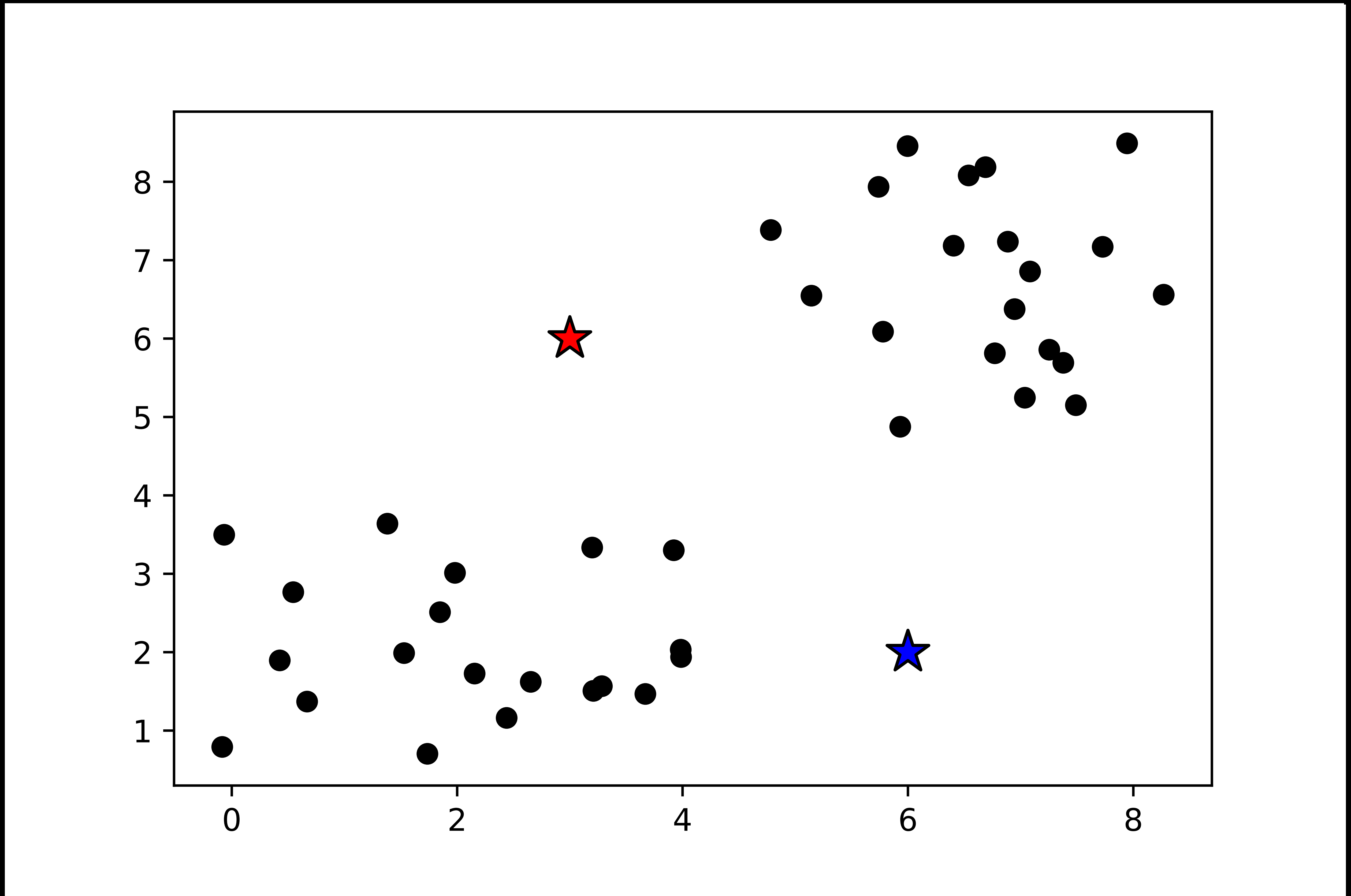
K-Means Clustering



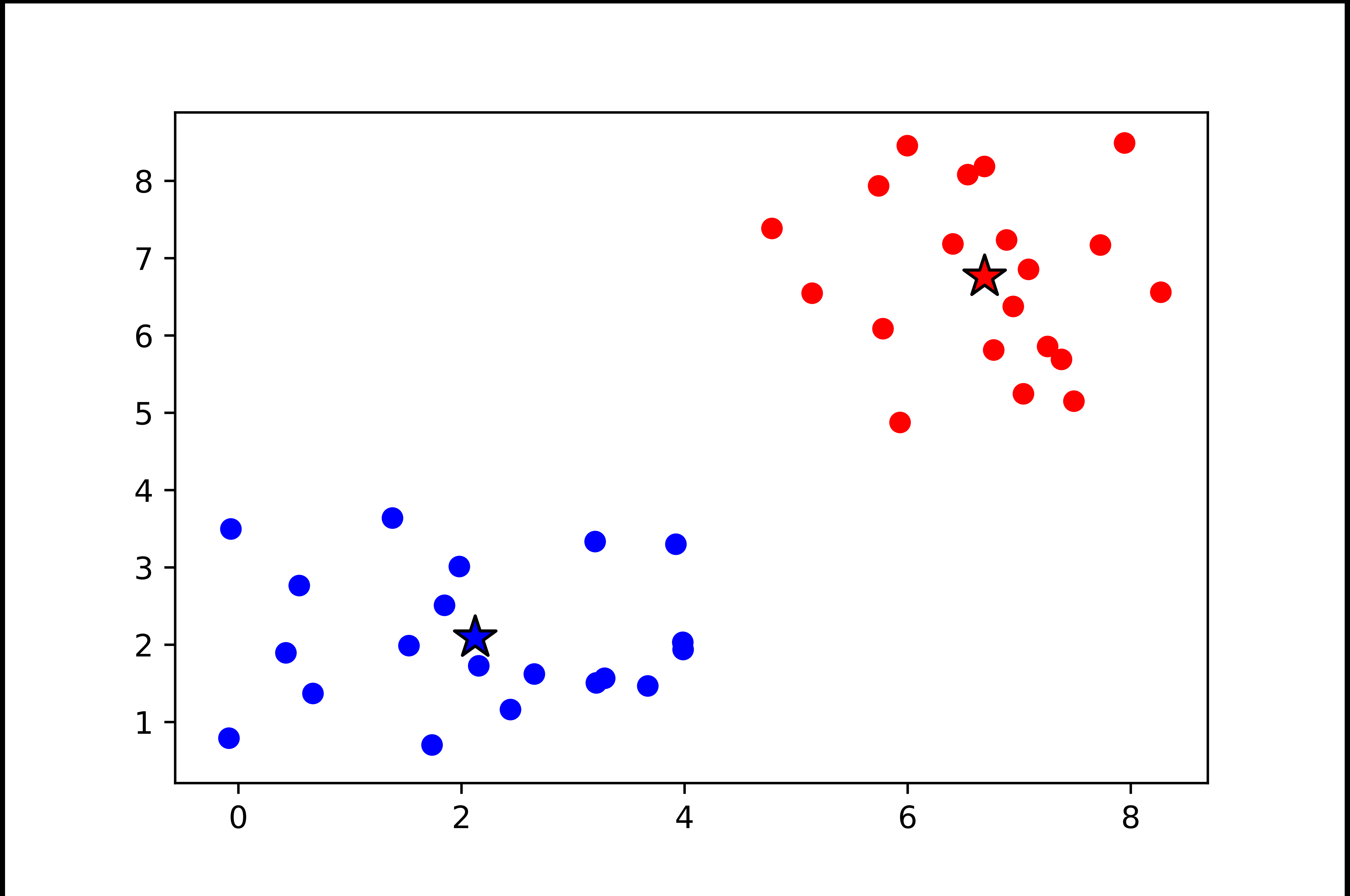
- Randomly initialise cluster centroids

- Calculate distance between each point and the cluster centroids

- Assign points to nearest cluster
- Calculate mean position of points in each cluster

- Set new cluster centroids equal to mean

- Repeat



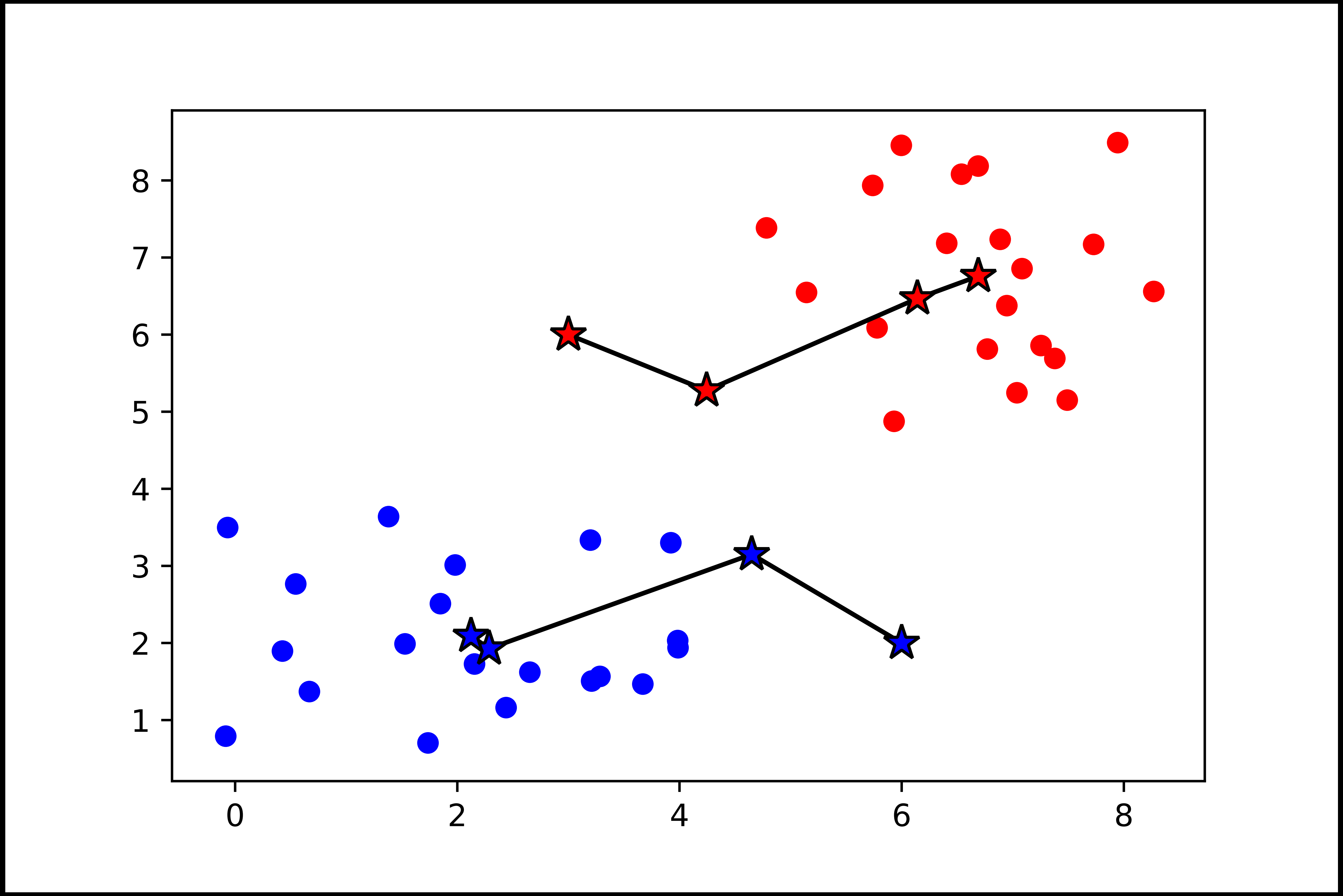
- Randomly initialise cluster centroids

- Calculate distance between each point and the cluster centroids

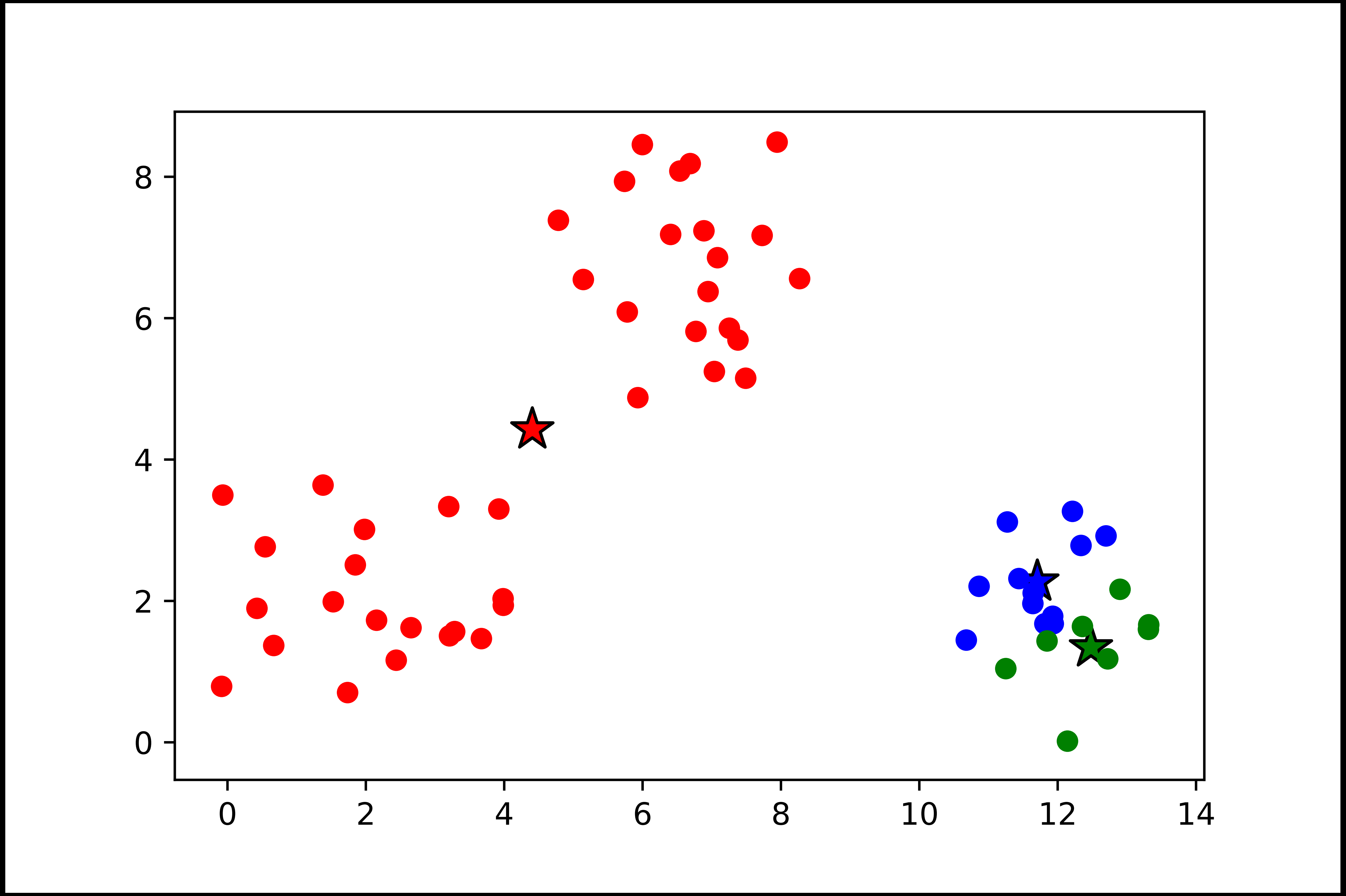
- Assign points to nearest cluster
- Calculate mean position of points in each cluster

- Set new cluster centroids equal to mean

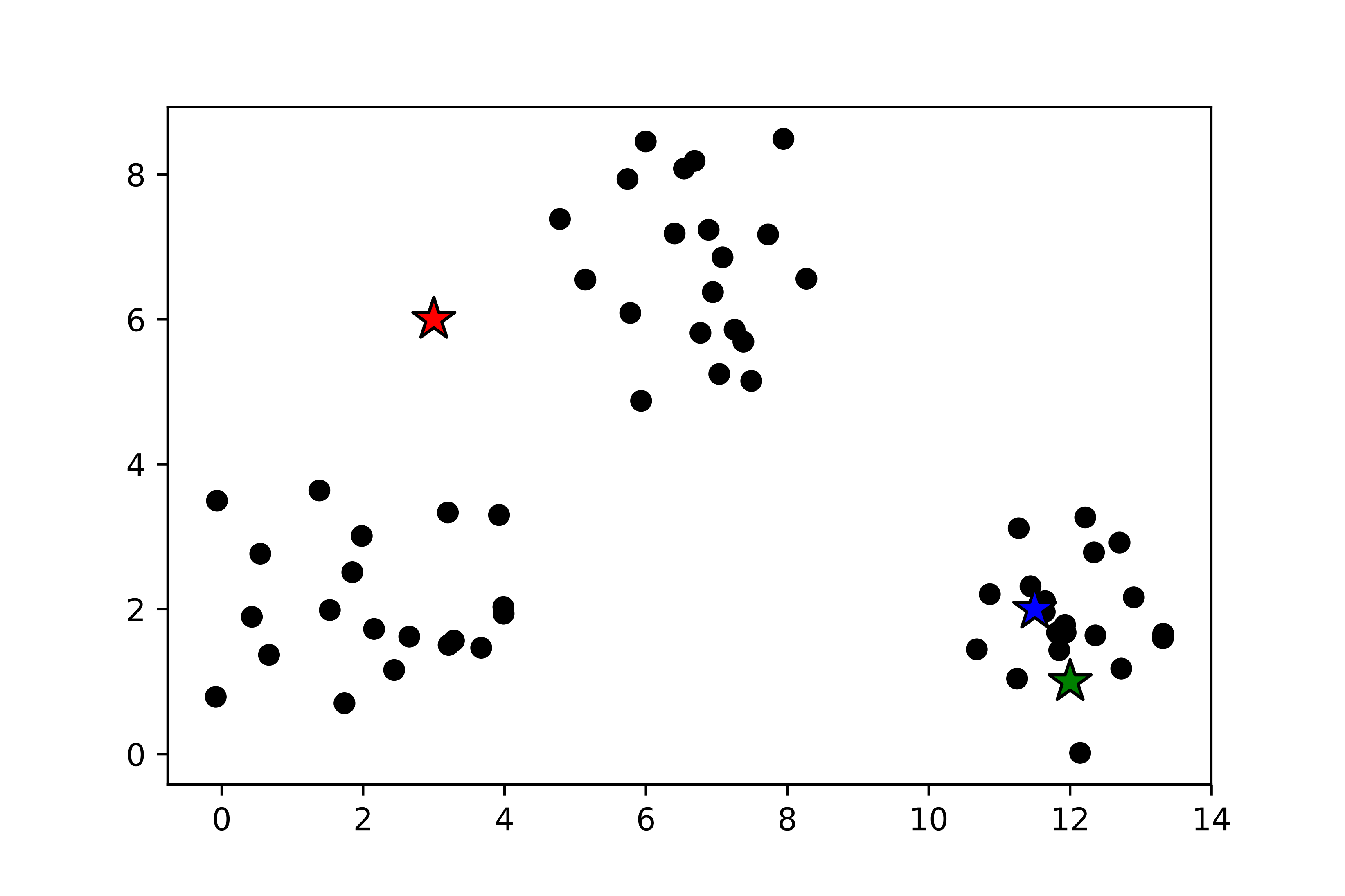
- Repeat

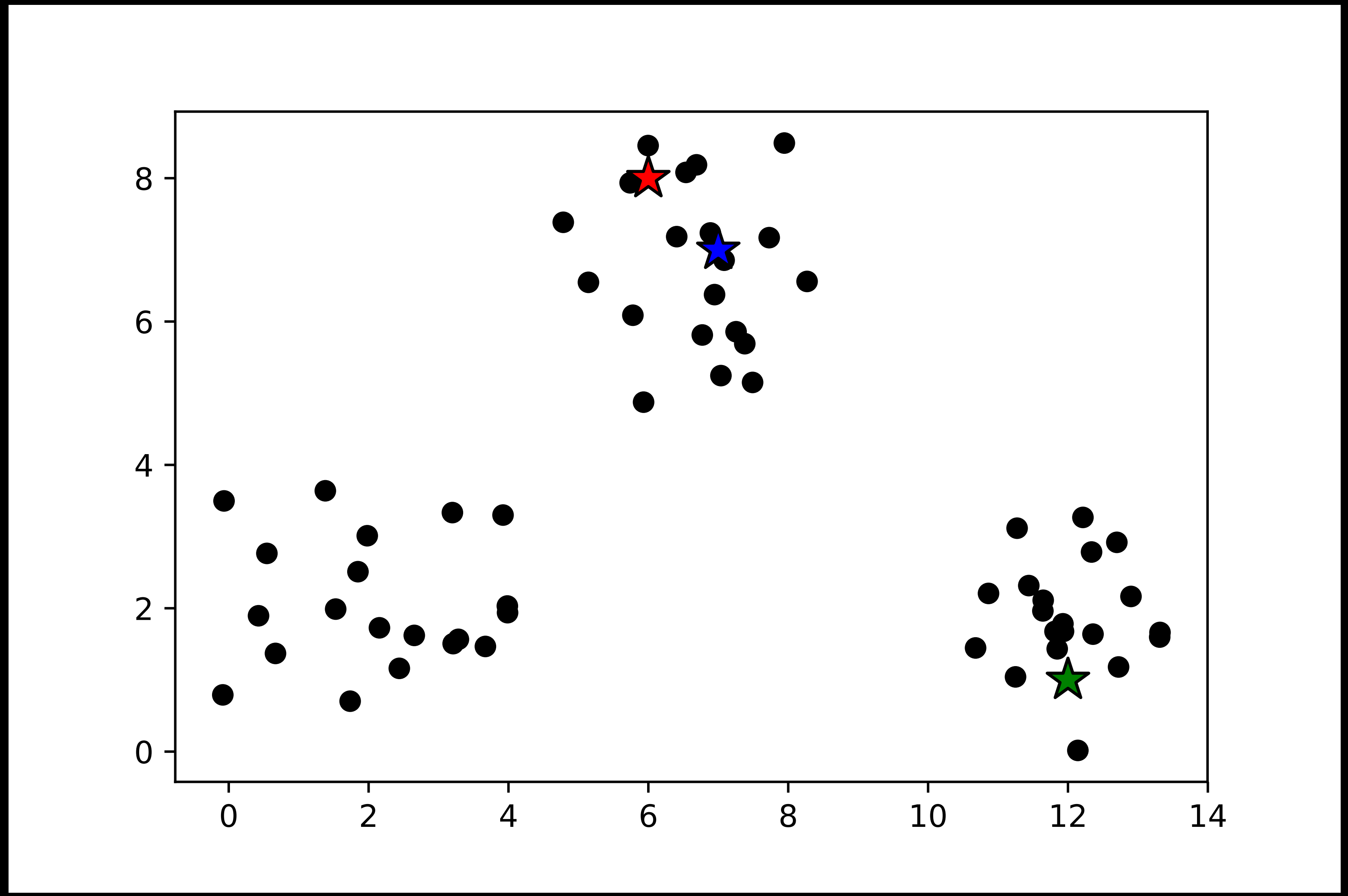


$$J = \frac{1}{m}\sum_{i=0}^{m} \left| x_i - \mu_{c^{(i)}} \right|^2$$



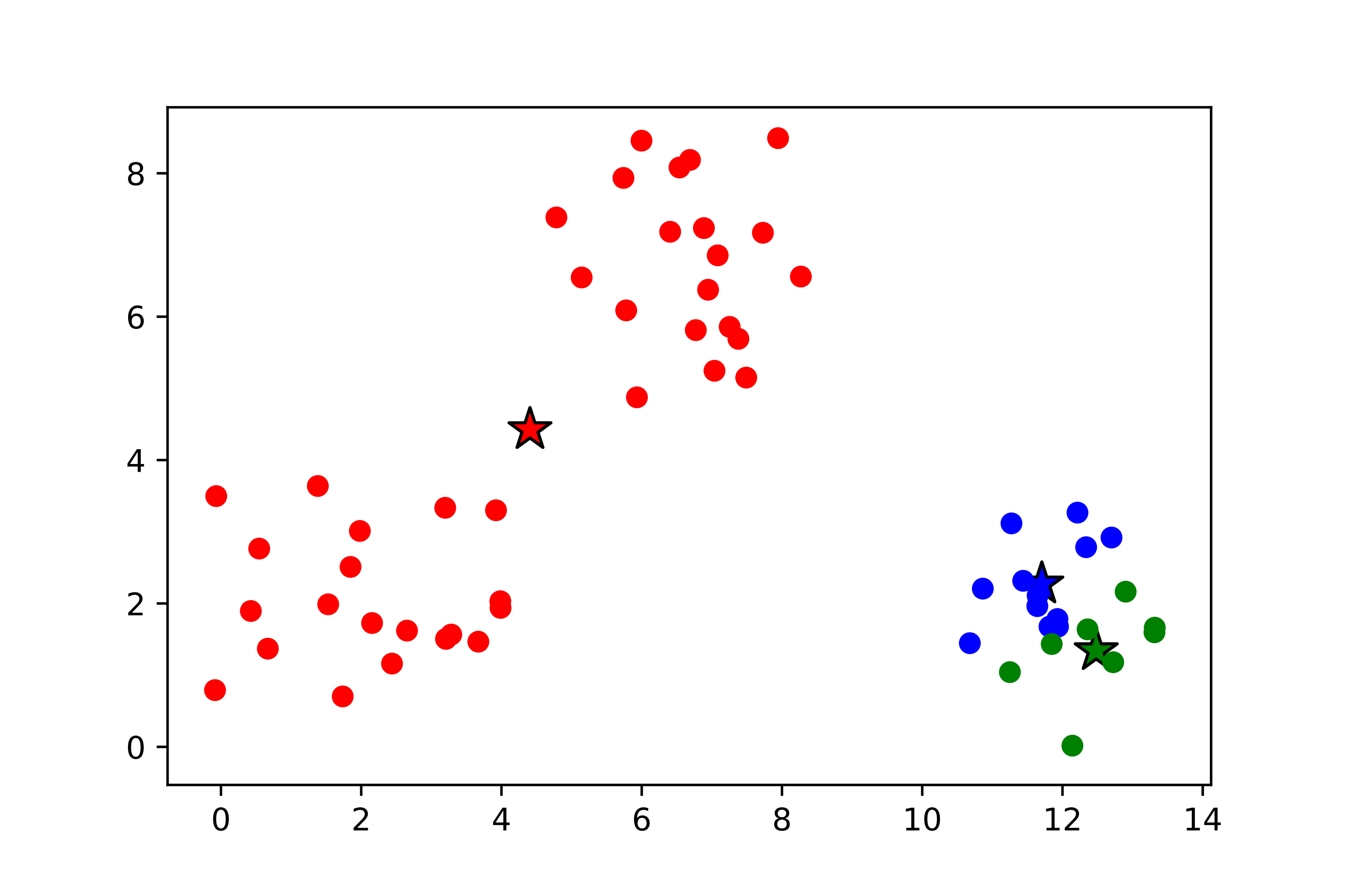
Possible to end up in local minimum

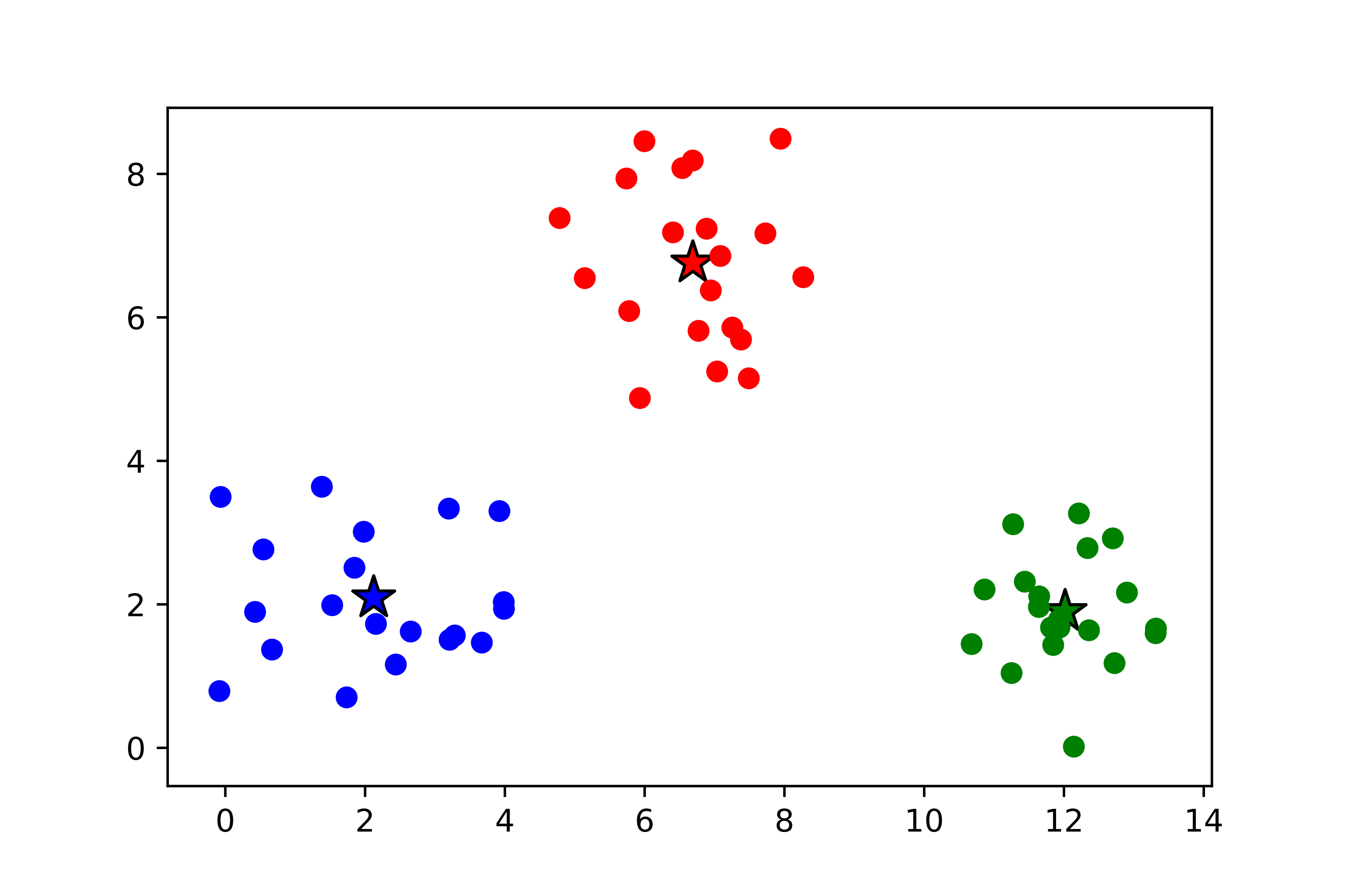


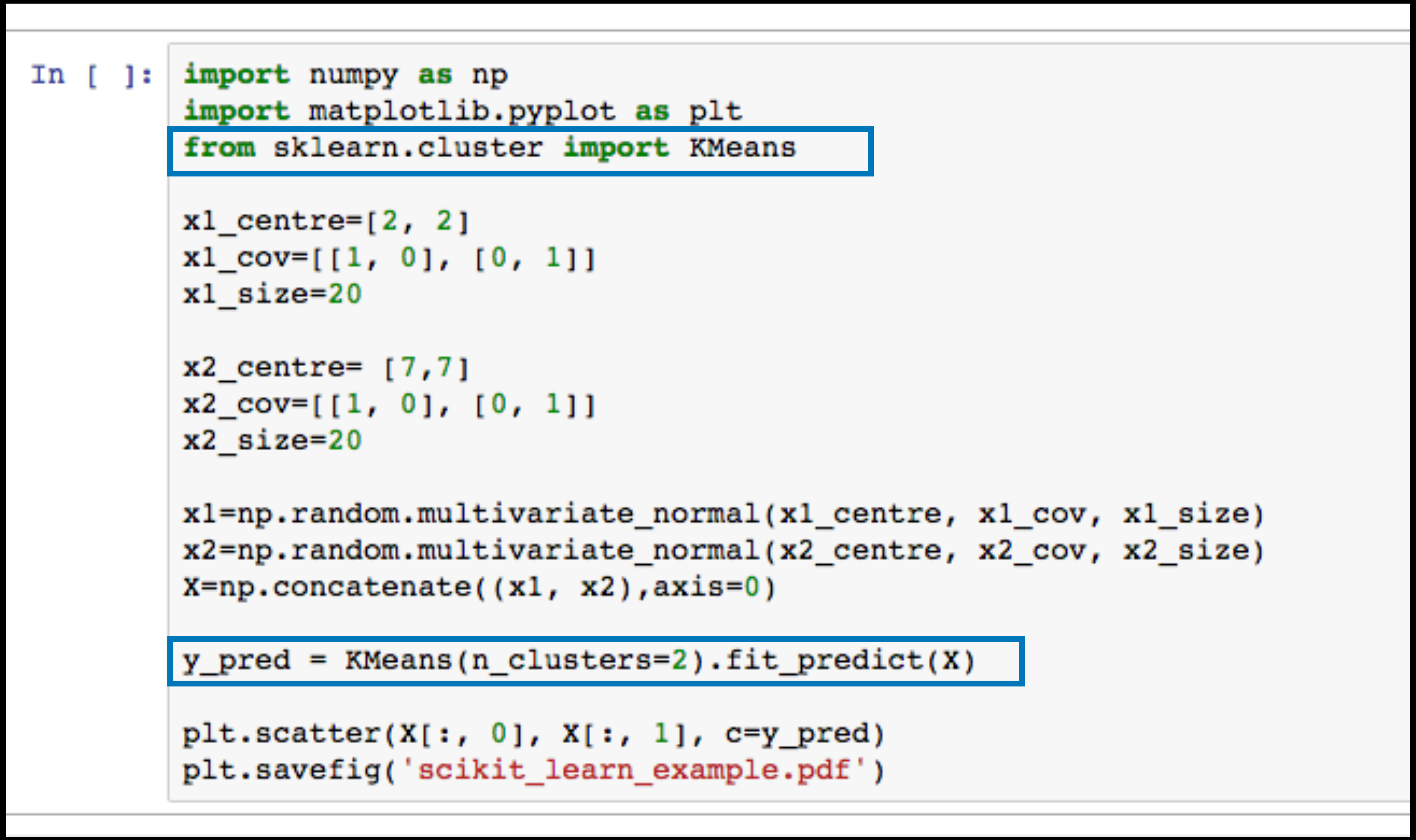


$J \approx 3$

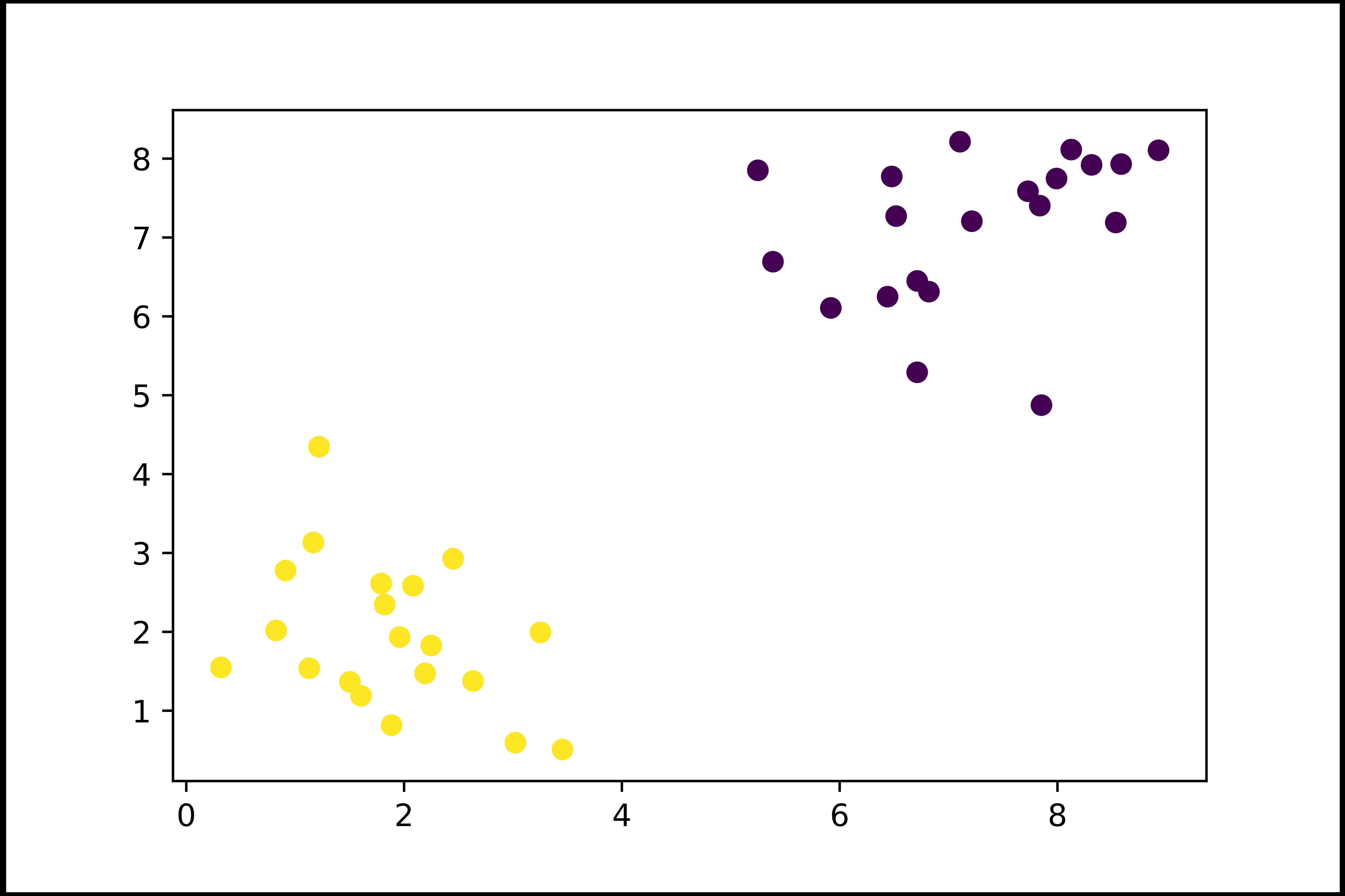
$J \approx 1$







```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

x1_centre=[2, 2]
x1_cov=[[1, 0], [0, 1]]
x1_size=20

x2_centre= [7,7]
x2_cov=[[1, 0], [0, 1]]
x2_size=20

x1=np.random.multivariate_normal(x1_centre, x1_cov, x1_size)
x2=np.random.multivariate_normal(x2_centre, x2_cov, x2_size)
X=np.concatenate((x1, x2),axis=0)

y_pred = KMeans(n_clusters=2).fit_predict(X)

plt.scatter(X[:, 0], X[:, 1], c=y_pred)
plt.savefig('scikit_learn_example.pdf')
```



http://scikit-learn.org/stable/modules/clustering.html#k-means

https://www.coursera.org/learn/machine-learning