# Homework-H3.4_k-Means_Clustering

October 16, 2020

## 1 Homework H3.4

By: Markus Limbacher & Lucas Krauter

### 1.1 Preparations

#### 1.1.1 Import of libraries

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
import copy
# to check the time of execution, import function time
import time
# check versions of libraries
print('pandas version is: {}'.format(pd.__version__))
print('numpy version is: {}'.format(np.__version__))
```

```
pandas version is: 1.0.3
numpy version is: 1.18.3
```

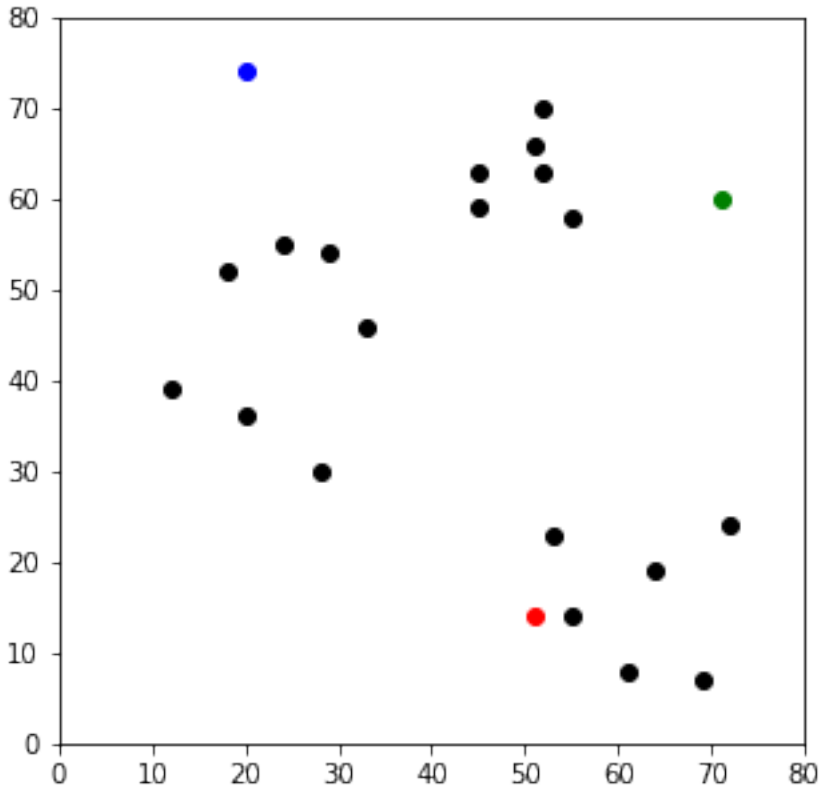
#### 1.1.2 Dataset

[2]:
```
# Dataset declaration
df = pd.DataFrame({
    'x': [12, 20, 28, 18, 29, 33, 24, 45, 45, 52, 51, 52, 55, 53, 55, 61, 64,␣
 ↪69, 72],
    'y': [39, 36, 30, 52, 54, 46, 55, 59, 63, 70, 66, 63, 58, 23, 14, 8, 19, 7,␣
 ↪24]
})

np.random.seed(42)

# Number of clusters ==> k
k = 3
# centroids[i] = [x, y]
centroids = {
```

```
    i+1: [np.random.randint(0, 80), np.random.randint(0, 80)]
    for i in range(k)
}
```

### 1.1.3 Display dataset

[3]:
```
fig = plt.figure(figsize=(5, 5))
plt.scatter(df['x'], df['y'], color='k')
colmap = {1: 'r', 2: 'g', 3: 'b'}
for i in centroids.keys():
    plt.scatter(*centroids[i], color=colmap[i])
plt.xlim(0, 80)
plt.ylim(0, 80)
plt.show()
```



## 1.2 K-Means manually

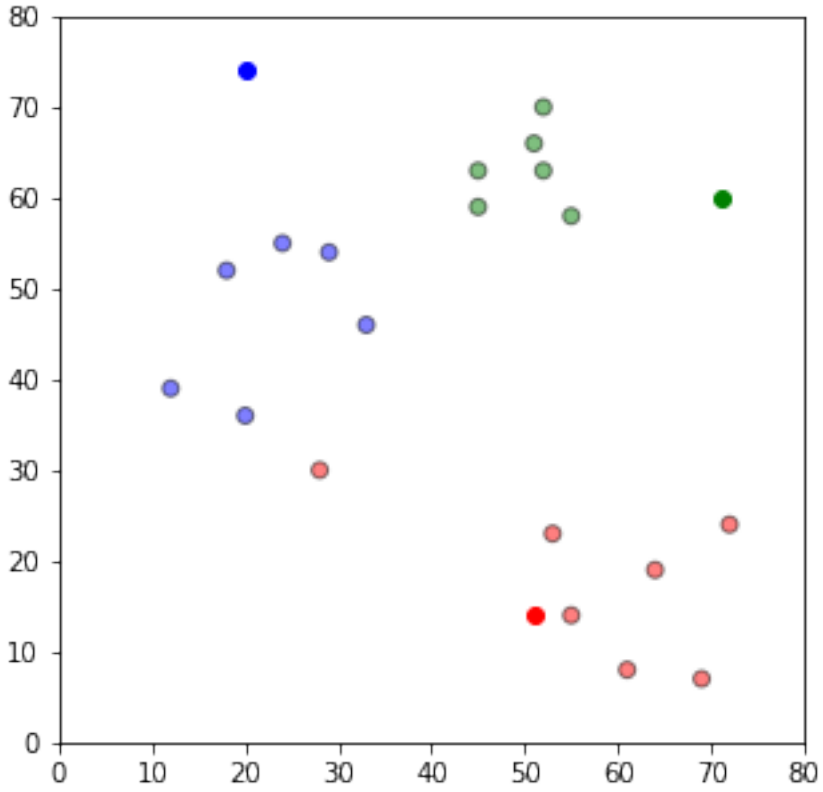
### 1.2.1 Assignment Stage

[4]:
```
# Function to determine closest centroid
def assignment(df, centroids):

    # Iterating over every centroid
    for i in centroids.keys():
        # distance function: sqrt((x1 - x2)^2 - (y1 - y2)^2)
        df['distance_from_{}'.format(i)] = (
            np.sqrt(
                (df['x'] - centroids[i][0]) ** 2
                + (df['y'] - centroids[i][1]) ** 2
            )
        )
    centroid_distance_cols = ['distance_from_{}'.format(i) for i in centroids.
 ↪keys()]
    df['closest'] = df.loc[:, centroid_distance_cols].idxmin(axis=1)
    df['closest'] = df['closest'].map(lambda x: int(x.lstrip('distance_from_')))
    df['color'] = df['closest'].map(lambda x: colmap[x])
    return df

df = assignment(df, centroids)
print(df)
```

| | x | y | distance_from_1 | distance_from_2 | distance_from_3 | closest | color |
|---|---|---|---|---|---|---|---|
| 0 | 12 | 39 | 46.324939 | 62.625873 | 35.902646 | 3 | b |
| 1 | 20 | 36 | 38.013156 | 56.364883 | 38.000000 | 3 | b |
| 2 | 28 | 30 | 28.017851 | 52.430907 | 44.721360 | 1 | r |
| 3 | 18 | 52 | 50.328918 | 53.600373 | 22.090722 | 3 | b |
| 4 | 29 | 54 | 45.650849 | 42.426407 | 21.931712 | 3 | b |
| 5 | 33 | 46 | 36.715120 | 40.496913 | 30.870698 | 3 | b |
| 6 | 24 | 55 | 49.091751 | 47.265209 | 19.416488 | 3 | b |
| 7 | 45 | 59 | 45.398238 | 26.019224 | 29.154759 | 2 | g |
| 8 | 45 | 63 | 49.365980 | 26.172505 | 27.313001 | 2 | g |
| 9 | 52 | 70 | 56.008928 | 21.470911 | 32.249031 | 2 | g |
| 10 | 51 | 66 | 52.000000 | 20.880613 | 32.015621 | 2 | g |
| 11 | 52 | 63 | 49.010203 | 19.235384 | 33.837849 | 2 | g |
| 12 | 55 | 58 | 44.181444 | 16.124515 | 38.483763 | 2 | g |
| 13 | 53 | 23 | 9.219544 | 41.146081 | 60.745370 | 1 | r |
| 14 | 55 | 14 | 4.000000 | 48.703183 | 69.462220 | 1 | r |
| 15 | 61 | 8 | 11.661904 | 52.952809 | 77.698134 | 1 | r |
| 16 | 64 | 19 | 13.928388 | 41.593269 | 70.434367 | 1 | r |
| 17 | 69 | 7 | 19.313208 | 53.037722 | 83.006024 | 1 | r |
| 18 | 72 | 24 | 23.259407 | 36.013886 | 72.138755 | 1 | r |

### 1.2.2 Display modified dataset with color assigned to closest centroid.

[5]:
```
def displayDataset(df, centroids):
    fig = plt.figure(figsize=(5, 5))
    plt.scatter(df['x'], df['y'], color=df['color'], alpha=0.5, edgecolor='k')
    for i in centroids.keys():
        plt.scatter(*centroids[i], color=colmap[i])
    plt.xlim(0, 80)
    plt.ylim(0, 80)
    plt.show()

displayDataset(df, centroids)
```



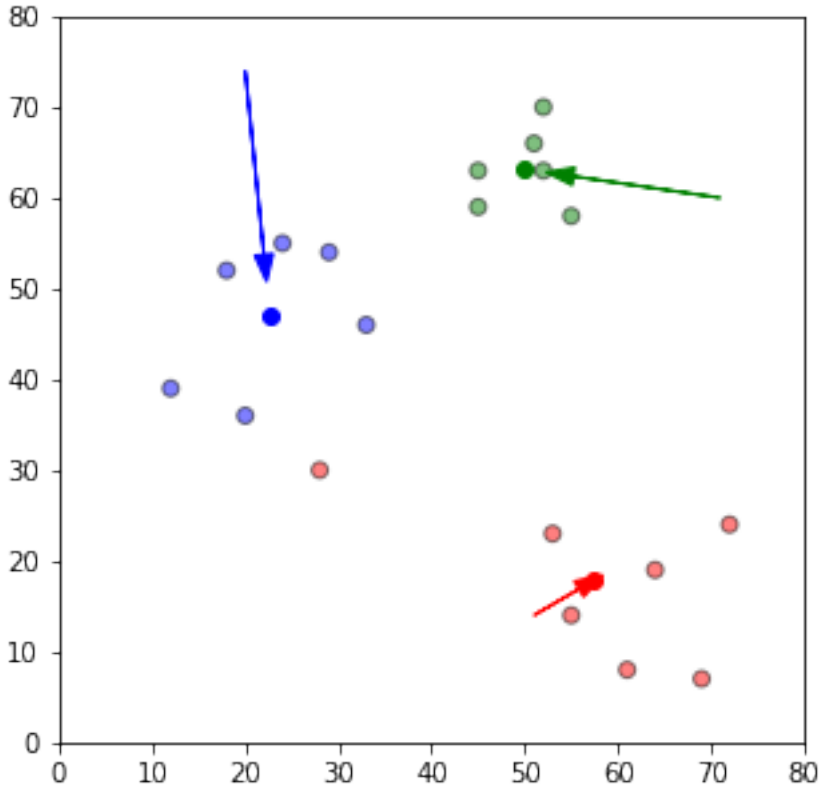

### 1.2.3 Update Stage

[6]:
```
# Copies current centroids for demonstration purposes
old_centroids = copy.deepcopy(centroids)

# Calculate mean from each seperate cluster as new centroids
def update(k):
    for i in centroids.keys():
```

```
        centroids[i][0] = np.mean(df[df['closest'] == i]['x'])
        centroids[i][1] = np.mean(df[df['closest'] == i]['y'])
    return k

centroids = update(centroids)
```

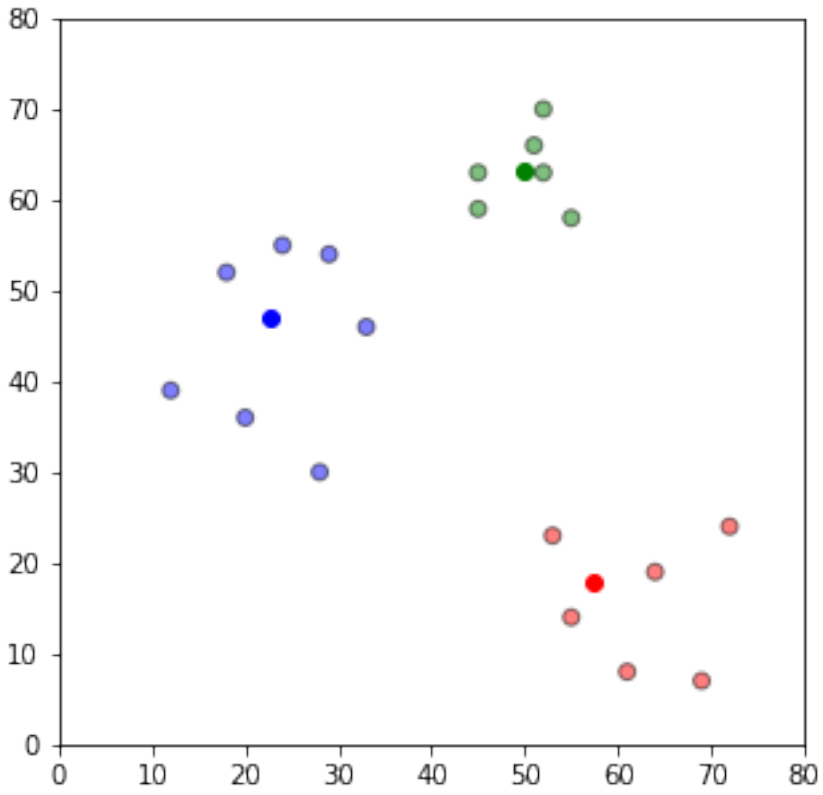
### 1.2.4 Display updated centroids

[7]:
```
fig = plt.figure(figsize=(5, 5))
ax = plt.axes()
plt.scatter(df['x'], df['y'], color=df['color'], alpha=0.5, edgecolor='k')
for i in centroids.keys():
    plt.scatter(*centroids[i], color=colmap[i])
plt.xlim(0, 80)
plt.ylim(0, 80)

for i in old_centroids.keys():
    old_x = old_centroids[i][0]
    old_y = old_centroids[i][1]
    dx = (centroids[i][0] - old_centroids[i][0]) * 0.75
    dy = (centroids[i][1] - old_centroids[i][1]) * 0.75
    ax.arrow(old_x, old_y, dx, dy, head_width=2, head_length=3, fc=colmap[i],␣
 ↪ec=colmap[i])
plt.show()
```



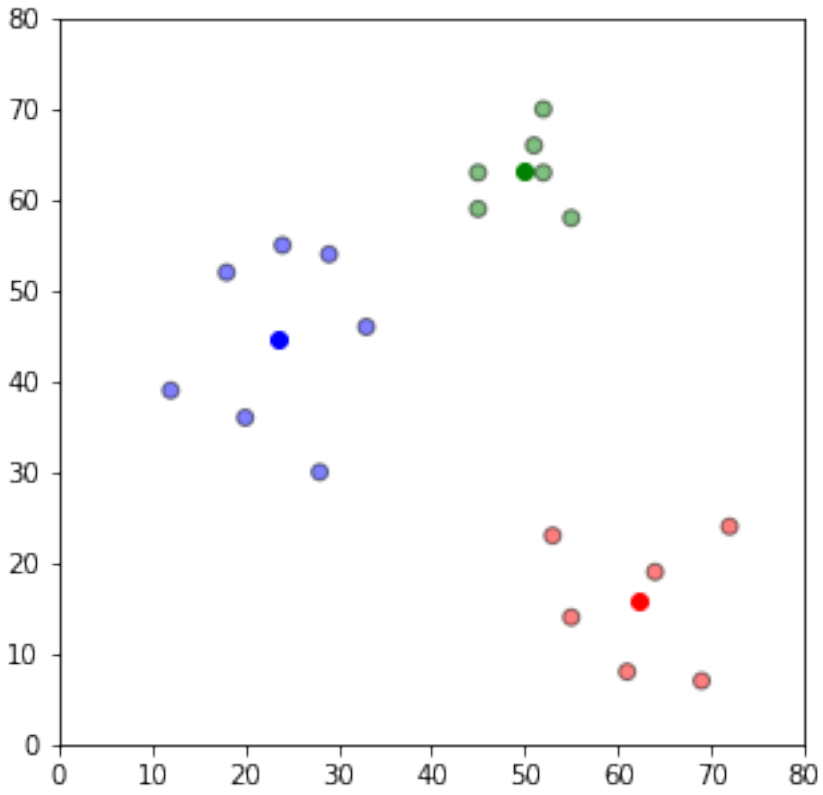

### 1.2.5 Repeat Assigment

[8]:
```
df = assignment(df, centroids)

# Plot results
displayDataset(df, centroids)
```




### 1.2.6 Loop until all assigned categories don't change any more

[9]:
```
while True:
    closest_centroids = df['closest'].copy(deep=True)
    centroids = update(centroids)
    df = assignment(df, centroids)
    if closest_centroids.equals(df['closest']):
        break

displayDataset(df, centroids)
```




### 1.3 K-Means using scikit-learn

#### 1.3.1 Preparations

[10]:
```
# Dataset
df = pd.DataFrame({
    'x': [12, 20, 28, 18, 29, 33, 24, 45, 45, 52, 51, 52, 55, 53, 55, 61, 64,␣
 ↪69, 72],
    'y': [39, 36, 30, 52, 54, 46, 55, 59, 63, 70, 66, 63, 58, 23, 14, 8, 19, 7,␣
 ↪24]
})

# Import K-Means library
from sklearn.cluster import KMeans
```

#### 1.3.2 Executing K-Means function

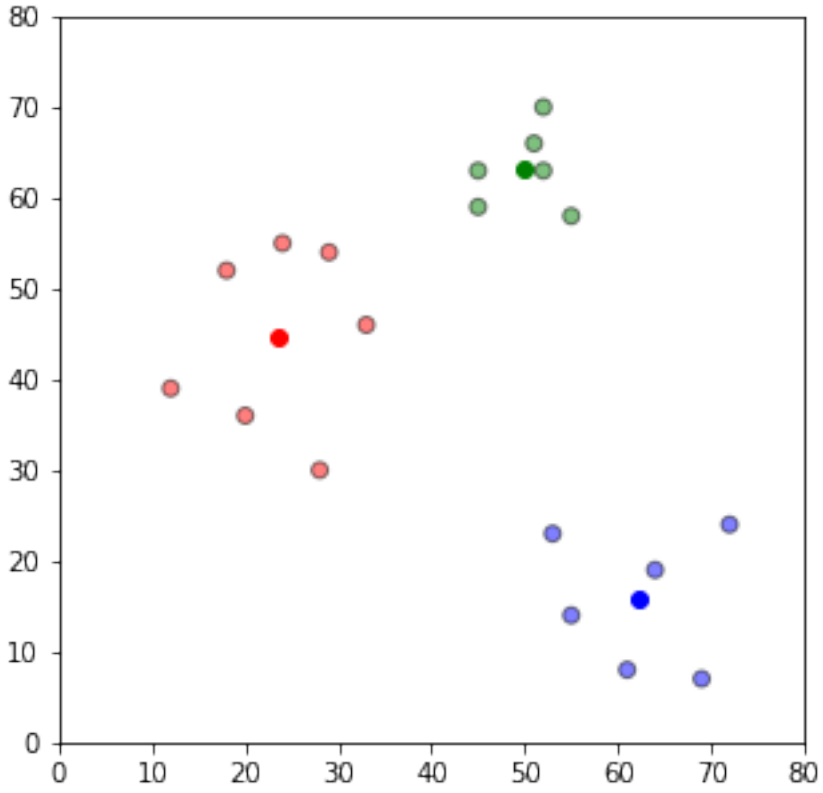
[11]:
```
# 3 Clusters
kmeans = KMeans(n_clusters=3)

# Fitting K-Means modell
```

```
print(kmeans.fit(df))
```

```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

[12]:
```
# Using modell to label dataset
labels = kmeans.predict(df)
centroids = kmeans.cluster_centers_
```

[13]:
```
# Display result
fig = plt.figure(figsize=(5, 5))

colors =list( map(lambda x: colmap[x+1], labels))


plt.scatter(df['x'], df['y'],color=colors, alpha=0.5, edgecolor='k')
for idx, centroid in enumerate(centroids):
    plt.scatter(*centroid, color=colmap[idx+1])
plt.xlim(0, 80)
plt.ylim(0, 80)
plt.show()
```



[14]:
```
# print current date and time
print("date & time:",time.strftime("%d.%m.%Y %H:%M:%S"))
print ("**** End of Homework-H3.4_k-Means_Clustering ****")
```

```
date & time: 16.10.2020 23:01:20
**** End of Homework-H3.4_k-Means_Clustering ****
```