The k-means Clustering Algorithm

1 Motivation

- Given a training set $D = \{x_1, x_2, \cdots, x_m\}$, we want to group the data into a few cohesive "clusters." Here, $x_i \in R^n$ as usual; but no labels y_i are given.
- So, this is an **unsupervised learning** problem.

2 Algorithm

- The algorithm contains the following steps.

1. Initialize cluster centroids $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^n$ randomly.
2. Repeat until convergence:
   - For every i, set,

$$c^{(i)} := \arg\min_j \left\| x^{(i)} - \mu_j \right\|^2$$

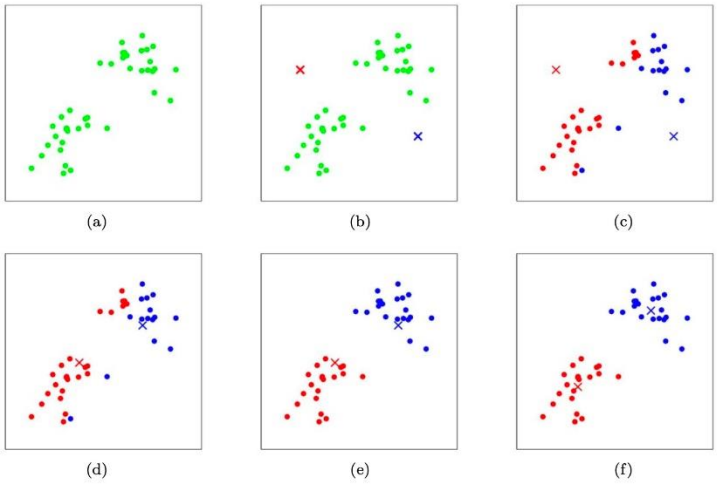
   - For each j, set,

$$\mu_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)} = j\}}$$

   - where $1\{\cdot\}$ represents the indicator function which returns 1 if its argument is true, otherwise 0. In other words, the indicator of a true statement is equal to 1 while that of a false statement is equal to 0.

3 Initialize parameters:

a) k: the number of clusters we want to find
b) μ_j: the mean of the jth cluster, where $j = 1,2, \cdots, k$
c) To initialize the cluster centroids (in step 1 of the algorithm above), we could choose k training examples randomly, and set the cluster centroids to be equal to the values of these k examples.



4 Convergence

- Let us define the distortion function as

$$J(c,\mu) = \sum_{i=1}^{m} \left\| x^{(i)} - \mu_{c^{(i)}} \right\|^2$$

- The distortion function J is decreasing with respect to both x_i and μ_j in two steps.
- However, the distortion function J is not convex. The k-mean cluster algorithm may converge to different local optimal solutions.
- How to solve the issue of local optimal? - run the algorithm multiple times and pick the best one with the lowest distortion value $J(c,\mu)$.

5 Example

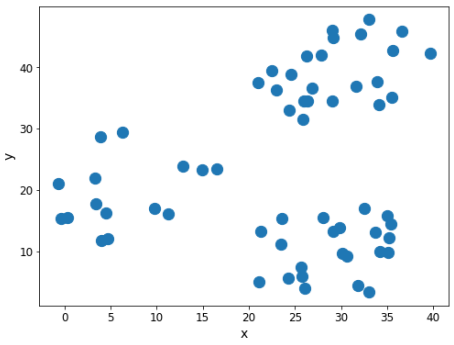
https://www.dominodatalab.com/blog/getting-started-with-k-means-clustering-in-python

https://www.w3schools.com/python/python_ml_k-means.asp

- Import data and display

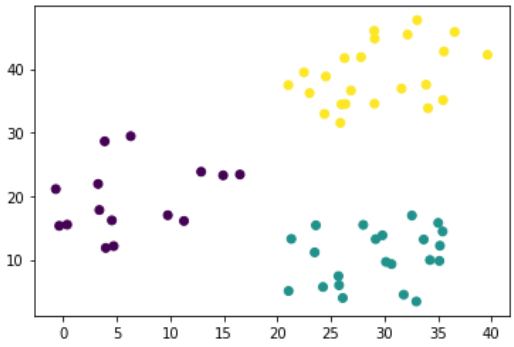
```
'''
-----------------------------------------------
Show data in two-dimension
'''
customcmap = ListedColormap(["crimson", "mediumblue", "darkmagenta"])

fig, ax = plt.subplots(figsize=(8, 6))
plt.scatter(x=blobs['x'], y=blobs['y'], s=150,
            #c=blobs['cluster'].astype('category'),
            cmap = customcmap)
ax.set_xlabel(r'x', fontsize=14)
ax.set_ylabel(r'y', fontsize=14)
plt.xticks(fontsize=12)
plt.yticks(fontsize=12)
plt.show()
```



- Train the clustering algorithm and show results

```
'''
---------------------------------
Cluster data into three classes
'''
kmeans = KMeans(n_clusters=3)
kmeans.fit(data)
inertias.append(kmeans.inertia_)
plt.scatter(data[:,0], data[:,1], c=kmeans.labels_)
plt.show()
```



Reference

https://aman.ai/cs229/kmc/

https://www.dominodatalab.com/blog/getting-started-with-k-means-clustering-in-python

https://www.w3schools.com/python/python_ml_k-means.asp