University of Illinois Fall 2018

ECE 544NA: Pattern Recognition
Lecture 16: October 18

Lecturer: Alexander Schwing Scribe: Yu Meng

1 Overview

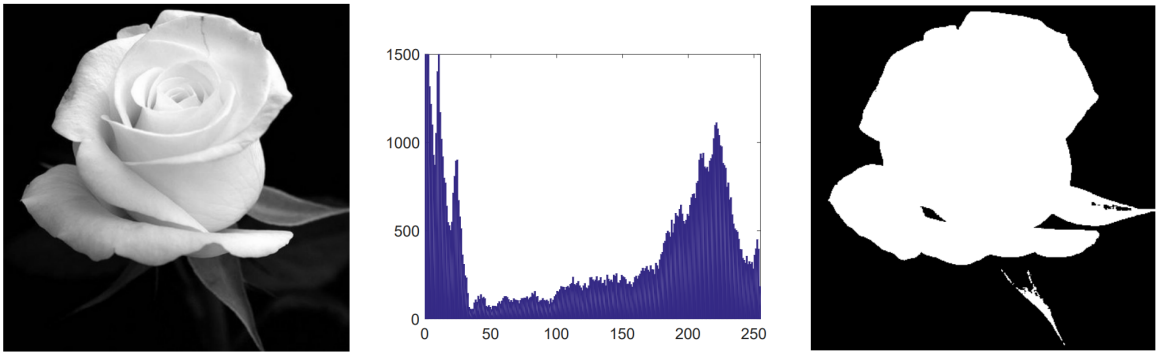
This lecture introduces K-Means, an unsupervised clustering algorithm which aims to partition N data points into K clusters. The goals of the lecture includes:

- Understanding clustering concept
- Getting to know K-Means

The following sections are organized as below: Section 2 provides an introduction to clustering tasks; Section 3 describes the K-Means algorithm in details; Section 4 provides some extensions based on standard K-Means algorithm; Section 5 enumerates some applications of K-Means clustering algorithm; Section 6 summarizes the entire lecture.

2 Introduction to Clustering

In general, clustering refers to tasks that group a set of data points so that samples from the same group are more similar to each other than those from different groups, **without labeled data**.



(a) Original Image (b) Pixel Intensity Frequency (c) Clustering Based on Intensity

Figure 1: Image Segmentation Using Clustering.

Figure 1 shows a case where we use clustering to find the flower in the image without labeled data: For a grayscale image, we can characterize each pixel using its intensity value. The regions corresponding to the flower tend to have higher intensity values; while background pixels usually have lower intensity values. Therefore, we can use clustering based on pixel intensity to group the pixels corresponding to the flower together, as shown in Figure 1(c).

3 K-Means Algorithm

3.1 An Informal Description

Algorithm 1: K-Means Standard Algorithm (Lloyd's Algorithm).

Input: A set of data points $\mathcal{X} = \{x^{(i)}\}|_{i=1}^{N}$; number of clusters K.
Output: Cluster assignments $\mathcal{R} = \{r_i\}|_{i=1}^{N}$; cluster centers $\mathcal{M} = \{\mu_k\}|_{k=1}^{K}$.

$\mathcal{M} \leftarrow$ randomly pick K points (Initialization);
while $\mathcal{R}$ *changes from last iteration* **do**
  $\mathcal{R} \leftarrow$ assign each data point to the closest cluster center based on some metric (Assignment Step);
  $\mathcal{M} \leftarrow$ average of assigned points of the corresponding clusters (Update Step);
Return $\mathcal{R}$, $\mathcal{M}$;

```python
import numpy as np

def kmeans(data, k):
  n = data.shape[0]
  c = data.shape[1]

  # Initialize centers randomly
  centers = np.random.randn(k, c)
  clusters_old = np.zeros(n)
  distances = np.zeros((n,k))

  while True:

    # Measure the distance to every center
    for i in range(k):
      distances[:,i] = np.linalg.norm(data - centers[i], axis=1)

    # Assignment Step: assign data points to closest center
    clusters_new = np.argmin(distances, axis = 1)

    # Update Step: calculate mean for every cluster and update the center
    for i in range(k):
      centers[i] = np.mean(data[clusters_new == i], axis=0)

    # Check convergence
    if np.array_equal(clusters_old, clusters_new):
      break
    clusters_old = np.copy(clusters_new)

  return clusters_new, centers
```

Algorithm 1 gives an informal description of the K-Means (Lloyd's) algorithm. The algorithm first picks K random points as cluster centers, and iterate between an assignment step and an update step, until no points change cluster assignments. The above script provides the python implementation of K-Means (Lloyd's) algorithm.

Figure 2 demonstrates the cluster assignment change during K-Means process applied on a set of 2D points. At the beginning, two cluster centers are randomly selected. During the iterations, the data points are assigned to the closer cluster center, and the cluster centers are updated based on the assigned points. The process terminates when no points change cluster assignment.

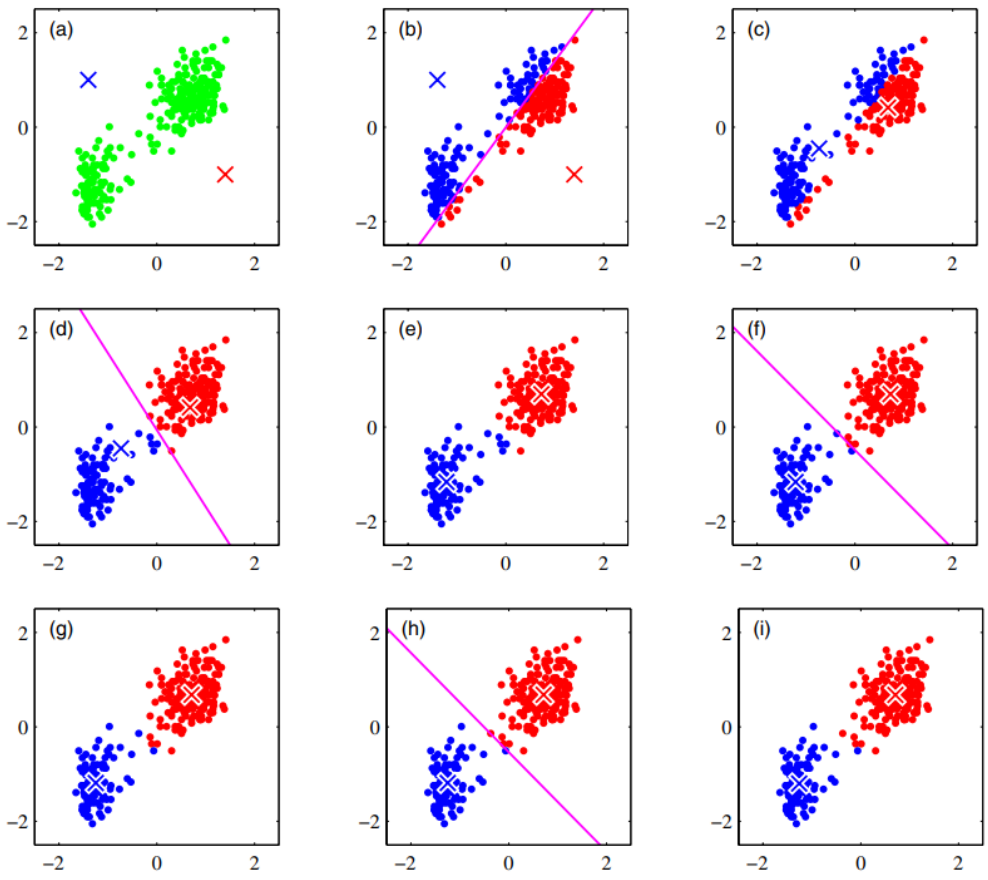


Figure 2: K-Means Process on an Example of 2D points.

3.2 Cost Function

When Euclidean distance is chosen as the distance measure, K-Means algorithm minimizes the sum of squared Euclidean distance between each point and its assigned cluster center. The cost function is as below:

$$\min_{\mu} \min_{r} \sum_{i \in \mathcal{D}} \sum_{k=1}^{K} \frac{1}{2} r_{ik} \|x^{(i)} - \mu_k\|_2^2 \quad s.t. \begin{cases} r_{ik} \in \{0,1\} & \forall i,k \\ \sum_{k=1}^{K} r_{ik} = 1 & \forall i \end{cases} \tag{1}$$

where $r_{ik} = 1$ if $x^{(i)}$ is assigned to cluster k, and $r_{ik} = 0$ otherwise.

3.3 Optimization

Finding the optimal solution for (1) is NP-hard [2, 5]. When data point dimension d and number of clusters K are fixed, the problem can be solved in $O(N^{dK+1})$ [7].

An approximative and faster algorithm, Lloyd's algorithm (Algorithm 1), applies an alternate optimization which minimizes Equation (1) with respect to μ and r in turns assuming the other one is fixed. Specifically, the alternate optimization iterates between the following two steps:

- Optimize with respect to r given μ:

We assign each point to the closest cluster, i.e.

$$r_{ik} = \begin{cases} 1 & \text{if } k = \text{argmin}_{k \in \{1,\dots,K\}} \|x^{(i)} - \mu_k\|_2^2 \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

- Optimize with respect to μ given r:
  We compute the partial derivative of the cost function in Equation (1) with respect to μ, and set it to zero, i.e., let

$$\nabla_{\mu_k} = \sum_{i \in \mathcal{D}} r_{ik}(x^{(i)} - \mu_k) = 0,$$

  we obtain

$$\mu_k = \frac{\sum_{i \in \mathcal{D}} r_{ik} x^{(i)}}{\sum_{i \in \mathcal{D}} r_{ik}} \tag{3}$$

3.4 Properties of the Algorithm

- Local minimum is found, but there is no guarantee that the global optimal is found [6].
- The algorithm is guaranteed to converge in a finite number of iterations.
  - Non-increasing objective
    In Equations (3) and (2), r_{ik} and μ_k are chosen to minimize the objective given the other one is fixed. Therefore, the objective will either stays the same or decrease, but cannot increase.
  - Iteration bound
    A trivial upper bound $O(K^N)$ can be easily observed because no partition of points into clusters is ever repeated during the course of the algorithm. A lower bound is found and proved in [3] that Lloyd's algorithm requires $2^{\Omega(\sqrt{N})}$ iterations to converge in the worst case.
- Running time per iteration consists of two parts:
  - Assign data points to the closest cluster center;
    Since computing Equation (2) requires $O(d)$ where d is the data point dimension, the total running time for computing every r_{ik} is $O(KNd)$.
  - Update the cluster centers to be the average of the assigned points.
    Since every data point will be used in computation once when Equation (3) is computed for all clusters, the running time is $O(N)$.

  Overall, the running time of the algorithm is $O(KNd)$ per iteration.

4 Extensions

4.1 K-Means++

K-Means algorithm can only find local optimal, and the clustering result depends heavily on initialization. K-Means++ addresses this issue by specifying a procedure to initialize the cluster centers before proceeding with the standard K-Means optimization algorithm. The major idea is to make the initial cluster centers far away from each other, and then apply the standard K-Means algorithm. The procedure is shown in Algorithm 2. The script below provides a python implementation of K-Means++ method for initializing cluster centers.

Algorithm 2: K-Means++ Algorithm.

Input: A set of data points $\mathcal{X} = \{x^{(i)}\}|_{i=1}^{N}$; number of clusters K.
Output: Cluster assignments $\mathcal{R} = \{r_i\}|_{i=1}^{N}$; cluster centers $\mathcal{M} = \{\mu_k\}|_{k=1}^{K}$.

$\mu_1 \leftarrow$ randomly select one point $x^{(i)} \in \mathcal{X}$;
for $i \leftarrow 2$ *to* K **do**
  $\mu_i \leftarrow$ select one point $x^{(i)} \in \mathcal{X}$ with probability proportional to $\min_{1 \le k \le i-1} \|x^{(i)} - \mu_k\|_2^2$;
$\mathcal{M} \leftarrow \{\mu_k\}|_{k=1}^{K}$;
$\mathcal{R}, \mathcal{M} \leftarrow$ Standard K-Means Algorithm (using $\mathcal{M}$ as initialization);
Return $\mathcal{R}$, $\mathcal{M}$;

```python
import numpy as np

def kpp_init(data, k):
  n = data.shape[0]
  centers = []

  # Randomly select the first cluster center
  select = np.random.choice(n)
  centers.append(data[select])

  for i in range(1, k):
    # Compute distances between each data point and each current cluster center
    distances = np.zeros((n, i))
    for j in range(i):
      distances[:, j] = np.linalg.norm(data - centers[j], axis=1)
    distances = np.min(distances, axis=1)

    # Select a new center to make it far away from the current ones
    prob = distances/np.sum(distances)
    select = np.random.choice(n, p=prob)
    centers.append(data[select])

  return np.array(centers)
```

4.2 Feature Space and Distance Measures

K-Means algorithm can use the following distance measures:

- Euclidean (most commonly used, as described in Section 3);
- Cosine, usually cosine dissimilarity is used as the distance measure;

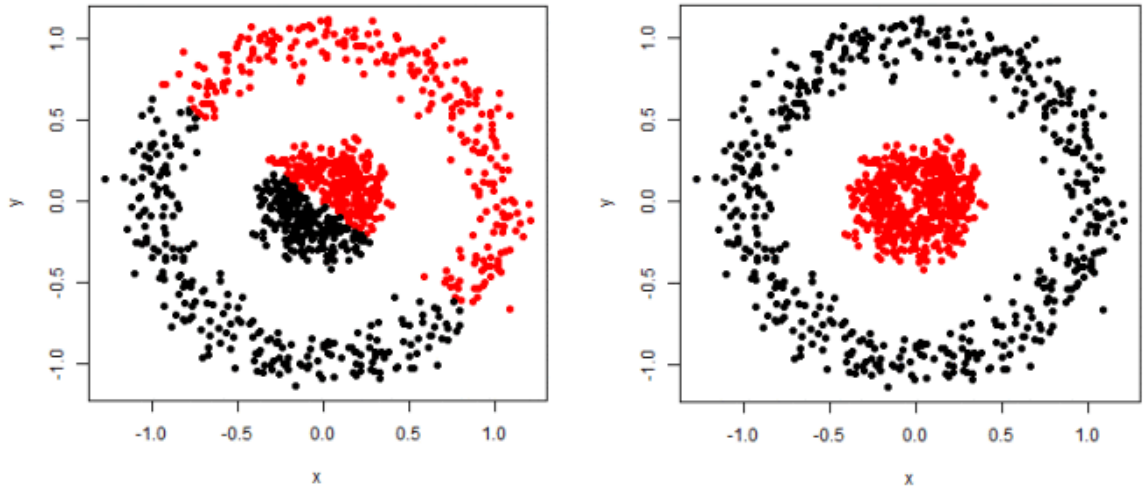
$$d_{cos}(x^{(i)}, \mu_k) = 1 - \cos(x^{(i)}, \mu_k) = 1 - \frac{x^{(i)\top}\mu_k}{\|x^{(i)}\|_2 \|\mu_k\|_2}.$$

  In some cases, the data points are normalized unit-length vectors (e.g. normalized document vectors are usually used in document clustering to account for different document length). Under those settings, using cosine dissimilarity and using Euclidean distance becomes equiv-

alent:

$$
\begin{aligned}
d_{euclidean}(x^{(i)}, \mu_k) &= \|x^{(i)} - \mu_k\|_2^2 \\
&= \|x^{(i)}\|_2^2 - 2x^{(i)\top}\mu_k + \|\mu_k\|_2^2 \\
&= 2 - 2x^{(i)\top}\mu_k \qquad \left(\|x^{(i)}\|_2 = \|\mu_k\|_2 = 1\right) \\
&= 2\left(1 - \frac{x^{(i)\top}\mu_k}{\|x^{(i)}\|_2\|\mu_k\|_2}\right) \\
&= 2d_{cos}(x^{(i)}, \mu_k)
\end{aligned}
$$

- Non-linear, achieved by using a kernel function $\phi(x^{(i)})$ to map data points from original space to the transformed space.



(a) Standard K-Means

(b) Kernel K-Means

Figure 3: Comparison Between Standard K-Means and Kernel K-Means on 2D Concentric Data.

As shown in Figure 3, when the 2D data points are distributed as concentric circles, standard K-Means fails to detect the two natural clusters (Figure 3(a)). By applying a polynomial kernel to the original data points,

$$\kappa(x, y) = (x^\top y)^2,$$

the data points will be mapped via a non-linear transformation to feature space where points can be well separated (Figure 3(b)).

Generally, standard K-Means does not work well for clusters with non-convex shapes.

4.3 Evaluations

- Generative: The measure focuses on how well are points reconstructed from the clusters. Reconstruction error (distortion) is defined as [8]

$$\frac{1}{N}\sum_{i\in\mathcal{D}}\|x^{(i)} - \mu_{z_i}\|_2^2,$$

where z_i is the cluster index that $x^{(i)}$ is assigned to.

- Discriminative: The measure focuses on how well the clusters correspond to labels. Purity is defined as [4]

$$\frac{1}{N}\sum_{m\in\mathcal{M}}\max_{c\in\mathcal{C}}|m\cap c|,$$

where $\mathcal{M}$ is the set of clusters, and $\mathcal{C}$ is the set of classes.

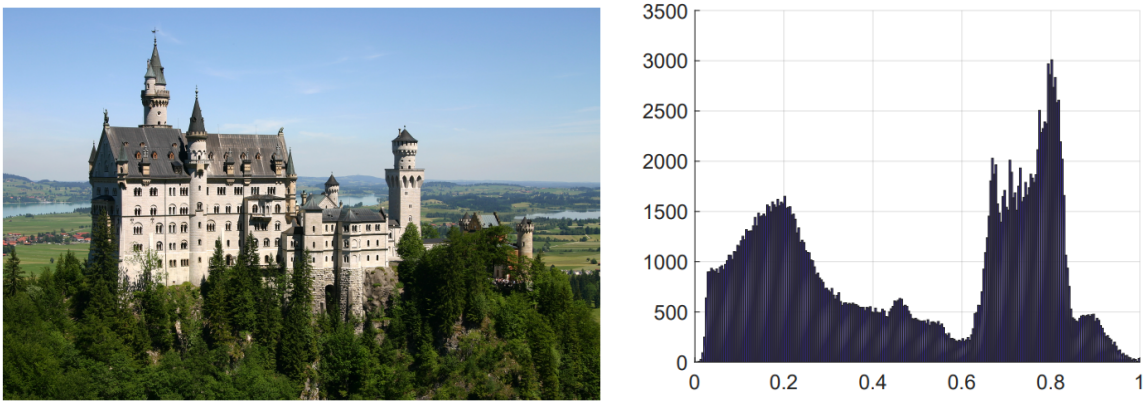
Note that neither of the measures penalizes having many clusters. When assigning each data point to its own cluster, we can achieve the minimum reconstruction error 0 and maximum purity value 1.

5 Applications

5.1 Clustering in Image

Figure 4(a) shows the original image and the pixel intensity frequency in the image. In section 2, we apply a clustering algorithm on a grayscale image based on the pixel intensity. Here, we can use a similar method by first changing the original image to grayscale (i.e., $x^{(i)} \in \mathbb{R}$), and then use K-Means to detect similar regions in the image. When we set $K = 2$ and $K = 3$, the results are shown in Figure 5(a) and Figure 5(b), respectively.

Another way for image segmentation is to directly apply K-Means algorithm in color space (i.e., $x^{(i)} \in \mathbb{R}^3$) based on the pixel intensity in the 3 color channels. Since the feature space becomes more complicated than in grayscale, the results generated by K-Means are more unstable (i.e., we observe completely different segmentation results by running K-Means multiple times).

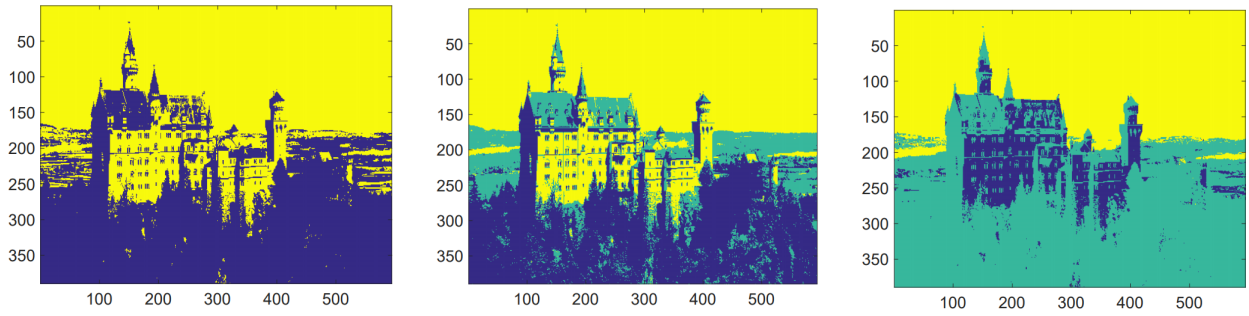


(a) Orignial Image (b) Pixel Intensity Frequency

Figure 4: Original Image and Pixel Intensity Frequency.

5.2 Superpixel Segmentation

In many cases, not only do we want to consider pixel intensity for image segmentation, but also we would like to take into account spacial relationships between pixels. For example, pixels that are close to each other are more likely to be of the same object than pixels that are far away. Therefore, we want to guarantee spatial smoothness by augmenting the feature space to contain spatial coordinates in addition to intensities.



(a) Grayscale 2 clusters (b) Grayscale 3 clusters (c) Color Space 3 clusters

Figure 5: Applying K-Means on the Original Image (Grayscale and Color Space).

Figure 6 shows the superpixel segmentation results by clustering in 5D feature space (3D lab color space and 2D spatial coordinates), i.e.,

$$x^{(i)} = \begin{bmatrix} l & a & b & x & y \end{bmatrix}^\top.$$

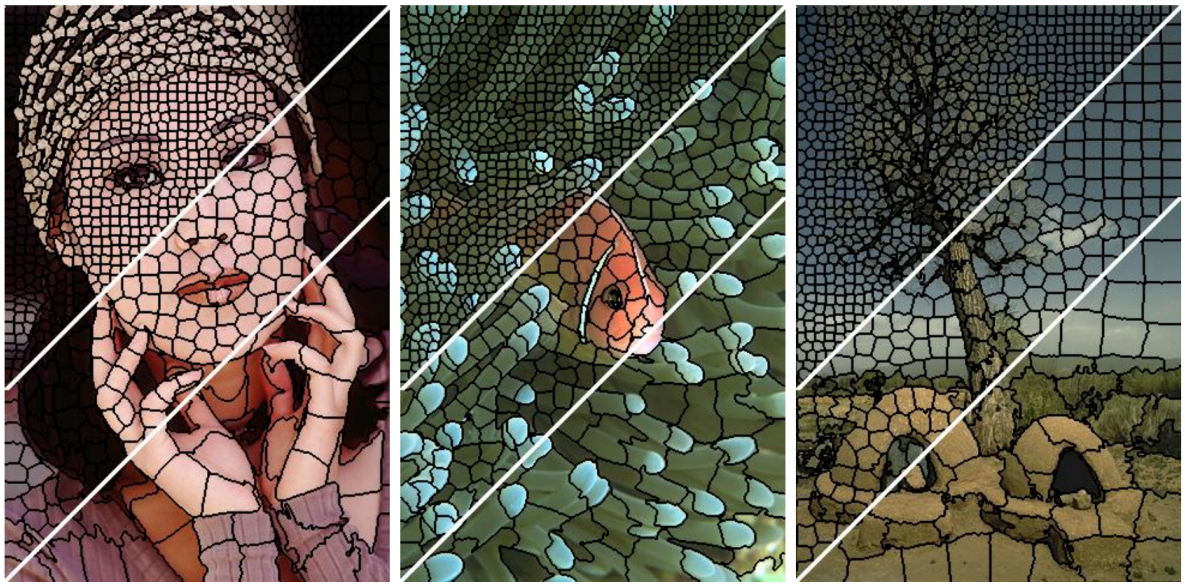


Figure 6: Superpixel Segmentation.

[1] applies a distance metric that treats color dimensions and spatial dimensions differently. Specifically,

$$\begin{aligned} d_c &= \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2}, \\ d_s &= \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}, \\ d' &= \sqrt{\left(\frac{d_c}{N_c}\right)^2 + \left(\frac{d_s}{N_s}\right)^2}, \end{aligned}$$

where N_c and N_s are the maximum color and spatial distances within a cluster, respectively.

Initial cluster centers are spaced regularly on the image and slightly perturbed to avoid edges, by moving cluster centers to the lowest gradient position in a 3×3 neighborhood [1].

6 Summary

In this lecture, we introduce an unsupervised clustering algorithm, K-Means, along with its properties and extensions. In summary, it has the following pros and cons:

- Pros
  - Simple;
  - Easy to implement.
- Cons
  - Need to choose the number of clusters K;
  - Sensitive to outliers: Outliers will also be assigned to specific clusters, which distorts the corresponding cluster centers;
  - Can get stuck in local minima: K-Means are only guaranteed to converge to local minimum, and the result can be arbitrarily bad/wrong;
  - All cluster centers have same parameters (non-adaptive);
  - Can be slow: Running time is $O(KNd)$ per iteration, and it might take many iterations to converge.

It has the following extensions and applications:

- Extensions
  - K-Means++: A better way to initialize cluster centers;
  - Feature Space and Distance Measures: K-Means can use different distance measures, including non-linear ones.
  - Evaluations: Both generative and discriminative measures can be used for evaluating the clustering quality.
- Applications
  - Clustering in Image: K-Means can be used to perform image segmentation on both grayscale and color space images;
  - Superpixel Segmentation: Spacial smoothness can be taken into account for clustering features.

References

[1] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Süsstrunk, et al. Slic superpixels compared to state-of-the-art superpixel methods. *IEEE transactions on pattern analysis and machine intelligence*, 34(11):2274–2282, 2012.

[2] D. Aloise, A. Deshpande, P. Hansen, and P. Popat. Np-hardness of euclidean sum-of-squares clustering. *Machine learning*, 75(2):245–248, 2009.

[3] D. Arthur and S. Vassilvitskii. How slow is the k-means method? In *Proceedings of the twenty-second annual symposium on Computational geometry*, pages 144–153. ACM, 2006.

[4] D. M. Christopher, R. Prabhakar, and S. Hinrich. Introduction to information retrieval. *An Introduction To Information Retrieval*, 151(177):5, 2008.

[5] M. Garey, D. Johnson, and H. Witsenhausen. The complexity of the generalized lloyd-max problem (corresp.). *IEEE Transactions on Information Theory*, 28(2):255–256, 1982.

[6] J. A. Hartigan and M. A. Wong. Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108, 1979.

[7] M. Inaba, N. Katoh, and H. Imai. Applications of weighted voronoi diagrams and randomization to variance-based k-clustering. In *Proceedings of the tenth annual symposium on Computational geometry*, pages 332–339. ACM, 1994.

[8] C. Robert. Machine learning, a probabilistic perspective, 2014.