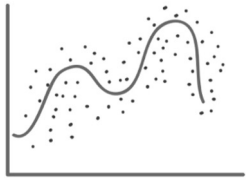
Machine Learning

Lecture 15: Clustering

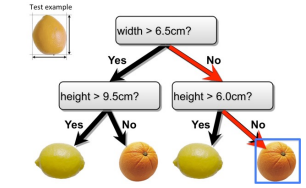
Fall 2023

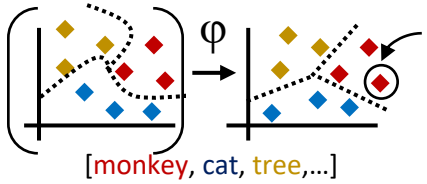
Instructor: Xiaodong Gu

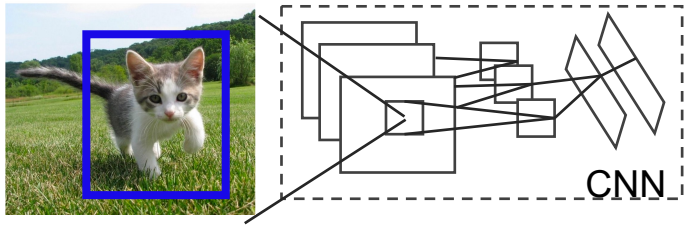
What we have learned so far?



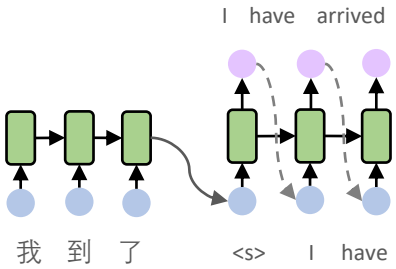
Regression







Classification



Generation

All previous lectures involve data labels (supervised learning)

What if we have no labels?

VISITORS	CONTACTS	LEADS	MQLS	BI_USER	CUSTOMERS	EXPECTED_ARR
14K	2.0K	1.650K	76.0	78.0	2	2.6M
↑ 37.2%	↑ 19.1%	↑ 21.0%	↑ 35.7%	↑ 69.6%	↑ N/A	↑ 2.3%

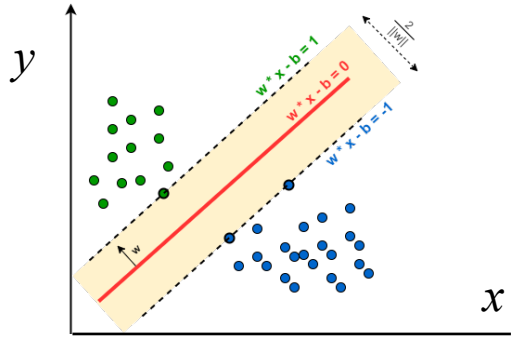
Medium | Source | Group Channel

GROUP CHANNEL	SOURCE	VISITORS	% Δ	LEADS	% Δ	LEADS TO MQLS CR	% Δ	MQLs	% Δ
Other	webinar	5.996	31.1% ↑	152	76.7% ↑	8%	69.7% ↑	12	200.0% ↑
Organic Traffic	yandex	3.424	18.8% ↑	102	18.6% ↑	2%	-15.7% ↓	2	0.0%
Referral Traffic	leadmagnet	918	9.08% ↑	1.172	18.1% ↑	4%	0.0% ↑	52	18.2% ↑
Brand Traffic	leadmagnet	722	65.6% ↑	2	58.3% ↑	5%	0.0% ↑	2	0.0%
Offline	Event	346	-3.4% ↓	2	-66.7% ↓	0%	0.0% ↑	0	0.0%
Referral Traffic	evernote.com	202	260.7% ↑	6	-88.9% ↓	0%	0.0% ↑	0	0.0%

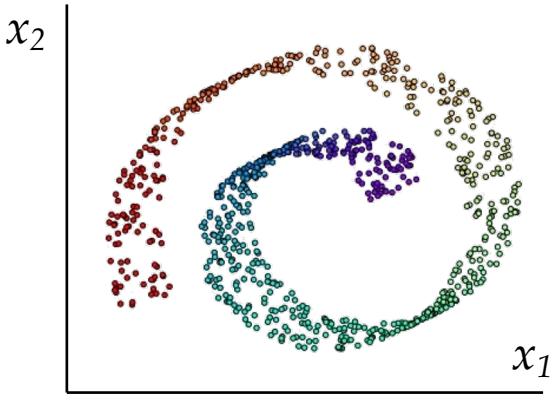
Can we still learn something from the data?

Supervised vs Unsupervised Learning

- Machine Learning
  - Supervised Learning
  - **Unsupervised Learning**
    - Clustering
    - Dimensional Reduction
    - Density Estimation
    - …
  - Reinforcement Learning



Supervised: (have labels)



Unsupervised: (No labels)

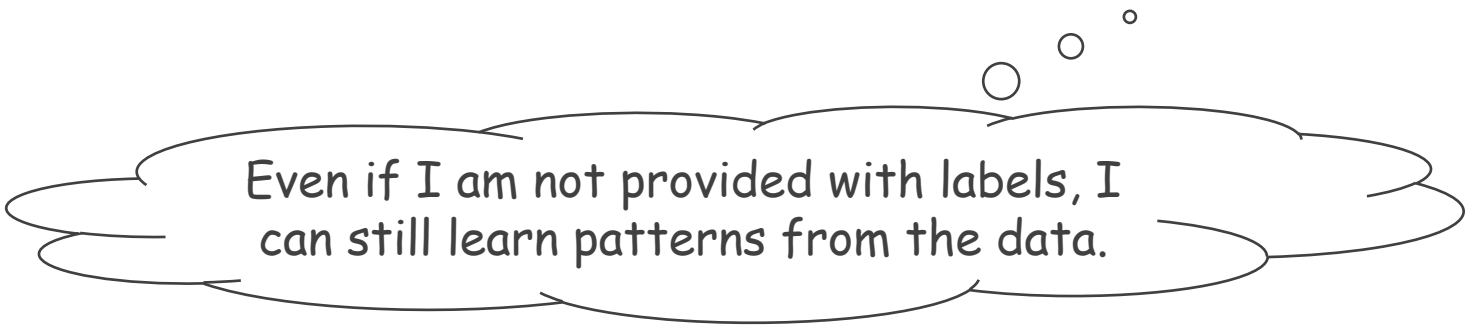
Supervised vs Unsupervised Learning

Supervised Learning

discover patterns in the data that relate data attributes **with a target (class)** attribute.

Unsupervised Learning

The data have **no target attribute**. We want to explore the data to find some **intrinsic structures**.



Unsupervised Learning

Clustering

Learning similarity

- K-means (✓)
- Gaussion Mixture

Generative

Density estimation

- GAN
- VAE
- Diffusion

Dimensional Reduction

Data compression

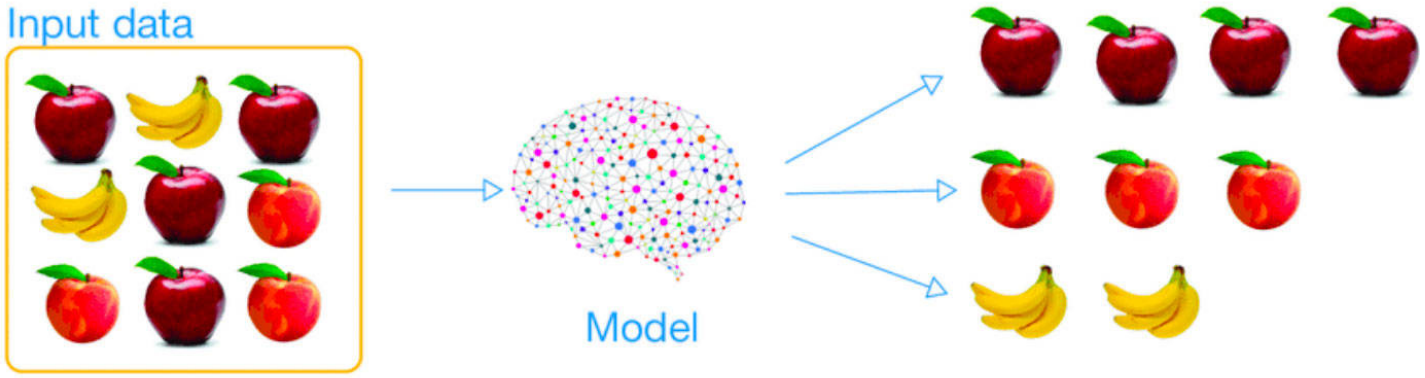
- Principal component analysis
- Auto-encoder

Today

- Clustering
- K-means

What is Clustering?

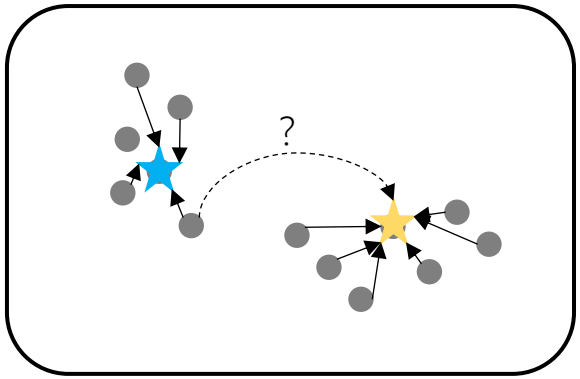
- **Clustering** = the organization of unlabeled data into similarity groups (called clusters).
- A cluster (簇) is a collection of data items which are "**similar**" between them, and "**dissimilar**" to data items in other clusters.



Q: How can we characterize "Similar" and "Dissimilar"?

How to Represent Clusters ?

- Suppose we have figured out the clusters for a dataset (e.g., by eyes), how can we represent them?



– For each cluster, elect a "**center**" point which best represents the whole cluster.

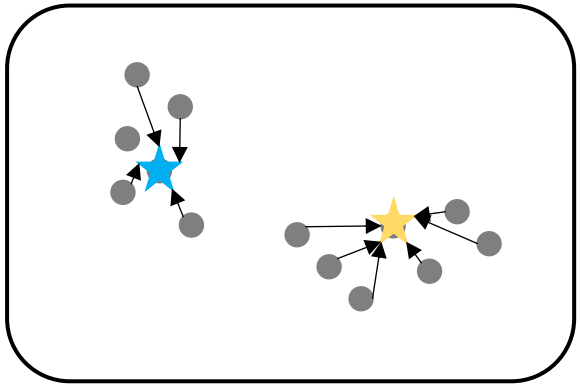
– Each data instance is assigned with a "label" indicating its **membership** to each cluster.

"Similar" or "Dissimilar"?

- The centroid must be more **similar** to instances in the represented cluster than other instances.
- Instances within a cluster are more **similar** to the centroid of the same cluster than centroids of other clusters.

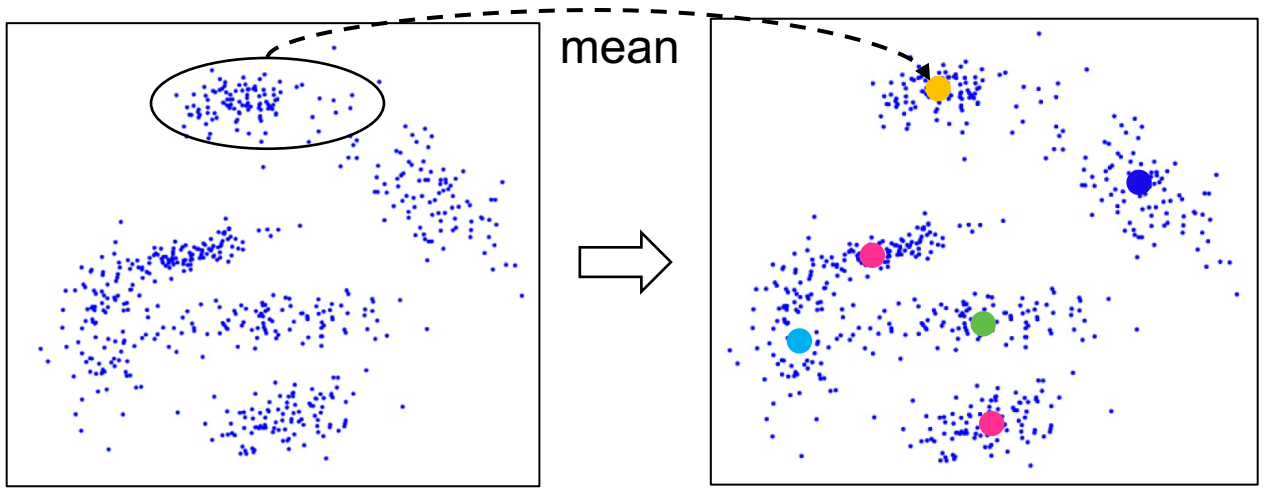
A Machine Learning Perspective

- What can be optimized in this "center-membership" framework?
  - The "**centers**" should be elected according to the pairwise distances of the instances in the same cluster. [**center update**]
  - Having obtained the centroids, we need to optimize the "**membership**" (label) of each data instance. [**label update**]
  - The above two step can be repeated alternatively.



K-Means [MacQueen, 1967]

- The *k*-means algorithm partitions the given data into ***k*** clusters, where the **center** for each cluster is defined as the **mean** of all instances within the cluster, called **centroid**.
- Let $D = \{x_1,\ldots,x_N\}$ be a data set, where $x_i \in \mathbb{R}^d$ is a d dimensional vector. We want to partition D into C clusters. We assign each x_i with an indicating variable $m_i \in \{1,\ldots,C\}$.



Steps

1. Choose k (random) data points (seeds) to be the initial centroids, cluster centers.
2. Assign each data point to the closest centroid.

$$L_2(x, \mu^k) = \|x - \mu^k\| = \sqrt{\sum_{m=1}^{d} (x_i - \mu_m^k)^2}$$

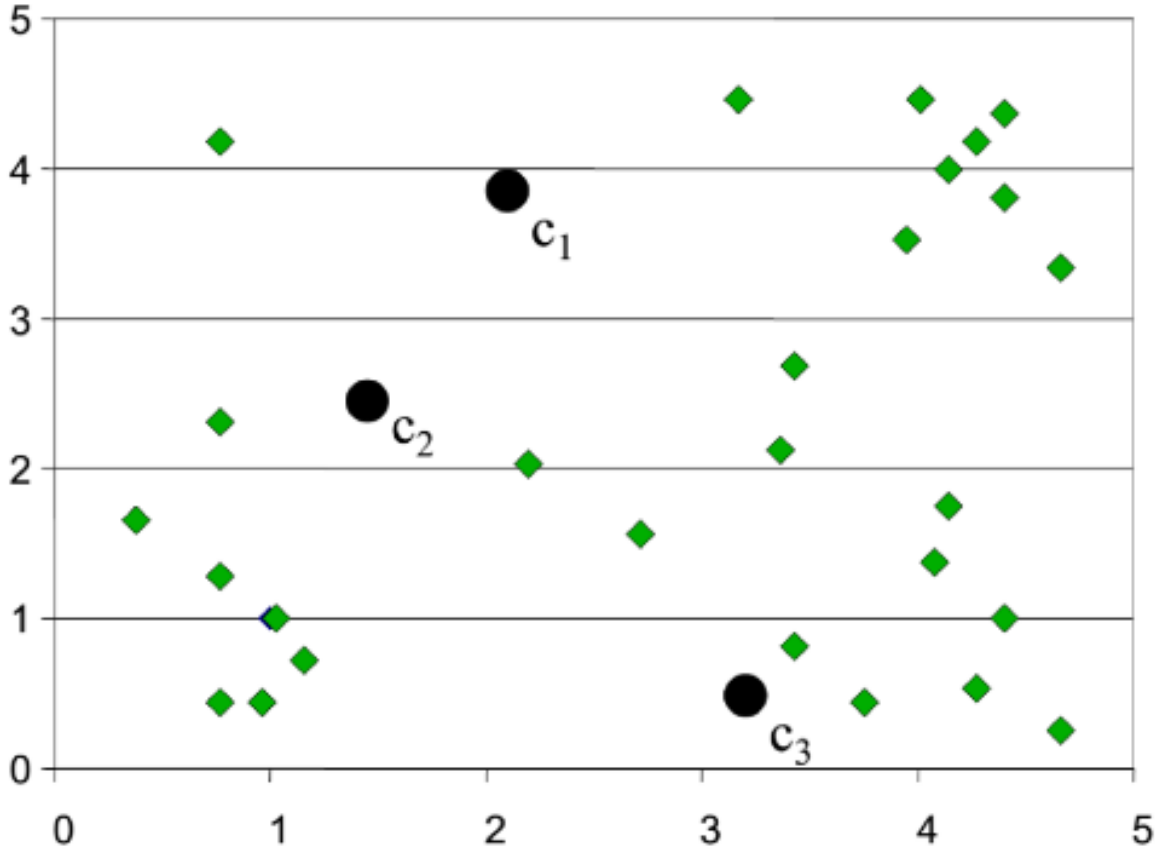
3. Re-compute the centroids using the current cluster memberships.

$$\mu^k = \frac{1}{C_k} \sum_{x \in C_k} x$$

4. Repeat steps 2 and 3 until a convergence criterion is met.

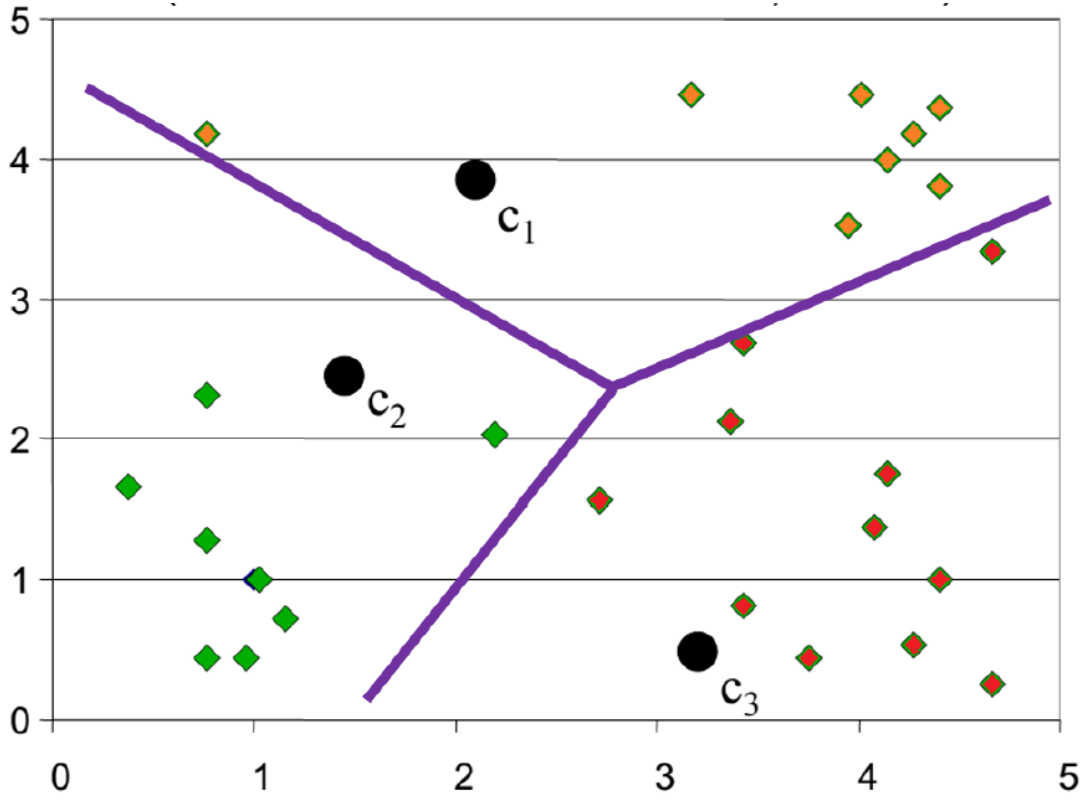
Example – Step1

Randomly initialize the cluster centers.



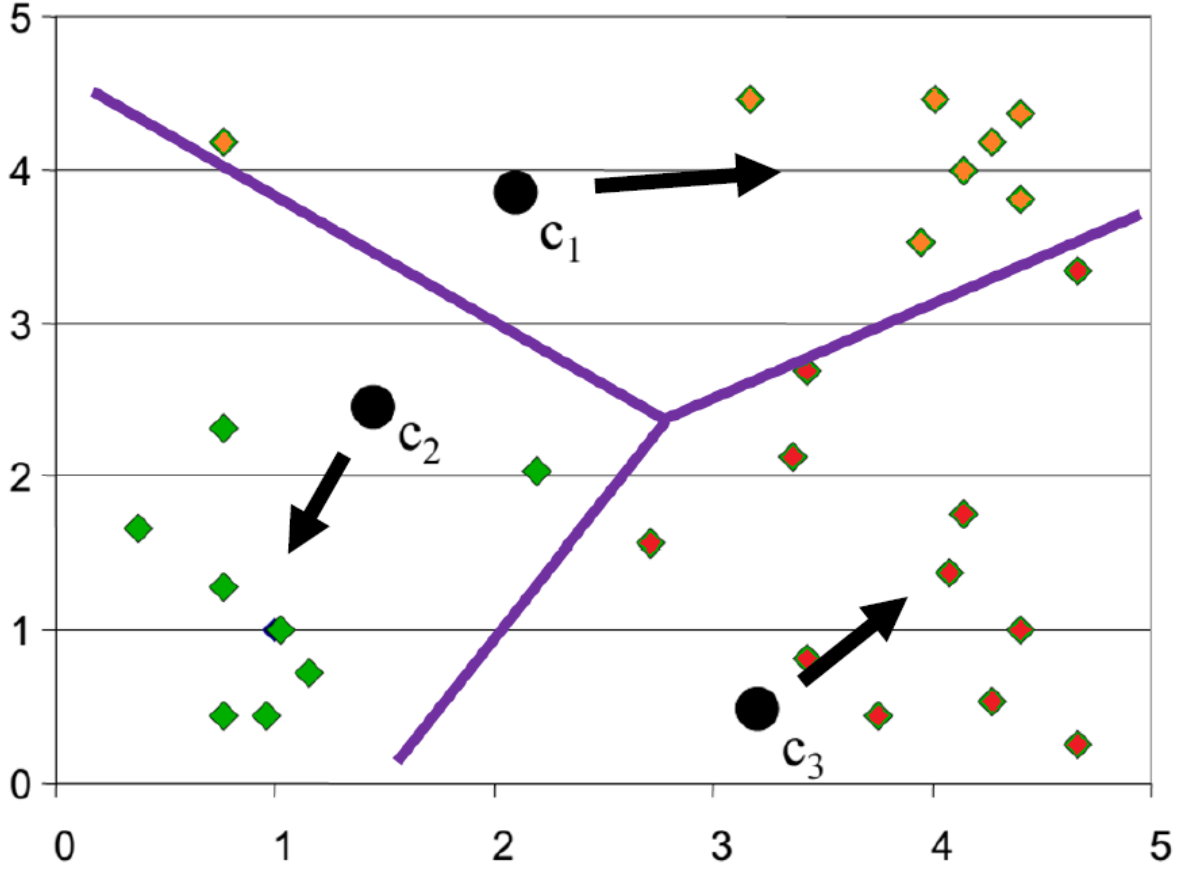
Example – Step2

Determine cluster membership for each input
("winner-takes-all" inhibitory circuit)



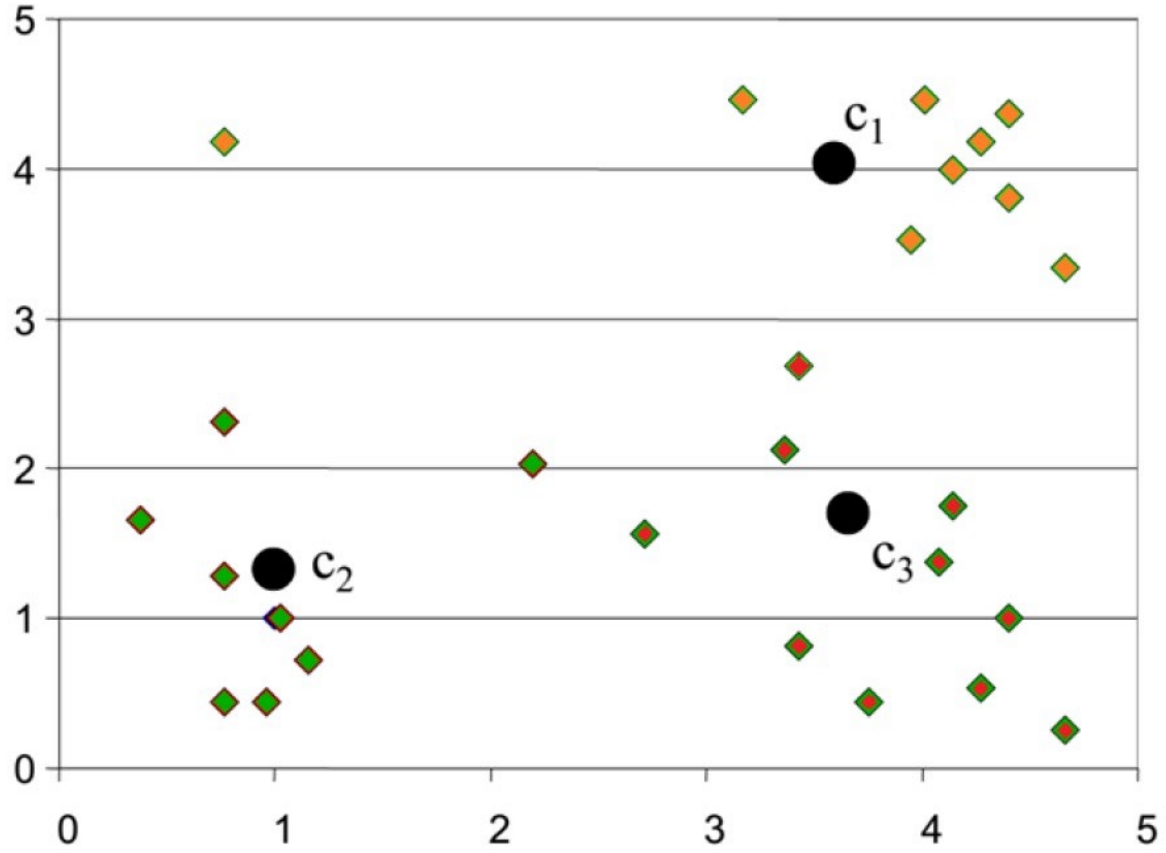
Example – Step3

Re-estimate cluster centers (adapt synaptic weights)



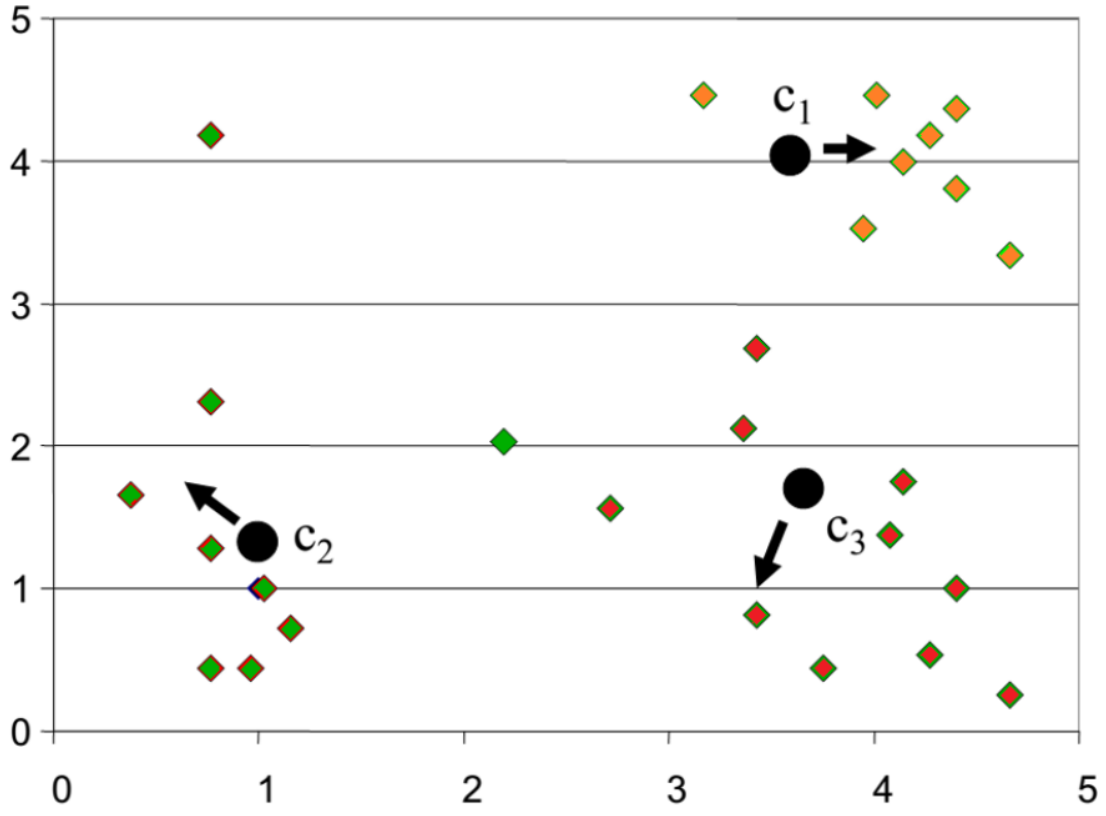
Example

Result of the first iteration



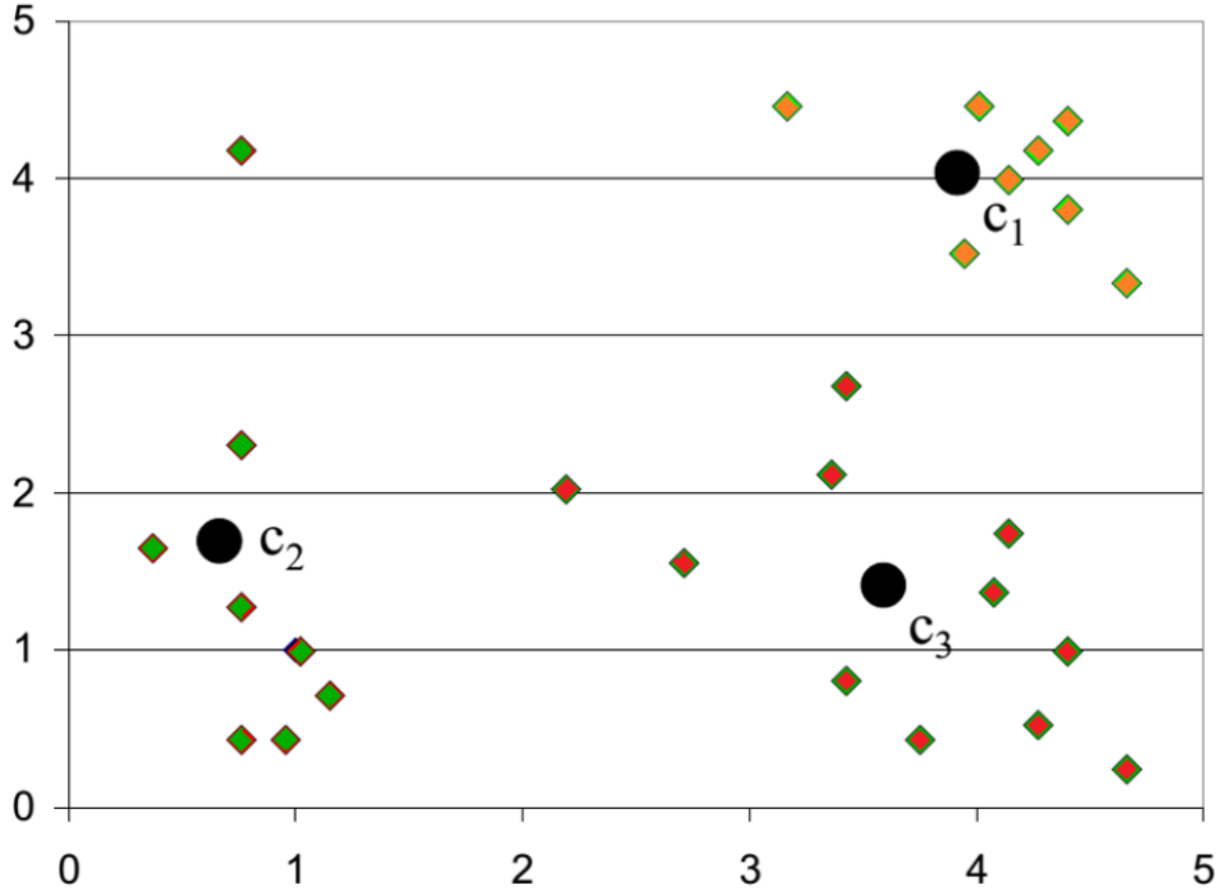
Example

The second iteration



Example

Result of the second iteration

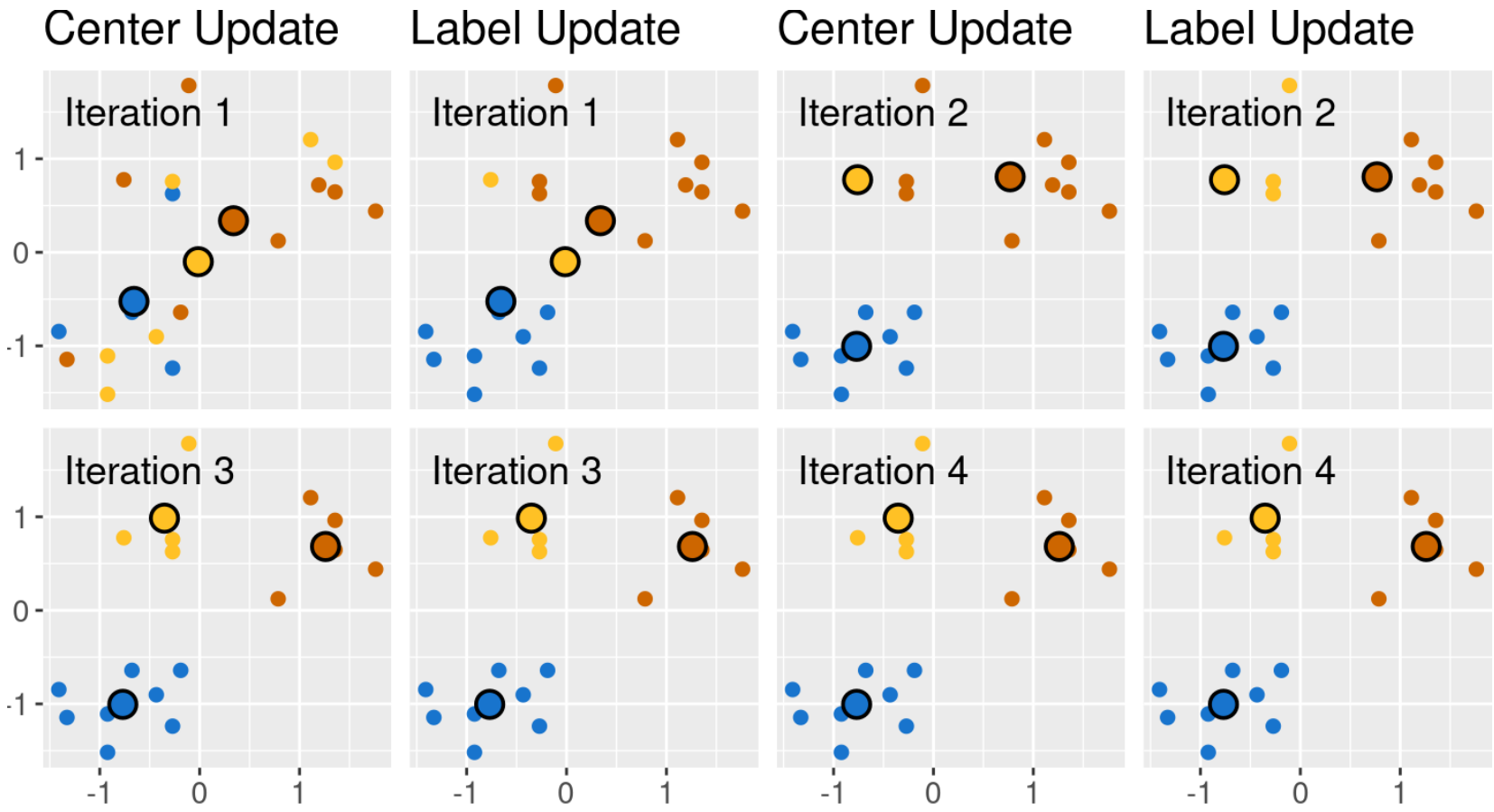


Convergence Criterion

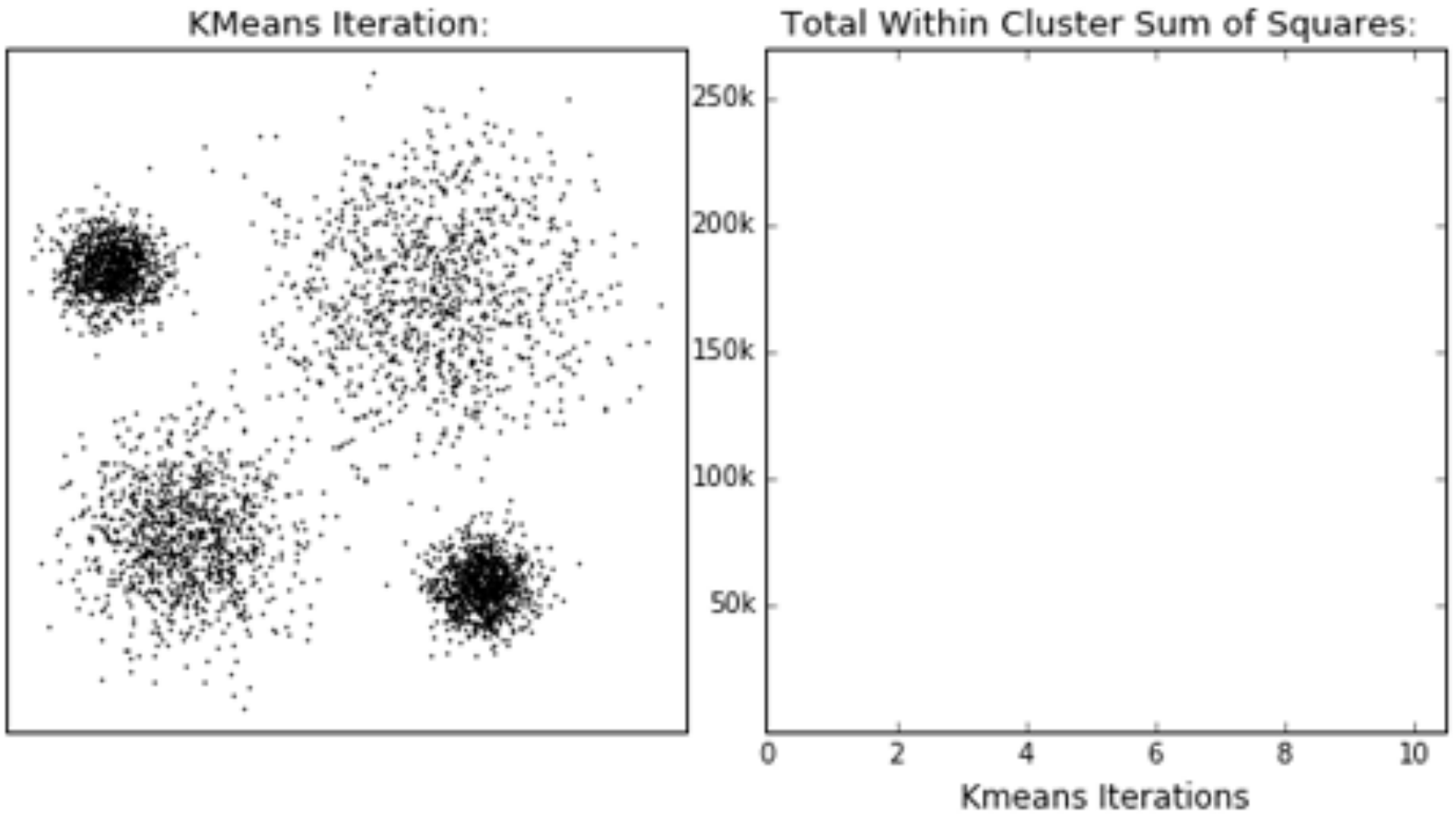
- No (or minimum) re-assignments of data points to different clusters, *or*
- No (or minimum) change of centroids, or
- Minimum decrease in the **sum of squared error** (SSE):

$$\min_{\{\mu^k\}_{k=1}^K} \sum_{k=1}^{K} \sum_{x \in C_k} L(x - \mu^k) \qquad \mu^k = \frac{1}{C_k} \sum_{x \in C_k} x$$

Example – A Big Picture



Example: Live K-Means



Game Time

Which food should I taste first for each cuisine?

Time Complexity

- Assume computing distance between two instances is $O(d)$ where d is the dimensionality of the vectors.
- **Reassigning clusters**: $O(knd)$ distance computations.
- **Computing centroids**: Each instance vector gets added once to some centroid: $O(nd)$.
- Assume these two steps are each done once for I iterations: $O(Iknd)$

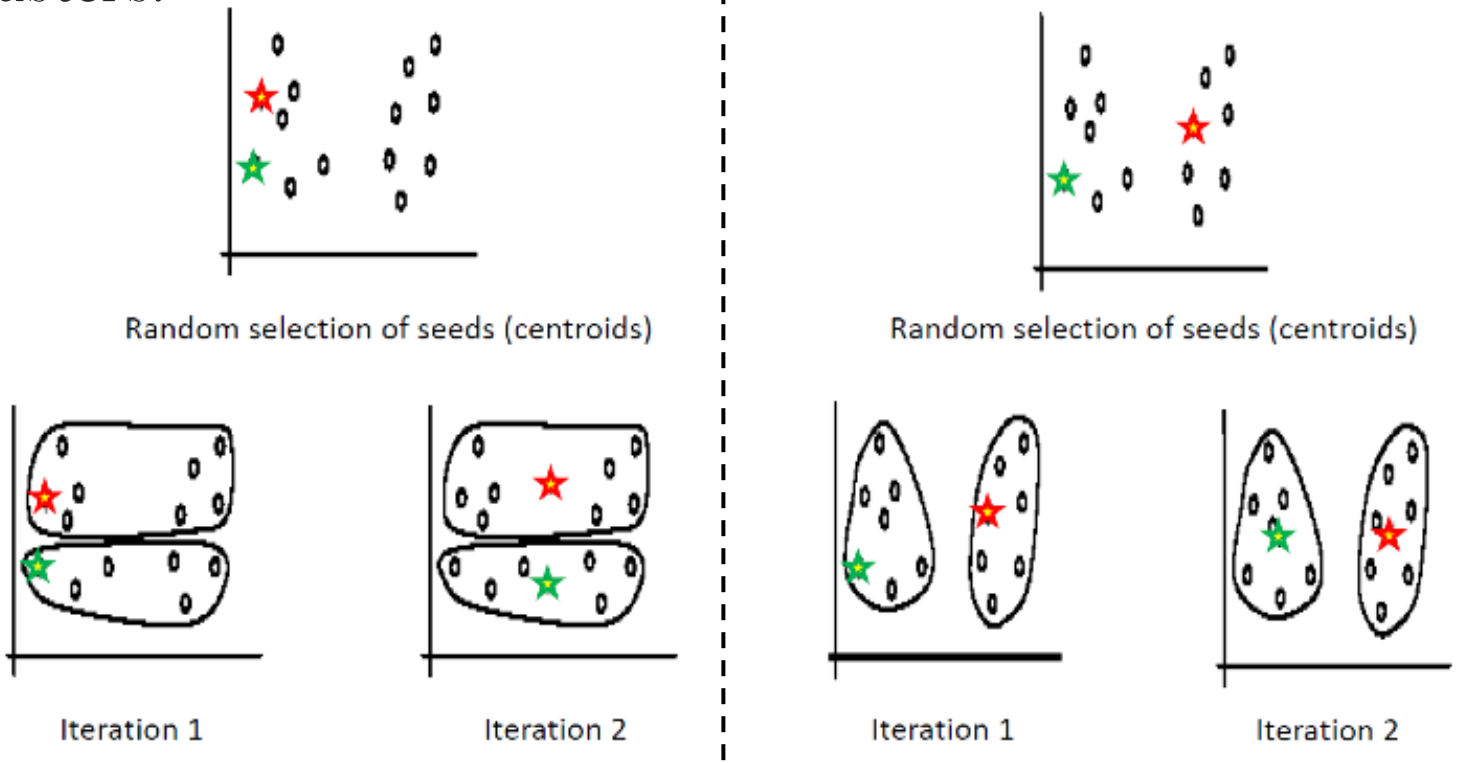
Local Optimum

- Finding the global optimum is NP-hard.
- The k-means algorithm is guaranteed to converge to a local optimum.

Q: What causes the uncertainty?

Sensitivity to Initial Centroids (Seeds)

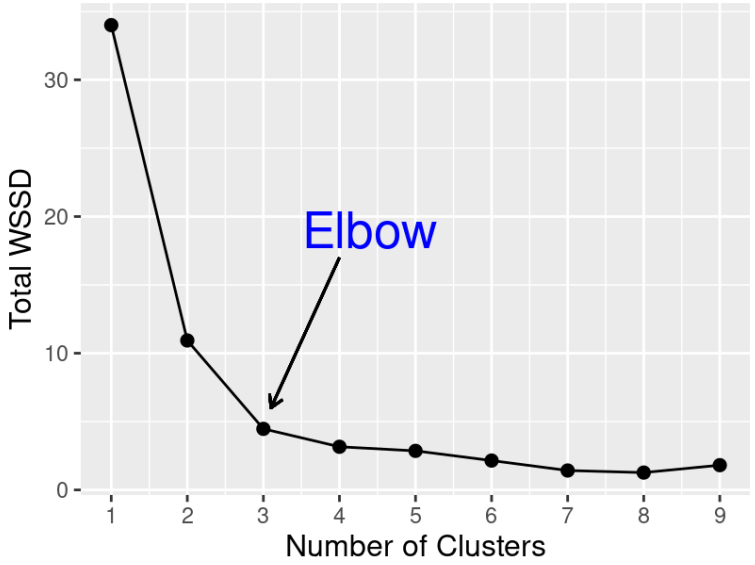
- Results can vary based on random seed selection.
- Some seeds can result in slow convergence or sub-optimal clusters.



How to choose the value of K?

- The Elbow method is the best way to find K.
- For each K, the *within-sum-of-squares* (WSS) is defined as the sum of the squared distance between each member of the cluster and its centroid.
- K with the least amount of WSS is taken as the optimum.

$$WSS = \sum_{i=1}^{m} (x_i - c_i)^2$$



How to choose the initial centroids?

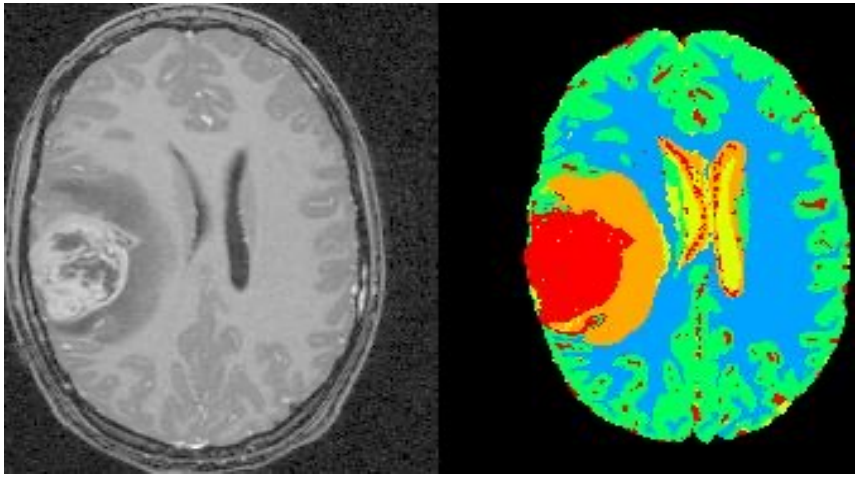
- Select good seeds using a heuristic or the results of another method.

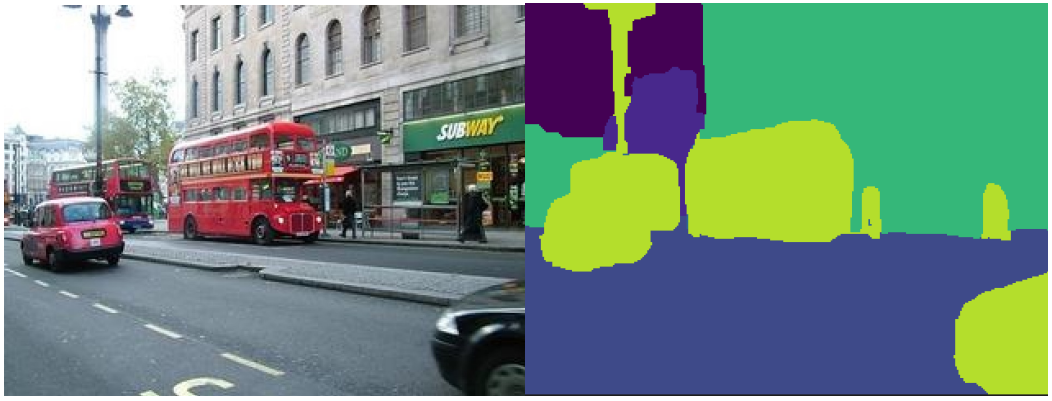
Example: k-means++

1. Choose one center uniformly at random from among the data points.
2. For each data point x, compute D(x), the distance between x and the nearest center that has already been chosen.
3. Choose one new data point at random as a new center, using a weighted probability distribution where a point x is chosen with probability proportional to D(x)^2 .
4. Repeat Steps 2 and 3 until k centers have been chosen.
5. Proceed using standard k-means clustering.

Applications of K-Means

Image Segmentation:





Applications of K-Means

Image Compression: replace each pixel value with its nearby centroid

Pros and Cons

Advantages:

- **Simple**: easy to understand and implement,
- **Efficient**: k-means is considered a linear algorithm.

Pros and Cons

Disadvantages:

- The algorithm is only applicable if the **mean** is defined.
- The user needs to specify k.
- The algorithm is sensitive to **outliers**.

What if we only know the **pairwise distances** of data?

TIME for Some Coding

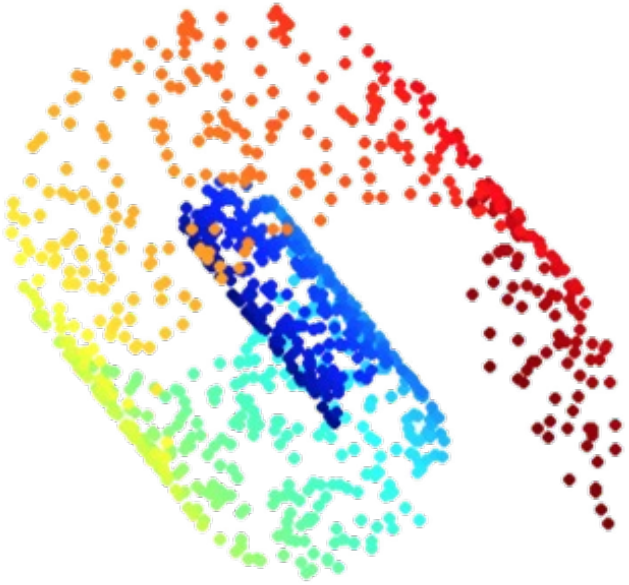
https://www.kaggle.com/code/adinishad/kmeans-clustering-from-scratch

https://www.kaggle.com/code/satishgunjal/tutorial-k-means-clustering

What's Next?

Dimensionality Reduction

Looks 3D



Actually 2D