MACHINE PROBLEM 2

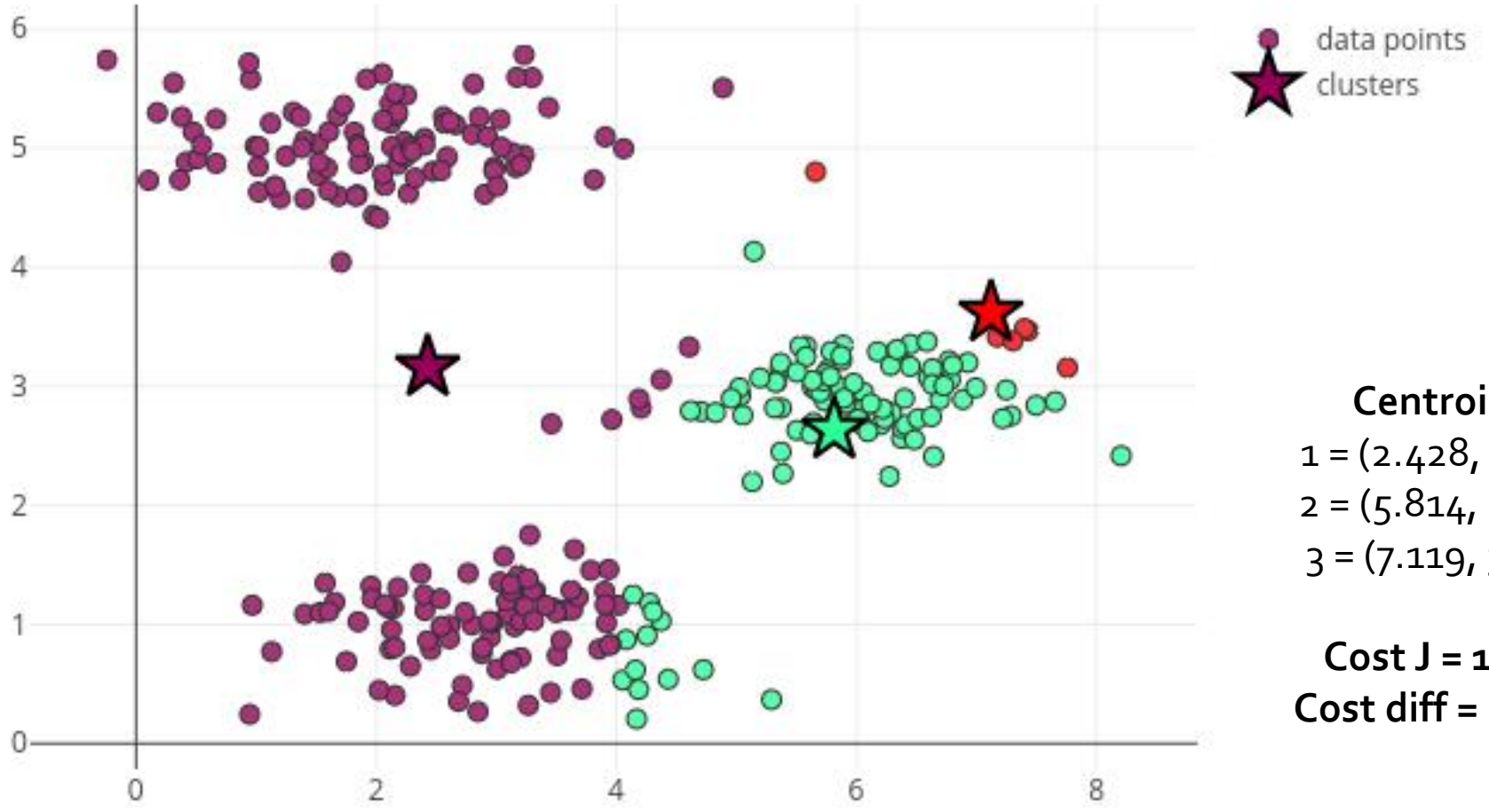
Image Compression Using K-Means

PART ONE

The K-Means Algorithm

***Implementation by Jose Arniel Pama**

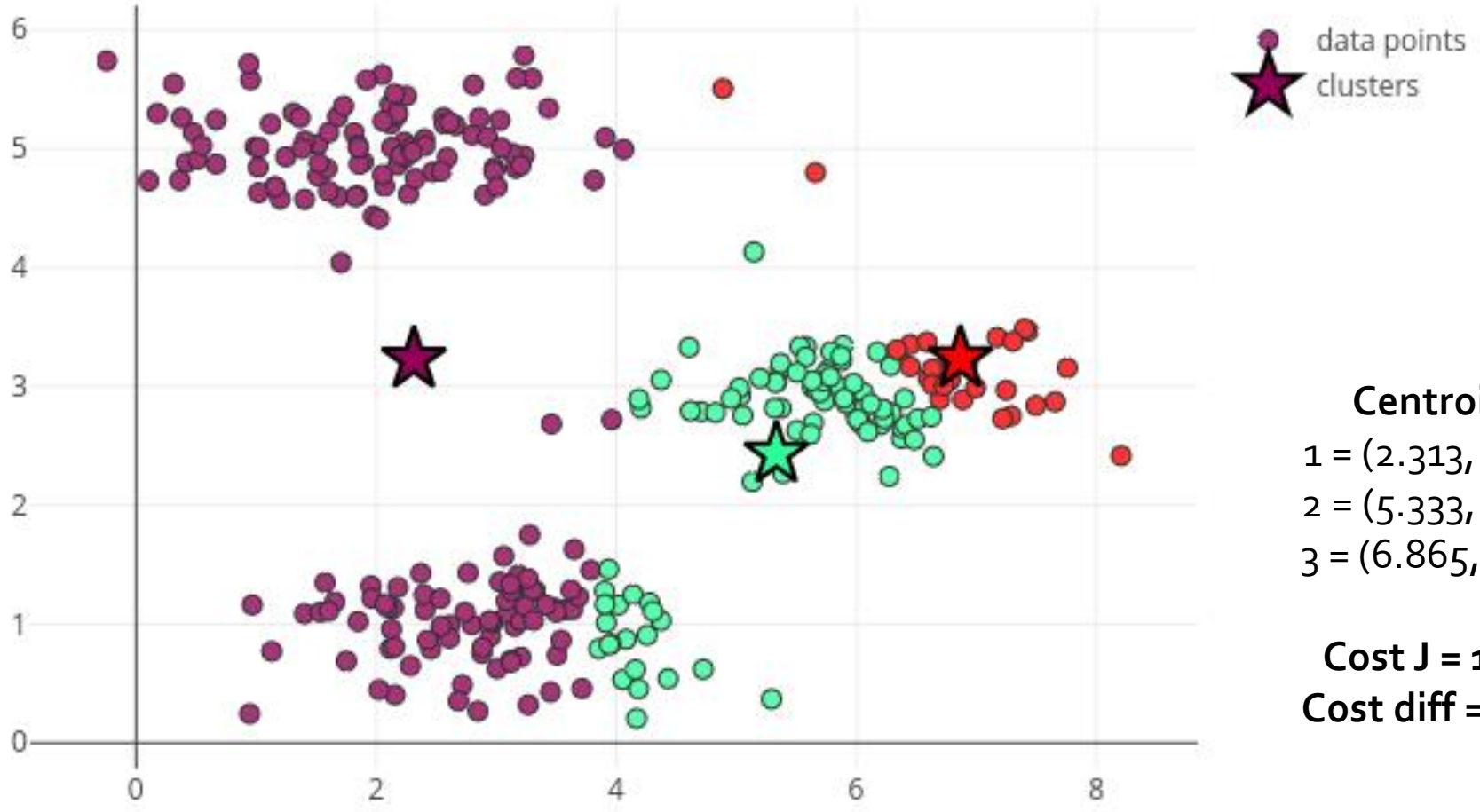
Iteration 1



Centroids:
1 = (2.428, 3.158)
2 = (5.814, 2.634)
3 = (7.119, 3.617)

Cost J = 1.907
Cost diff = -1.907

Iteration 2



Centroids:

1 = (2.313, 3.228)

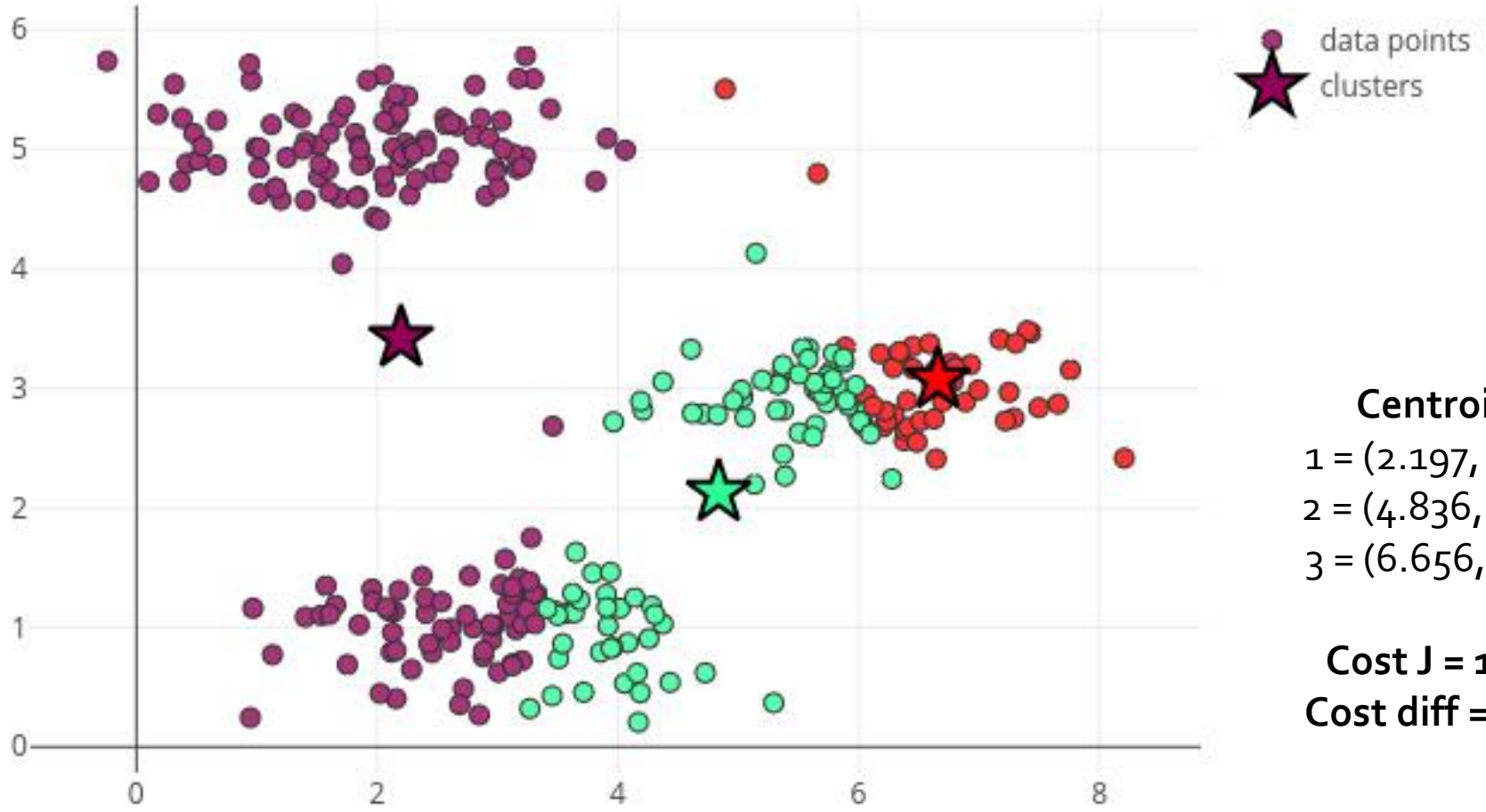
2 = (5.333, 2.432)

3 = (6.865, 3.232)

Cost J = 1.695

Cost diff = 0.211

Iteration 3



Centroids:

1 = (2.197, 4.421)

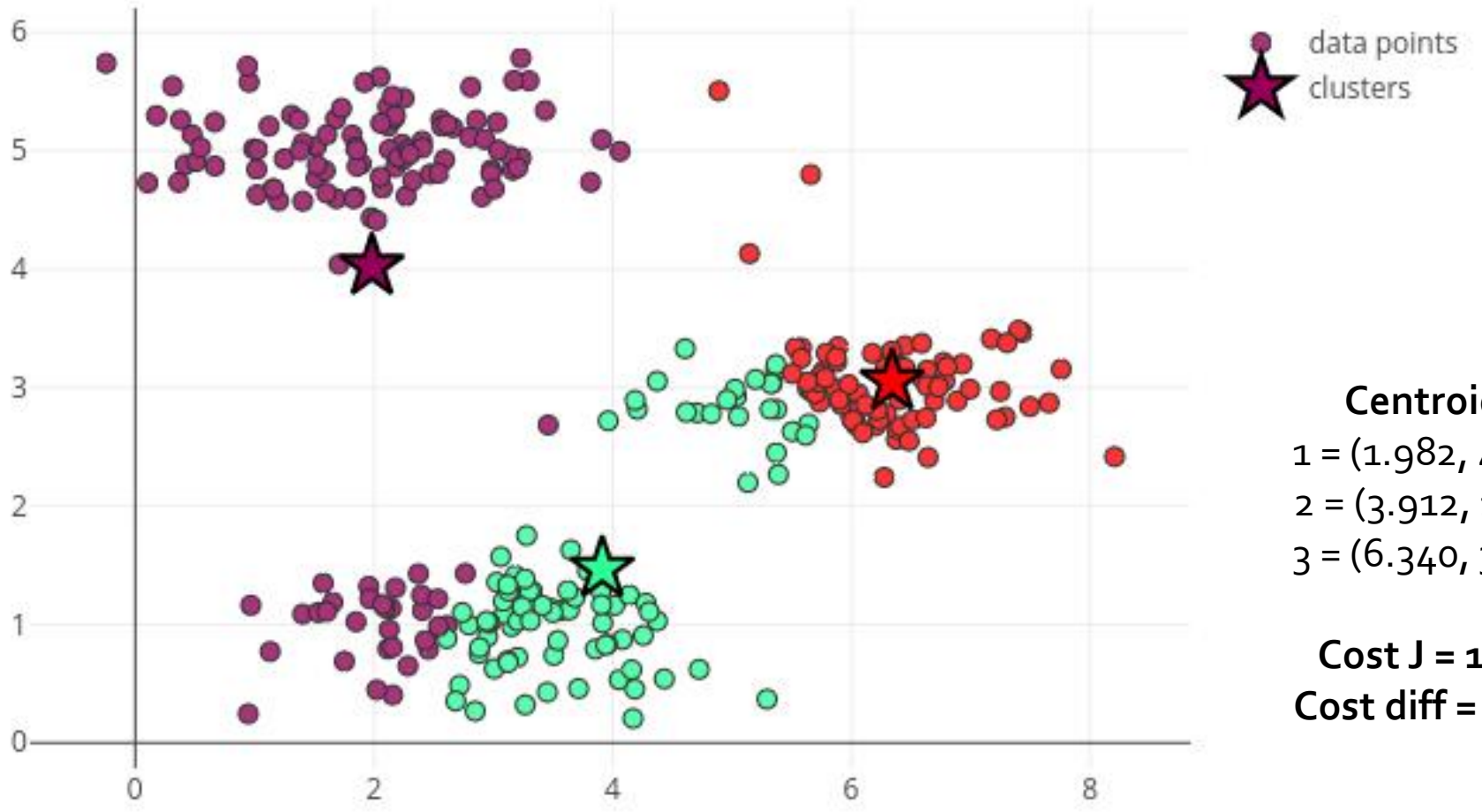
2 = (4.836, 2.129)

3 = (6.656, 3.075)

Cost J = 1.649

Cost diff = 0.047

Iteration 4



Centroids:

1 = (1.982, 4.025)

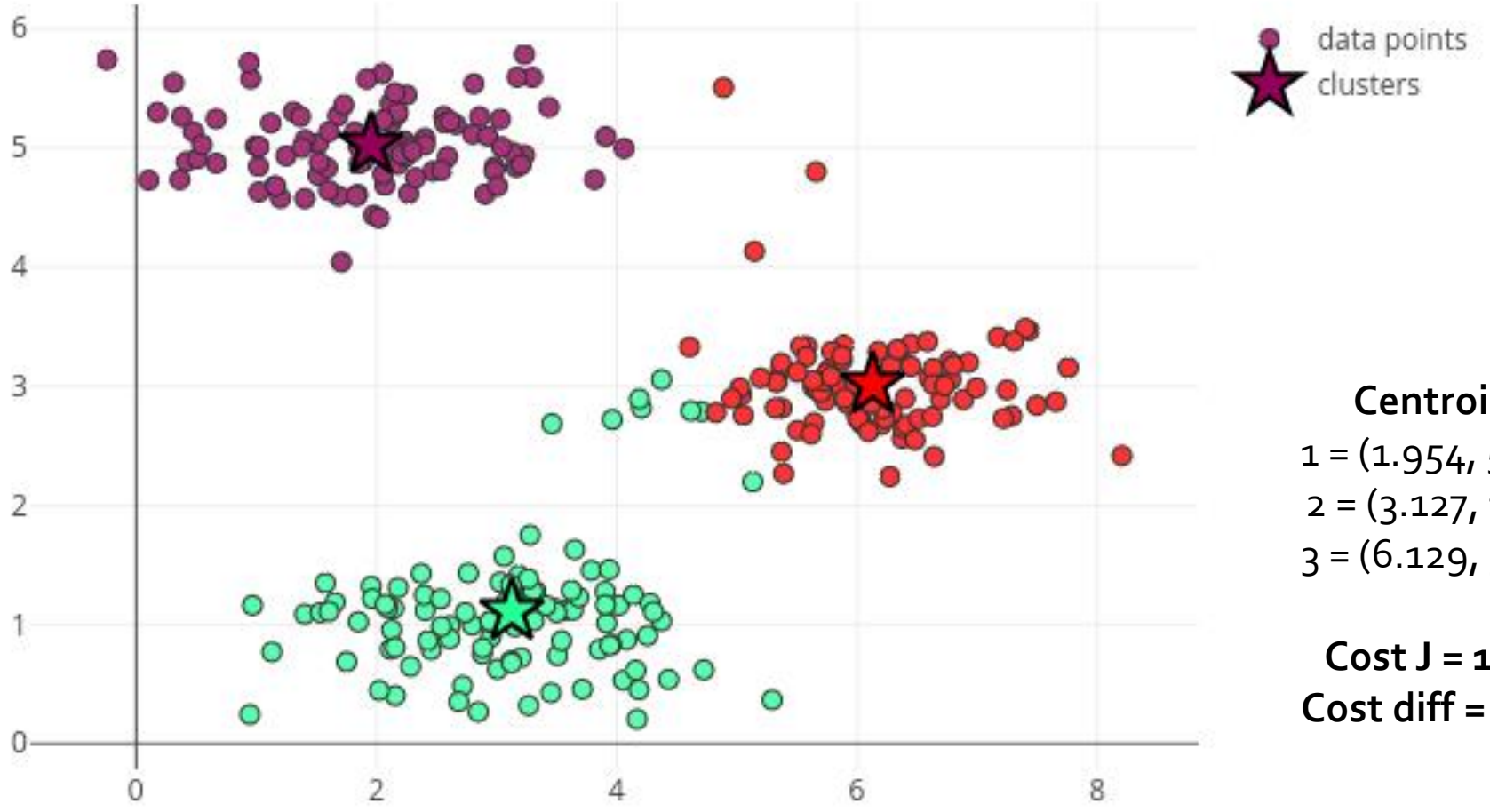
2 = (3.912, 1.470)

3 = (6.340, 3.054)

Cost J = 1.535

Cost diff = 0.114

Iteration 5



Centroids:

1 = (1.954, 5.026)

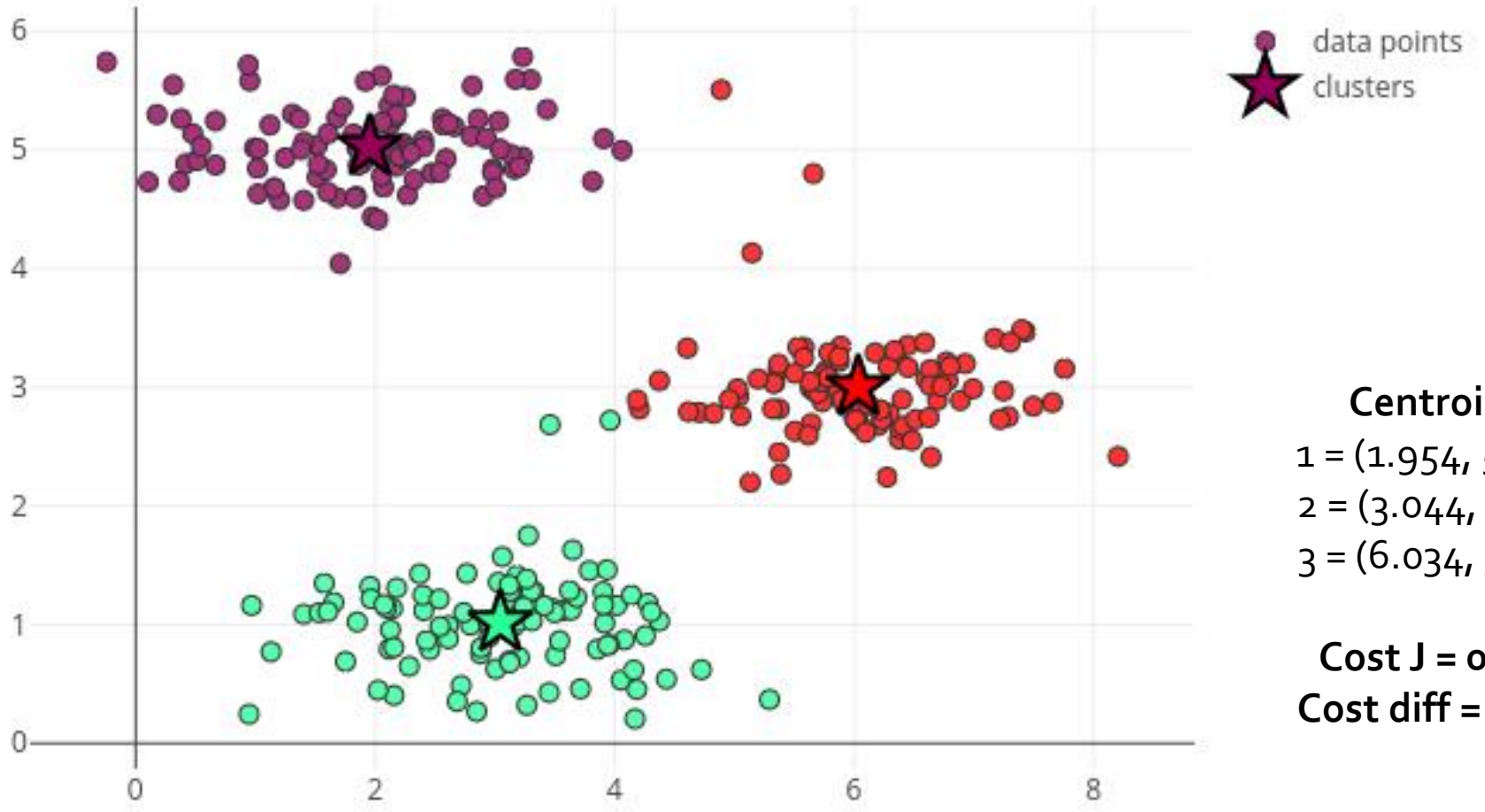
2 = (3.127, 1.112)

3 = (6.129, 3.016)

Cost J = 1.095

Cost diff = 0.439

Iteration 6



Centroids:

1 = (1.954, 5.026)

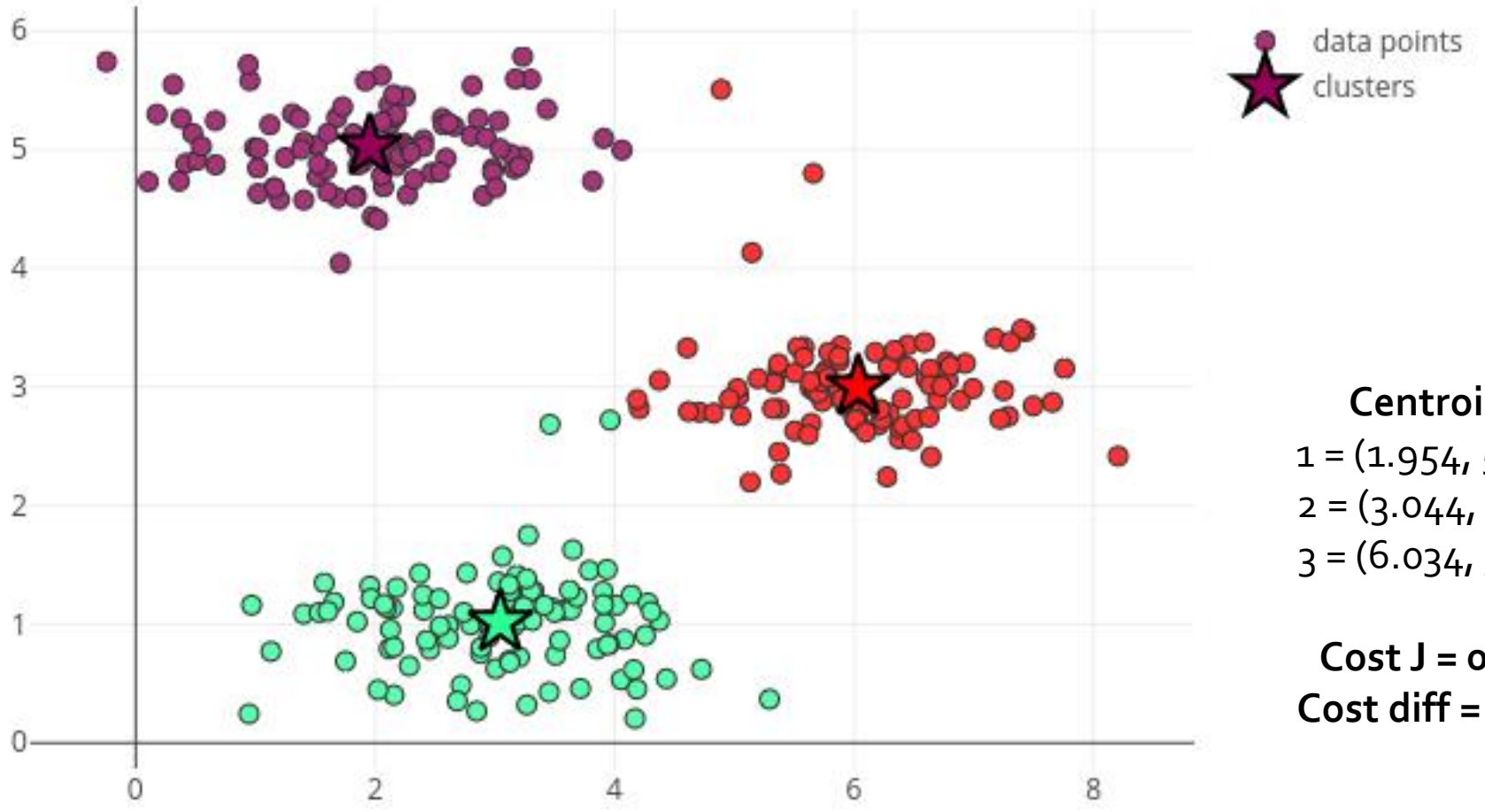
2 = (3.044, 1.015)

3 = (6.034, 3.001)

Cost J = 0.798

Cost diff = 0.297

Iteration 7



Centroids:

1 = (1.954, 5.026)

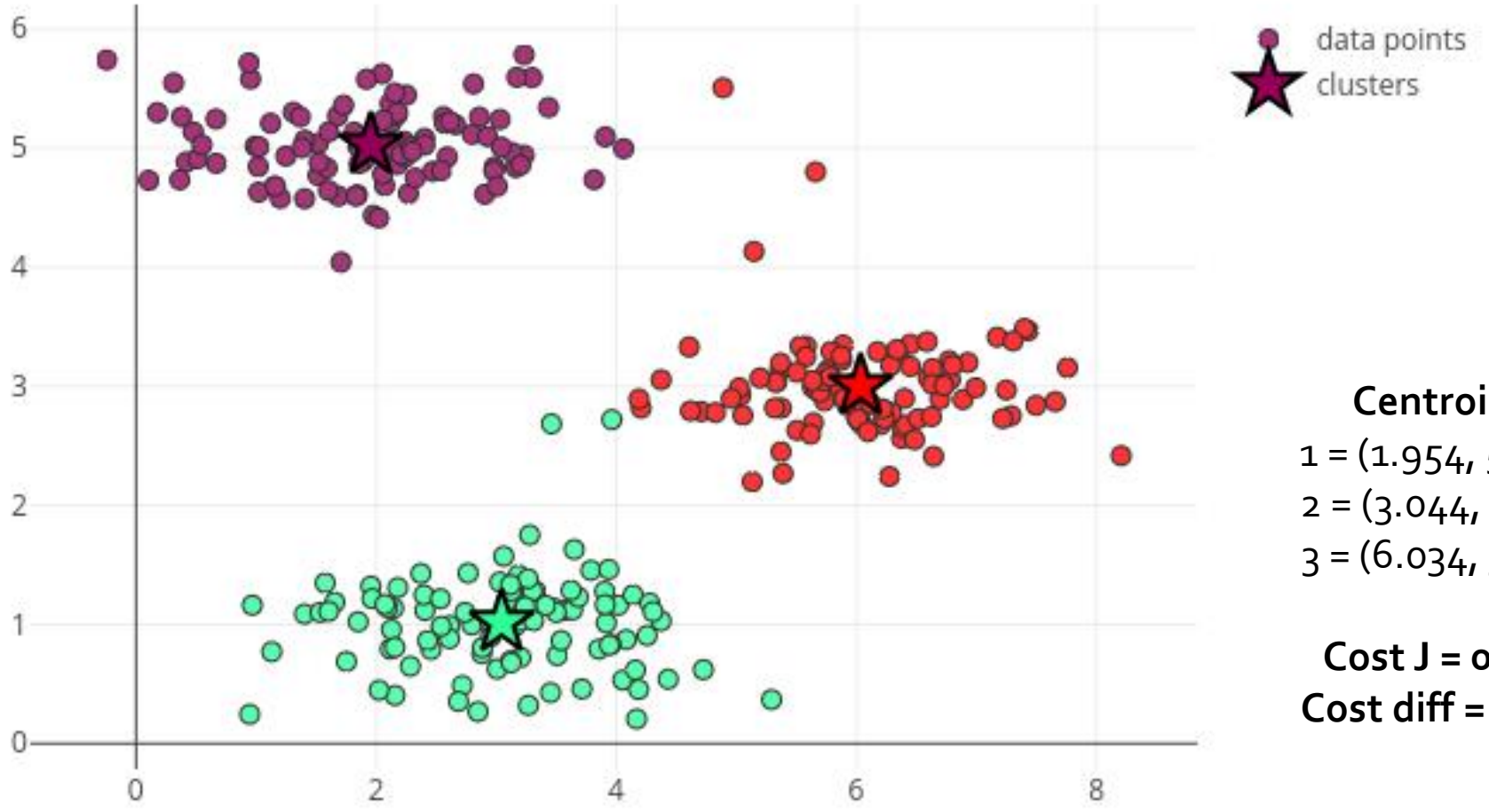
2 = (3.044, 1.015)

3 = (6.034, 3.001)

Cost J = 0.794

Cost diff = 0.004

Iteration 8



Centroids:

1 = (1.954, 5.026)

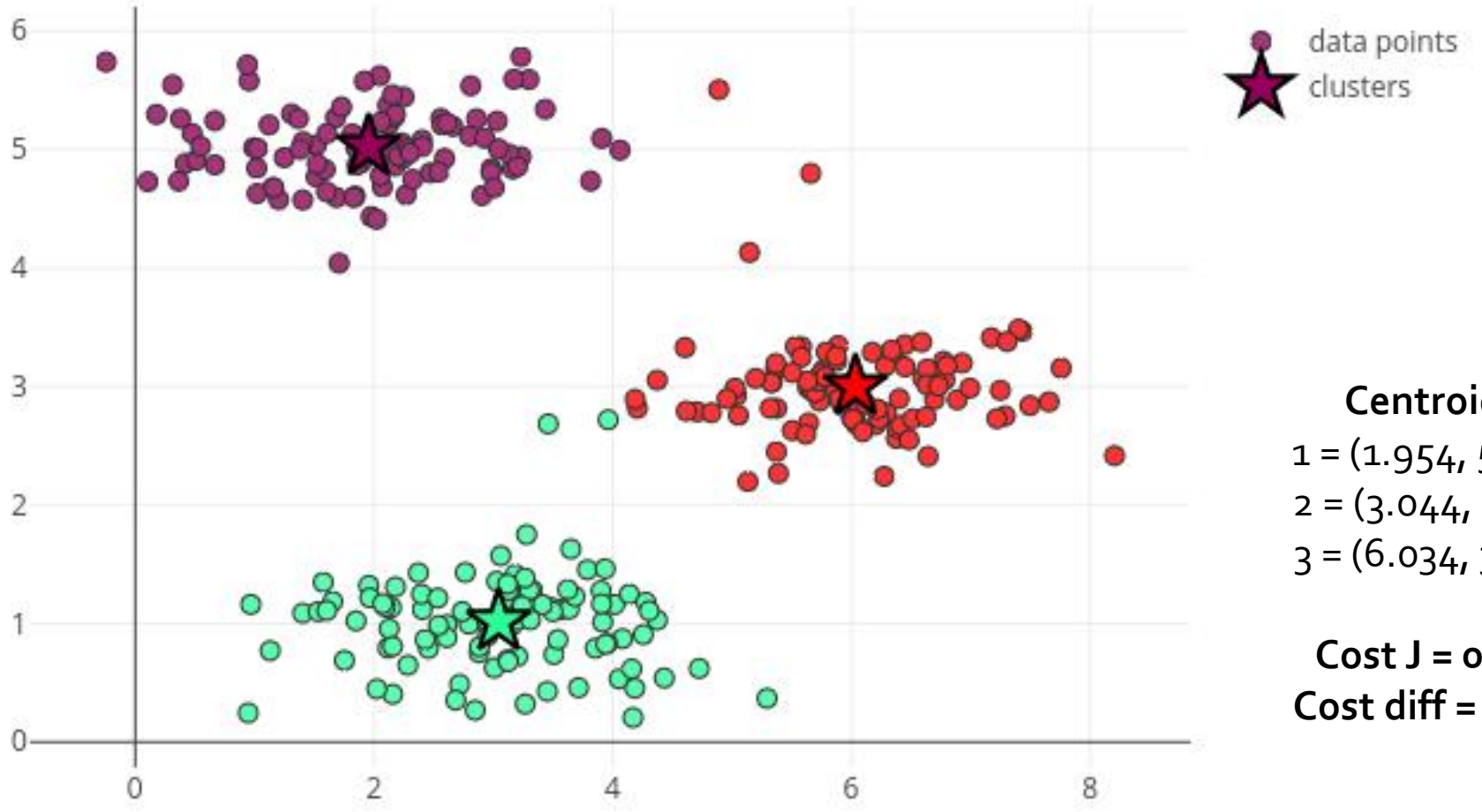
2 = (3.044, 1.015)

3 = (6.034, 3.001)

Cost J = 0.794

Cost diff = 0.000

Iteration 9



Centroids:

1 = (1.954, 5.026)

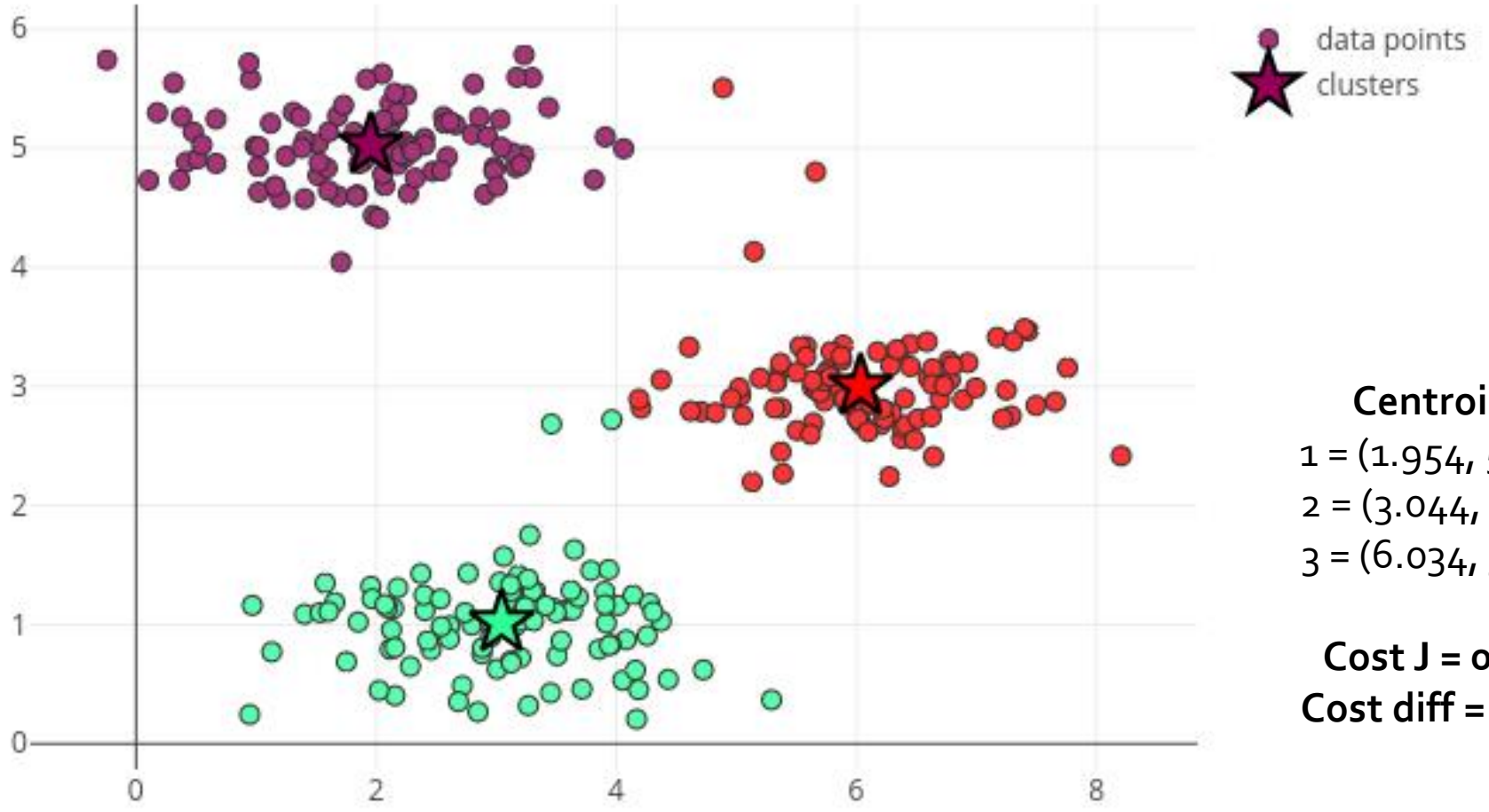
2 = (3.044, 1.015)

3 = (6.034, 3.001)

Cost J = 0.794

Cost diff = 0.000

Iteration 10



Centroids:

1 = (1.954, 5.026)

2 = (3.044, 1.015)

3 = (6.034, 3.001)

Cost J = 0.794

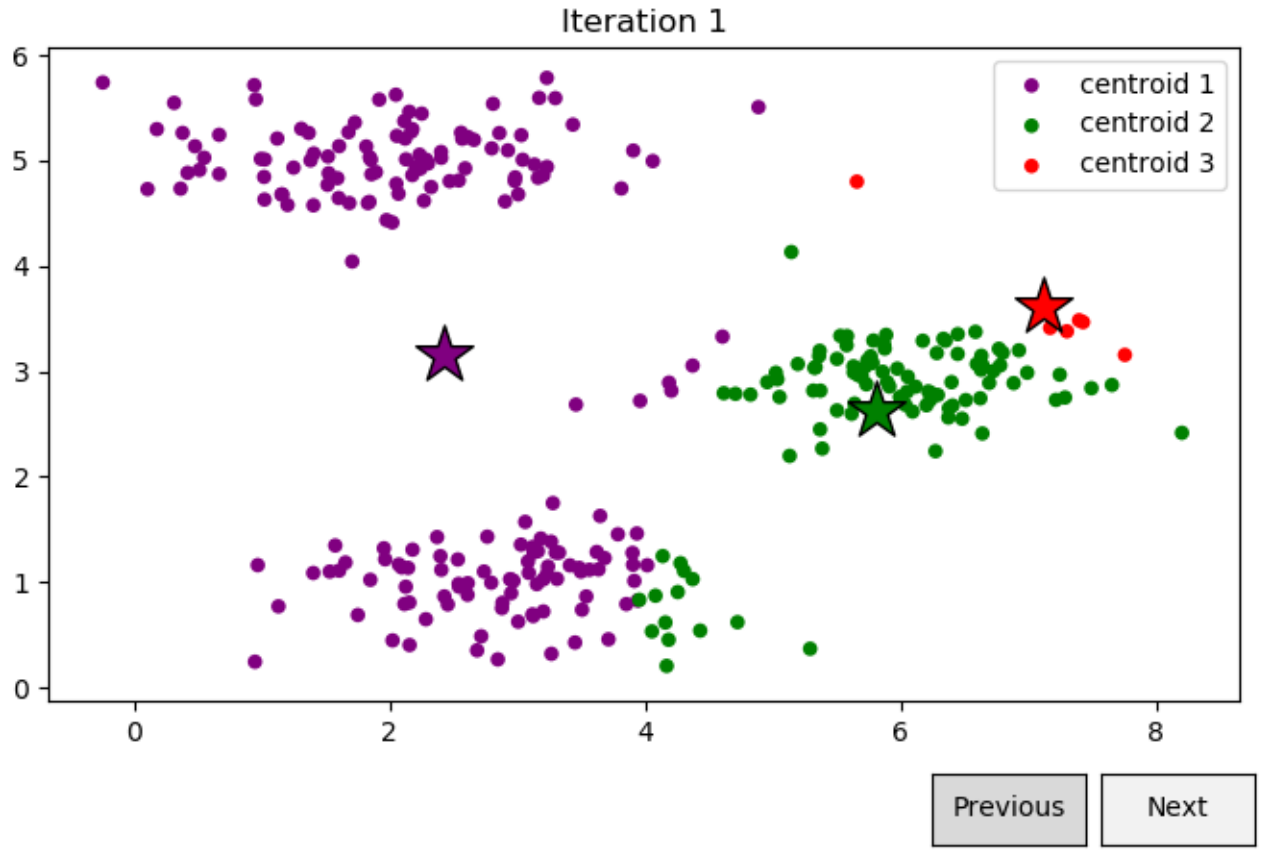
Cost diff = 0.000

PART ONE

The K-Means Algorithm

***Implementation by Arvin Bonganay**

Iteration 1



Centroids:

1 = (2.428, 3.158)

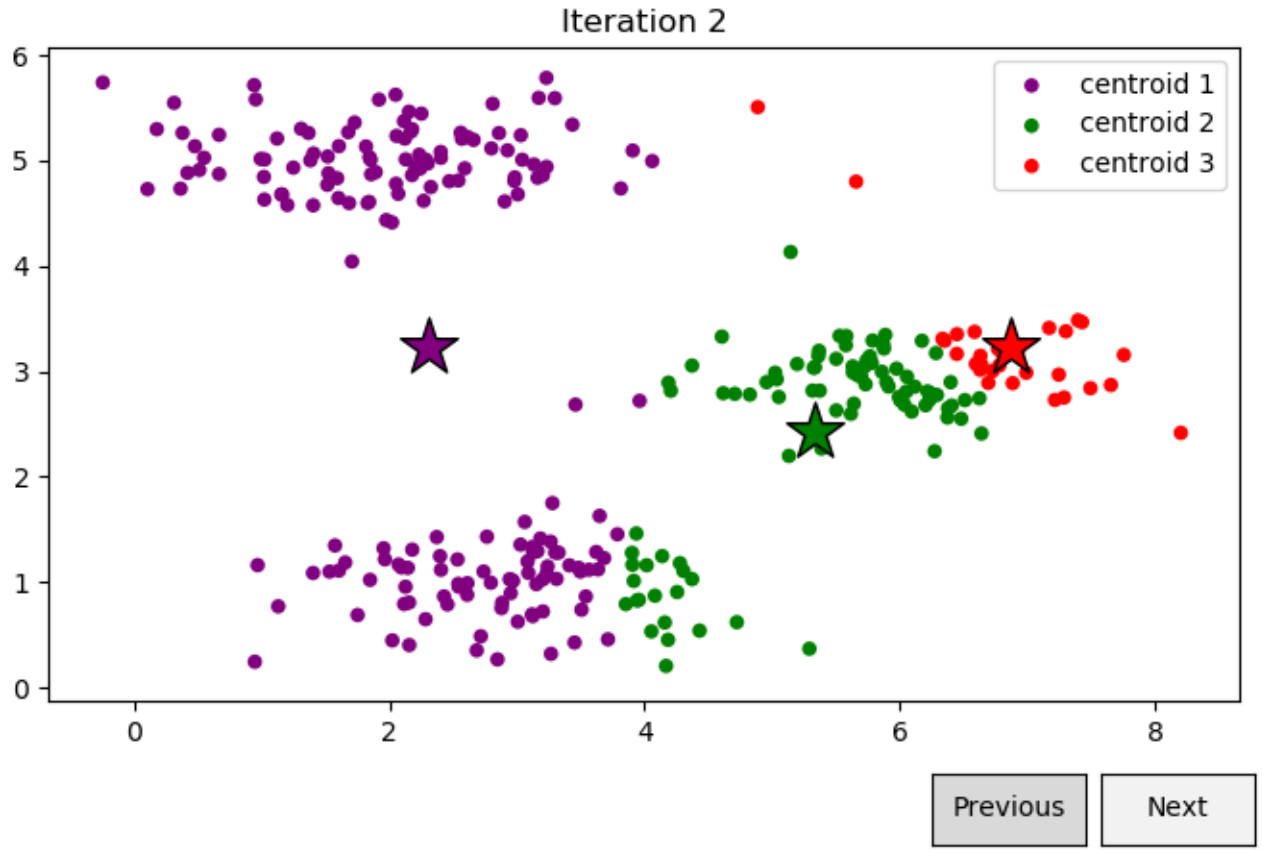
2 = (5.814, 2.634)

3 = (7.119, 3.617)

Cost J = 1.907

Cost diff = -1.907

Iteration 2



Centroids:

1 = (2.313, 3.228)

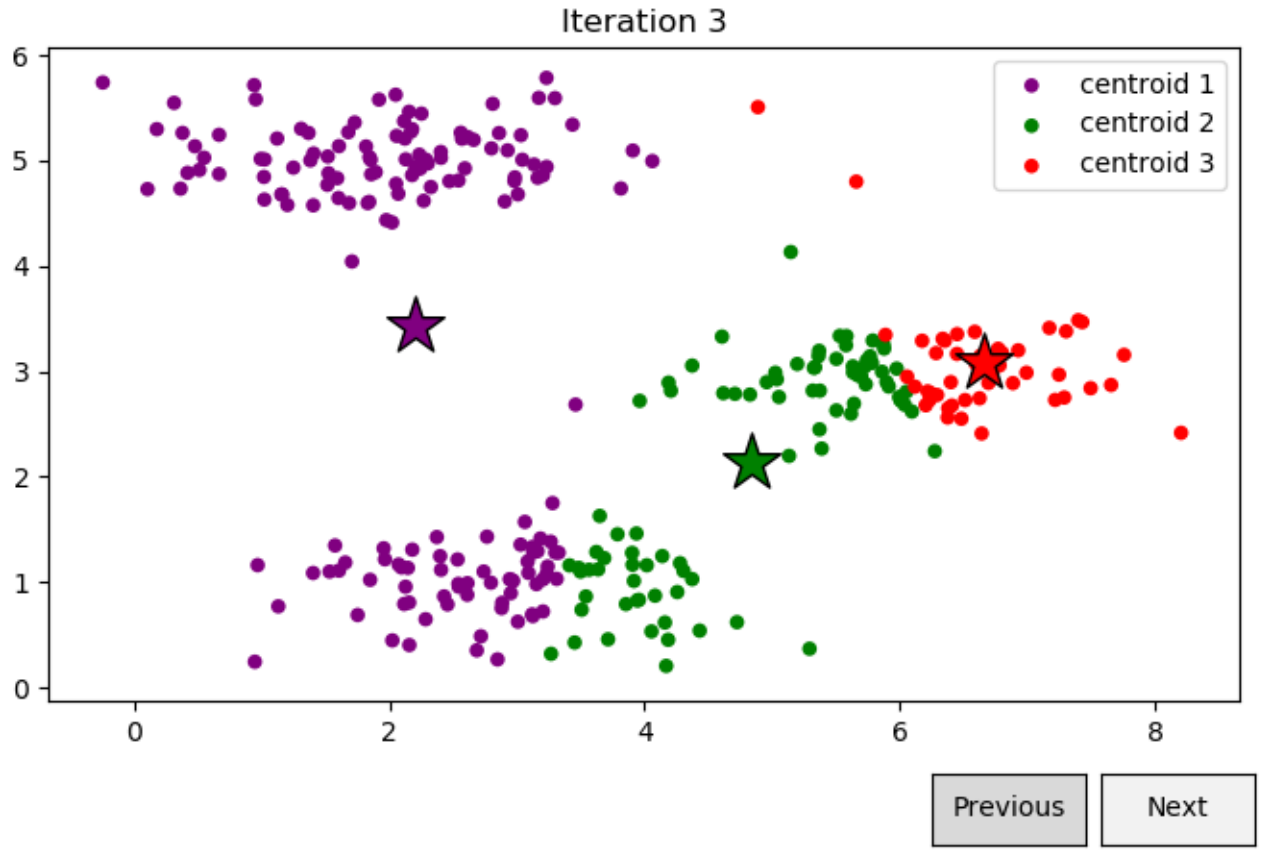
2 = (5.333, 2.432)

3 = (6.865, 3.232)

Cost J = 1.695

Cost diff = 0.211

Iteration 3



Centroids:

1 = (2.197, 4.421)

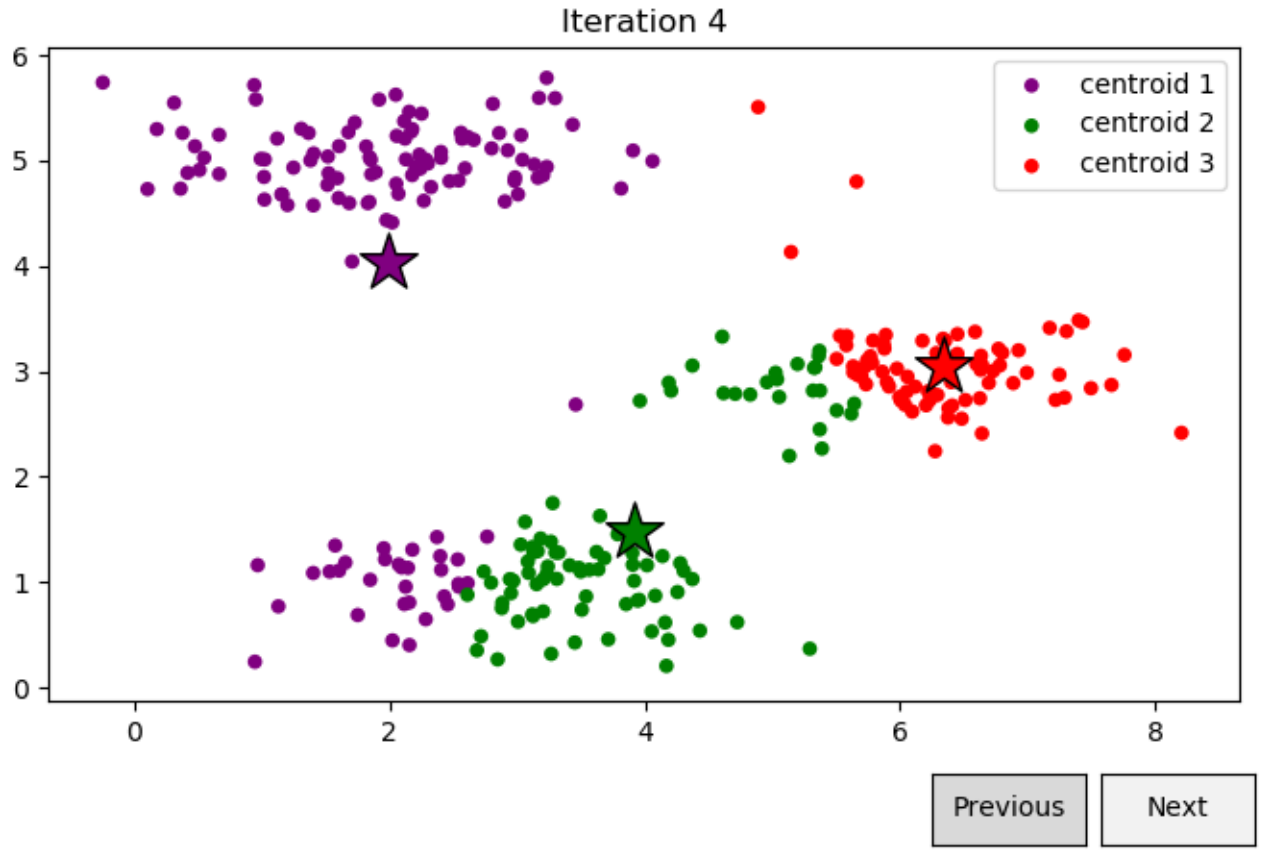
2 = (4.836, 2.129)

3 = (6.656, 3.075)

Cost J = 1.649

Cost diff = 0.047

Iteration 4



Centroids:

1 = (1.982, 4.025)

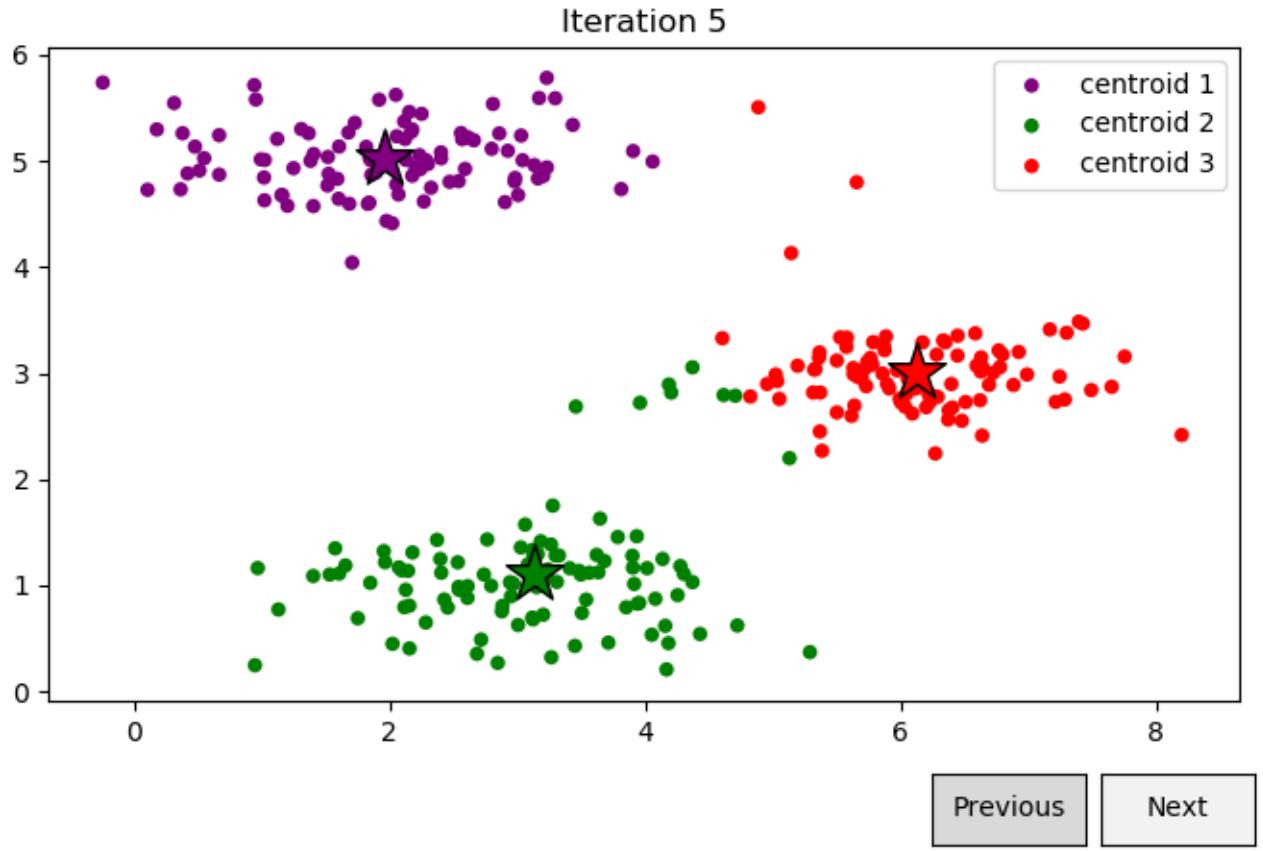
2 = (3.912, 1.470)

3 = (6.340, 3.054)

Cost J = 1.535

Cost diff = 0.114

Iteration 5



Centroids:

1 = (1.954, 5.026)

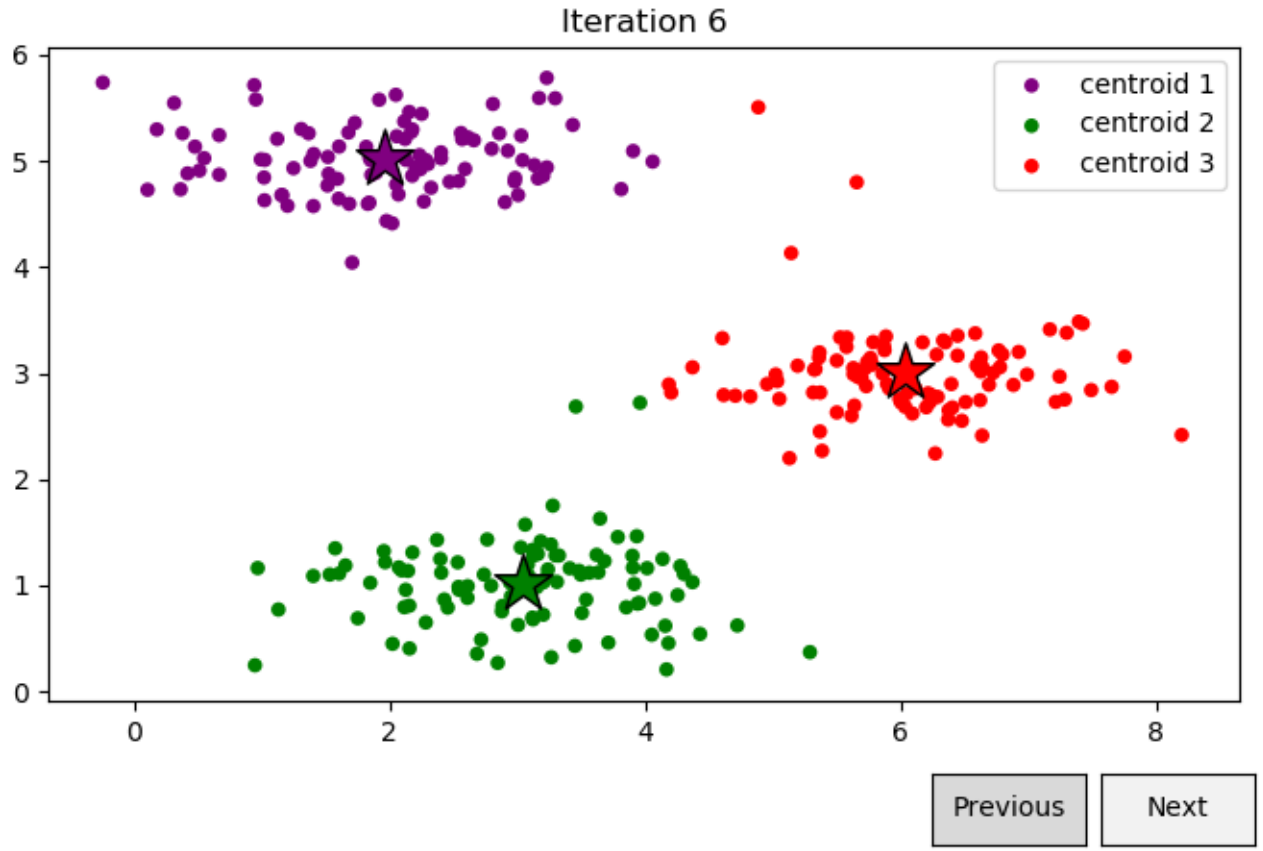
2 = (3.127, 1.112)

3 = (6.129, 3.016)

Cost J = 1.095

Cost diff = 0.439

Iteration 6



Centroids:

1 = (1.954, 5.026)

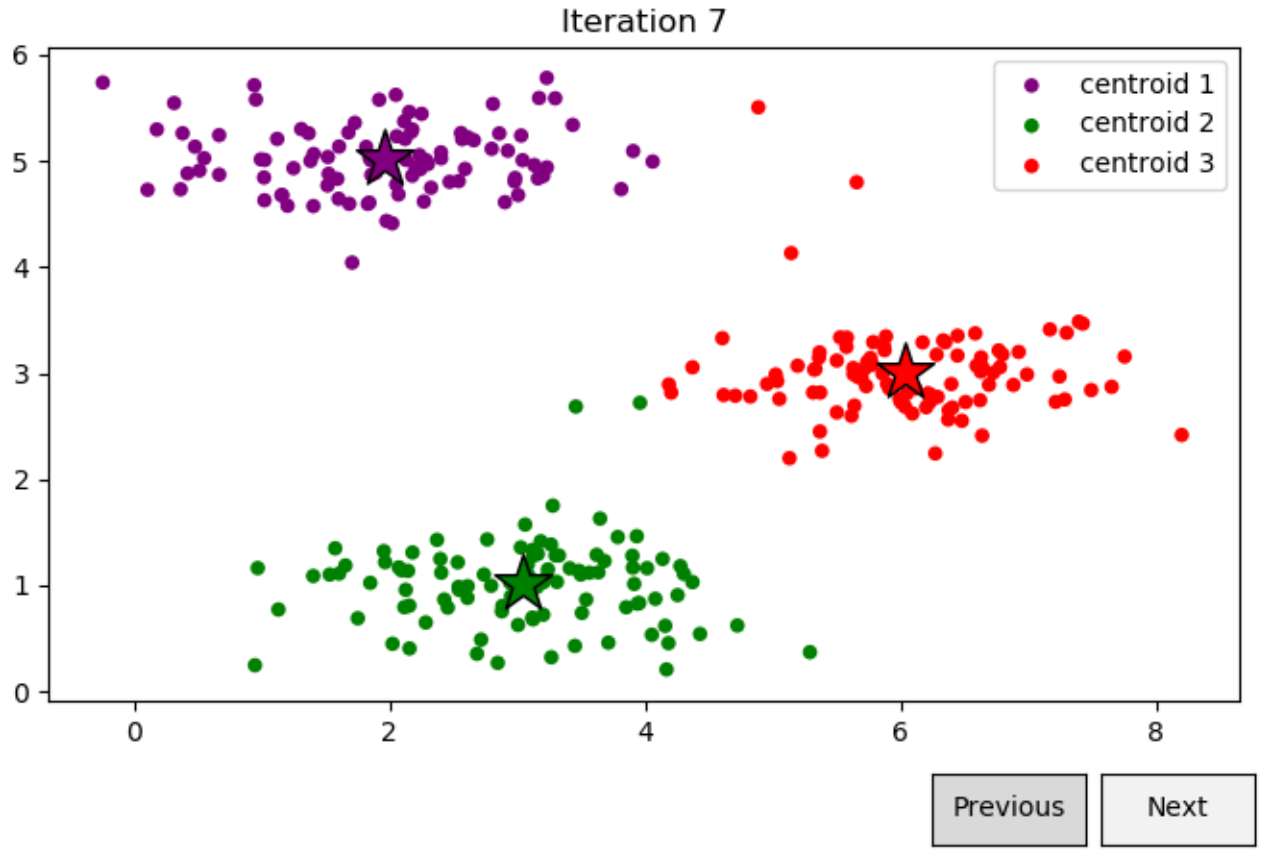
2 = (3.044, 1.015)

3 = (6.034, 3.001)

Cost J = 0.798

Cost diff = 0.297

Iteration 7



Centroids:

1 = (1.954, 5.026)

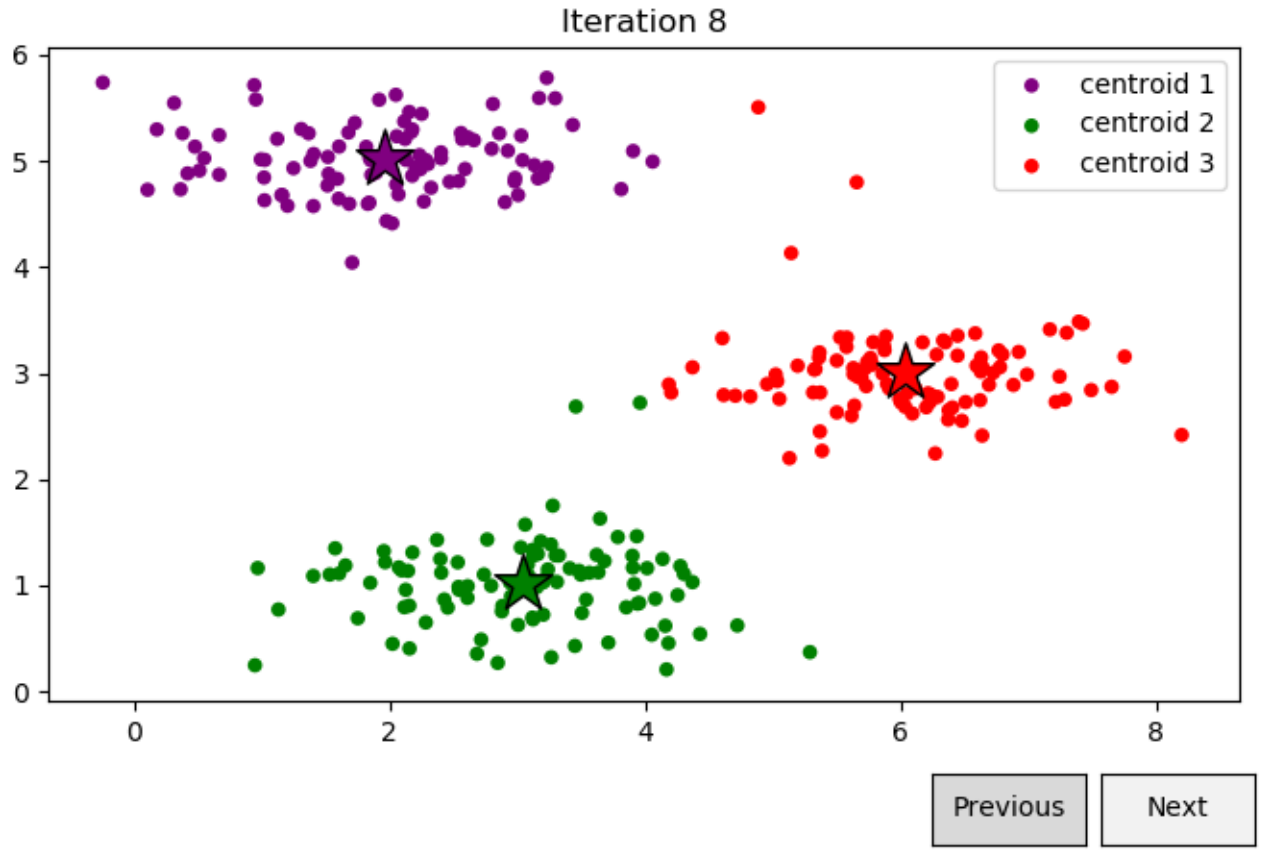
2 = (3.044, 1.015)

3 = (6.034, 3.001)

Cost J = 0.794

Cost diff = 0.004

Iteration 8



Centroids:

1 = (1.954, 5.026)

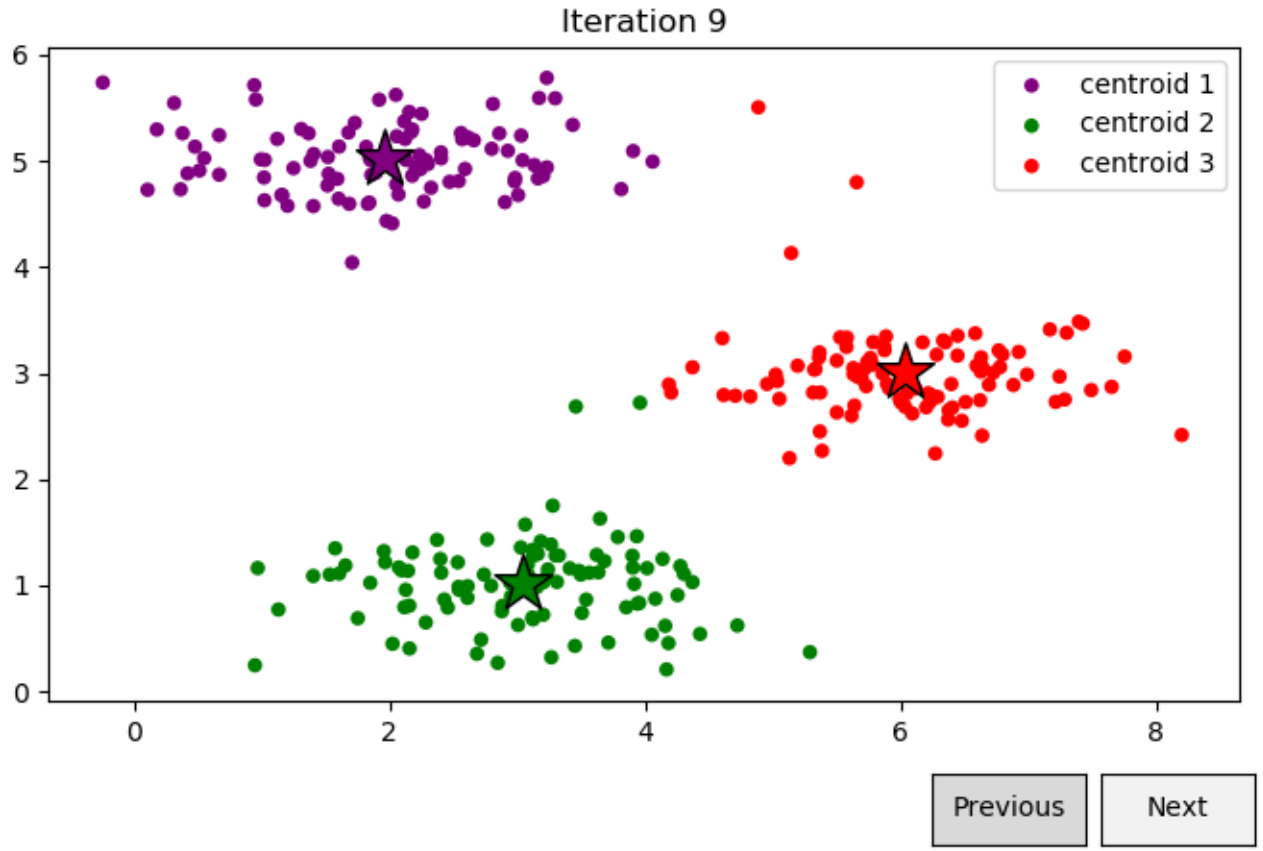
2 = (3.044, 1.015)

3 = (6.034, 3.001)

Cost J = 0.794

Cost diff = 0.000

Iteration 9



Centroids:

1 = (1.954, 5.026)

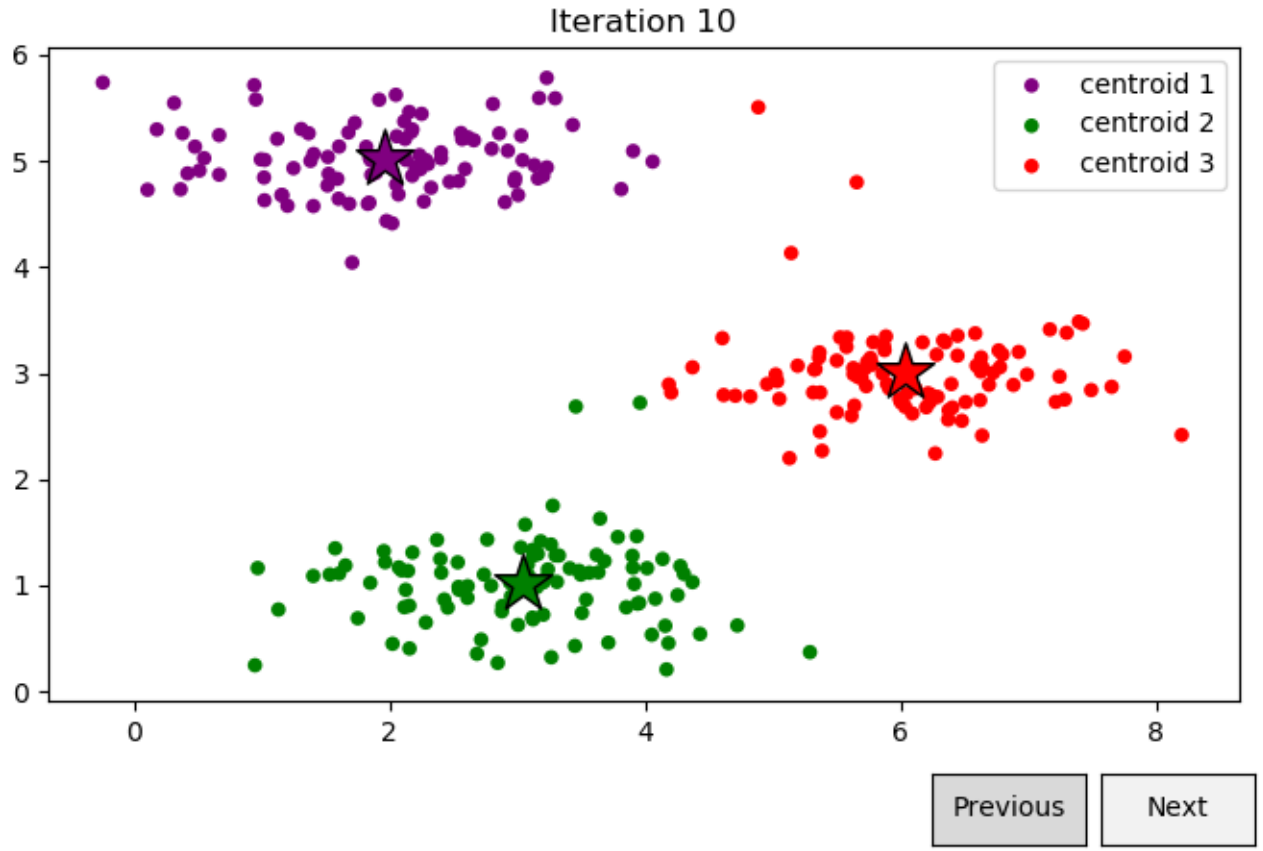
2 = (3.044, 1.015)

3 = (6.034, 3.001)

Cost J = 0.794

Cost diff = 0.000

Iteration 10



Centroids:

1 = (1.954, 5.026)

2 = (3.044, 1.015)

3 = (6.034, 3.001)

Cost J = 0.794

Cost diff = 0.000

Interpretation

- As the number of iteration progresses, the centroids get closer and closer to their ideal values—that is, they stopped at the values where they are closest to the different data sets they encompass.
- In the case of our data set, the centroid values stopped changing in the 6^{th} iteration until last.
- The centroids reached their optimum values during the 8^{th} iteration when there was already no difference between the previous cost and the current cost (i.e. zero cost difference) .
- The algorithm divided the data set into three clusters. Although each cluster may not contain the same number of data, it is guaranteed that the distance between each data point and the center of the cluster it belongs has reached its minimum in the final iteration.

PART TWO

Image Compression

Pama's Implementation

Original Image

Compressed Image

Bonganay's Implementation

Original Image

Compressed Image

Interpretation

- The compression algorithm works since it has reduced the possible RGB values from 16384 (i.e. 128 x 128) to 16 RGB values only.
- Although it is effective, the actual image compression process consumes a significant time and space complexity. The k-means algorithm for instance is applied to find 16 clusters for 16384 RGB values in 10 iterations, among other transformation processes (e.g. converting from image to RGB, then converting RGB to final compressed image).
- The final compressed image only has 16 possible colors or RGB combinations but it still appears similar to the original image. It is because the centroid values are closest to the actual values.
- Our two implementations have produced somewhat different RGB intensities (the other appears lighter over the other). This may be attributed to the randomization of the initial centroids.

Contributions

- **Bonganay, Arvin**
  - Implemented Part One
    - Used mathplotlib (offline) in generating the scatter plot
  - Implemented Part Two

- **Pama, Jose Arniel**
  - Implemented Part One
    - Used plot.ly (online) in generating the scatter plot
  - Implemented Part Two
  - Crafted the documentation

***CMSC 170 afternoon class**