AI and Machine Learning

Zhiyun Lin

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

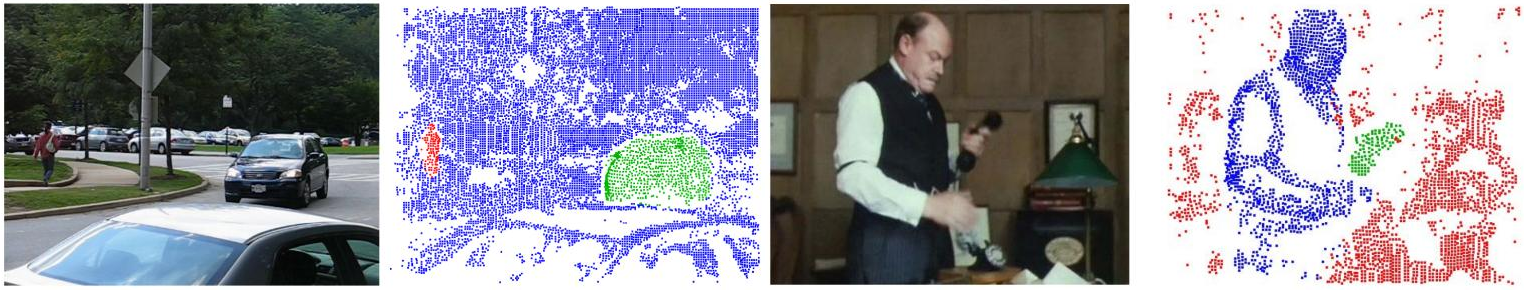
Clustering

- K-means
- Soft K-means

Motivating Example

- Determine groups of people in image above
  - based on clothing styles
  - gender, age, etc

Motivating Example



- Determine moving objects in videos

Unsupervised Learning

- **Supervised learning** algorithms have a clear goal: produce desired outputs for given inputs.
  - You are given $\{(x^{(i)}, t^{(i)})\}$ during training (inputs and targets)
- Goal of **unsupervised learning** algorithms (no explicit feedback whether outputs of system are correct) less clear.
  - You are given the inputs $\{x^{(i)}\}$ during training, labels are unknown.

Unsupervised Learning

- Tasks to consider:
  - Reduce dimensionality
  - Find clusters
  - Model data density
  - Find hidden causes
- Key utility
  - Compress data
  - Detect outliers
  - Facilitate other learning

Major Types

- Primary problems, approaches in unsupervised learning fall into three classes:

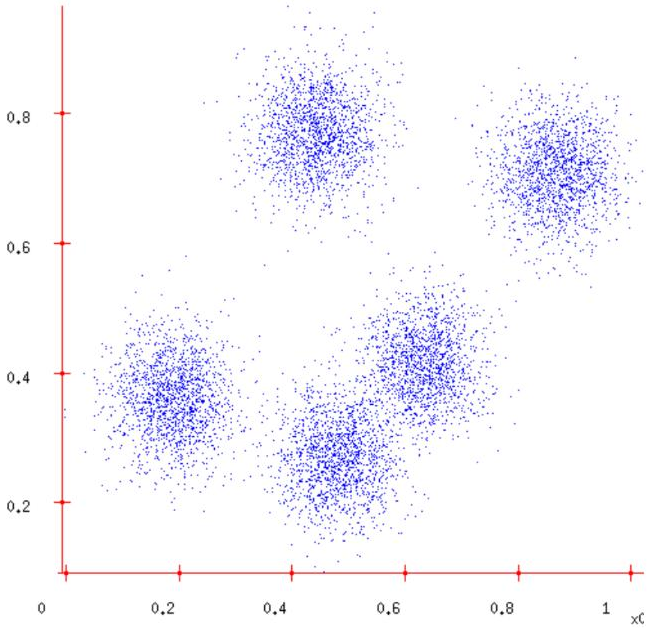
  i. **Dimensionality reduction:** represent each input case using a small number of variables (e.g., principal components analysis, factor analysis, independent components analysis)

  ii. **Clustering:** represent each input case using a prototype example (e.g., k-means, mixture models)

  iii. **Density estimation:** estimating the probability distribution over the data space

Clustering

- Grouping N examples into K clusters: one of canonical problems in unsupervised learning

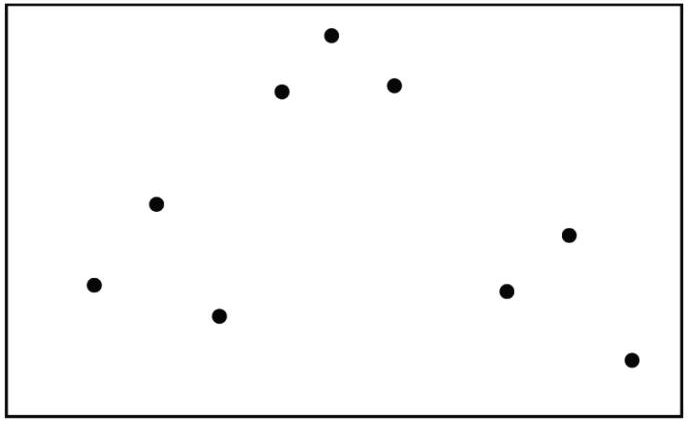


> Motivation: prediction; lossy compression; outlier detection

Clustering

- We assume that the data was generated from a number of different classes. The aim is to cluster data from the same class together.
  - How many classes?
  - Why not put each data point into a separate class?
- What is the objective function that is optimized by sensible clustering?

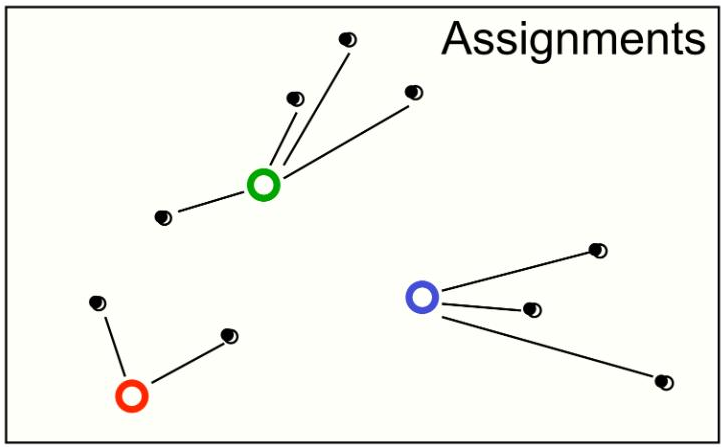
Clustering

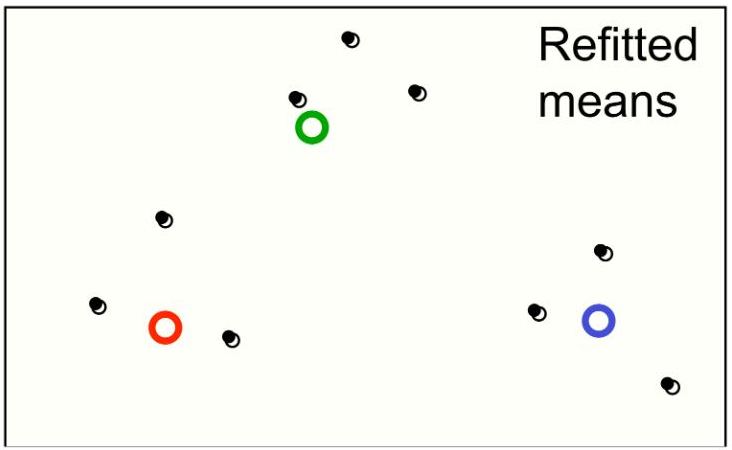


- Assume the data $\{x^{(1)}, \ldots, x^{(N)}\}$ lives in a Euclidean space, $x^{(n)} \in \mathbb{R}^d$.
- Assume the data belongs to K classes (patterns)
- How can we identify those classes (data points that belong to each class)?

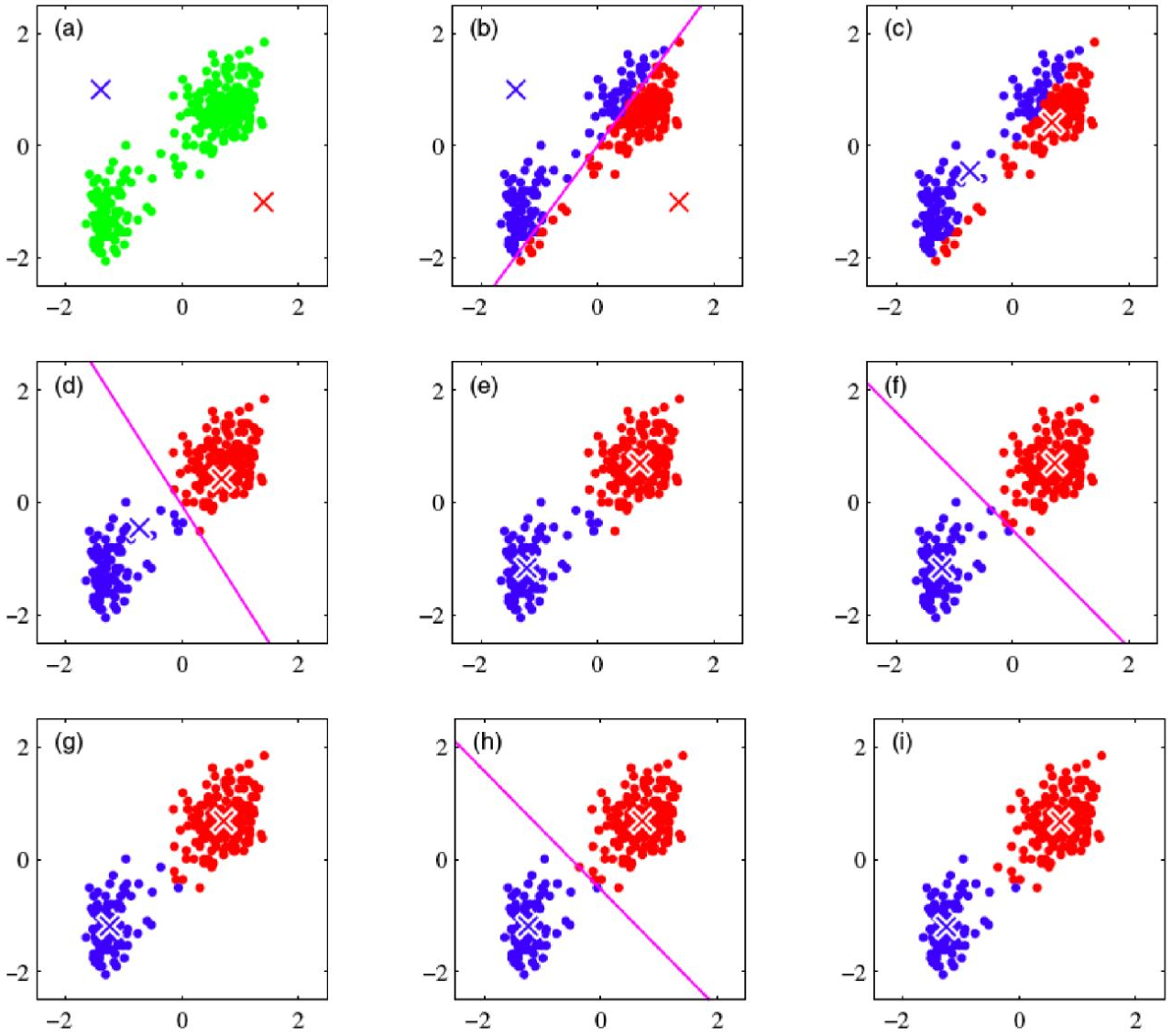
K-means

- **Initialization**: randomly initialize cluster centers
- The algorithm iteratively alternates between two steps:
  - **Assignment step**: Assign each data point to the closest cluster
  - **Refitting step**: Move each cluster center to the center of gravity of the dataassigned to it





K-means



K-means Objective

- What is actually being optimized?

> K-means Objective:
>
> Find cluster centers m and assignments r to minimize the sum of squared distance of data points $\{x^{(n)}\}$ to their assigned cluster centers

$$\min_{\{m\},\{r\}} J(\{m\},\{r\}) = \min_{\{m\},\{r\}} \sum_{n=1}^{N}\sum_{k=1}^{K} r_k^{(n)} \|m_k - x^{(n)}\|^2$$

$$\text{s. t.} \sum_k r_k^{(n)} = 1, \forall n, \quad \text{where} \quad r_k^{(n)} \in \{0,1\},\ \forall k, n$$

where $r_k^{(n)} = 1$ means that $x^{(n)}$ is assigned to cluster k with center m_k.

The K-means Algorithm

- **Optimization method** is a form of coordinate descent ("block coordinate descent")
  - Fix centers, optimize assignments (choose cluster whose mean is closest)
  - Fix assignments, optimize means (average of assigned datapoints)

The K-means Algorithm

- **Initialization**: Set K cluster means $m_1, \cdots, m_k$ to random values
- Repeat until convergence (until assignments do not change):

1. **Assignment**: Each data point $x^{(n)}$ assigned to nearest mean

$$\hat{k}^n = \arg\min_k d(m_k, x^{(n)})$$

> for example, $L2$ norm:
>
> $$\hat{k}^n = \arg\min_k ||m_k - x^{(n)}||^2$$

and **Responsibilities** (1 of k encoding)

$$r_k^{(n)} = 1 \longleftrightarrow \hat{k}^{(n)} = k$$

The K-means Algorithm

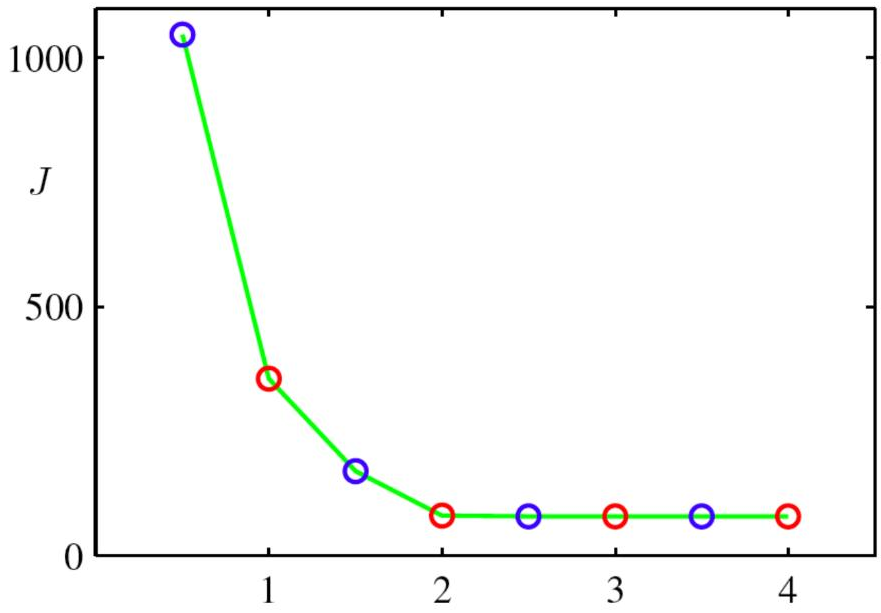
2. **Update**: Model parameters, means are adjusted to match sample means of data points they are responsible for:

$$m_k = \frac{\sum_n r_k^{(n)} x^{(n)}}{\sum_n r_k^{(n)}}$$

K-means for Vector Quantization

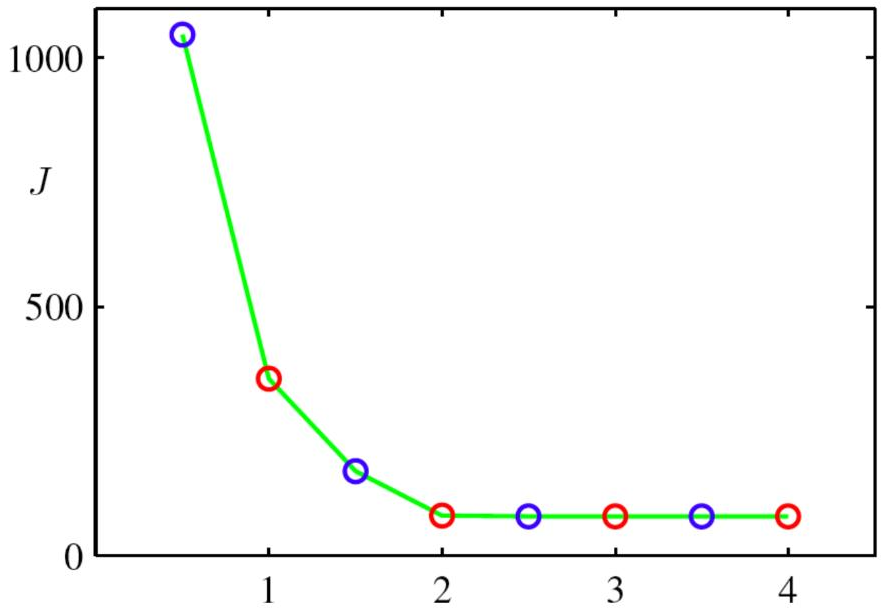
Why K-means Converges

- Whenever an assignment is changed, the sum squared distances J of data points from their assigned cluster centers is reduced.
- Whenever a cluster center is moved, J is reduced.



Why K-means Converges

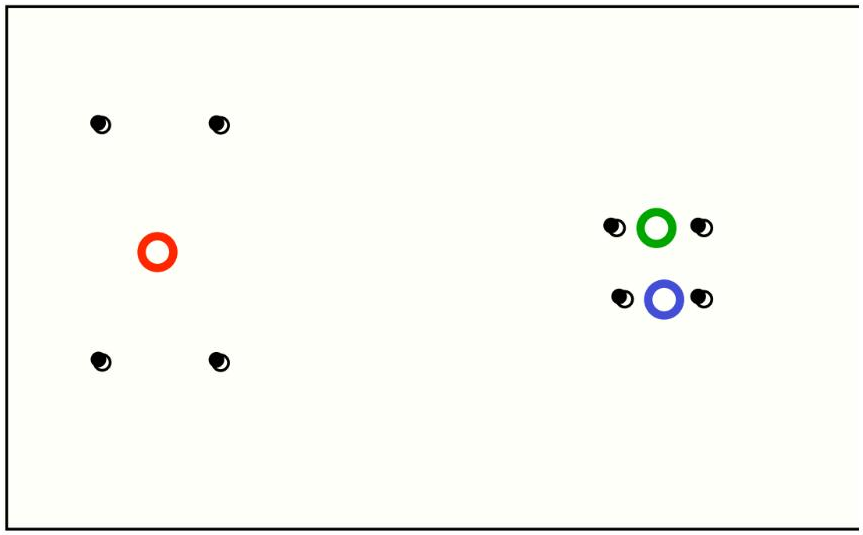
- **Test for convergence**: If the assignments do not change in the assignment step, we have converged (to at least a local minimum)
- K-means cost function after each E step (blue) and M step (red). The algorithm has converged after the third M step



Local Minima

- The objective J is non-convex (so coordinate descent on J is not guaranteed to converge to the global minimum)

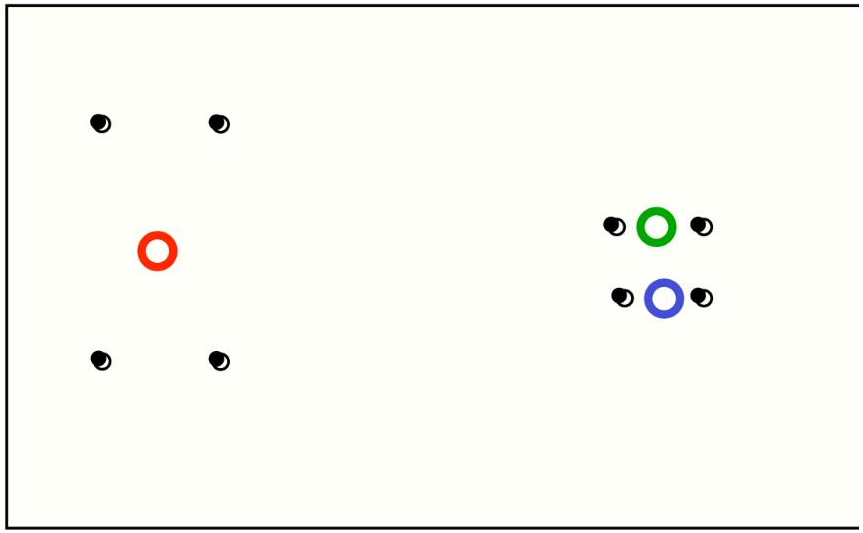
A bad local optimum



Local Minima

- There is nothing to prevent k-means getting stuck at local minima.
- We could try many random starting points
- We could try non-local split-and-merge moves:
  - Simultaneously **merge** two nearby clusters
  - and **split** a big cluster into two

A bad local optimum



K-means++

- K-means++ is an improvement over the standard K-means clustering algorithm
  - aims to improve the initial placement of cluster centers, leading to better quality clusters and faster convergence.
- The K-means++ algorithm is designed to reduce the likelihood of getting stuck in poor local minima, which is a common issue with the standard K-means algorithm.

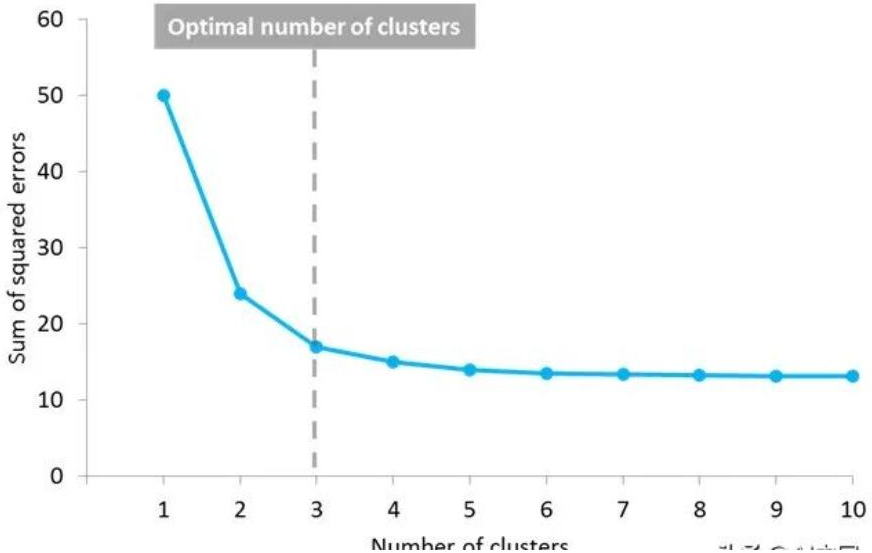
K-means++

1. **Initialize One Center**:
   - Choose the first center point m_1 randomly from the data points.
   - For each data point $x^{(n)}$, calculate the distance $D(x^{(n)})$ to the nearest center point that has already been chosen.
2. **Select the Next Center** (Farthest point):
   - For each data point $x^{(n)}$, calculate the probability $p(x^{(n)})$

$$p(x^{(n)}) = \frac{D(x^{(n)})^2}{\sum_j D(x^{(j)})^2}$$

   - Choose the next center m_{k+1} from the data points according to $p(x^{(n)})$.
3. **Repeat Step 2** until the desired number of centers K is reached.
4. **Run K-means algorithm**

Which K is optimal?



- A commonly used method is to test different K and measures the resulting **SSE** J.
- The value of K is chosen where an **increase leads to a very small decrease** in SSE, and a **decrease leads to a sharp increase**
- This point, which defines the optimal K, is known as the **"elbow point"**.

Mini-Batch K-Means

- Mini-Batch K-Means is a variant of the K-Means algorithm that is more **efficient for large datasets** by processing a small random sample of the data at each iteration instead of the entire dataset.

1. **Initialize Centers**:
   - Randomly select a subset of data points, called a **mini-batch**.
   - Initialize cluster centers from the mini-batch.
2. **Assign Clusters**:
   - For each data point in the mini-batch, assign it to the nearest center.
3. **Update Centers**:
   - Update the positions of the centers by calculating the mean of the points assigned to each cluster in the mini-batch.

Mini-Batch K-Means

4. **Repeat Steps 2 and 3**:
   - Repeat steps 2 and 3 for a fixed number of iterations or until the centers converge (i.e., the change in the positions of the centers is below a certain threshold).
5. **Expand Mini-Batch Size** (optional):
   - Optionally, increase the size of the mini-batch over time to improve the accuracy of the center updates.
6. **Refinement**:
   - After the initial convergence with mini-batches, the algorithm can be refined by running a few iterations of the standard K-Means algorithm on the entire dataset

Mini-Batch K-Means

7. **Convergence**:
   - The algorithm converges when the assignments of the data points to the clusters do not change significantly, or the change in the within-cluster sum of squares is below a certain threshold.

> Mini-Batch K-Means is particularly useful for large-scale clustering tasks where processing the entire dataset at once is computationally expensive or impractical.

Soft K-means

- Instead of making hard assignments of data points to clusters, we can make **soft assignments**.
- One cluster may have a responsibility of 0.7 for a datapoint and another may have a responsibility of 0.3.
  - Allows a cluster to use more information about the data in the refitting step.
  - What happens to our convergence guarantee?
  - How do we decide on the soft assignments?

Soft K-means Algorithm

- **Initialization**: Set K means $\{m_k\}$ to random values
- Repeat until convergence (until assignments do not change):

1. **Assignment**: Each data point n given soft "degree of assignment" to each cluster mean k, based on responsibilities

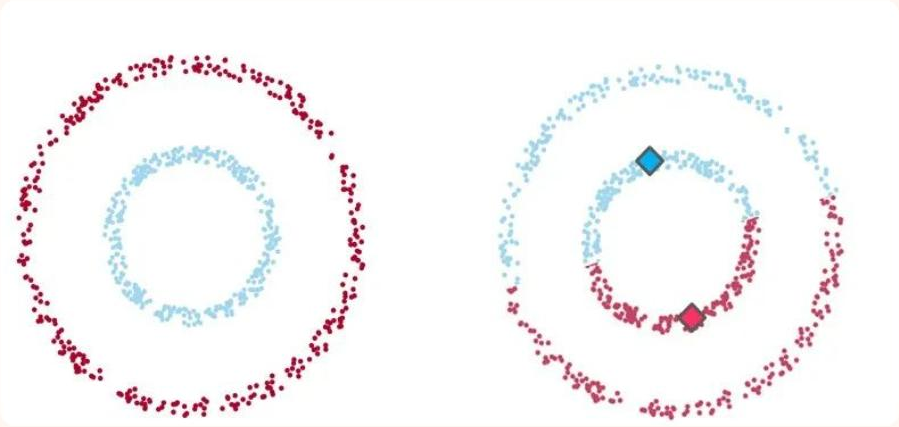
$$r_k^{(n)} = \frac{\exp[-\beta d(m_k, x^{(n)})]}{\sum_j \exp[-\beta d(m_j, x^{(n)})]}$$

2. **Update**: Model parameters, means, are adjusted to match sample means of data points they are responsible for:

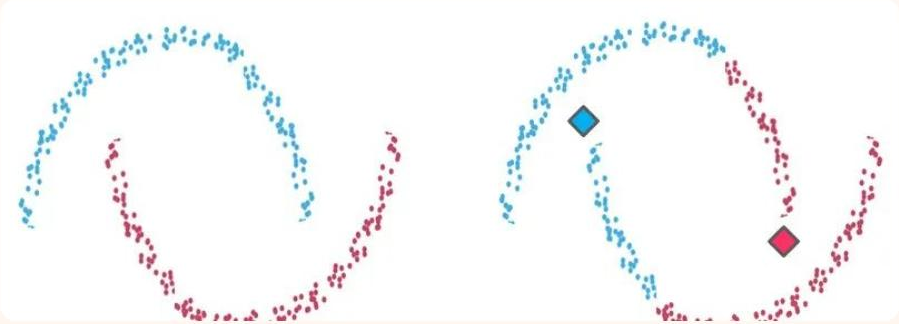
$$m_k = \frac{\sum_n r_k^{(n)} x^{(n)}}{\sum_n r_k^{(n)}}$$

Perform poorly on non-convex data

- Example 1:

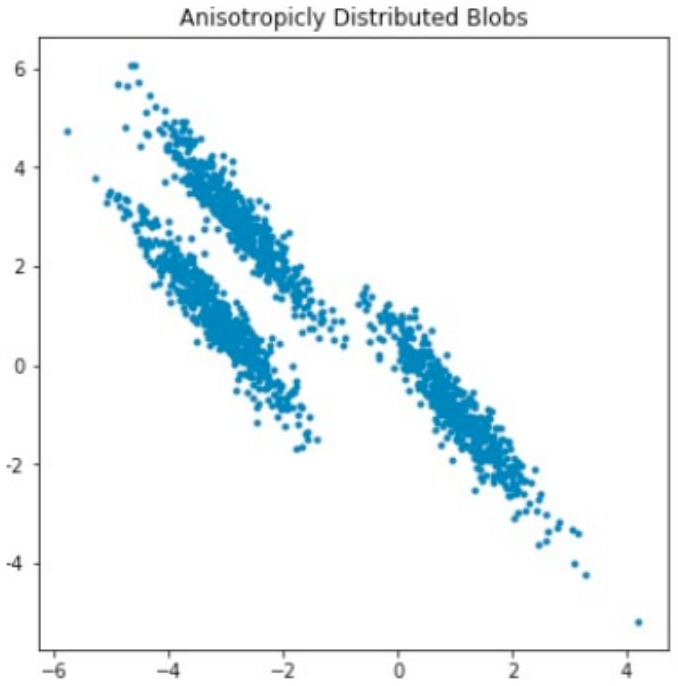


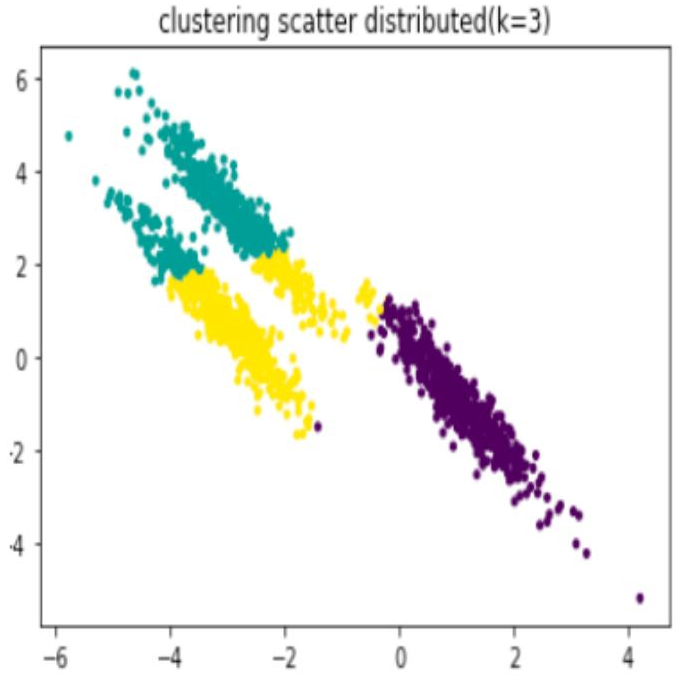
- Example 2:



Perform poorly on anisotropic data

- Anisotropic data





Perform poorly on data with size variation

- If the number of samples in each cluster varies greatly, the clustering performance is poor