K-Means Clustering

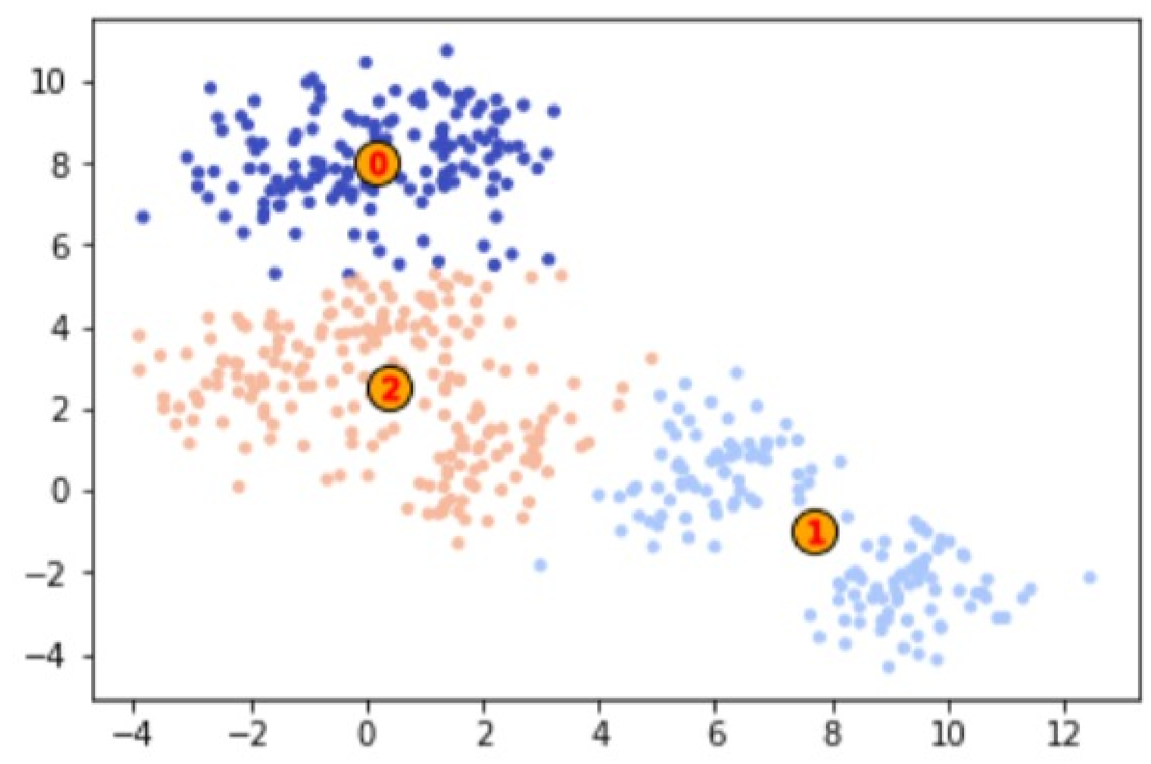
In [2]:

```python
# import K-Means
from sklearn.cluster import KMeans

# import numpy and pandas
import pandas as pd
import numpy as np

# for visualization
from matplotlib import pyplot as plt
from matplotlib import cm


#========================================================================== READ DATA
df = pd.read_csv("Example1.csv", sep = '\t')


#========================================================================== IMPLEMENT K-MEANS

# no. of clusters
clusters = 3

# --------------------------------- KMeans parameters
# n_clusters = no. of clusters
# n_init = no. of iterations
# tol = tolerance (minimum close to 0)
# random_state = seed for pseudo number

km = KMeans(n_clusters=clusters, n_init=50, tol=1e-10, random_state=1234).fit(df)


# get centers
centers = km.cluster_centers_
#print(centers)

# get the labels/clusters
labels = km.predict(df)
#print(labels)

# assign each data point to its labels/cluster
df['label'] = labels

# group data points to by its labels/cluster
groups = df.groupby('label')
print(groups.size) # print how many instances each cluster have

# inertia manually
# sum_of_squares = 0

# for name, group in groups:
#     arr = np.array(group)[:,:2]
#     for x in arr:
#         sum_of_squares += np.sum((x-centers[name])**2)

# print(sum_of_squares)

inertia = km.inertia_
print(inertia)

#========================================================================== VISUALIZATION

# colors
colors = cm.coolwarm(np.array(labels).astype(float)/clusters)

# graph figure
fig, ax = plt.subplots()

# plot data to graph figure
ax.scatter(df['A'], df['B'], marker='o', s=10, c=colors)

# attach centers to the graph
ax.scatter(centers[:,0], centers[:,1], s=200, c='orange', edgecolor = 'k')
for i,c in enumerate(centers):
    ax.scatter(c[0], c[1], marker='$%d$'%i, c='red', s=50)

plt
```

```
<bound method GroupBy.size of <pandas.core.groupby.DataFrameGroupBy object at 0x000001AFCA3976A0>>
2784.0682340858402
```

Out[2]:

```
<module 'matplotlib.pyplot' from 'C:\\Users\\Abe-r\\Anaconda3\\lib\\site-packages\\matplotlib\\pyplot.py'>
```

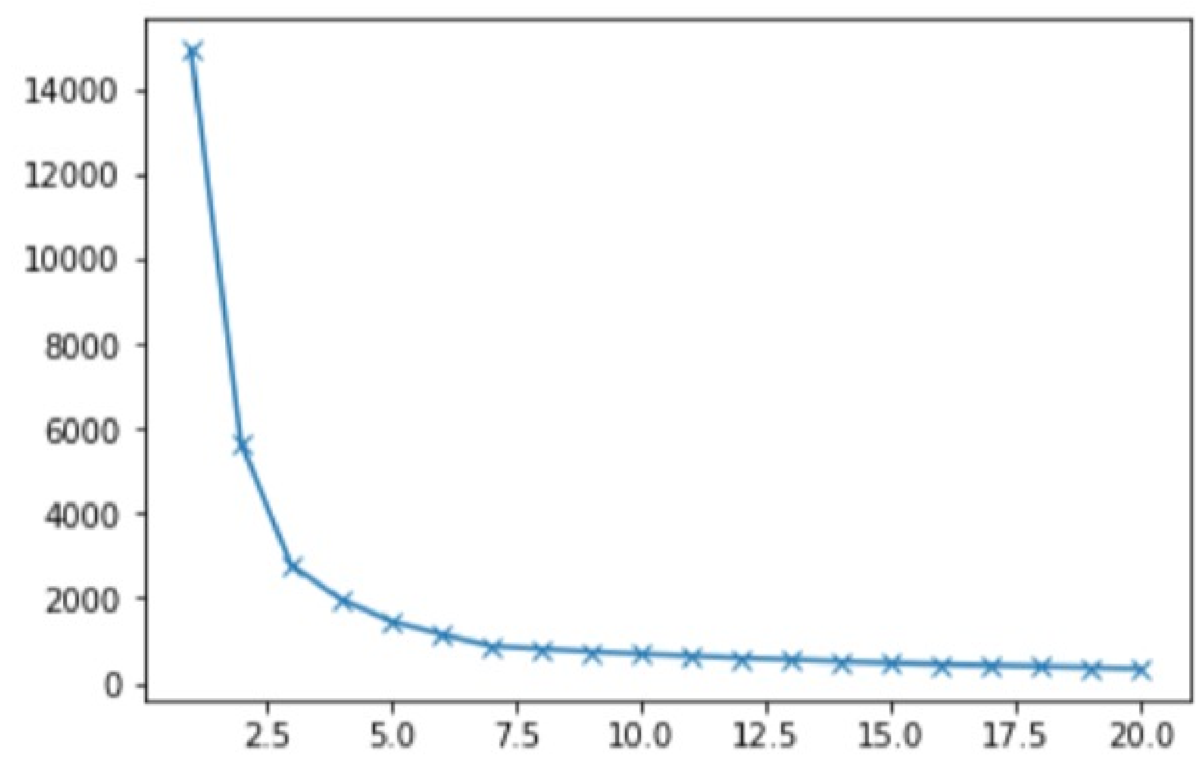


Validation for K-Means

Elbow - getting k using a graph which has the mose curve

In [3]:

```python
# list of inertia of k-means from k = 1 until k = 20
inertias = []

# loop to get inertia where k = 1 to 20
for i in range(1, 21) :
    km = KMeans(n_clusters=i, n_init=50, tol=1e-10, random_state=1234).fit(df)
    inertias.append(km.inertia_)

# graph all inirtias with respect to k
fig2,ax2 = plt.subplots()
ax2.plot(range(1,21), inertias, 'x-')

plt.show()

#print(inertias)
```



Sillouette - more precise way of getting k, closest to 1 is most advisable

In [6]:

```python
from sklearn.metrics import silhouette_score

scores = []
for i in range(2, 8) :
    km = KMeans(n_clusters=i, n_init=50, tol=1e-10, random_state=1234).fit_predict(df)
    sc = silhouette_score(df, km)
    print("For k = %d, the score is %g"%(i, sc))
    scores.append(sc)

print("\nk =", scores.index(max(scores)) + 2, "since", round(max(scores), 4) ,"is the closest score to 1")
```

```
For k = 2, the score is 0.523647
For k = 3, the score is 0.538974
For k = 4, the score is 0.502279
For k = 5, the score is 0.512795
For k = 6, the score is 0.51385
For k = 7, the score is 0.493014

k = 3 since 0.539 is the closest score to 1
```