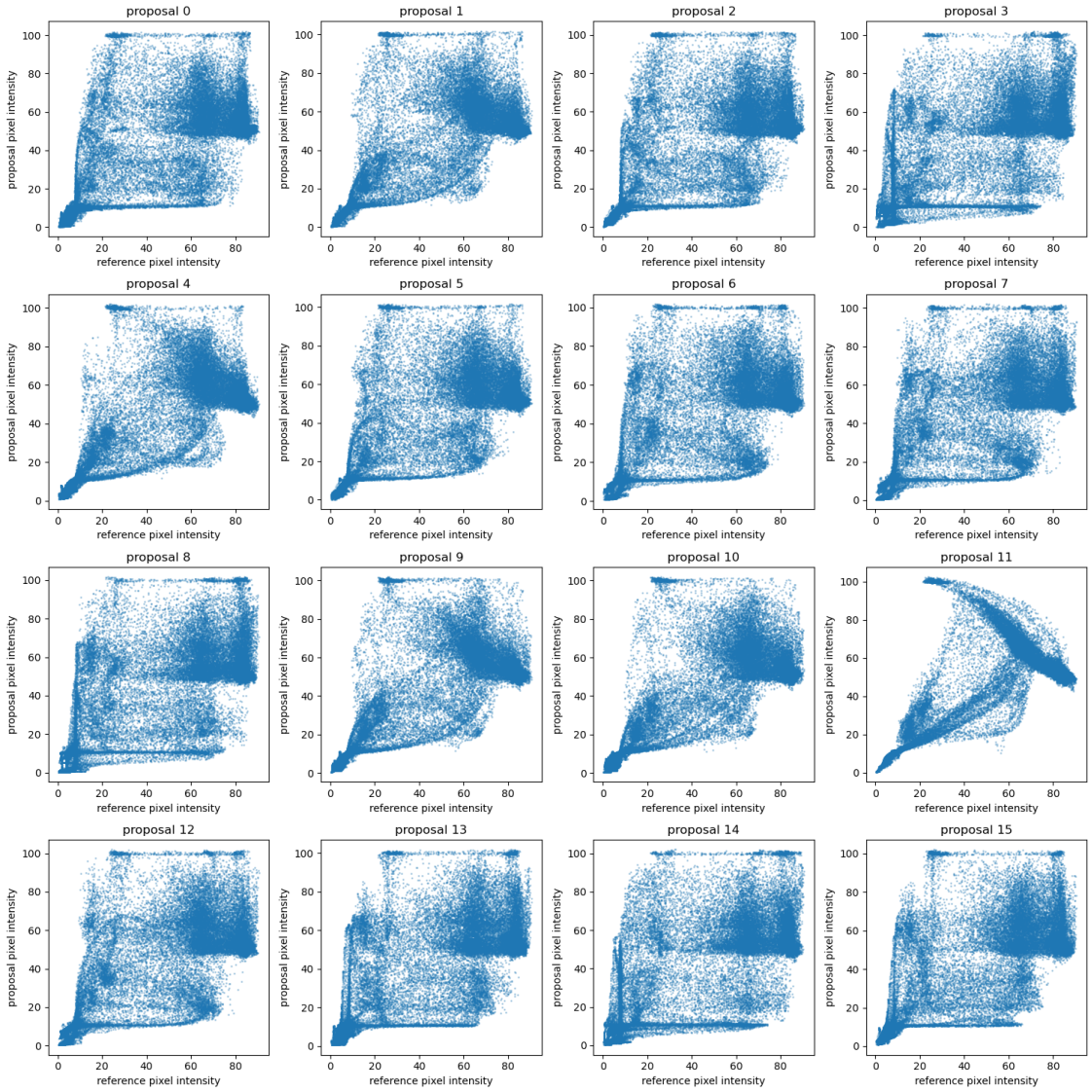
### 1.3 3. Mutual Information for Image Matching

[ ]:
```
# load the reference image and the proposals
reference = np.load('data/t1_reference.npy')
proposals = np.load('data/t2_registration_proposals.npy')

plt.imshow(reference, cmap='gray')
plt.title('reference image, T1 modality')
plt.show();

plt.imshow(proposals[0], cmap='gray')
plt.title('first proposal image, T2 modality')
plt.show();
```


reference image, T1 modality
0
25
50
75
100
125
150
175
0
50
100
150
200




[ ]:
```
# create 16 subplots in order to plot all proposals in one figure
fig, axs = plt.subplots(4, 4, figsize=(15, 15))

for i, (ax, proposal) in enumerate(zip(axs.flatten(), proposals)):
    # TODO: make a scatterplot of the reference pixel intensities vs the pixel␣
 ↪intensities in the i-th proposal
    #       do not forget to label the axes
    ax.scatter(reference,proposal,s=1,alpha=0.3)
    ax.set_xlabel("reference pixel intensity")
    ax.set_ylabel("proposal pixel intensity")
    ax.set_title(f'proposal {i}')
plt.tight_layout()
plt.show();
```



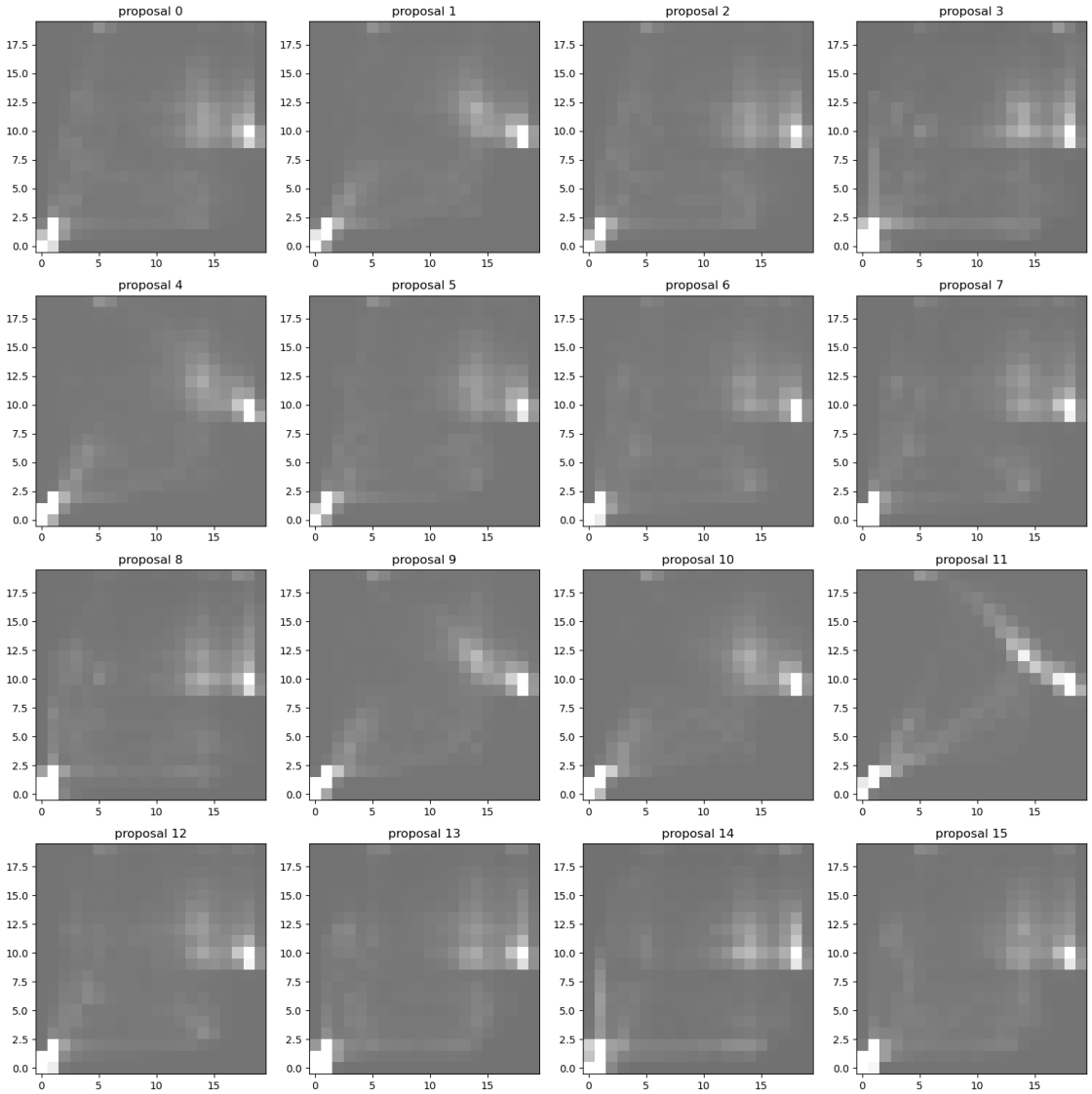

**Qualitative differences in reference and proposals:**

- We see that the both the images do not have sam intensities and the same respective pixel position. If this would have been the case all the points would have been along the diagonal of the plot.
- From the spreadof the points we can see that at lower pixel values of reference image we have higher intensity at the correspoinding pixel in the proposal image and this is seen vice versa too. This explains the general inversion of intensity as seen in the plots in previous cell.
- We also see an overdensity at the top right part of the graph, which shows that for high intensity values in the reference images, the pixels in the proposal image have almost a constant intensity lying between 60 to 40. It may also show misalignment of images

- Both the images agree at exteremely low intensities which correspond to the pixels in the black background of the two images.

(b)

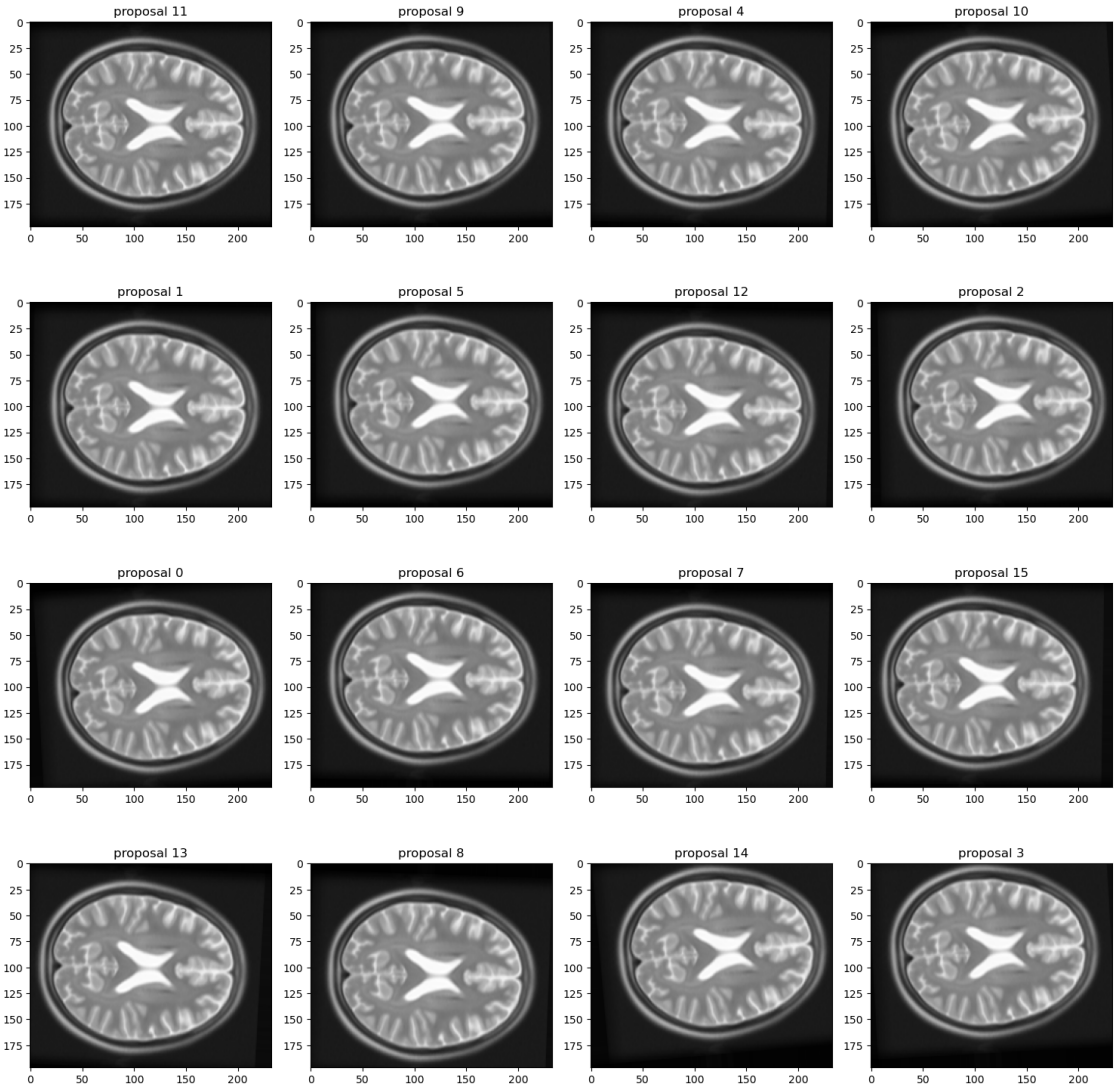
[ ]:
```
# TODO: For each proposal, create a 2D histograms of pixel intensities of it␣
 ↪and the reference
#       (Choose a sensible range and bins for the histogram)
# Hint: np.histogram2d, flatten the arrays as pixel positions don't matter

histograms = [np.histogram2d(reference.flatten(),proposal.flatten(),bins=20)[0]␣
 ↪for proposal in proposals]

# TODO: plot the histograms

# create 16 subplots in order to plot all proposals in one figure
fig, axs = plt.subplots(4, 4, figsize=(15, 15))

for i, (ax, hist) in enumerate(zip(axs.flatten(), histograms)):
    # TODO: plot the histogram for the i'th proposal.
    mean = np.mean(hist.T)
    sigma = np.std(hist.T)
    ax.imshow(hist.T,␣
 ↪origin='lower',vmax=mean+(3*sigma),vmin=mean-(3*sigma),cmap='gray')
    ax.set_title(f'proposal {i}')

plt.tight_layout()
plt.show();
```




[ ]:
```
from sklearn.metrics import mutual_info_score as mi

# TODO: for each histogram, compute the mutual information score
MI = np.
 ↪array([mi(labels_true=reference,labels_pred=proposal,contingency=hist)for␣
 ↪(hist,proposal) in zip(histograms,proposals)])
# TODO: order the proposal by MI, plot them in this order and include the␣
 ↪scores in the titles of the plots.
sort_ind = np.argsort(MI)[::-1]
```

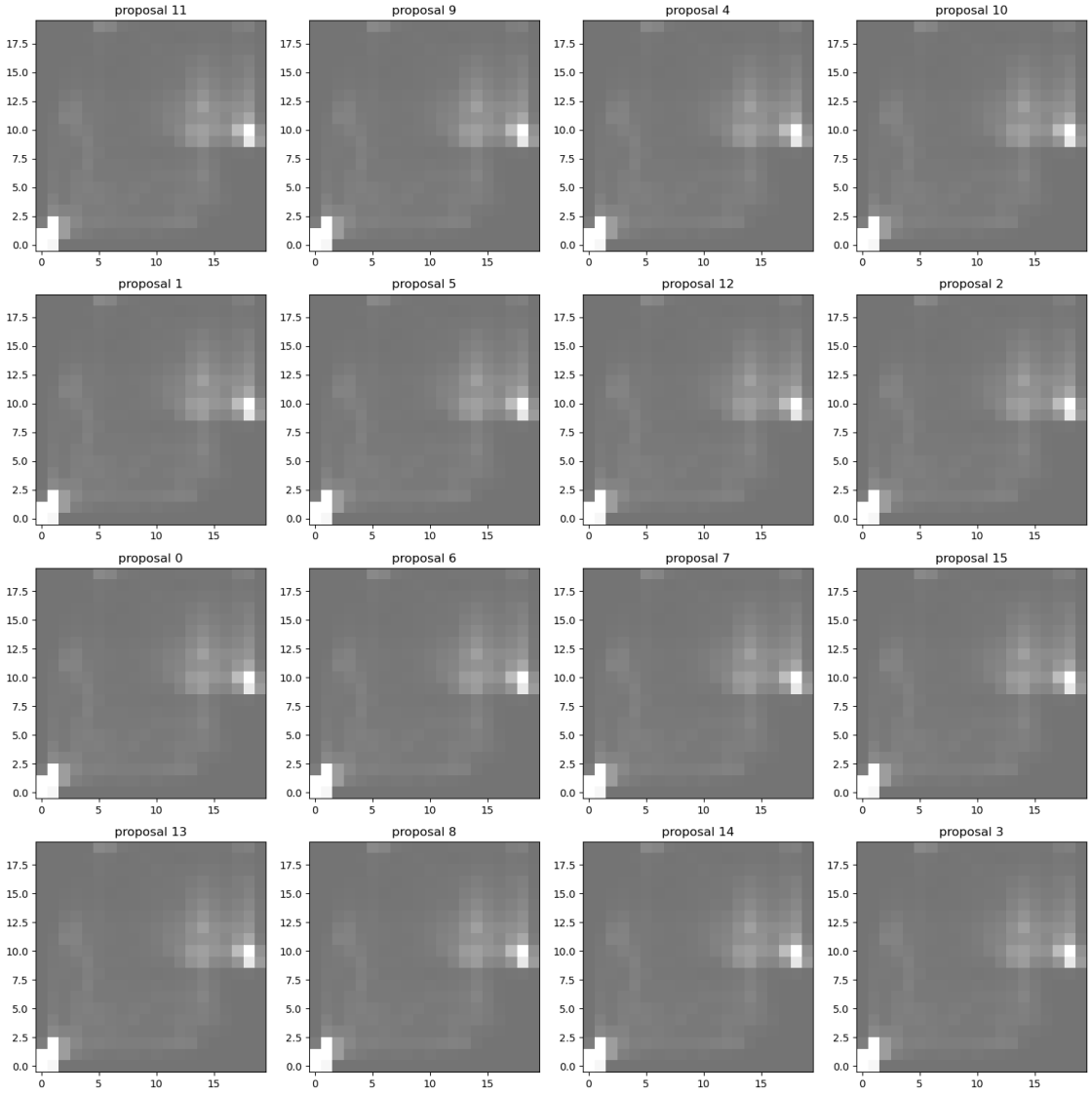
### 1.3.1 Plotting from highest MI to lowest MI

[ ]:
```
fig, axs = plt.subplots(4, 4, figsize=(15, 15))

for j, (ax, i) in enumerate(zip(axs.flatten(), sort_ind)):
    ax.imshow(proposals[i], cmap='gray')
    ax.set_title(f'proposal {i}')

plt.tight_layout()
plt.show();
```




[ ]:
```
fig, axs = plt.subplots(4, 4, figsize=(15, 15))

for j, (ax, i) in enumerate(zip(axs.flatten(), sort_ind)):
```

```
    mean = np.mean(hist.T)
    sigma = np.std(hist.T)
    ax.imshow(hist.T,␣
 ↪origin='lower',vmax=mean+(3*sigma),vmin=mean-(3*sigma),cmap='gray')
    ax.set_title(f'proposal {i}')

plt.tight_layout()
plt.show();
```




**We know that Mutual information is a metric from the joint (2D) histogram. The metric is high when the signal is highly concentrated in few bins, and low when the signal is spread across**

**many bins as seen in proposal 11. We see that for the proposals with lower MI the signal is less concentrated into a small number of bins, leading to the drop in the mutual information.**