Assignment 3

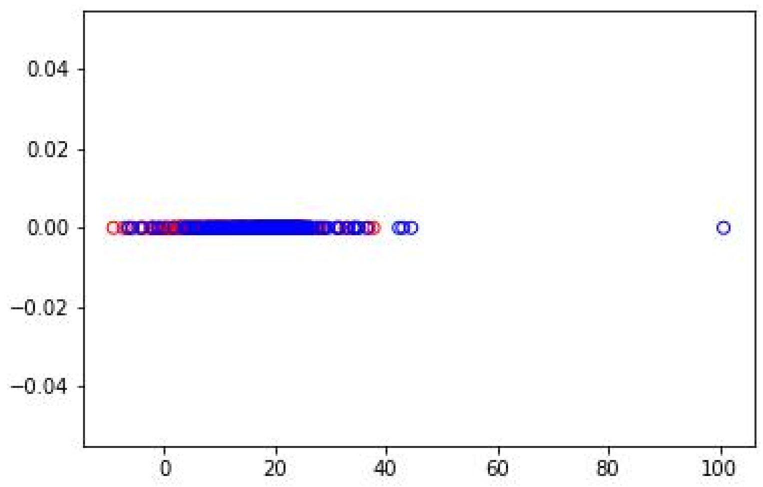
Description

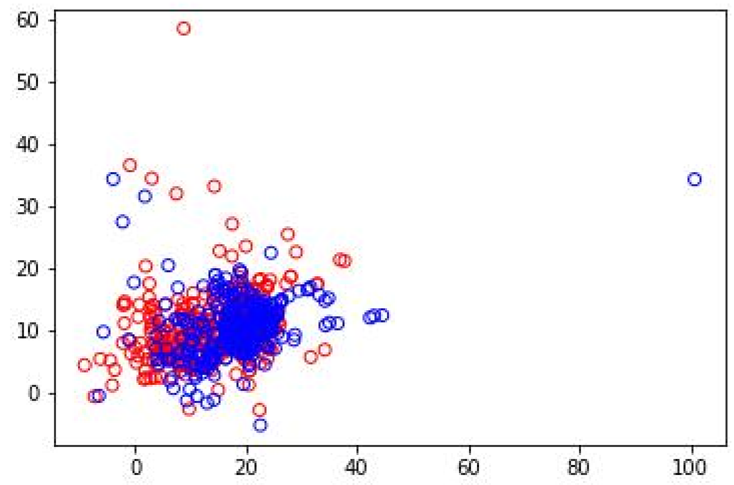
1. Apply BSCA algorithm on the dataset(s) you chose for week 2 assignment. Specifically, apply BSCA on the new 2-D and 1-D representation of the datasets after applying PCA and LDA respectively (from week 2). At the end analyze your findings.
2. Experiment BSCA using three different values for the threshold of dissimilarity (α) and three different values for the maximum number of clusters (M).

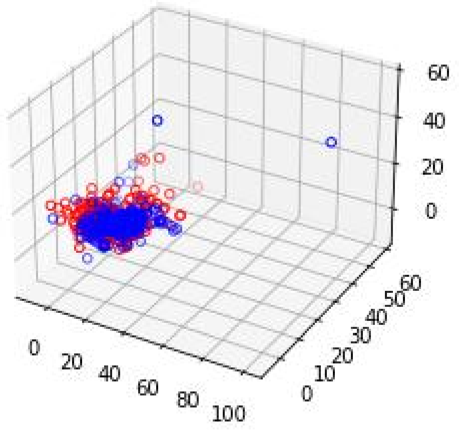
Assignment 2 Review and Update

Originally I believed that my LDA results from assignment 2 were bad because I had failed to implement the LDA algorithm correctly.

```
In [1430]: from IPython.display import Image, display
            display(Image(filename='../week2/lda_outputs/LDA_1_axes_measurements.jp
            g') )
            display(Image(filename='../week2/lda_outputs/LDA_2_axes_measurements.jp
            g') )
            display(Image(filename='../week2/lda_outputs/LDA_3_axes_measurements.jp
            g') )
```

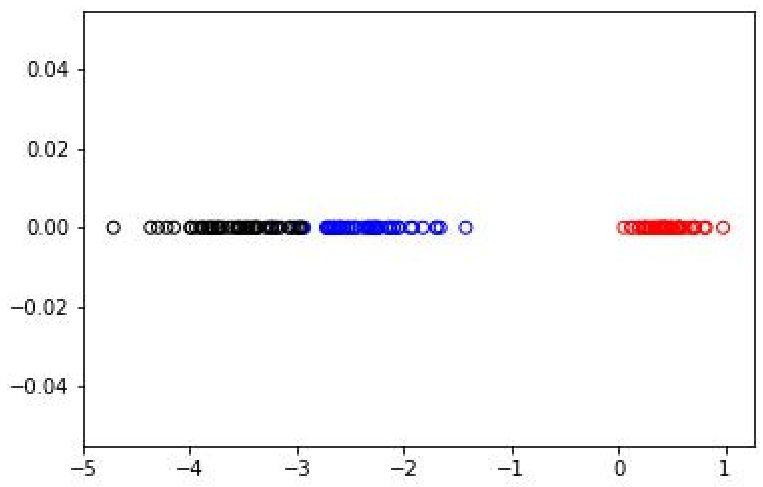


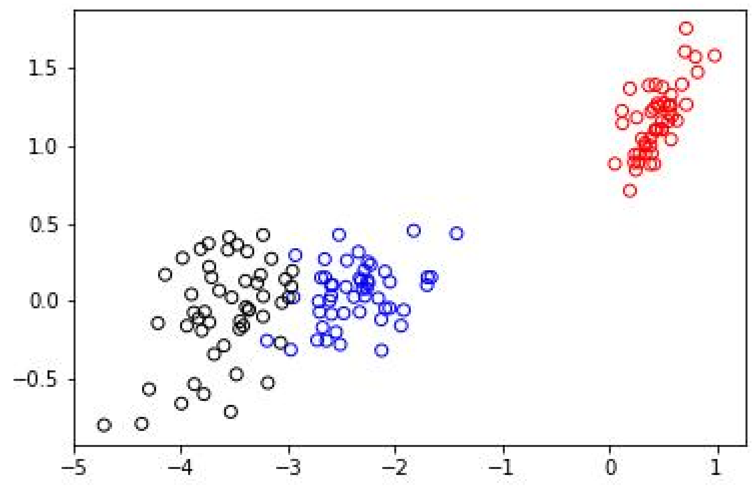


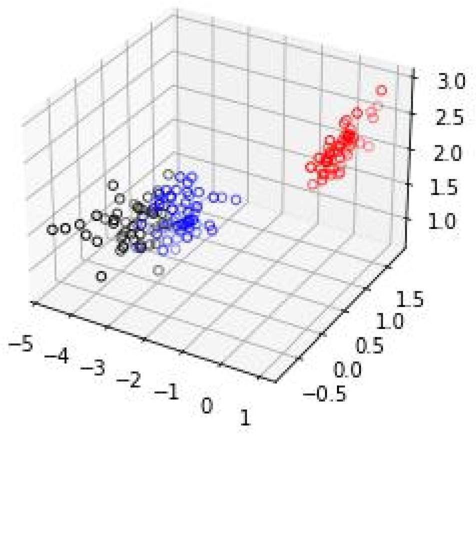


So I re-applied my script to the iris dataset.

In [1431]:
```
from IPython.display import Image, display
display(Image(filename='../week2/lda_outputs/LDA_1_axes_iris.jpg') )
display(Image(filename='../week2/lda_outputs/LDA_2_axes_iris.jpg') )
display(Image(filename='../week2/lda_outputs/LDA_3_axes_iris.jpg') )
```


0.04
0.02
0.00
−0.02
−0.04
−5
−4
−3
−2
−1
0
1


1.5
1.0
0.5
0.0
−0.5
−5
−4
−3
−2
−1
0
1

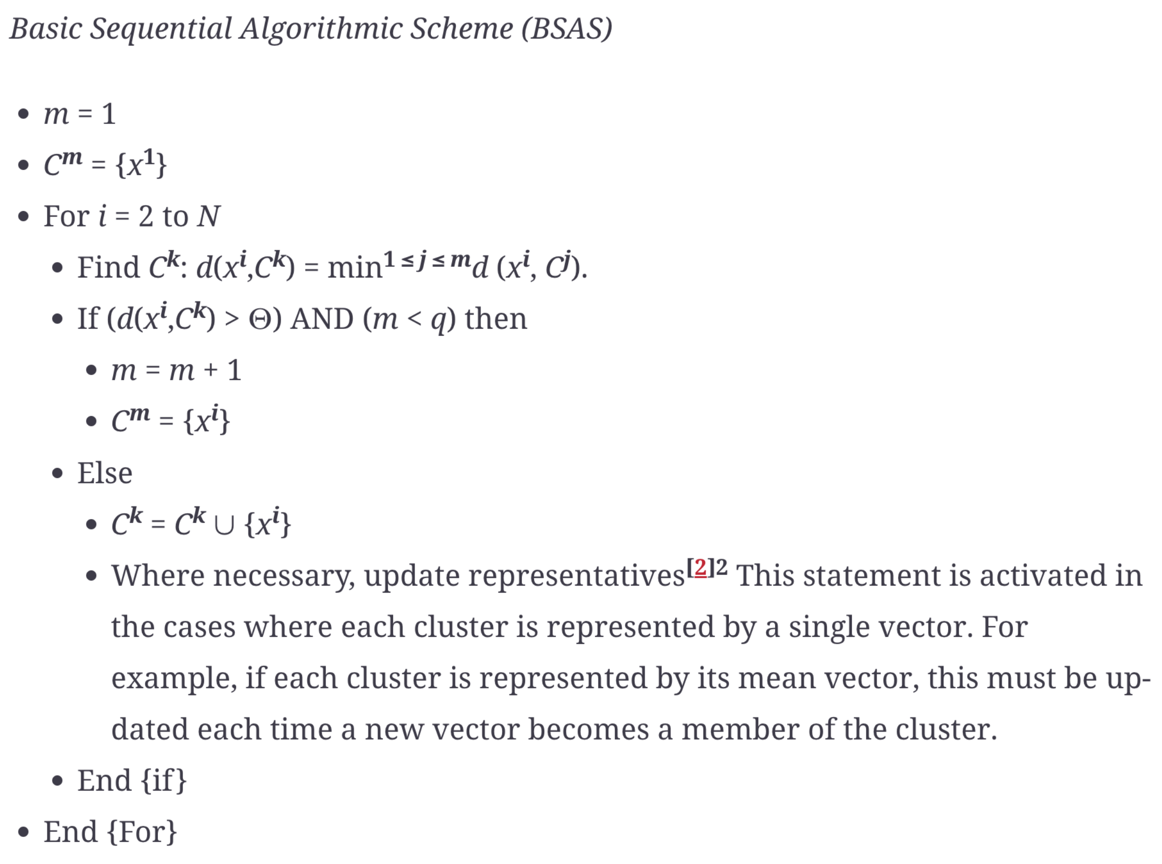

3.0
2.5
2.0
1.5
1.0
1.5
1.0
0.5
0.0
−0.5
−5
−4
−3
−2
−1
0
1

We can see that with this dataset, the LDA algorithm yielded great results. On the previous dataset I had believed that there would be some combination of body measurements that could be used to separate the male and female classes; it seems that this intuition was wrong. I am going to go forward and apply BCSA to the iris dataset so that the results are clearer, and I'll also make my function generalizable in case I would like to apply it to other datasets.

BSCA Technique

The Basic Sequential Clustering Algorithm (BSCA) or Basic Sequential Algorithm Scheme (BSAS) is a simple single pass technique that runs through each of the feature vectors and assigns them a cluster. The following is a description of the algorithm via our textbook (https://learning.oreilly.com/library/view/pattern-recognition-4th/9781597492720/kindle_split_134.html#:~:text=be%20stated%20as%3A-,Basic,-Sequential%20Algorithmic%20Scheme):

```
In [1432]: display(Image(filename='bsas_image.png') )
```



Basic Sequential Algorithmic Scheme (BSAS)

```
• $m = 1$
• $C^m = \{x^1\}$
• For $i = 2$ to $N$
    • Find $C^k$: $d(x^i, C^k) = \min_{1 \le j \le m} d(x^i, C^j)$.
    • If $(d(x^i, C^k) > \Theta)$ AND $(m < q)$ then
        • $m = m + 1$
        • $C^m = \{x^i\}$
    • Else
        • $C^k = C^k \cup \{x^i\}$
        • Where necessary, update representatives[2]2 This statement is activated in the cases where each cluster is represented by a single vector. For example, if each cluster is represented by its mean vector, this must be updated each time a new vector becomes a member of the cluster.
    • End {if}
• End {For}
```

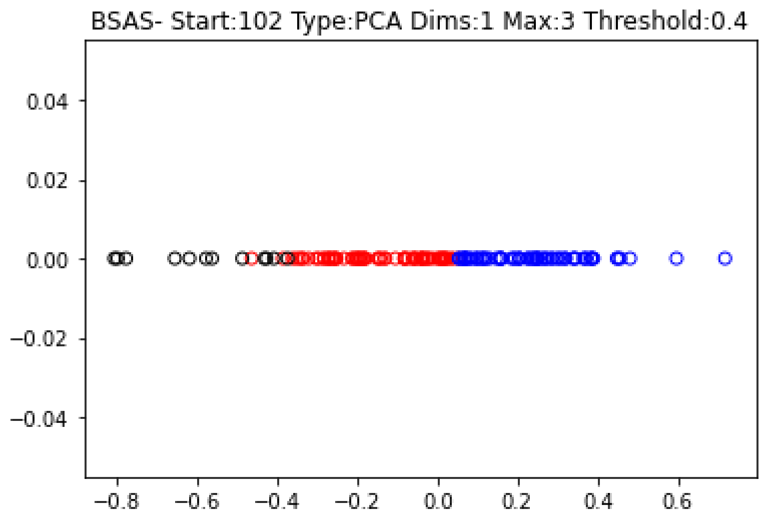
We start with a single cluster containing the first feature vector. Then for each subsequent feature vector we find the minimum distance between the feature vector and the representative vector for the clusters. If the minimum distance is within our desired threshold, we add the vector to the cluster associated with the minimum distance. If it is not within the threshold and we have not reached our cluster maximum, we create a new cluster containing the feature vector.

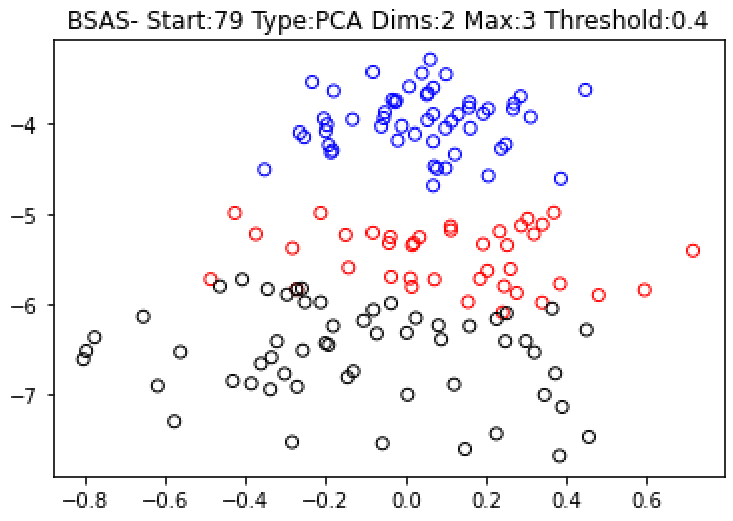
In [1433]:
```
import pandas as pd
import pprint as pp
import numpy as np
from matplotlib import pyplot as plt
import random
```

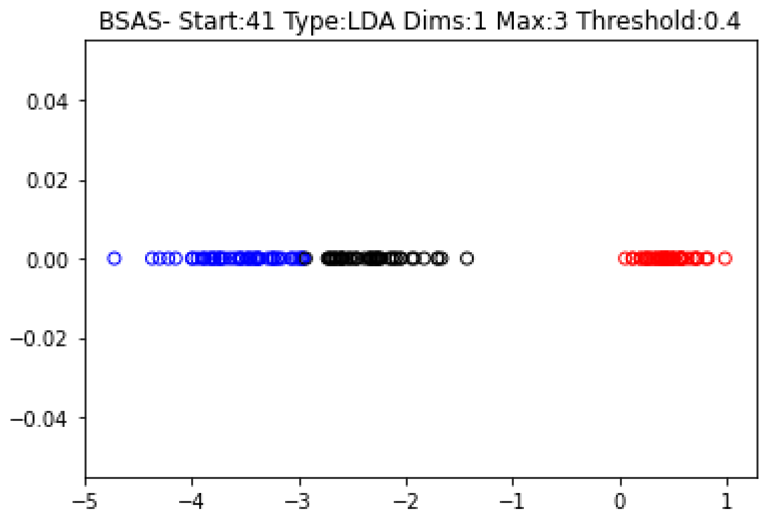
In [1434]:
```
def cluster_means(clusters):
    means = []
    for cluster in clusters:
        means.append(cluster.mean(axis=0))
    return means

def BCSA(normalized_data, threshold, max_clusters, dims=1, label=''):
    normalized_data.reset_index(drop=True, inplace=True)
    normalized_data = normalized_data.to_numpy()
    starter = random.randrange(0, len(normalized_data), 1)
    clusters = [np.array([normalized_data[starter , :]])]
    normalized_data = np.delete(normalized_data, starter, 0)
    np.random.shuffle(normalized_data)
    rep_clusters = cluster_means(clusters)
    for i in range(0, len(normalized_data)):
        feature = normalized_data[i, :]
        # Manhattan Distance
        distances = []
        for cluster in rep_clusters:
            distances.append(np.sum(np.abs(np.subtract(feature, cluster
))))
        min_index = np.argmin(distances)
        if distances[min_index] > threshold and len(clusters) < max_clus
ters:
            clusters.append(np.array([feature]))
        else:
            clusters[min_index] = np.concatenate((clusters[min_index], n
p.array([feature])), axis=0)
        rep_clusters = cluster_means(clusters)
    colors = ['r', 'b', 'k', 'g', 'c', 'm', 'y']

    if dims == 1:
        fig, ax = plt.subplots()
        for index, cluster in enumerate(clusters):
            color = colors[index]
            ax.scatter(cluster[:,0], len(cluster) * [0], edgecolors=colo
r, facecolors='none')
            ax.set_title(f'BSAS {label} {dims} dims')
    if dims == 2:
        fig, ax = plt.subplots()
        for index, cluster in enumerate(clusters):
            color = colors[index]
            ax.scatter(cluster[:,0], cluster[:, 1], edgecolors=color, fa
cecolors='none')
    ax.set_title(f'BSAS- Start:{starter} Type:{label} Dims:{dims} Max:{m
ax_clusters} Threshold:{threshold}')
    plt.savefig(f'BSAS_Type_{label}_Dims_{dims}_Max_{max_clusters}_Thres
hold_{threshold}_Start_{starter}.jpg')
    plt.show()
```

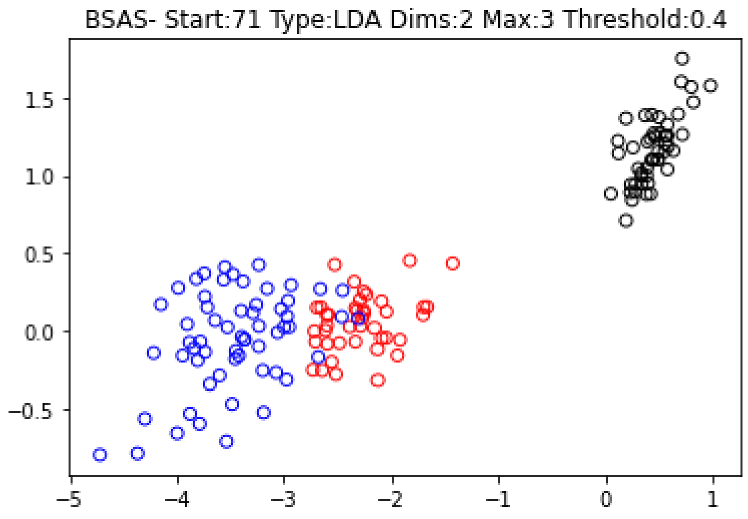
In [1435]:
```
pca_1 = pd.read_csv('../week2/pca_outputs/PCA_1_components_iris.csv')
pca_1 = pca_1.drop(columns=pca_1.columns[0])

pca_2 = pd.read_csv('../week2/pca_outputs/PCA_2_components_iris.csv')
pca_2 = pca_2.drop(columns=pca_2.columns[0])

lda_1 = pd.read_csv('../week2/lda_outputs/LDA_1_axes_iris.csv')
lda_1 = lda_1.drop(columns=lda_1.columns[0])

lda_2 = pd.read_csv('../week2/lda_outputs/LDA_2_axes_iris.csv')
lda_2 = lda_2.drop(columns=lda_2.columns[0])
```

In [1436]:
```
BCSA(pca_1, .4, 3, 1, label='PCA')
BCSA(pca_2, .4, 3, 2, label='PCA')
BCSA(lda_1, .4, 3, 1, label='LDA')
BCSA(lda_2, .4, 3, 2, label='LDA')
```


BSAS- Start:102 Type:PCA Dims:1 Max:3 Threshold:0.4
0.04
0.02
0.00
−0.02
−0.04
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6


BSAS- Start:79 Type:PCA Dims:2 Max:3 Threshold:0.4
−4
−5
−6
−7
−0.8
−0.6
−0.4
−0.2
0.0
0.2
0.4
0.6

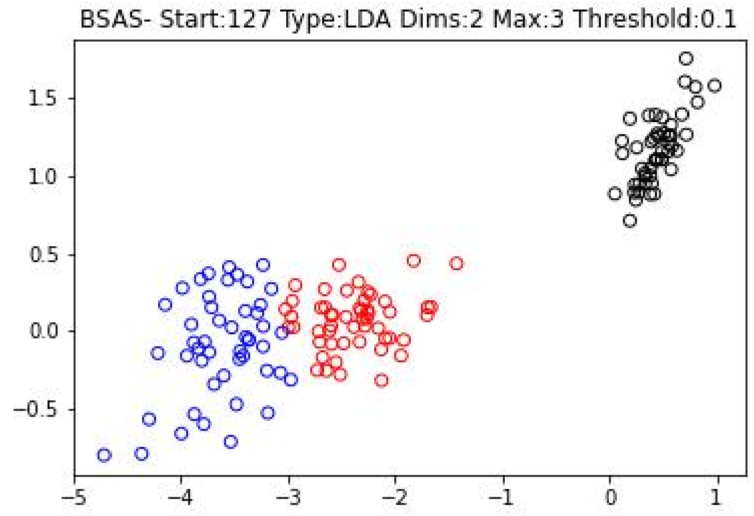

BSAS- Start:41 Type:LDA Dims:1 Max:3 Threshold:0.4
0.04
0.02
0.00
−0.02
−0.04
−5
−4
−3
−2
−1
0
1

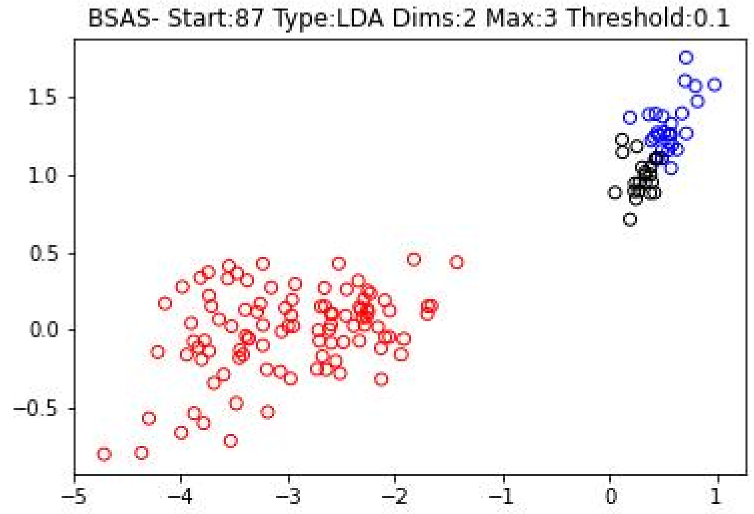


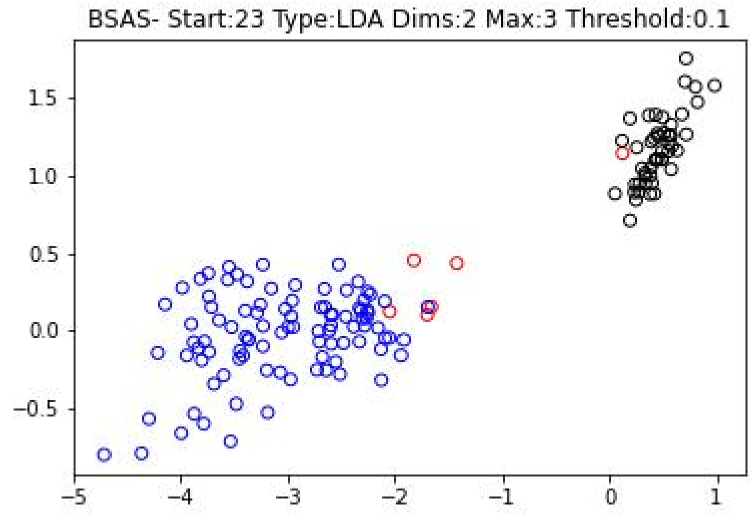
Results

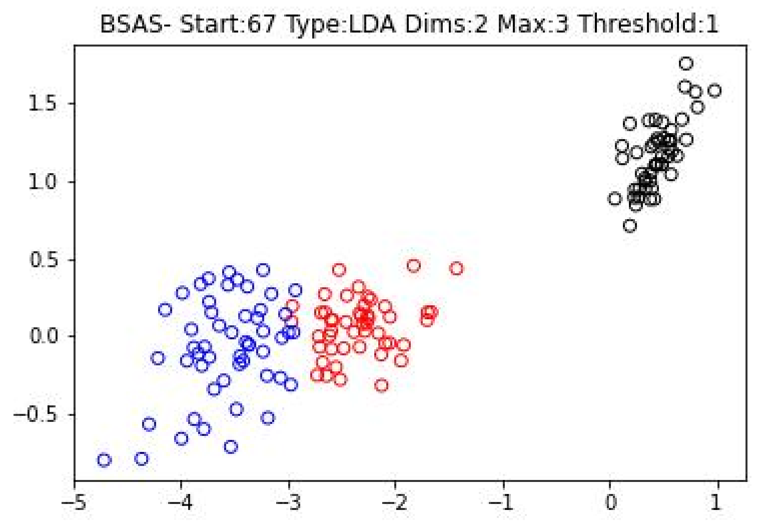
We can see from the graphs above that the algorithm does establish good separation between clusters, but that it does not always match the classes that we are aware of beforehand. I included a step to randomize the starting point that is passed to the initial cluster, and to shuffle the datapoints to see what the effects of the sequence are. The following graphs show 3 different results of running the algorithm on the 2 dimensional LDA datapoints.

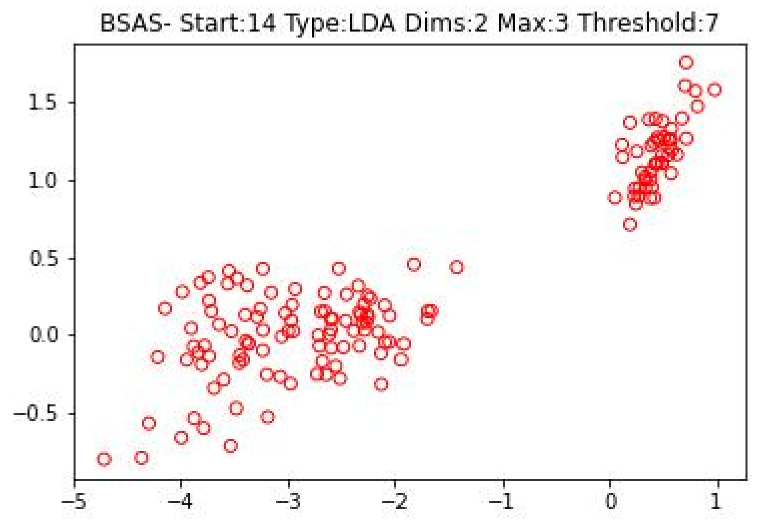
In [1437]:
```
display(Image(filename='BSAS_Type_LDA_Dims_2_Max_3_Threshold_0.1_Start_1
27.jpg') )
display(Image(filename='BSAS_Type_LDA_Dims_2_Max_3_Threshold_0.1_Start_8
7.jpg') )
display(Image(filename='BSAS_Type_LDA_Dims_2_Max_3_Threshold_0.1_Start_2
3.jpg') )
display(Image(filename='BSAS_Type_LDA_Dims_2_Max_3_Threshold_1_Start_67.
jpg') )
display(Image(filename='BSAS_Type_LDA_Dims_2_Max_3_Threshold_7_Start_14.
jpg') )
```


BSAS- Start:127 Type:LDA Dims:2 Max:3 Threshold:0.1
1.5
1.0
0.5
0.0
−0.5
−5
−4
−3
−2
−1
0
1


BSAS- Start:87 Type:LDA Dims:2 Max:3 Threshold:0.1
1.5
1.0
0.5
0.0
−0.5
−5
−4
−3
−2
−1
0
1


BSAS- Start:23 Type:LDA Dims:2 Max:3 Threshold:0.1
1.5
1.0
0.5
0.0
−0.5
−5
−4
−3
−2
−1
0
1





We can see that the results are heavily dependent on the sequence. In the first image we get results that closely represent the known classes. The second image has good separation, but does not match the known classes. The third image shows relatively poor separation overall. The fourth image demonstrates that we can achieve good separation through a range of thresholds, and the fifth shows the results of increasing the threshold to encompass the entire range of values.

Conclusion

From these results, we can conclude that BCSA is capable of yielding good separation, but it is heavily dependent on the sequence that it is performed in. For my dataset there is a wide range of potential thresholds that still yield good separation. It is clear that BCSA is not going to consistently give good results without modification.