Assignment06

May 8, 2019

20132915 Nam, Geun Woo

[K-means clustering on the spatial domain]

Apply K-means algorithm to the regular grid of a spatial domain in two dimension with varying number of clusters.

The spatial domain can be represented by two matrices where one matrix represents the horizontal index and the other matrix represents the vertical index.

Define a distance between each spatial point (x_i, y_i) and a centroid (c_x^k, c_y^k) for cluster k using L2-norm square and L1-norm.

Visualize the result using color coding scheme that distinguishes different clusters.

Observe the trajectory of centroid during the optimization and the shape of the clusters depending on the distance.

Input: 1. Number of rows 2. Number of columns 3. Number of clusters 4. L2-distance, L1-distance

Visualization: Visualize image with cluster labels.

```
In [3]: import numpy as np
        import matplotlib.pyplot as plt
```

```
In [4]: # Read the image file
        img = plt.imread('../../exercise/python/data/tiger.jpeg')
        img_rows = img.shape[0]
        img_cols = img.shape[1]
```

```
In [5]: # Make a matrix for rows and columns and initialize.
        matrix_row = np.zeros((img_rows, img_cols))
        for i in range(0, img_rows):
            for j in range(0, img_cols):
                matrix_row[i][j] = i
        matrix_col = np.zeros((img_rows, img_cols))
        for i in range(0, img_rows):
            for j in range(0, img_cols):
                matrix_col[i][j] = j
```

```
In [6]: # A matrix for storing labels.
        label_map = np.zeros((img_rows, img_cols))
```

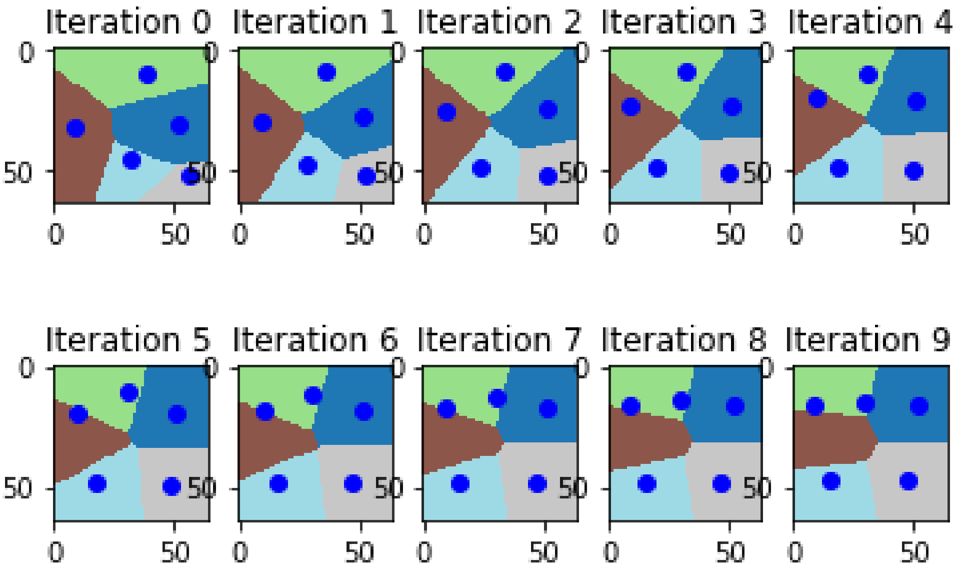
1 K = 5, L2 norm

In [7]:

```
# Define K and make random centroids
K = 5
centroid_row = np.random.randint(0, img_rows, (K, 1))
centroid_col = np.random.randint(0, img_cols, (K, 1))
centroid = np.hstack([centroid_row, centroid_col])
centroid_list = []

# K-Means clustering by L2 norm
distance_list = []
count_list = np.zeros((K))

for iteration in range(0, 10):
    for i in range(0, img_rows):
        for j in range(0, img_cols):
            # Compare the distance with each centroid.
            for iterator in range(0, K):
                distance_list.append(np.sqrt((i-centroid[iterator,0])**2+\
                                             (j-centroid[iterator,1])**2))
            # Choose the label that has the minimum distance and update the label.
            label_map[i][j] = np.argmin(distance_list)
            # Re-initialize distance_list
            distance_list = []


    # Update centroids.
    centroid.fill(0)
    # Iterate through the image label map matrix.
    for i in range(0, img_rows):
        for j in range(0, img_cols):
            centroid[int(label_map[i][j]),0] += i
            centroid[int(label_map[i][j]),1] += j
            count_list[int(label_map[i][j])] += 1

    centroid[:,0] = np.true_divide(centroid[:,0], count_list)
    centroid[:,1] = np.true_divide(centroid[:,1], count_list)

    # Reinitialize count_list
    count_list.fill(0)

    centroid_list.append(centroid)

    print("%d th iteration" % iteration)

    # Visualize image
    plt_sub = plt.subplot(2, 5, iteration+1)
    plt_sub.title.set_text("Iteration %d" % iteration)
```

```
            plt_sub.imshow(label_map,cmap='tab20')
            # Visualize centroids
            plt_sub.scatter(centroid[:,0], centroid[:,1], c='b')
```

```
0 th iteration
1 th iteration
2 th iteration
3 th iteration
4 th iteration
5 th iteration
6 th iteration
7 th iteration
8 th iteration
9 th iteration
```

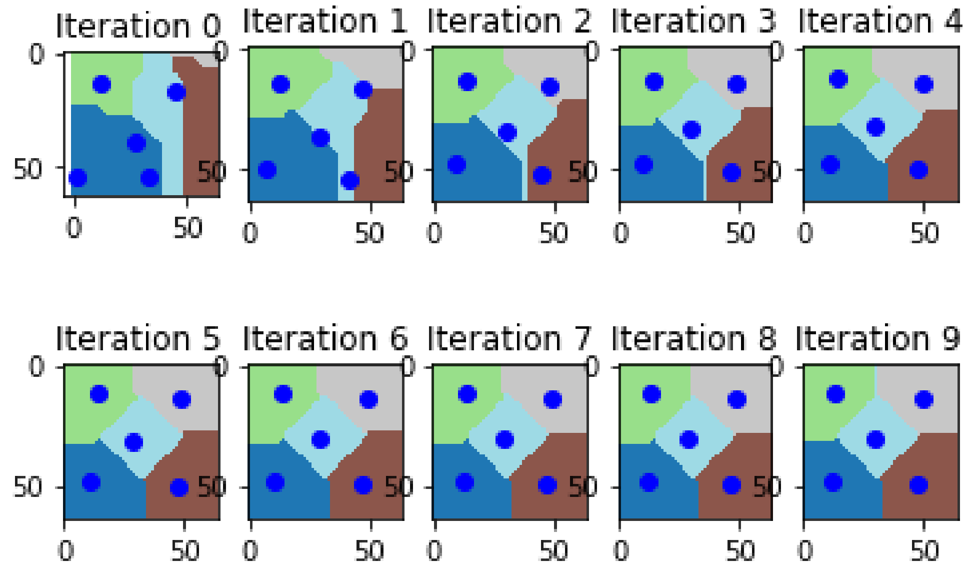


2 K = 5, L1 norm

```
In [8]: # Define K and make random centroids
        K = 5
        centroid_row = np.random.randint(0, img_rows, (K, 1))
        centroid_col = np.random.randint(0, img_cols, (K, 1))
        centroid = np.hstack([centroid_row, centroid_col])
        centroid_list = []

        # K-Means clustering by L2 norm
        distance_list = []
        count_list = np.zeros((K))

        for iteration in range(0, 10):
            for i in range(0, img_rows):
                for j in range(0, img_cols):
                    # Compare the distance with each centroid.
                    for iterator in range(0, K):
                        distance_list.append(np.abs(i-centroid[iterator,0])+\
                                             np.abs(j-centroid[iterator,1]))
                    # Choose the label that has the minimum distance and update the label.
                    label_map[i][j] = np.argmin(distance_list)
                    # Re-initialize distance_list
                    distance_list = []


            # Update centroids.
            centroid.fill(0)
            # Iterate through the image label map matrix.
            for i in range(0, img_rows):
                for j in range(0, img_cols):
                    centroid[int(label_map[i][j]),0] += i
                    centroid[int(label_map[i][j]),1] += j
                    count_list[int(label_map[i][j])] += 1

            centroid[:,0] = np.true_divide(centroid[:,0], count_list)
            centroid[:,1] = np.true_divide(centroid[:,1], count_list)

            # Reinitialize count_list
            count_list.fill(0)

            centroid_list.append(centroid)

            print("%d th iteration" % iteration)

            # Visualize image
            plt_sub = plt.subplot(2, 5, iteration+1)
            plt_sub.title.set_text("Iteration %d" % iteration)
            plt_sub.imshow(label_map,cmap='tab20')
            # Visualize centroids
            plt_sub.scatter(centroid[:,0], centroid[:,1], c='b')
```

```
0 th iteration
1 th iteration
2 th iteration
3 th iteration
4 th iteration
5 th iteration
6 th iteration
```

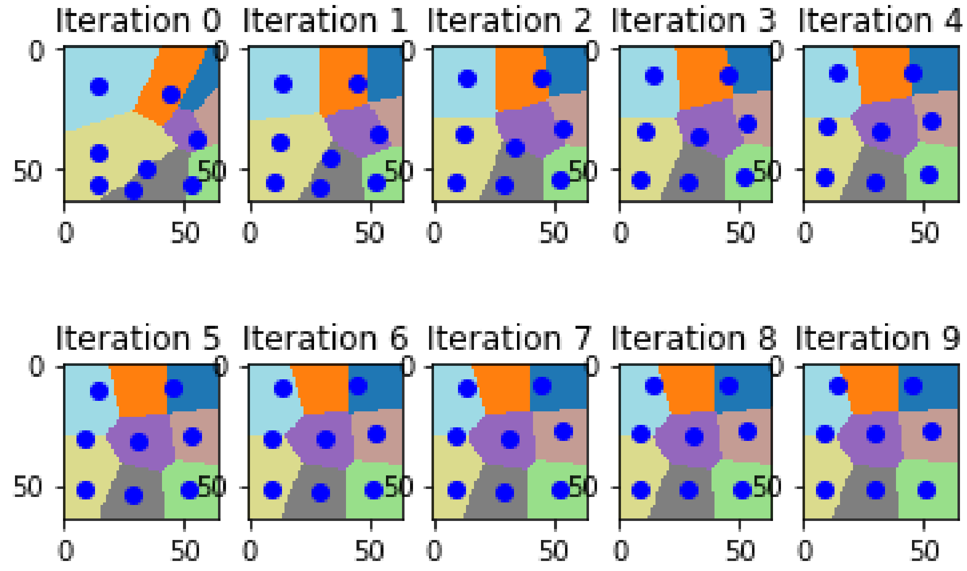
```
7 th iteration
8 th iteration
9 th iteration
```



3 K=8, L2 norm

```
In [9]: # Define K and make random centroids
        K = 8
        centroid_row = np.random.randint(0, img_rows, (K, 1))
        centroid_col = np.random.randint(0, img_cols, (K, 1))
        centroid = np.hstack([centroid_row, centroid_col])
        centroid_list = []

        # K-Means clustering by L2 norm
        distance_list = []
        count_list = np.zeros((K))

        for iteration in range(0, 10):
            for i in range(0, img_rows):
                for j in range(0, img_cols):
                    # Compare the distance with each centroid.
                    for iterator in range(0, K):
                        distance_list.append(np.sqrt((i-centroid[iterator,0])**2+\
                                                     (j-centroid[iterator,1])**2))
                    # Choose the label that has the minimum distance and update the label.
                    label_map[i][j] = np.argmin(distance_list)
                    # Re-initialize distance_list
                    distance_list = []


            # Update centroids.
            centroid.fill(0)
            # Iterate through the image label map matrix.
            for i in range(0, img_rows):
                for j in range(0, img_cols):
                    centroid[int(label_map[i][j]),0] += i
                    centroid[int(label_map[i][j]),1] += j
                    count_list[int(label_map[i][j])] += 1

            centroid[:,0] = np.true_divide(centroid[:,0], count_list)
            centroid[:,1] = np.true_divide(centroid[:,1], count_list)

            # Reinitialize count_list
            count_list.fill(0)

            centroid_list.append(centroid)

            print("%d th iteration" % iteration)

            # Visualize image
            plt_sub = plt.subplot(2, 5, iteration+1)
            plt_sub.title.set_text("Iteration %d" % iteration)
            plt_sub.imshow(label_map,cmap='tab20')
            # Visualize centroids
            plt_sub.scatter(centroid[:,0], centroid[:,1], c='b')
```

```
0 th iteration
1 th iteration
2 th iteration
3 th iteration
4 th iteration
5 th iteration
6 th iteration
7 th iteration
8 th iteration
9 th iteration
```

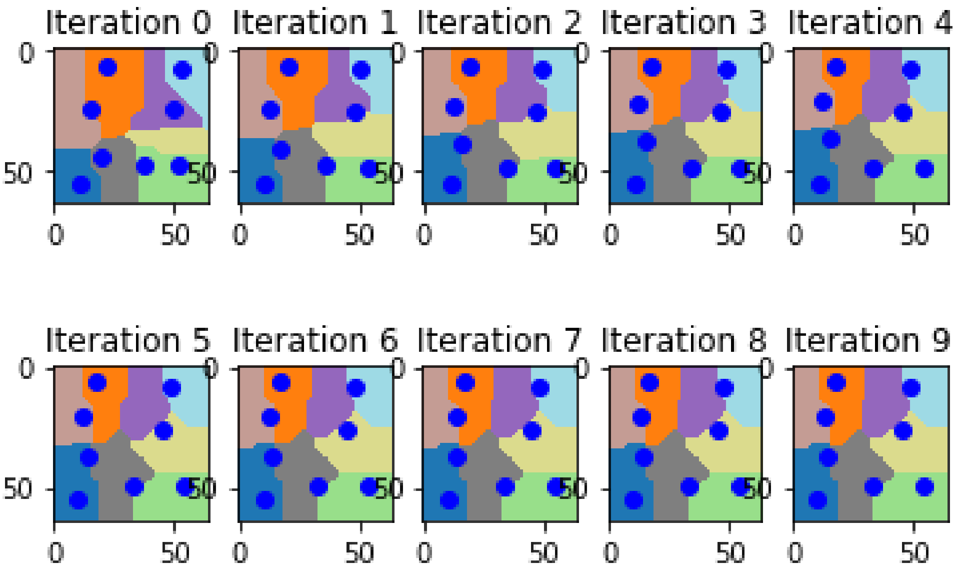


4 K=8, L1 norm

```
In [10]: # Define K and make random centroids
         K = 8
         centroid_row = np.random.randint(0, img_rows, (K, 1))
         centroid_col = np.random.randint(0, img_cols, (K, 1))
         centroid = np.hstack([centroid_row, centroid_col])
         centroid_list = []

         # K-Means clustering by L2 norm
         distance_list = []
         count_list = np.zeros((K))

         for iteration in range(0, 10):
             for i in range(0, img_rows):
                 for j in range(0, img_cols):
                     # Compare the distance with each centroid.
                     for iterator in range(0, K):
                         distance_list.append(np.abs(i-centroid[iterator,0])+\
                                             np.abs(j-centroid[iterator,1]))
                     # Choose the label that has the minimum distance and update the label.
                     label_map[i][j] = np.argmin(distance_list)
                     # Re-initialize distance_list
                     distance_list = []


             # Update centroids.
```

```
        centroid.fill(0)
        # Iterate through the image label map matrix.
        for i in range(0, img_rows):
            for j in range(0, img_cols):
                centroid[int(label_map[i][j]),0] += i
                centroid[int(label_map[i][j]),1] += j
                count_list[int(label_map[i][j])] += 1

        centroid[:,0] = np.true_divide(centroid[:,0], count_list)
        centroid[:,1] = np.true_divide(centroid[:,1], count_list)

        # Reinitialize count_list
        count_list.fill(0)

        centroid_list.append(centroid)

        print("%d th iteration" % iteration)

        # Visualize image
        plt_sub = plt.subplot(2, 5, iteration+1)
        plt_sub.title.set_text("Iteration %d" % iteration)
        plt_sub.imshow(label_map,cmap='tab20')
        # Visualize centroids
        plt_sub.scatter(centroid[:,0], centroid[:,1], c='b')
```

```
0 th iteration
1 th iteration
2 th iteration
3 th iteration
4 th iteration
5 th iteration
6 th iteration
7 th iteration
8 th iteration
9 th iteration
```

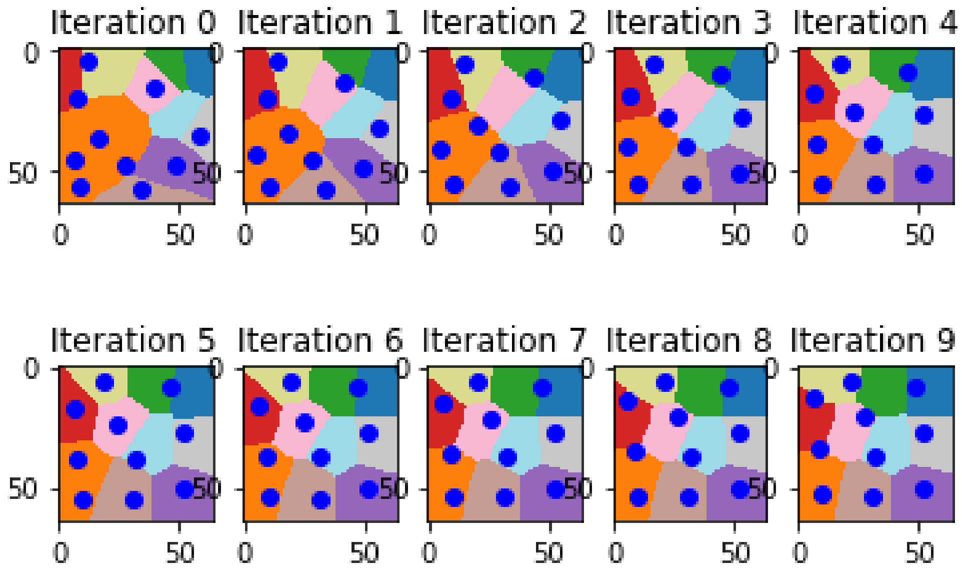


5 K=10, L2 norm

In [11]:
```
# Define K and make random centroids
K = 10
centroid_row = np.random.randint(0, img_rows, (K, 1))
centroid_col = np.random.randint(0, img_cols, (K, 1))
centroid = np.hstack([centroid_row, centroid_col])
centroid_list = []

# K-Means clustering by L2 norm
distance_list = []
count_list = np.zeros((K))

for iteration in range(0, 10):
    for i in range(0, img_rows):
        for j in range(0, img_cols):
            # Compare the distance with each centroid.
            for iterator in range(0, K):
                distance_list.append(np.sqrt((i-centroid[iterator,0])**2+\
                                             (j-centroid[iterator,1])**2))
            # Choose the label that has the minimum distance and update the label.
            label_map[i][j] = np.argmin(distance_list)
            # Re-initialize distance_list
            distance_list = []


    # Update centroids.
```

```
        centroid.fill(0)
        # Iterate through the image label map matrix.
        for i in range(0, img_rows):
            for j in range(0, img_cols):
                centroid[int(label_map[i][j]),0] += i
                centroid[int(label_map[i][j]),1] += j
                count_list[int(label_map[i][j])] += 1

        centroid[:,0] = np.true_divide(centroid[:,0], count_list)
        centroid[:,1] = np.true_divide(centroid[:,1], count_list)

        # Reinitialize count_list
        count_list.fill(0)

        centroid_list.append(centroid)

        print("%d th iteration" % iteration)

        # Visualize image
        plt_sub = plt.subplot(2, 5, iteration+1)
        plt_sub.title.set_text("Iteration %d" % iteration)
        plt_sub.imshow(label_map,cmap='tab20')
        # Visualize centroids
        plt_sub.scatter(centroid[:,0], centroid[:,1], c='b')
```

```
0 th iteration
1 th iteration
2 th iteration
3 th iteration
4 th iteration
5 th iteration
6 th iteration
7 th iteration
8 th iteration
9 th iteration
```

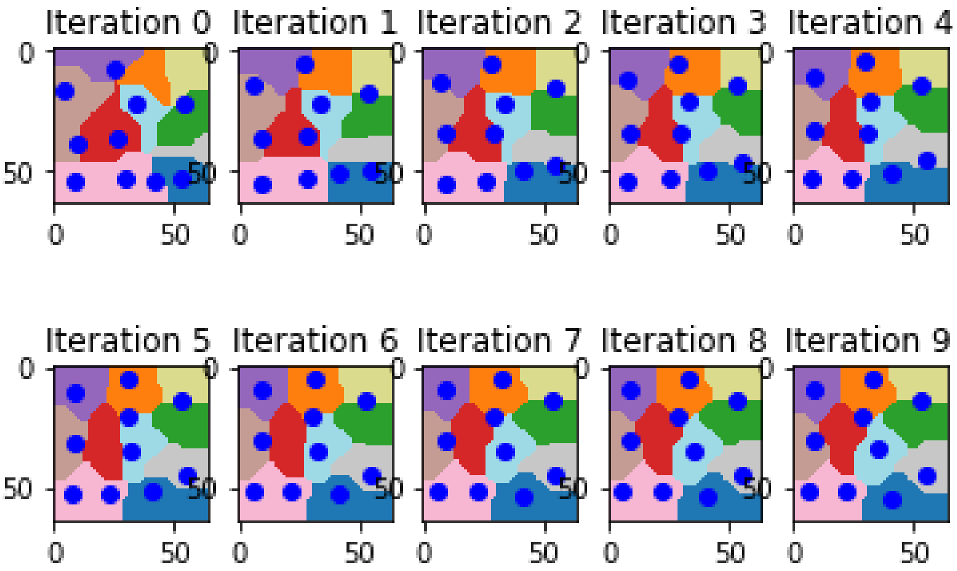


6 K=10, L1 norm

```
In [13]: # Define K and make random centroids
         K = 10
         centroid_row = np.random.randint(0, img_rows, (K, 1))
         centroid_col = np.random.randint(0, img_cols, (K, 1))
         centroid = np.hstack([centroid_row, centroid_col])
         centroid_list = []

         # K-Means clustering by L2 norm
         distance_list = []
         count_list = np.zeros((K))

         for iteration in range(0, 10):
             for i in range(0, img_rows):
                 for j in range(0, img_cols):
                     # Compare the distance with each centroid.
                     for iterator in range(0, K):
                         distance_list.append(np.abs(i-centroid[iterator,0])+\
                                                     np.abs(j-centroid[iterator,1]))
                     # Choose the label that has the minimum distance and update the label.
                     label_map[i][j] = np.argmin(distance_list)
                     # Re-initialize distance_list
                     distance_list = []


             # Update centroids.
```

```
            centroid.fill(0)
            # Iterate through the image label map matrix.
            for i in range(0, img_rows):
                for j in range(0, img_cols):
                    centroid[int(label_map[i][j]),0] += i
                    centroid[int(label_map[i][j]),1] += j
                    count_list[int(label_map[i][j])] += 1

            centroid[:,0] = np.true_divide(centroid[:,0], count_list)
            centroid[:,1] = np.true_divide(centroid[:,1], count_list)

            # Reinitialize count_list
            count_list.fill(0)

            centroid_list.append(centroid)

            print("%d th iteration" % iteration)

            # Visualize image
            plt_sub = plt.subplot(2, 5, iteration+1)
            plt_sub.title.set_text("Iteration %d" % iteration)
            plt_sub.imshow(label_map,cmap='tab20')
            # Visualize centroids
            plt_sub.scatter(centroid[:,0], centroid[:,1], c='b')
```

```
0 th iteration
1 th iteration
2 th iteration
3 th iteration
4 th iteration
5 th iteration
6 th iteration
7 th iteration
8 th iteration
9 th iteration
```

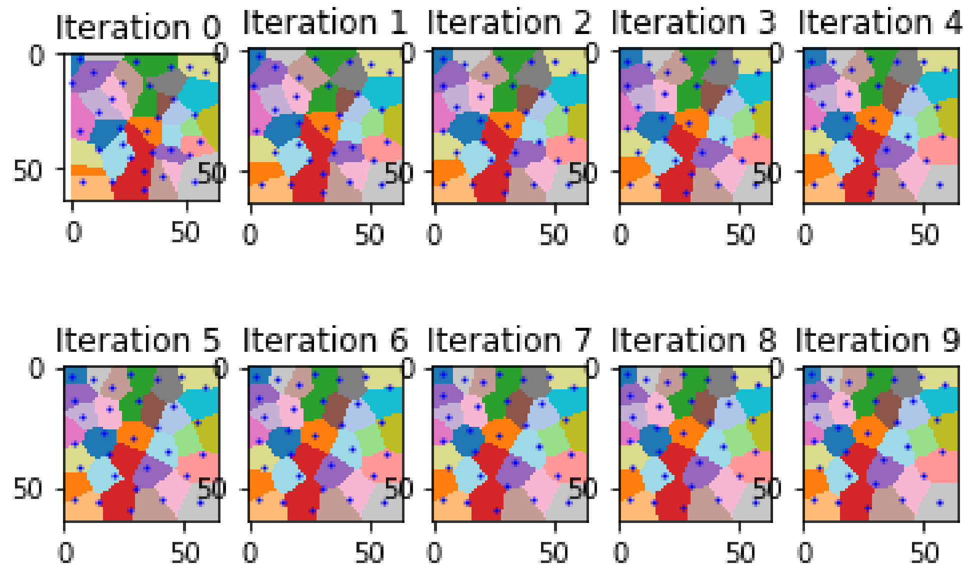


7 K = 50, L2 norm

```
In [15]: # Define K and make random centroids
         K = 30
         centroid_row = np.random.randint(0, img_rows, (K, 1))
         centroid_col = np.random.randint(0, img_cols, (K, 1))
         centroid = np.hstack([centroid_row, centroid_col])
         centroid_list = []

         # K-Means clustering by L2 norm
         distance_list = []
         count_list = np.zeros((K))

         for iteration in range(0, 10):
             for i in range(0, img_rows):
                 for j in range(0, img_cols):
                     # Compare the distance with each centroid.
                     for iterator in range(0, K):
                         distance_list.append(np.sqrt((i-centroid[iterator,0])**2+\
                                                      (j-centroid[iterator,1])**2))
                     # Choose the label that has the minimum distance and update the label.
                     label_map[i][j] = np.argmin(distance_list)
                     # Re-initialize distance_list
                     distance_list = []


             # Update centroids.
```

```
        centroid.fill(0)
        # Iterate through the image label map matrix.
        for i in range(0, img_rows):
            for j in range(0, img_cols):
                centroid[int(label_map[i][j]),0] += i
                centroid[int(label_map[i][j]),1] += j
                count_list[int(label_map[i][j])] += 1

        centroid[:,0] = np.true_divide(centroid[:,0], count_list)
        centroid[:,1] = np.true_divide(centroid[:,1], count_list)

        # Reinitialize count_list
        count_list.fill(0)

        centroid_list.append(centroid)

        print("%d th iteration" % iteration)

        # Visualize image
        plt_sub = plt.subplot(2, 5, iteration+1)
        plt_sub.title.set_text("Iteration %d" % iteration)
        plt_sub.imshow(label_map,cmap='tab20')
        # Visualize centroids
        plt_sub.scatter(centroid[:,0], centroid[:,1], s=1, c='b')
```

```
0 th iteration
1 th iteration
2 th iteration
3 th iteration
4 th iteration
5 th iteration
6 th iteration
7 th iteration
8 th iteration
9 th iteration
```

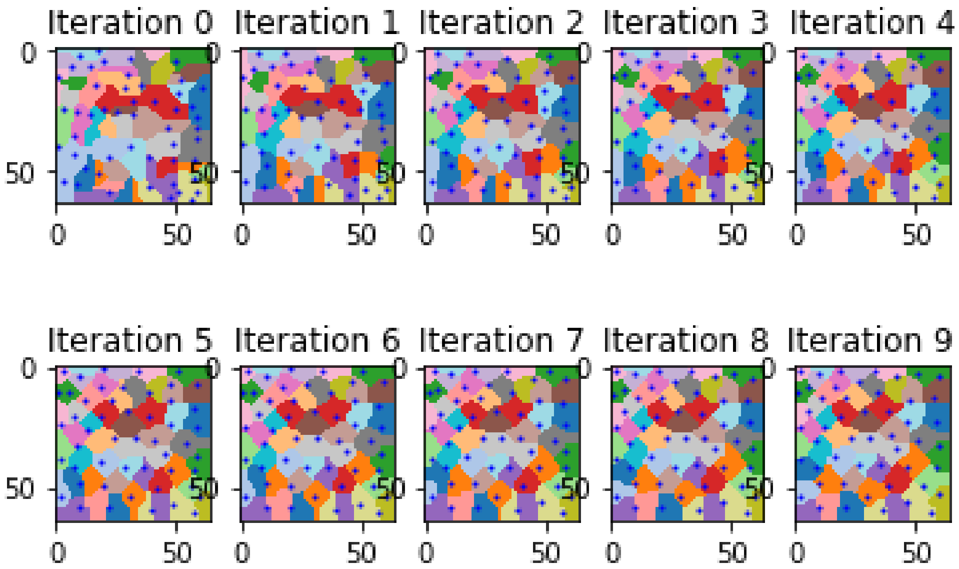


8 K = 50, L1 norm

```
In [16]: # Define K and make random centroids
         K = 50
         centroid_row = np.random.randint(0, img_rows, (K, 1))
         centroid_col = np.random.randint(0, img_cols, (K, 1))
         centroid = np.hstack([centroid_row, centroid_col])
         centroid_list = []

         # K-Means clustering by L2 norm
         distance_list = []
         count_list = np.zeros((K))

         for iteration in range(0, 10):
             for i in range(0, img_rows):
                 for j in range(0, img_cols):
                     # Compare the distance with each centroid.
                     for iterator in range(0, K):
                         distance_list.append(np.abs(i-centroid[iterator,0])+\
                                              np.abs(j-centroid[iterator,1]))
                     # Choose the label that has the minimum distance and update the label.
                     label_map[i][j] = np.argmin(distance_list)
                     # Re-initialize distance_list
                     distance_list = []


             # Update centroids.
```

```
        centroid.fill(0)
        # Iterate through the image label map matrix.
        for i in range(0, img_rows):
            for j in range(0, img_cols):
                centroid[int(label_map[i][j]),0] += i
                centroid[int(label_map[i][j]),1] += j
                count_list[int(label_map[i][j])] += 1

        centroid[:,0] = np.true_divide(centroid[:,0], count_list)
        centroid[:,1] = np.true_divide(centroid[:,1], count_list)

        # Reinitialize count_list
        count_list.fill(0)

        centroid_list.append(centroid)

        print("%d th iteration" % iteration)

        # Visualize image
        plt_sub = plt.subplot(2, 5, iteration+1)
        plt_sub.title.set_text("Iteration %d" % iteration)
        plt_sub.imshow(label_map,cmap='tab20')
        # Visualize centroids
        plt_sub.scatter(centroid[:,0], centroid[:,1], s=1, c='b')
```

```
0 th iteration
1 th iteration
2 th iteration
3 th iteration
4 th iteration
5 th iteration
6 th iteration
7 th iteration
8 th iteration
9 th iteration
```


Iteration 0
Iteration 1
Iteration 2
Iteration 3
Iteration 4
Iteration 5
Iteration 6
Iteration 7
Iteration 8
Iteration 9
0
50
0
50