

# K-Means Algorithm for Clustering

Liang Liang

# Categories of Machine Learning

- Supervised Learning

to model the relationship between measured features of data and some label associated with the data

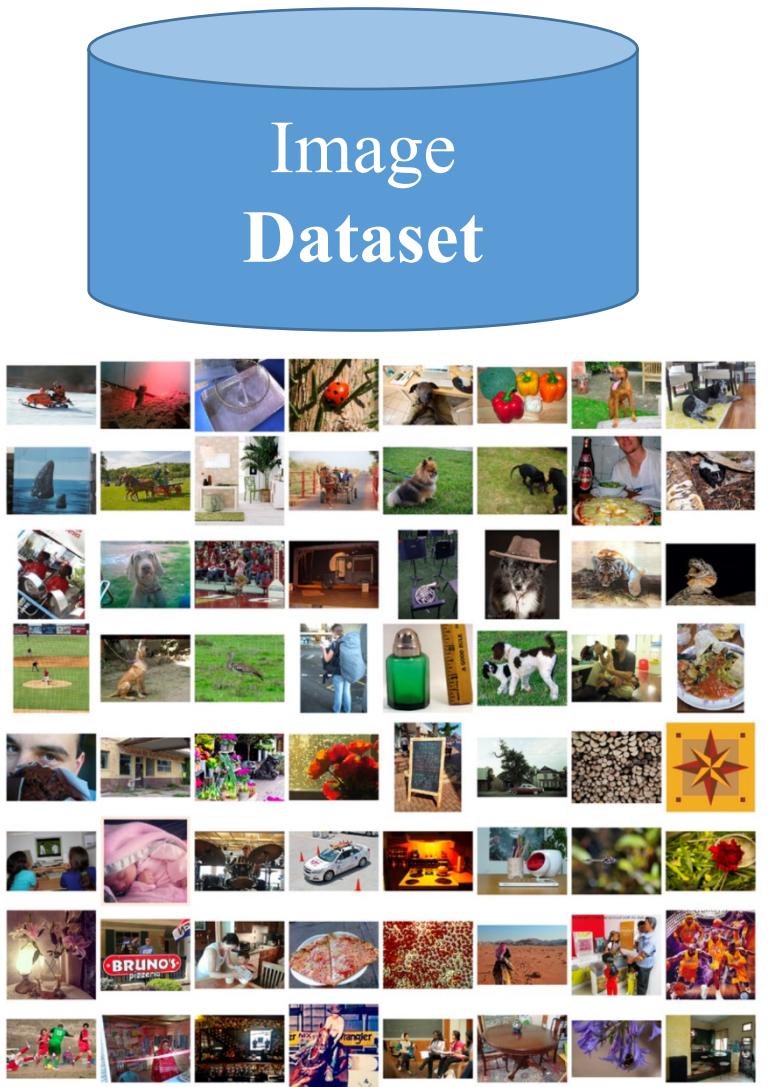
- Unsupervised Learning

to model the features of a dataset without reference to any label

- Reinforcement Learning

the goal is to develop a model (agent) that improves its performance based on interactions with the environment

# Cluster images



clustering



**Goal of clustering:**

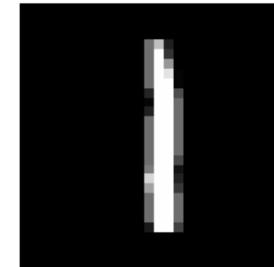
Divide objects into groups,  
and objects within a group  
are more similar than  
those outside the group

unsupervised learning

# Clustering handwritten digit images

72104149590690159784966540740131  
34727121174235124463556041957893  
74643070291732977627847361369314  
17696054992194873974449254767905  
85665781016467317182029955156034  
46546545144723271818185089250111  
09031642361113952945939036557227  
12841733887922415987230442419577  
28268577918180301994182129759264  
15829204002847124027433003196525  
12930410711215339786361381051315  
56185179462250656372088541140337  
61621928619525442838245031773797  
19219292049148184598837600302664  
95332391268056663882758961841259  
19754089910523789406395213136571  
22632654897130383193446421825488  
40023277087447969098046063548339  
33378082170654380963809968685786  
02402231975108462479309822927359  
18020511376712580371409186774349  
19317397691378336128585114931077  
07944855408215845040613326726931  
46254206217341054311749948402451  
16471942415538314568941538032512  
83440883317359632613607217142821  
79611248177480231310770355276692  
83522560829288887493066321322930  
05781446029147473988471212232383  
91740355865267663279117564951334  
78911691445406223151203812671623  
90152089

image



Clustering the images into ten groups/clusters

A cluster may correspond to a digit.

# Cluster customers - customer segmentation

- Assume you work in the credit card department of a bank  
your job title is data scientist
- to understand the behaviors of the customers (credit card holders) and improve marketing strategies, you may need to categorize the customers based on their characteristics (income, age, buying behavior, etc).
- Find the clusters/groups that contain valuable customers:  
e.g., high income but low annual spend.

# Cluster houses

<https://arxiv.org/pdf/1803.00919.pdf>

- Assume you work for a real-estate company as a data scientist
- Predict the sale prices of houses

**step-1:** make clusters of houses (sub-markets)

cluster houses built in different locations based on characteristics such as income of the house owner, house price, size, closeness to bay, etc

**step-2:** use a regression model to predict the sale price of a house in a sub-market/cluster, given the attributes/features of the house (e.g., size, number of bedrooms).

Previous  
2.2.  
Manifold...  
Next  
2.4.  
Biclustering  
Up  
2.  
Unsupervis...

scikit-learn v0.20.2  
Other versions

Please cite us if you use  
the software.

## 2.3. Clustering

2.3.1. Overview of clustering methods

2.3.2. K-means

- 2.3.2.1. Mini Batch K-Means

2.3.3. Affinity Propagation

2.3.4. Mean Shift

2.3.5. Spectral clustering

- 2.3.5.1. Different label assignment strategies

- 2.3.5.2. Spectral Clustering Graphs

2.3.6. Hierarchical clustering

- 2.3.6.1. Different linkage type:  
Ward, complete, average, and  
single linkage

- 2.3.6.2. Adding connectivity  
constraints

- 2.3.6.3. Varying the metric

2.3.7. DBSCAN

2.3.8. Birch

## 2.3. Clustering

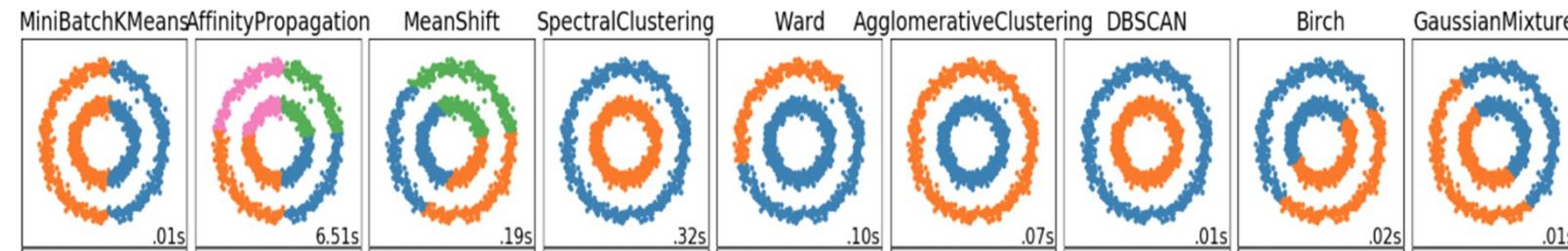
Clustering of unlabeled data can be performed with the module `sklearn.cluster`.

Each clustering algorithm comes in two variants: a class, that implements the `fit` method to learn the clusters on train data, and a function, that, given train data, returns an array of integer labels corresponding to the different clusters. For the class, the labels over the training data can be found in the `labels_` attribute.

### Input data

One important thing to note is that the algorithms implemented in this module can take different kinds of matrix as input. All the methods accept standard data matrices of shape `[n_samples, n_features]`. These can be obtained from the classes in the `sklearn.feature_extraction` module. For `AffinityPropagation`, `SpectralClustering` and `DBSCAN` one can also input similarity matrices of shape `[n_samples, n_samples]`. These can be obtained from the functions in the `sklearn.metrics.pairwise` module.

### 2.3.1. Overview of clustering methods



Next

[Prev](#) [Up](#) [Next](#)**scikit-learn 0.22.1**[Other versions](#)

Please [cite us](#) if you use the software.

## sklearn.cluster.KMeans

Examples using

### sklearn.cluster.KMeans

# sklearn.cluster.KMeans

```
class sklearn.cluster.KMeans(n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001,  
    precompute_distances='auto', verbose=0, random_state=None, copy_x=True, n_jobs=None,  
    algorithm='auto')
```

[\[source\]](#)

K-Means clustering.

Read more in the [User Guide](#).

---

**Parameters:** **n\_clusters : int, default=8**

The number of clusters to form as well as the number of centroids to generate.

**init : {'k-means++', 'random'} or ndarray of shape (n\_clusters, n\_features), default='k-means++'**

Method for initialization, defaults to 'k-means++':

'k-means++' : selects initial cluster centers for k-mean clustering in a smart way to speed up convergence. See section Notes in k\_init for more details.

'random': choose k observations (rows) at random from data for the initial centroids

[Toggle Menu](#)

## sklearn.cluster.KMeans

```
class sklearn.cluster.KMeans(n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001,  
    precompute_distances='auto', verbose=0, random_state=None, copy_x=True, n_jobs=None,  
    algorithm='auto')
```

[\[source\]](#)

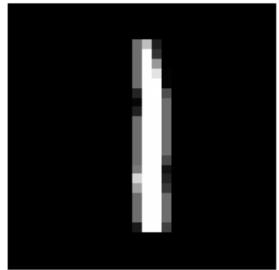
To use k-means, we need to understand the meaning of each parameter  
The default parameter values may NOT work for your application.  
You need to adjust the parameters

To understand the meanings of the parameters, we need to understand  
the algorithm of k-means

# K-means Algorithm for Clustering Objects

- Represent each object by a numerical vector
- Input to the k-means algorithm is a set of vectors  
we need to put those vectors into a 2D array (matrix/table)
- Output from the k-means algorithm is a set of clusters (groups)  
each cluster contains a subset of the vectors/objects  
the clusters are disjoint (do not share any vectors/objects)
- Clustering is based on the distance between two vectors  
we need a function to measure the distance(vectorA, vectorB)

# Represent an image by a vector



This image has  $28 \times 28$  pixels.

It is a matrix/ 2D array  $A \in \mathbb{R}^{28 \times 28}$

$$A = \begin{bmatrix} A_{0,0} & \dots & A_{0,27} \\ \dots & \dots & \dots \\ A_{27,0} & \dots & A_{27,27} \end{bmatrix} \quad \text{row-0} \quad \text{row-27}$$

$$\mathbf{x} = \begin{bmatrix} A_{0,0} \\ A_{0,1} \\ A_{0,2} \\ \vdots \\ \vdots \\ \vdots \\ \vdots \\ A_{27,27} \end{bmatrix} \quad \begin{array}{l} \text{the first row} \\ \text{the second row} \end{array}$$

$$\mathbf{x} \in \mathbb{R}^{784}$$

a vector ~an image ~ a data sample

# Represent a customer by a vector

Each **row** is a feature vector of a customer

	A	B	C
1	income	spend	customer_ID
2	233	150	1
3	250	187	2
4	204	172	3
5	236	178	4
6	354	163	5
7	192	148	6
8	294	153	7
9	263	173	8
10	199	162	9

$$x = \begin{bmatrix} ID \\ income \\ spend \end{bmatrix}$$

In many applications,  
*customer ID* is not useful,  
so we remove it

$$x = \begin{bmatrix} income \\ spend \end{bmatrix}$$

- $x_1$  : the 1<sup>st</sup> customer (1<sup>st</sup> row in the table)  
 $x_2$  : the 2<sup>nd</sup> customer (2<sup>nd</sup> row in the table)

# Vector Norm and Distance Measure

- In general, we can define a  $\ell_p$  norm ( $p \geq 1$ )

$$\|x\|_p = \left( \sum_{m=1}^M |x_{[m]}|^p \right)^{\frac{1}{p}}$$

It measures the length of the vector

- $\ell_2$  norm  $\|x\|_2 = \sqrt{\sum_{m=1}^M x_{[m]}^2} = \sqrt{x^T x}$  (Euclidean norm)

$$x = \begin{bmatrix} x_{[1]} \\ x_{[2]} \end{bmatrix} = \begin{bmatrix} 0.1 \\ 1.2 \end{bmatrix}, \text{ then } \|x\|_2 = \sqrt{0.1^2 + 0.2^2}$$

# Vector Norm and Distance Measure

- two vectors/points  $\mathbf{x}, \mathbf{y} \in \mathbb{R}^M$ , and the norm of  $\mathbf{x}$  is  $\ell_p$  norm  $\|\mathbf{x}\|_p$
- $\mathbf{x}$  is the feature vector of object-A
- $\mathbf{y}$  is the feature vector of object-B
- Then the distance between  $\mathbf{x}$  and  $\mathbf{y}$  is  $\|\mathbf{x} - \mathbf{y}\|_p$
- $\ell_2$  norm is used in k-means algorithm to measure distance

$$\mathbf{x} = \begin{bmatrix} 0.1 \\ 1.2 \end{bmatrix}, \quad \mathbf{y} = \begin{bmatrix} 0.2 \\ 2.1 \end{bmatrix}$$

the distance is  $\|\mathbf{x} - \mathbf{y}\|_2 = \sqrt{(0.1 - 0.2)^2 + (1.2 - 2.1)^2}$

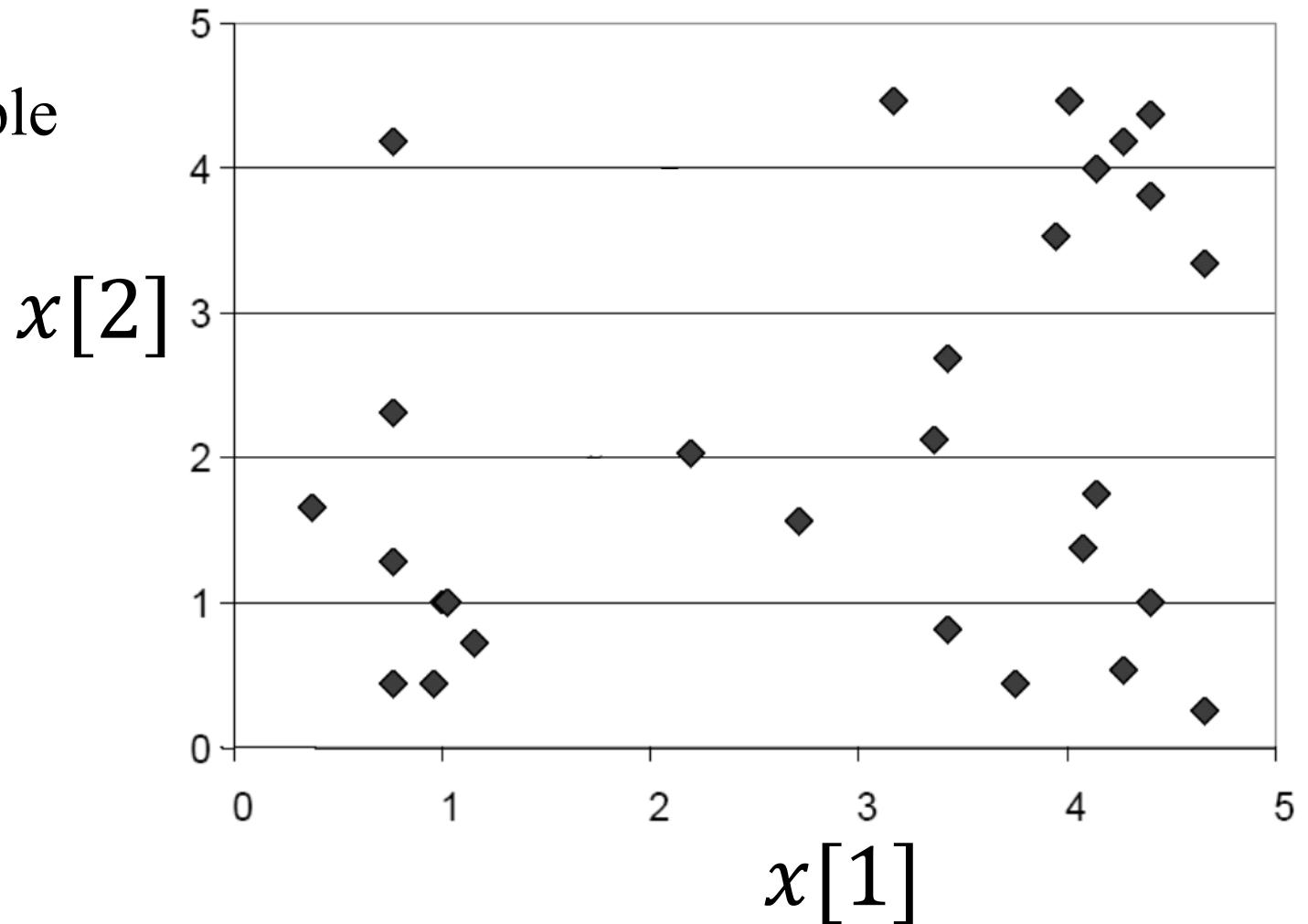
Run kmeans\_cust\_seg.ipynb

# Before clustering, a dataset of vectors/samples

a feature vector  $\mathbf{x}$  is a data sample

$$\mathbf{x} = \begin{bmatrix} x[1] \\ x[2] \end{bmatrix}$$

a data sample is also called  
a data point, i.e. a point in a  
*high* dimensional space



# Apply k-means algorithm: Initialization

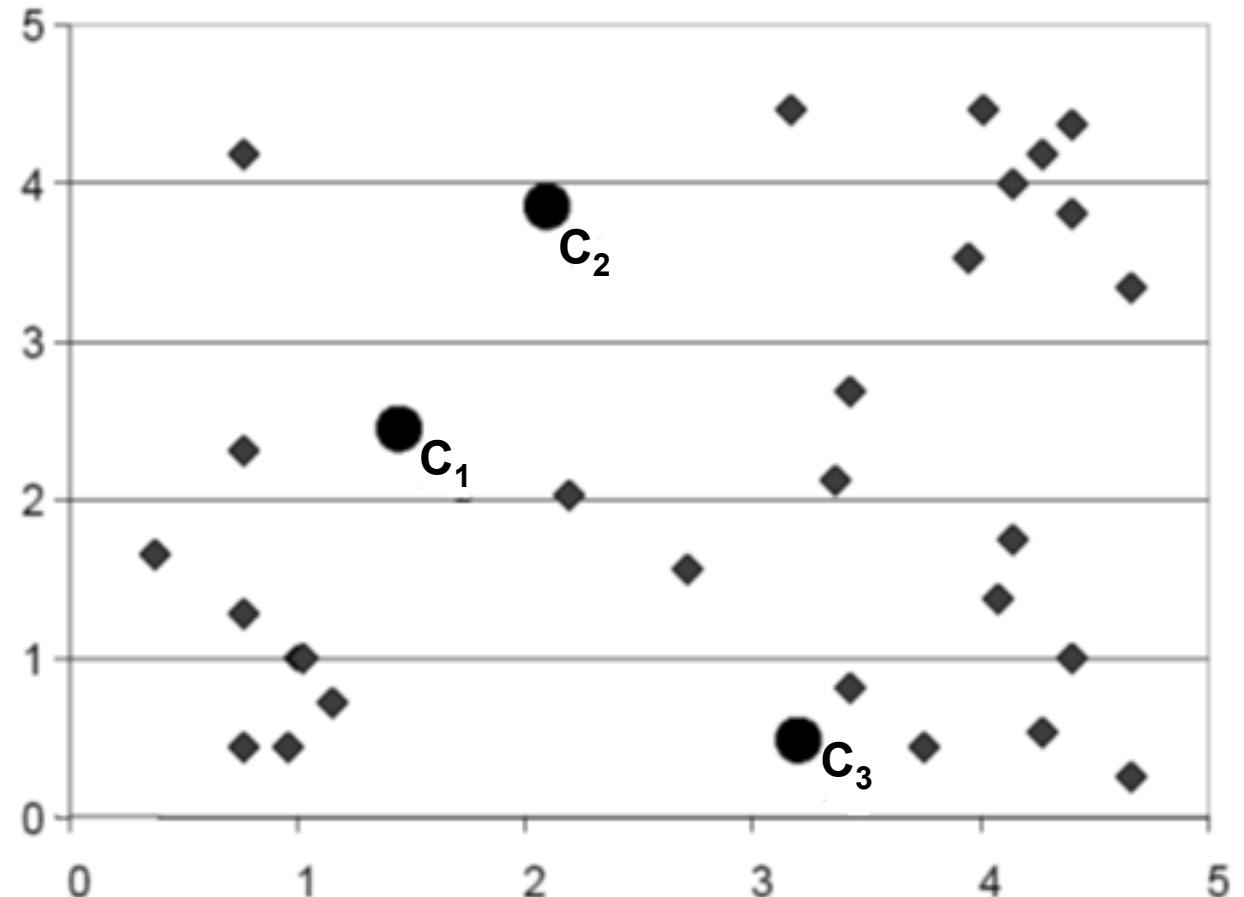
## Initialization:

(1) The user (you) sets the number of clusters  
e.g., 3

(2) The algorithm will randomly initialize the cluster centers/centroids.

A cluster center is a vector.

We get three random centers

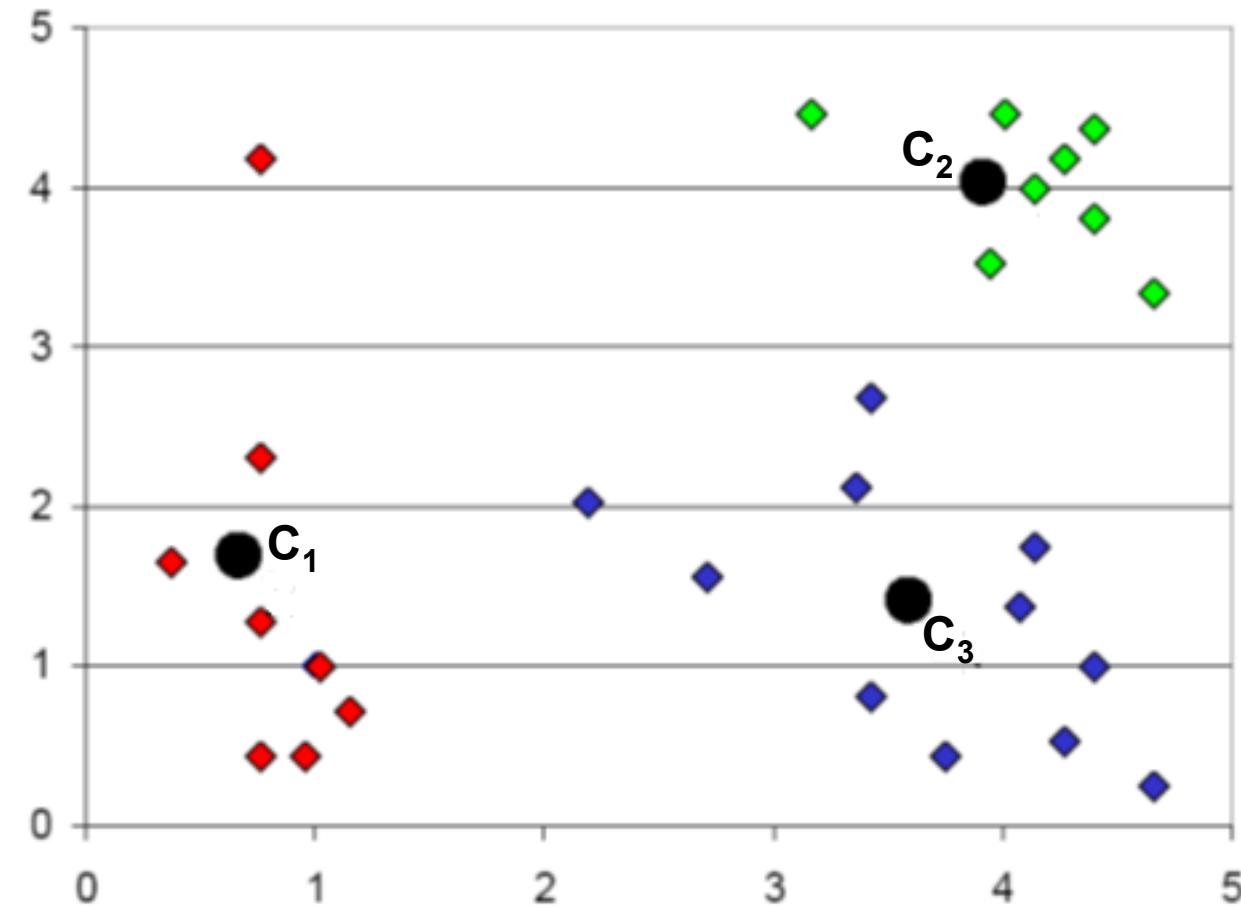


$c_1$ ,  $c_2$ ,  $c_3$  are initial cluster centers at three random locations

After k-means clustering, clusters/groups are formed

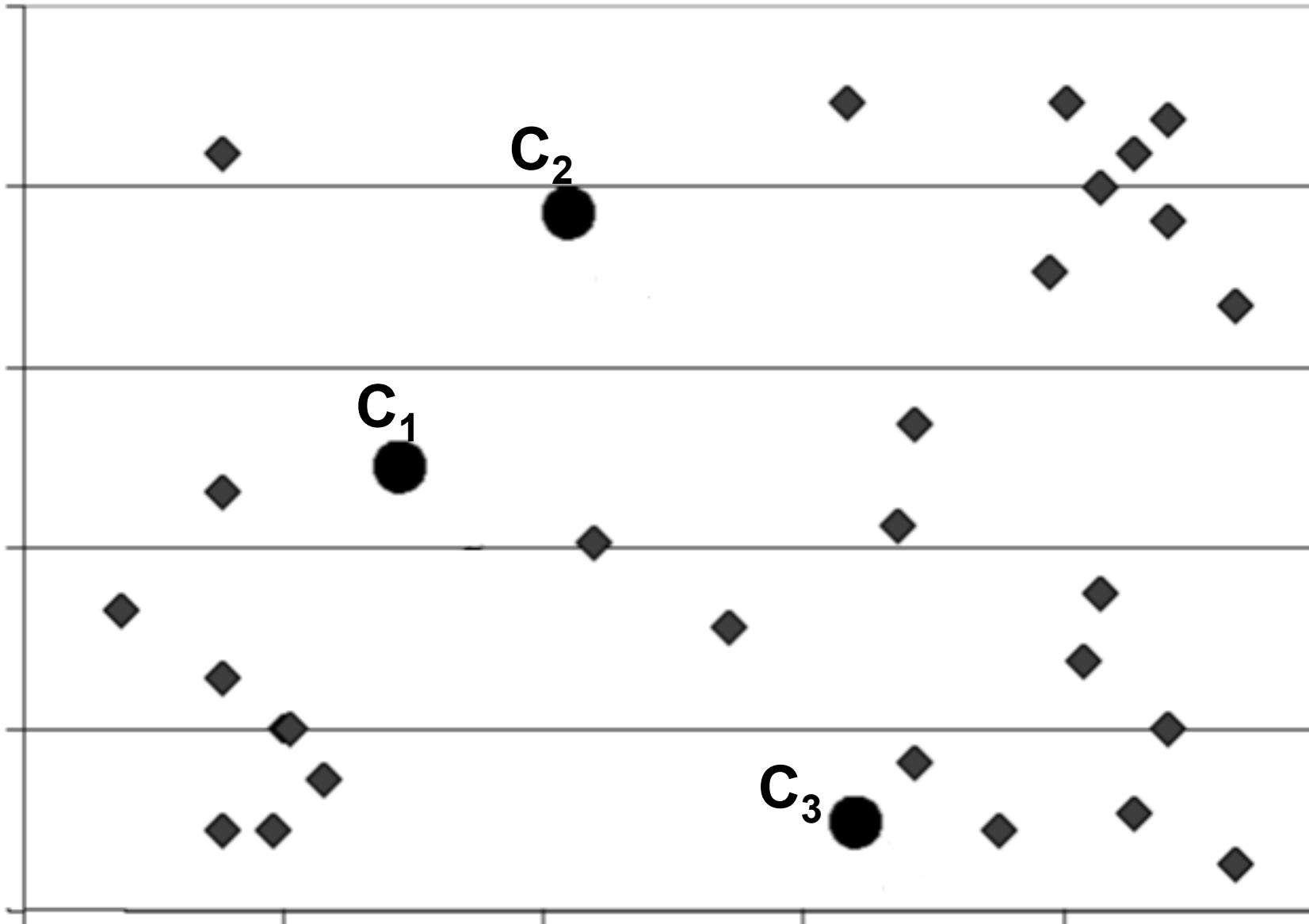
## After k-means clustering:

- The data points are assigned to the three clusters
    - red-cluster**
    - green-cluster**
    - blue-cluster**
  - Every data point has a cluster label that could be 1, 2, or 3
  - The final cluster centers are different from the initial centers

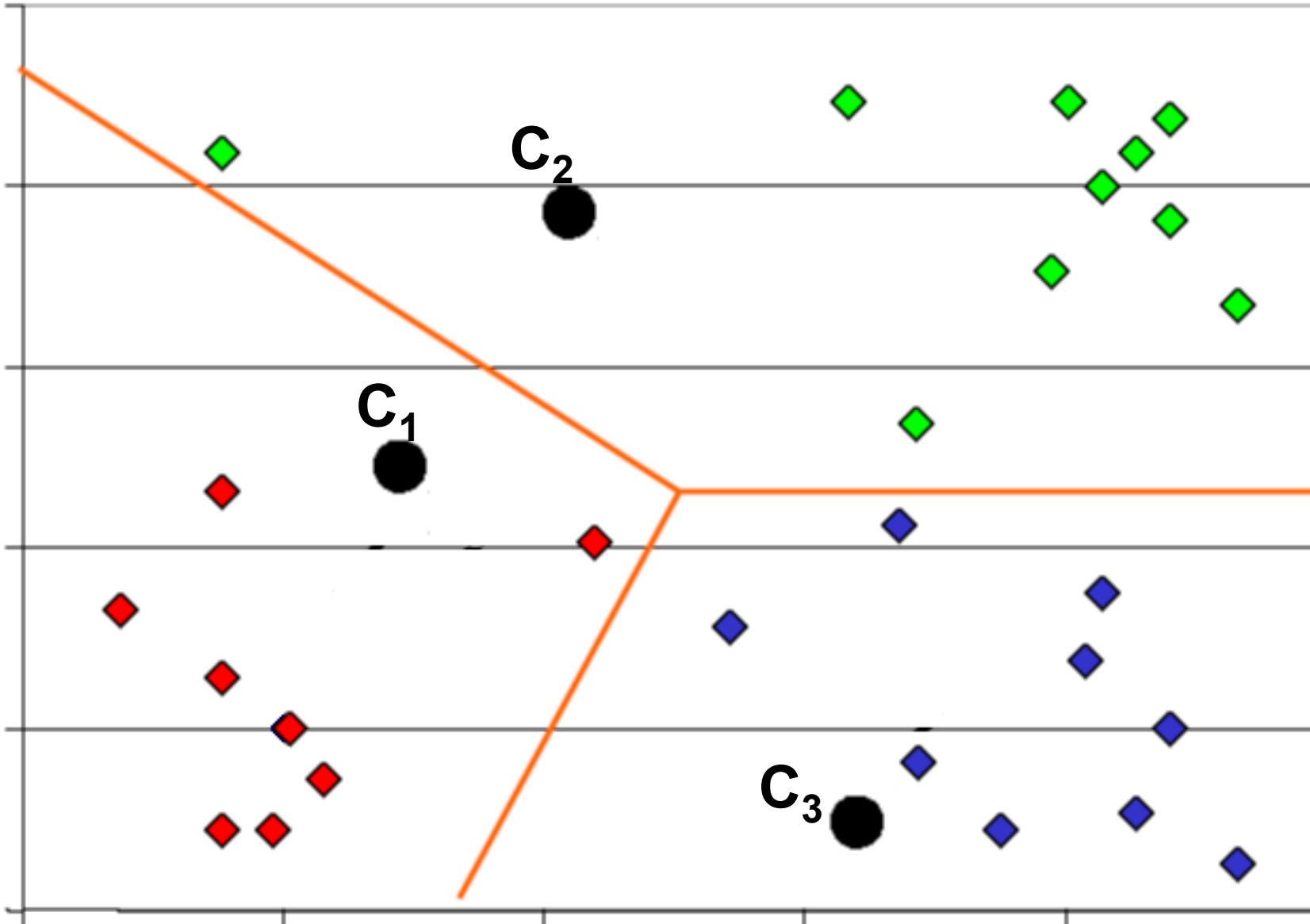


$c_1, c_2, c_3$  are the cluster centers

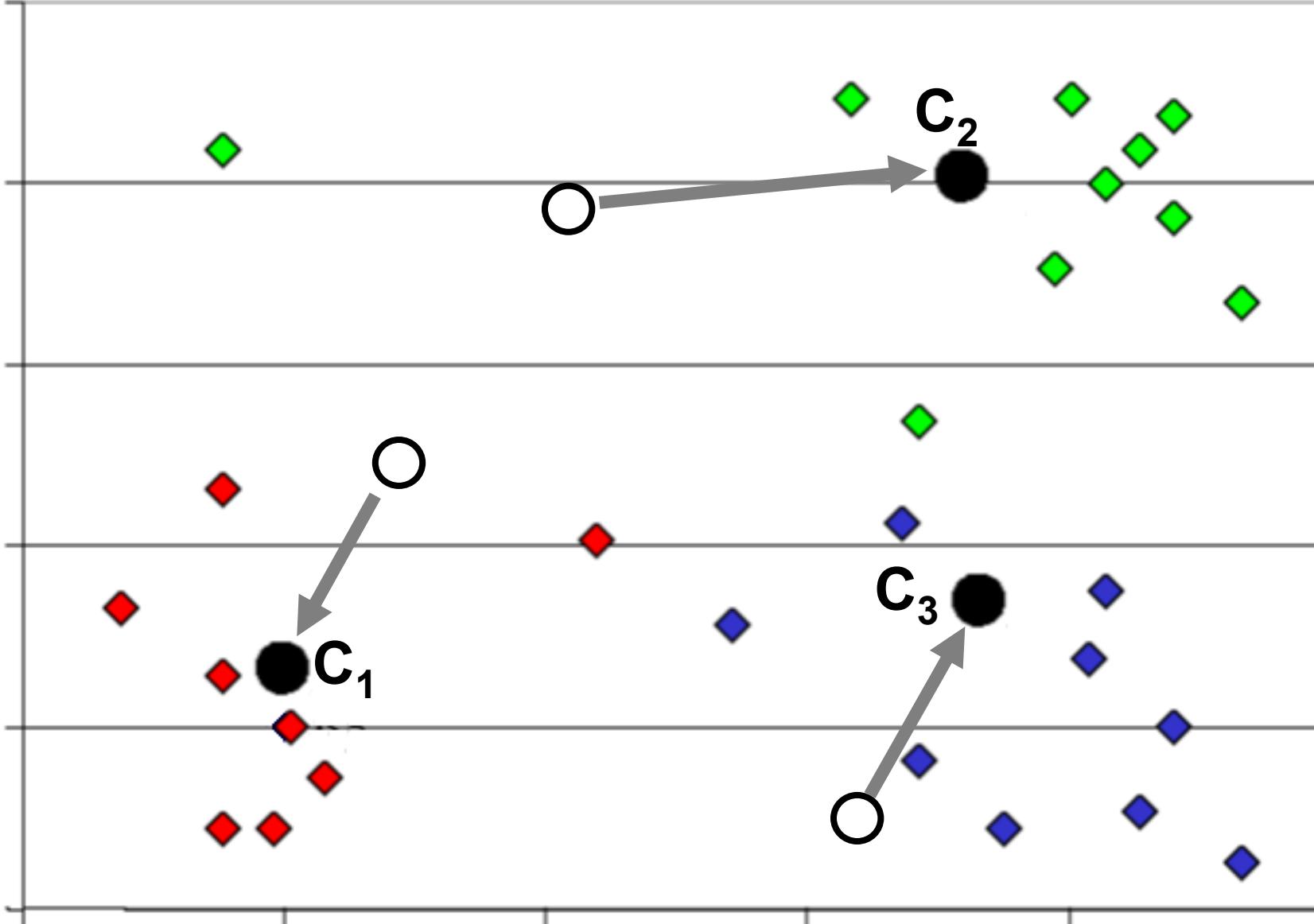
Initialization: the number of clusters and random locations of the cluster centers



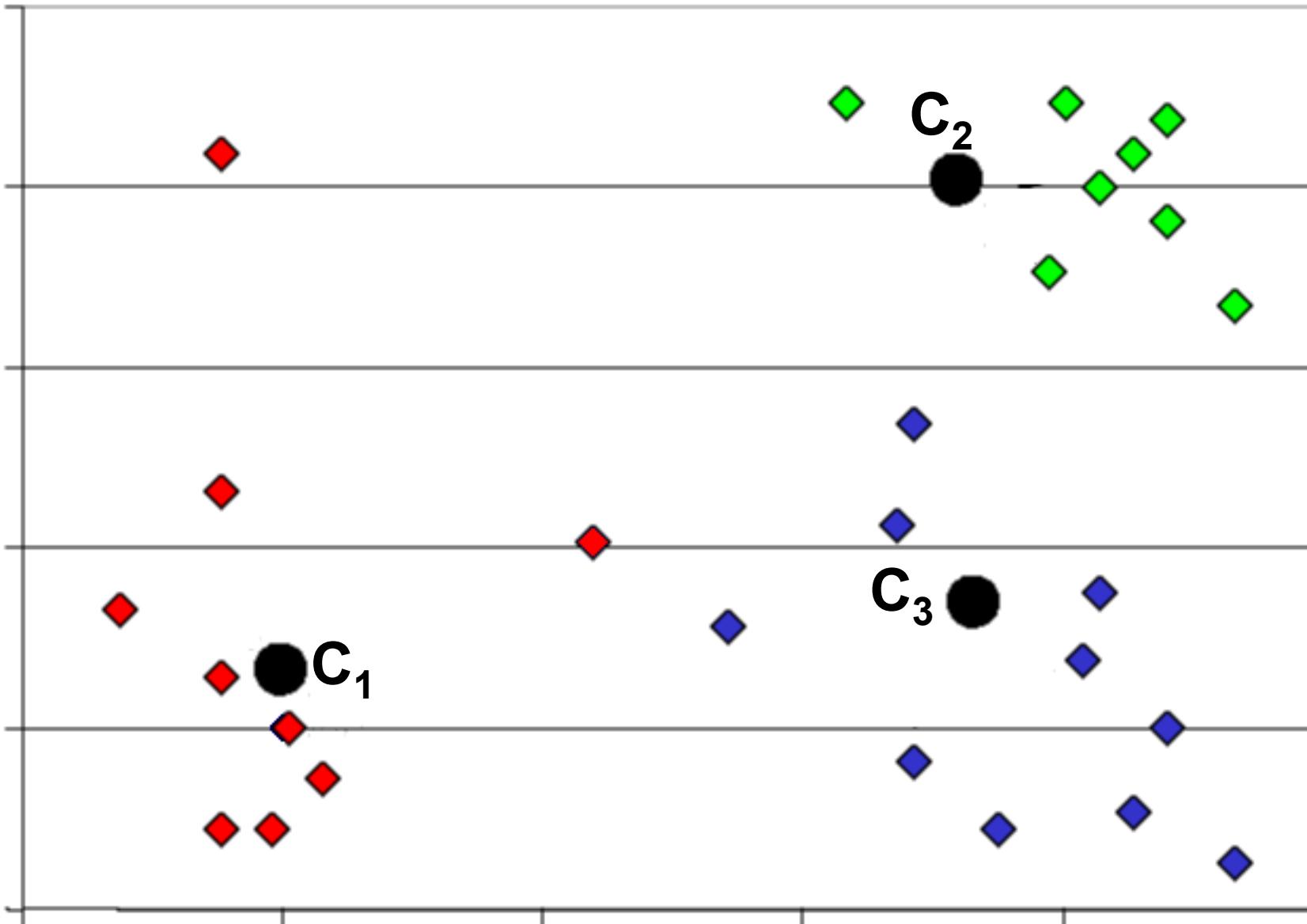
Assign Labels: assign each data point to the nearest cluster center



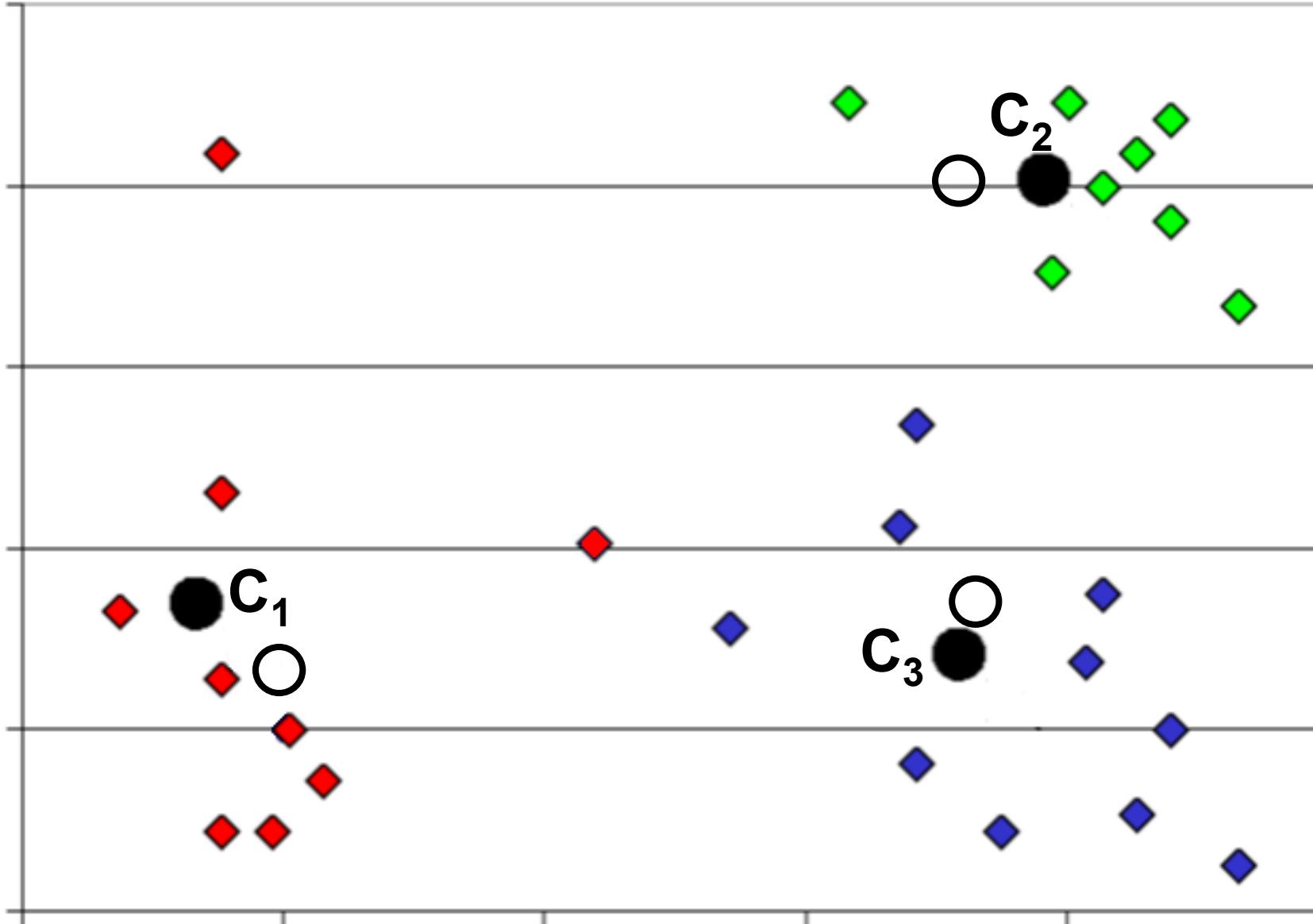
Update Centers: re-compute the center of each cluster



Assign Labels: assign each data point to the nearest cluster center



Update Centers: re-compute the center of each cluster



# two steps run iteratively in the k-means algorithm

- Update Centers

for each cluster, move the center vector  $C$  to the average location of the data points in the cluster

- Update Labels

for each data point, find the nearest cluster center and then attach a cluster label to the data point

Watch video: k-means\_clustering

Run kmeans\_raw.ipynb

# Formal statement of the k-means objective

- Given  $N$  data points  $\{x_1, \dots, x_N\}$   
 $x_n \in \mathcal{R}^M$  is a data point (feature vector) of an object
- Find  $K$  cluster centers,  $\{c_1, \dots, c_K\}$ ,  $c_k \in \mathcal{R}^M$ ,  $K \ll N$
- Assign each data point  $x_n$  to one cluster:
  - $\alpha(n)$  is the cluster label of the data point  $x_n$
  - $\alpha(n) = k$  states the data point  $x_n$  is assigned to the cluster-k
- The goal is to find the optimal clusters such that the objective/loss function is minimized:

$$L = \frac{1}{N} \sum_{n=1}^N \|x_n - c_{\alpha(n)}\|^2$$

the average "distance" (squared) from the data points to the corresponding centers

# Clustering is difficult in general

- Find  $K$  cluster centers  $\{c_1, \dots, c_K\} \in \mathcal{R}^M$  that minimize the loss

$$L = \frac{1}{N} \sum_{n=1}^N \|x_n - c_{\alpha(n)}\|^2$$

$\alpha(n) = k$  states  $x_n$  is assigned to the cluster- $k$

- It is a chicken-egg problem:
  - To make the assignment, we need to know the centers
  - To obtain the centers, we need to know the assignment (cluster labels)
- Brute-force search: try all possible assignments  $\{\alpha(1), \alpha(2), \dots, \alpha(N)\}$

Given  $N$  data points, there are  $K^N$  possible clustering results: computation cost is too high for a large dataset

# The k-means algorithm

- Initialization: the user inputs  $K$ , and the algorithm initializes random centers  $\{c_1, \dots, c_K\}, c_k \in \mathcal{R}^M$
- In each iteration:
  - step-1: assign each data point  $x_n$  to its nearest cluster

$$\alpha(n) = \arg \min_{k \in \{1, \dots, K\}} \|x_n - c_k\|^2$$

- step-2: move center  $c_k$  to the average location of the data points in cluster- $k$

$$c_k = \frac{1}{N_k} \sum_{n: \alpha(n)=k} x_n$$

where  $N_k$  is the number of data points in the cluster- $k$

```
[3]: ► import numpy as np
```

```
[4]: ► A=np.array([0.1, 0.2, 0.01, 1, 2])
      idx=np.argmin(A)
      idx
```

Out [4]: 2

$$\alpha(n) = \arg \min_{k \in \{1, \dots, K\}} \|x_n - c_k\|^2$$

$A_{[k]} = \|x_n - c_k\|^2$  is the squared-distance between  $x_n$  and  $c_k$

$$\alpha(n) = \arg \min([A_{[1]}, A_{[2]}, \dots, A_{[K]}])$$

Note: Element index starts from 0 in Python, but it starts from 1 in textbooks

# the k-means algorithm

- Loss:  $L = \frac{1}{N} \sum_{n=1}^N \|x_n - c_{\alpha(n)}\|^2$
- The goal of clustering is to minimize the loss
- For a cluster, the optimal cluster center is the average of the data points in the cluster.
$$c_k = \frac{1}{N_k} \sum_{n: \alpha(n)=k} x_n$$
- Why?

## the k-means algorithm

- Rewrite the loss function (it is a scalar function):

$$L = \frac{1}{N} \sum_{n=1}^N \sum_{k=1}^K \alpha_{(n,k)} \|x_n - c_k\|^2$$

a  $N$ -by- $K$  assignment matrix  $\begin{bmatrix} \alpha_{(1,1)} & \dots & \alpha_{(1,K)} \\ \dots & \dots & \dots \\ \alpha_{(N,1)} & \dots & \alpha_{(N,K)} \end{bmatrix}$ ,  $\alpha_{(n,k)} = 0 \text{ or } 1$

$\alpha_{(n,k)} = 1$  if and only if  $x_n$  is assigned to the cluster- $k$

$\sum_{k=1}^K \alpha_{(n,k)} = 1$  because  $x_n$  is assigned to only one cluster

## the k-means algorithm

- The loss function (it is a scalar/number):

$$L = \frac{1}{N} \sum_{n=1}^N \sum_{k=1}^K \alpha_{(n,k)} \|x_n - c_k\|^2$$

We want to minimize the loss: it is an optimization problem.

- Next, we show that the solution of this optimization problem is

$$c_k = \frac{1}{N_k} \sum_{n=1}^N \alpha_{(n,k)} x_n$$

which means the optimal cluster center is the average of the data points in the cluster

## Calculus (basic concept)

Two vectors  $\mathbf{x}, \mathbf{b} \in \mathbb{R}^M$ , let  $f(\mathbf{x}) = \mathbf{b}^T \mathbf{x}$ , then  $\frac{\partial f}{\partial \mathbf{x}} = \mathbf{b}$

$$\mathbf{x} = \begin{bmatrix} x_{[1]} \\ x_{[2]} \\ x_{[3]} \end{bmatrix}, \mathbf{b} = \begin{bmatrix} b_{[1]} \\ b_{[2]} \\ b_{[3]} \end{bmatrix}, \quad f(\mathbf{x}) = x_{[1]}b_{[1]} + x_{[2]}b_{[2]} + x_{[3]}b_{[3]}$$

$$\frac{\partial f}{\partial \mathbf{x}} \triangleq \begin{bmatrix} \frac{\partial f}{\partial x_{[1]}} \\ \frac{\partial f}{\partial x_{[2]}} \\ \frac{\partial f}{\partial x_{[3]}} \end{bmatrix} = \mathbf{b} \text{ because } \frac{\partial f}{\partial x_{[1]}} = b_{[1]}, \frac{\partial f}{\partial x_{[2]}} = b_{[2]}, \frac{\partial f}{\partial x_{[3]}} = b_{[3]}$$

$L = \frac{1}{N} \sum_{n=1}^N \sum_{k=1}^K \alpha_{(n,k)} \|x_n - c_k\|^2$ , k-means uses  $\ell_2$  norm

$$\|x_n - c_k\|^2 = (x_n - c_k)^T (x_n - c_k) = x_n^T x_n + c_k^T c_k - 2x_n^T c_k$$

$$\frac{\partial L}{\partial c_k} = \frac{1}{N} \sum_{n=1}^N 2\alpha_{(n,k)} (c_k - x_n)$$

$\frac{\partial L}{\partial c_k} = 0$  when the loss  $L$  reaches the minimum value

$$L = \frac{1}{N} \sum_{n=1}^N \sum_{k=1}^K \alpha_{(n,k)} \|x_n - c_k\|^2, \text{ k-means uses } \ell_2 \text{ norm}$$

$$\|x_n - c_k\|^2 = (x_n - c_k)^T (x_n - c_k) = x_n^T x_n + c_k^T c_k - 2x_n^T c_k$$

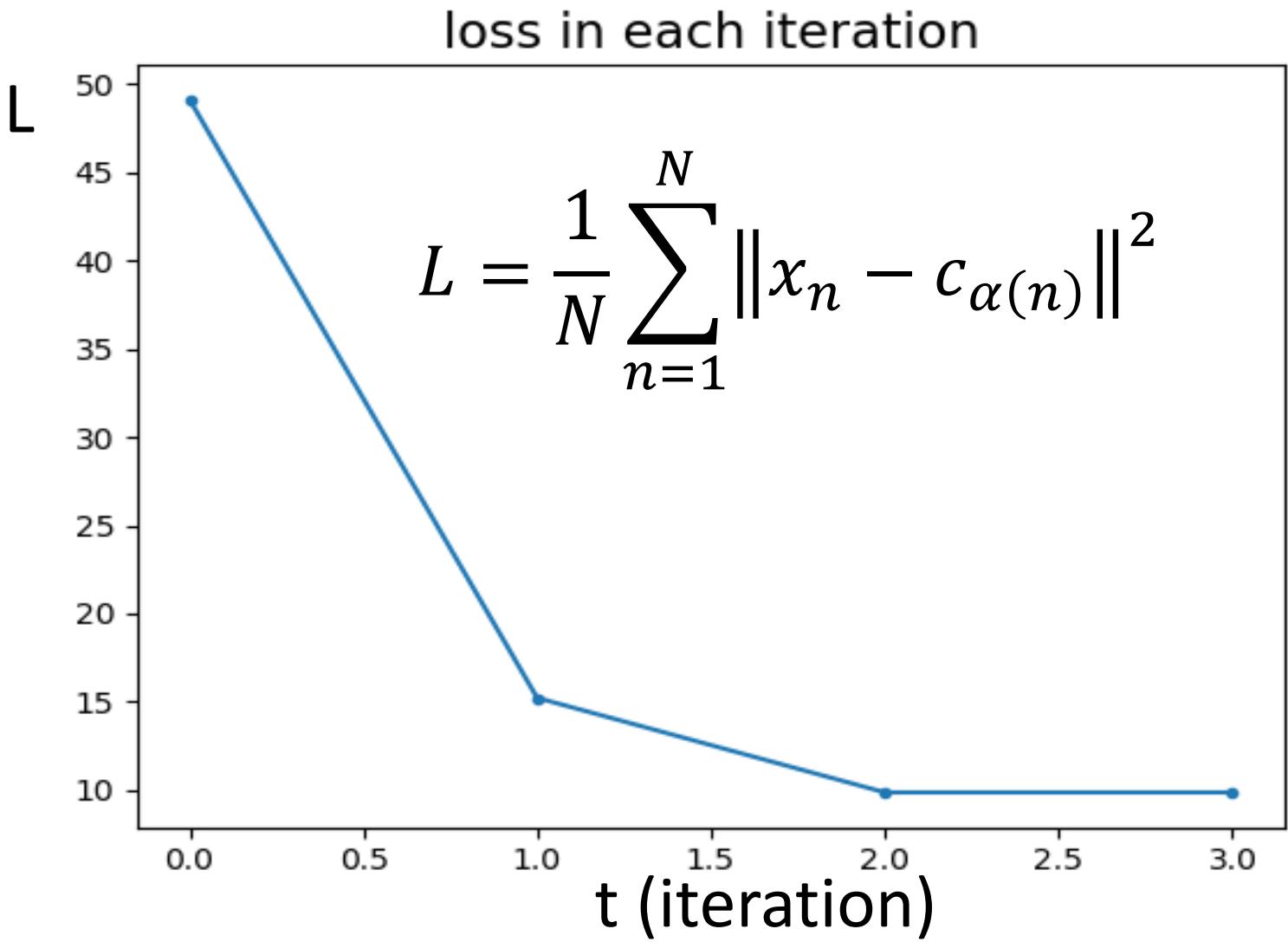
$$\frac{\partial L}{\partial c_k} = \frac{1}{N} \sum_{n=1}^N 2\alpha_{(n,k)} (c_k - x_n)$$

$\frac{\partial L}{\partial c_k} = 0$  when the loss  $L$  reaches the minimum value

we set  $\frac{\partial L}{\partial c_k} = 0$  and obtain:

$$c_k = \frac{1}{N_k} \sum_{n=1}^N \alpha_{(n,k)} x_n, \quad N_k \text{ the number of data points in the cluster-}k$$

Therefore, the optimal cluster center is the average of the data points in the cluster.



For our dataset, the algorithm converged after some iterations.  
Declare convergence if : after a number of iterations, the loss curve becomes flat: check if  $|L^{(t)} - L^{(t-1)}| < \epsilon$ , (e.g.,  $\epsilon = 0.0001$ )

## Question:

- Will the k-means algorithm always converge after a finite number of iterations for any dataset ?

Yes ? No ? Maybe ?

# Convergence of the k-means algorithm

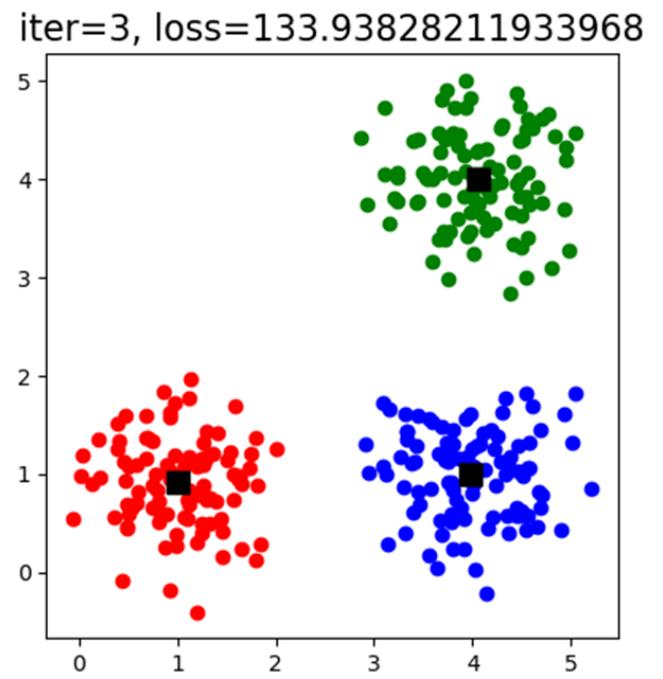
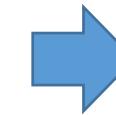
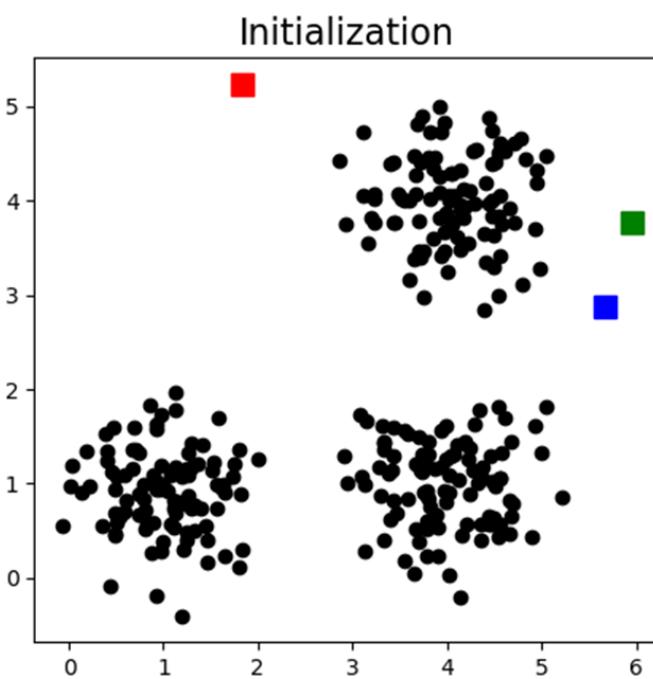
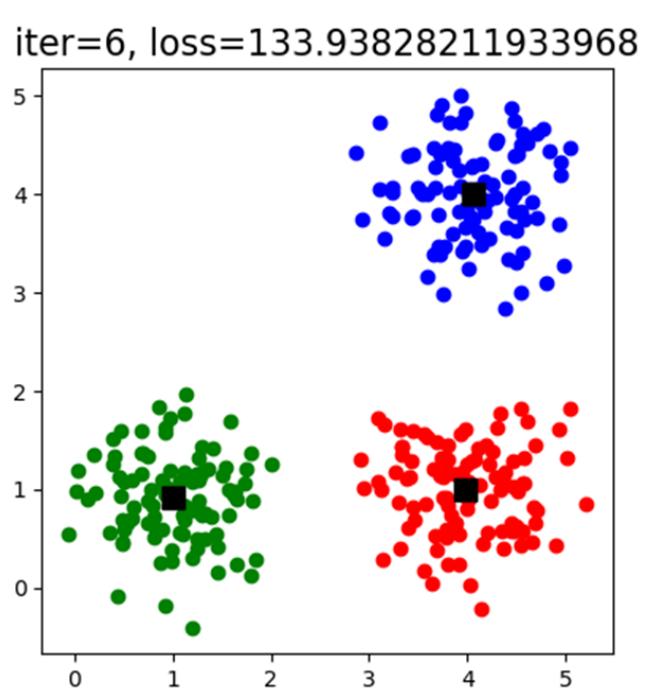
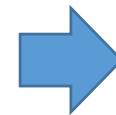
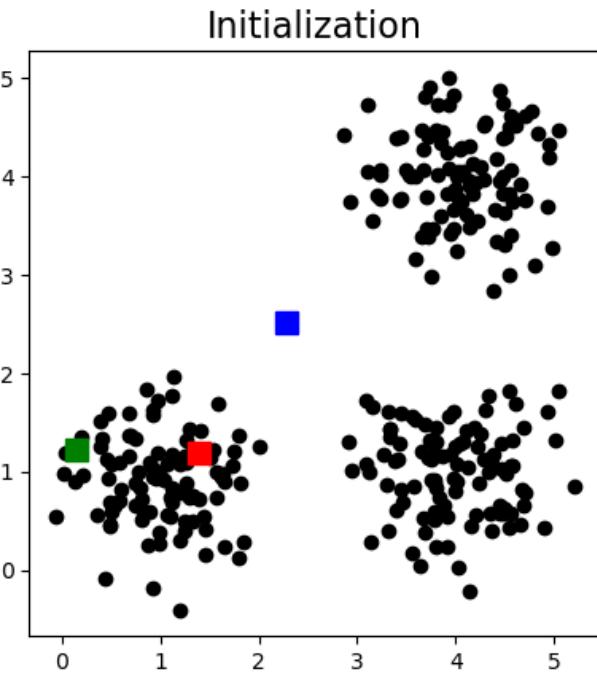
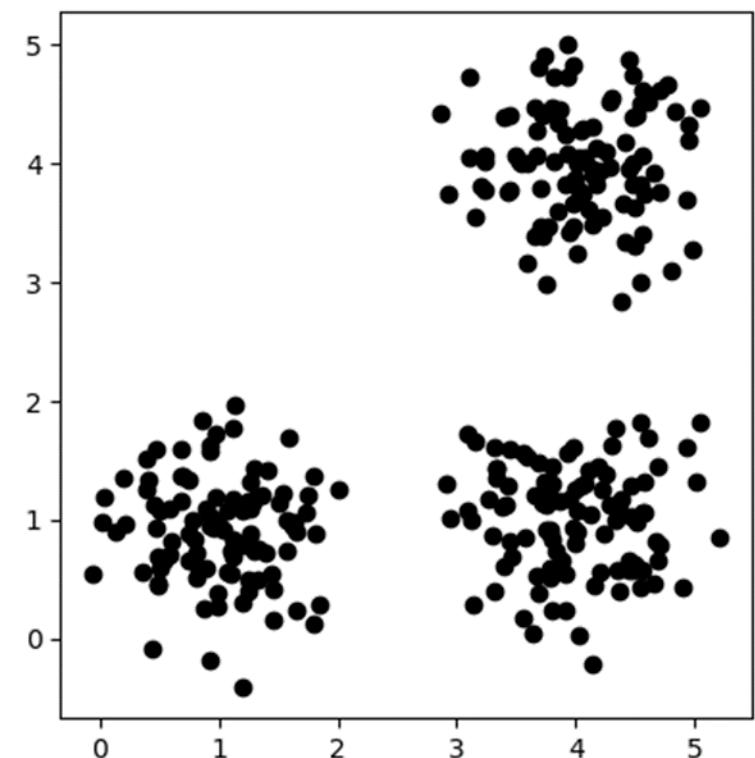
- loss:  $L = \frac{1}{N} \sum_{n=1}^N \|x_n - c_{\alpha(n)}\|^2$
- There is only a finite number of clustering results,  $K^N$ , so the loss only has a finite number of possible values.
- The loss will not increase in each iteration of the k-means algorithm
  - make assignment:  $\alpha(n) = \arg \min_{k \in \{1, \dots, K\}} \|x_n - c_k\|^2$
  - update centers:  $c_k = \arg \min_c \sum_{n: \alpha(n)=k} \|x_n - c\|^2$

## Question:

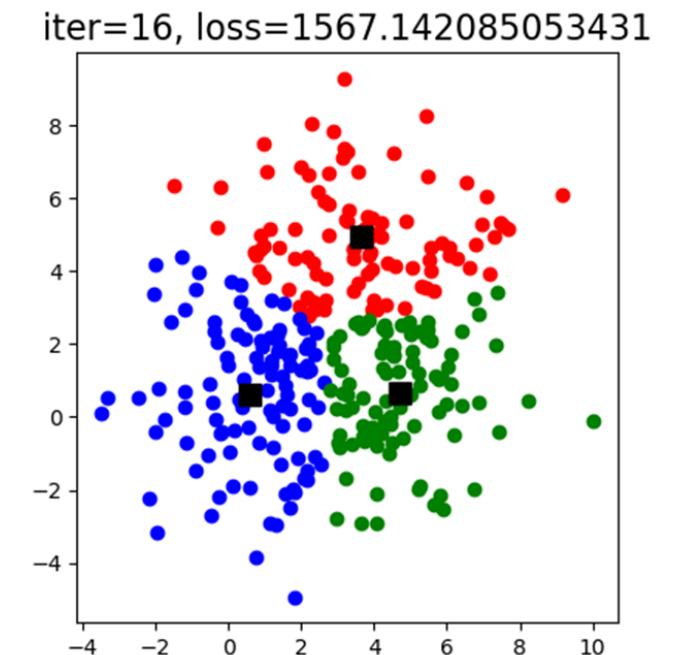
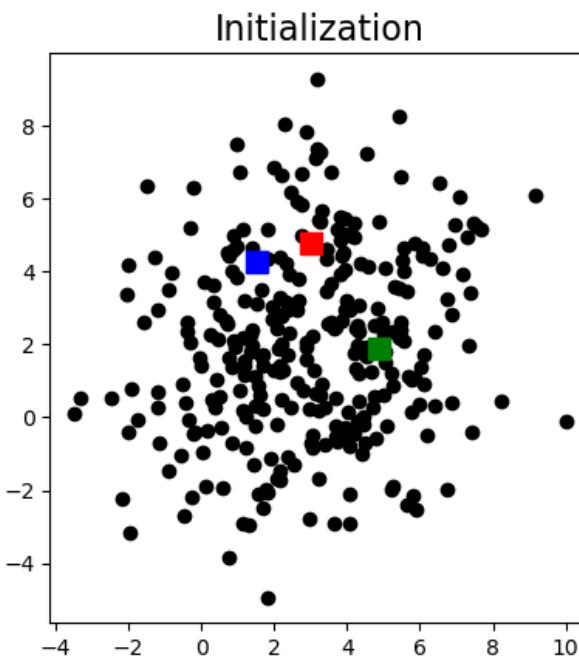
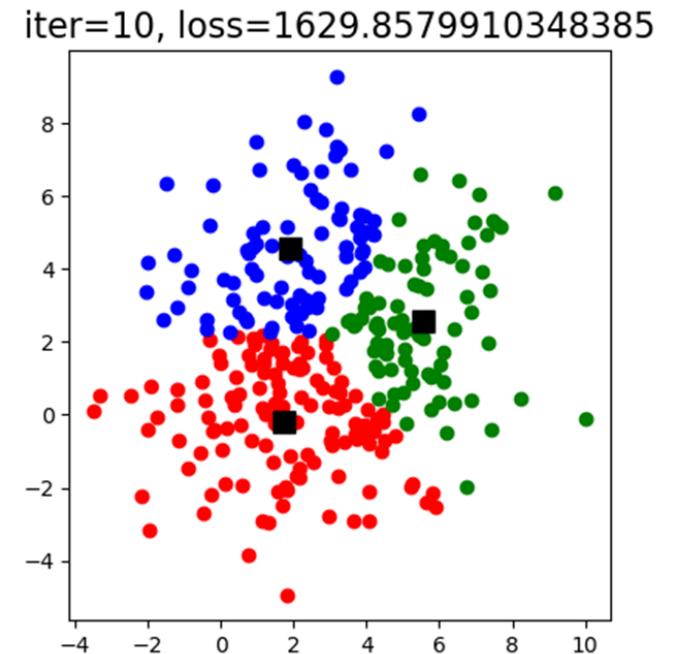
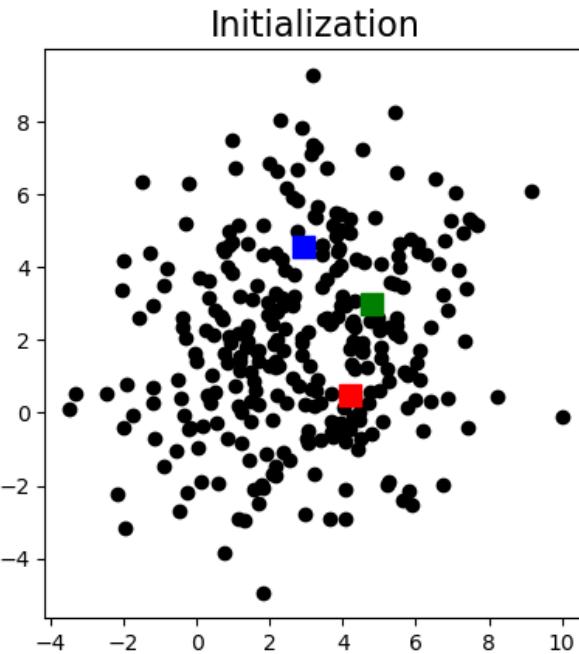
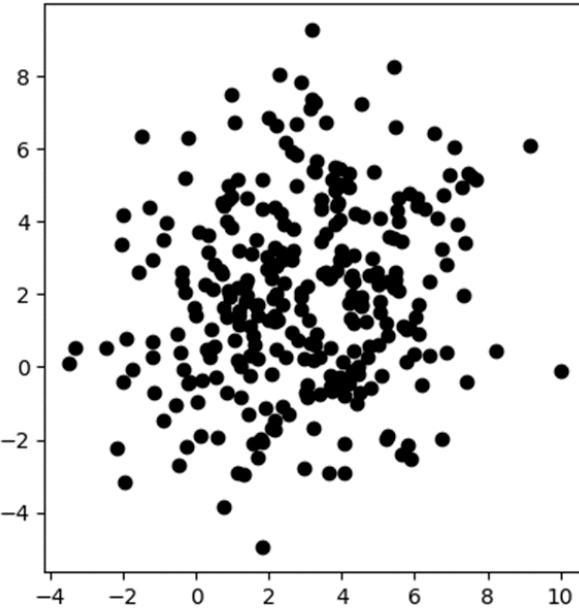
- Will different initialization lead to different clustering results?

Yes ? No ? Maybe ?

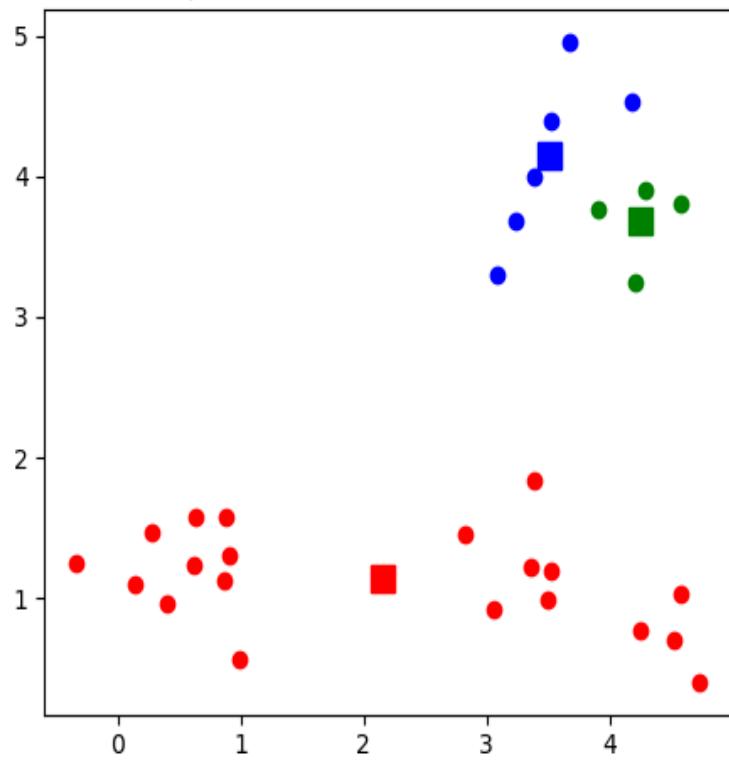
Centers/centroids are randomly initialized



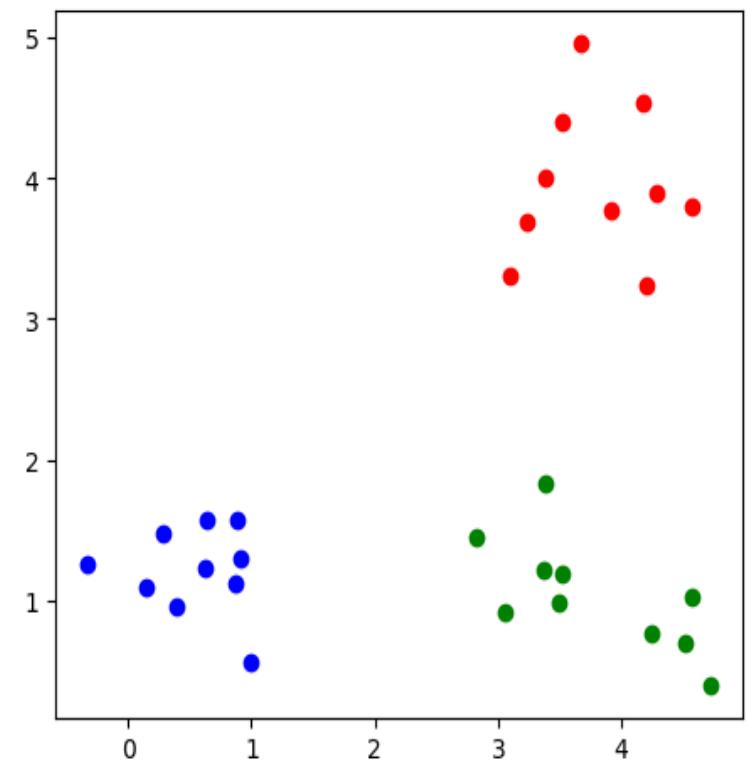
Centers are randomly initialized



a bad result from k-means



the best clustering result



## Clustering result of k-means algorithm could be random

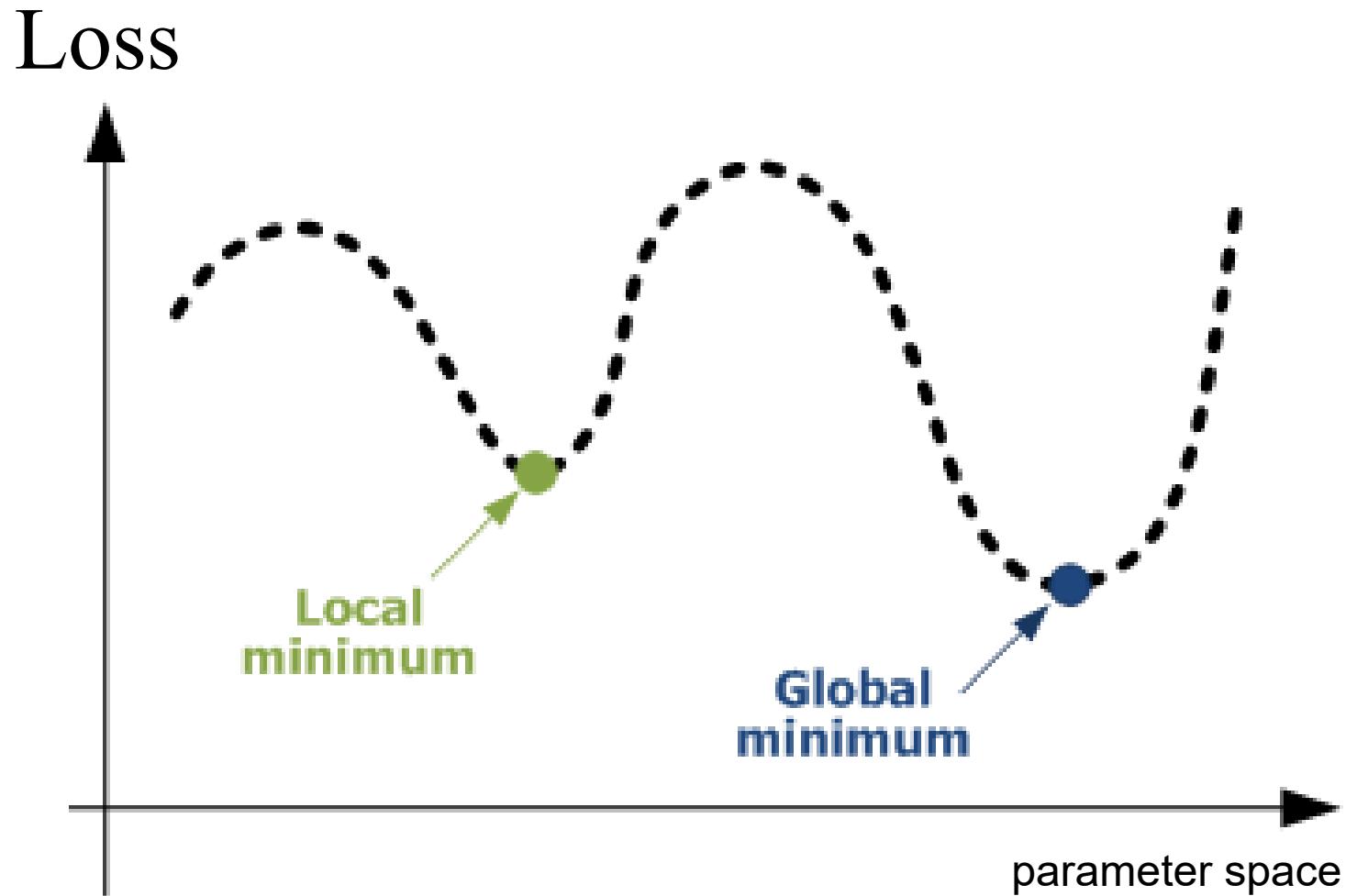
Clustering result is determined by data distribution and initialization  
(given the number of clusters K)

Initialization of the centers is random.

If the data points are well-separated, then different initializations could lead to the same clustering result (cluster ID may change)

If the data points are NOT well-separated, then different initializations could lead to different clustering results.

# k-means may just find a local optimal solution



# Better Initialization for k-means?

## k-means++: The Advantages of Careful Seeding

David Arthur and Sergei Vassilvitskii

### Abstract

The k-means method is a widely used clustering technique that seeks to minimize the average squared distance between points in the same cluster. Although it offers no accuracy guarantees, its simplicity and speed are very appealing in practice. By augmenting k-means with a simple, randomized seeding technique, we obtain an algorithm that is  $O(\log k)$ -competitive with the optimal clustering. Experiments show our augmentation improves both the speed and the accuracy of k-means, often quite dramatically.

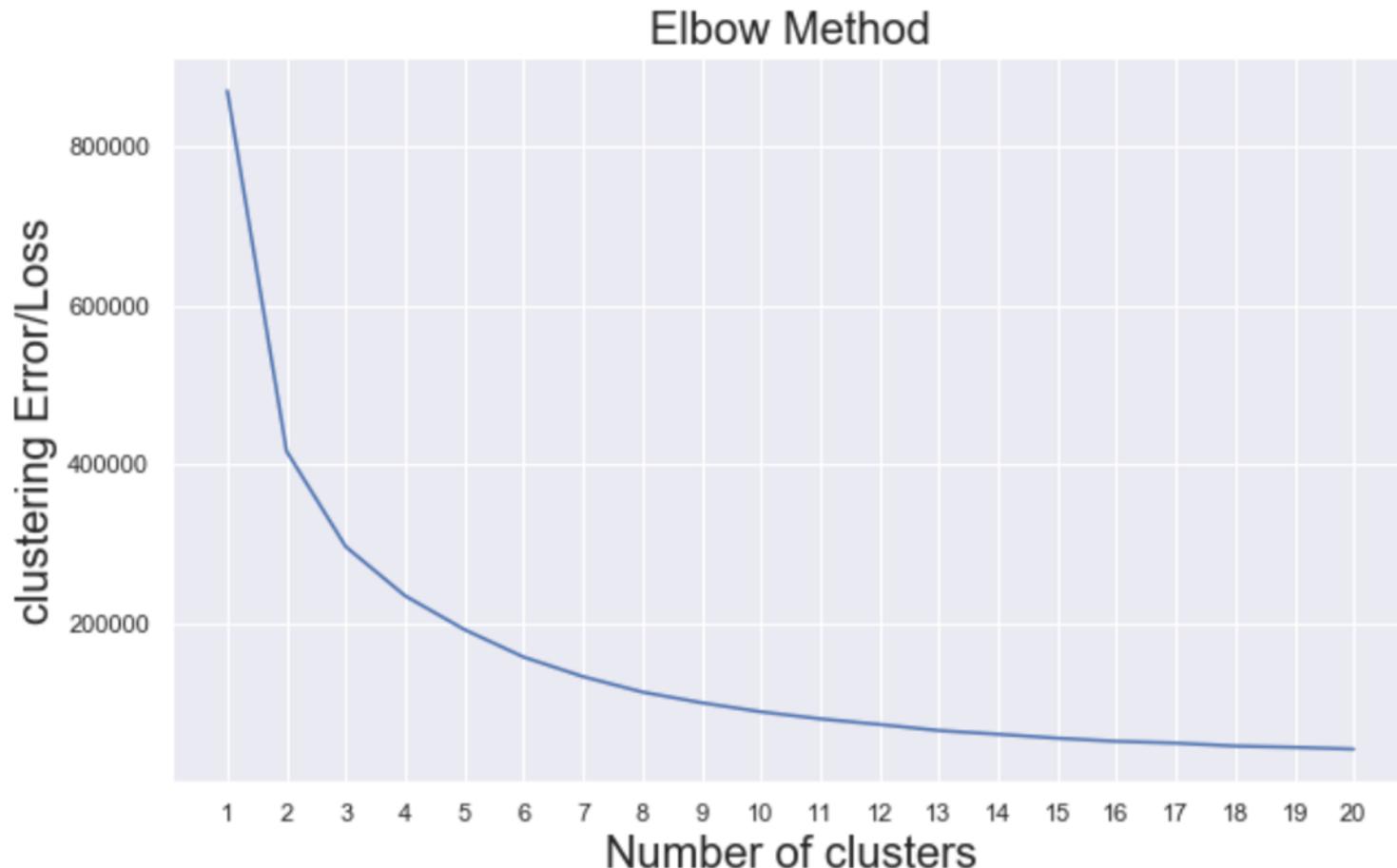
**k-means++** use a special algorithm to initialize centers from the data points.  
The initial centers will be far away to each other  
It helps to find the best solution

```
class sklearn.cluster.KMeans(n_clusters=8, init='k-means++', n_init=10, max_iter=300, tol=0.0001,  
    precompute_distances='auto', verbose=0, random_state=None, copy_x=True, n_jobs=None,  
    algorithm='auto')
```

[source]

# How many clusters ? (K = ?)

- From some prior knowledge about the application
- Try  $K = 2, 3, 4, 5, \dots$  plot the results and see which one is the best



# Empty Cluster ?

- Will the k-means algorithm output an empty cluster ?  
e.g., set  $K = 10$ , but the first cluster is empty (no data points in that cluster)
- Handle empty cluster:  
whenever an empty cluster is detected during the iterations in k-means, a new center will be generated randomly for this cluster

This method will reduce the chance of empty clusters.

It is implemented in sk-learn k-means

# Faster ?

## sklearn.cluster.MiniBatchKMeans

- In standard k-means, we need to use all data points to compute the centers and make assignments

$$loss = \frac{1}{N} \sum_{n=1}^N \|x_n - c_{\alpha(n)}\|^2$$

- Mini-batch k-means

7 2 1 0 4 1 4 9 5 9 0 1 5 9 7 8 4 9 6 6 5 4 0 7 4 0 1 3 1 3 4 7 2 7 1 2 1 1 7 4 2 3 5 1 2 4 4 6 3 5 5 6 0 4 1 9 5 7 8 4 3 7 4 6 4 3 0 7 0 2 9 1 7 3 2 9 7 1 6 2 7 8 4 7 3 6 1 3 6 9 3 1 4 1 7 6 9 6 0 5 4 9 9 2 1 9 4 8 7 3 9 7 9 4 4 9 2 5 4 7 6 7 9 0 5 8 5 6 6 5 7 8 1 0 1 6 4 6 7 3 1 7 1 8 2 0 2 9 9 5 5 1 5 6 0 3 4 4 6 5 1 5 6 5 1 4 4 7 2 2 7 1 1 0 1 8 1 5 6 0 2 1 7 5 0 1 1	Mini-batch 1
0 9 0 3 1 6 4 2 3 6 1 1 3 9 5 2 9 4 5 9 3 9 0 3 6 5 5 7 8 2 7 1 2 8 4 1 7 3 3 8 8 7 9 2 2 4 1 5 9 8 7 2 3 0 4 2 4 1 9 5 7 7 2 8 2 6 8 5 7 7 9 1 8 1 8 0 3 0 1 9 9 4 1 2 9 7 5 9 2 6 4 1 5 4 2 9 2 0 4 0 0 2 8 4 7 1 2 4 0 2 7 4 3 3 0 0 1 9 6 5 0 5 2 7 9 3 0 4 2 0 7 1 1 2 1 5 3 3 9 2 8 6 5 9 1 3 8 1 0 5 1 3 1 5 5 2 9 8 5 1 4 0 4 1 6 7 7 5 0 6 5 6 3 7 8 0 8 5 4 1 1 4 0 1 2 7	Mini-batch 2
6 1 0 2 7 1 4 8 0 1 9 5 4 3 7 4 2 8 5 8 2 4 5 0 0 1 1 1 0 1 1 1 9 2 1 4 2 9 2 0 4 9 1 4 8 1 8 4 5 9 8 8 3 7 6 0 0 3 0 2 6 6 1 9 3 3 2 3 9 1 2 6 8 0 5 6 6 6 3 8 2 7 5 8 9 6 1 8 4 1 2 5 9 1 9 7 5 4 0 8 9 9 1 0 5 2 3 7 8 9 4 0 6 3 9 5 2 1 3 1 3 6 5 7 9 2 2 6 3 2 6 5 4 8 9 7 1 3 0 3 8 3 1 9 3 4 6 4 6 2 1 8 2 5 4 8 8 4 0 0 2 3 2 7 7 6 8 7 4 7 9 6 9 0 9 8 0 4 6 0 6 3 5 4 9 3 3 9	Mini-batch 3

new center

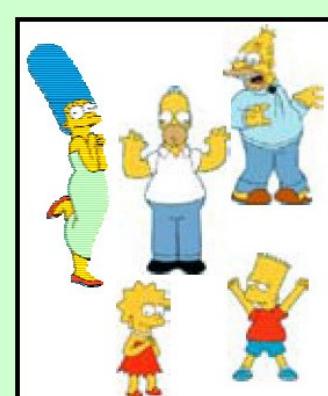
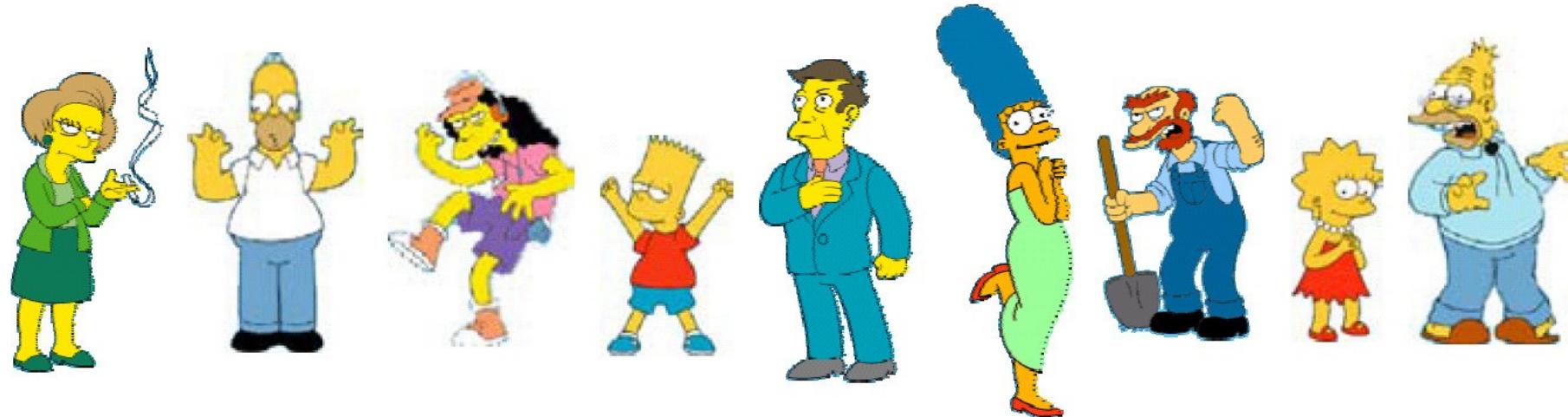
$$\mathbf{c} \leftarrow (1 - \eta)\mathbf{c} + \eta \mathbf{x}$$

from the previous mini-batch

learning rate

c is the center  
nearest to x

# Clustering is based on distance measure and feature vector



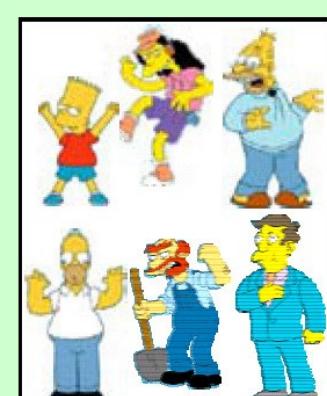
Simpson's Family



School Employees

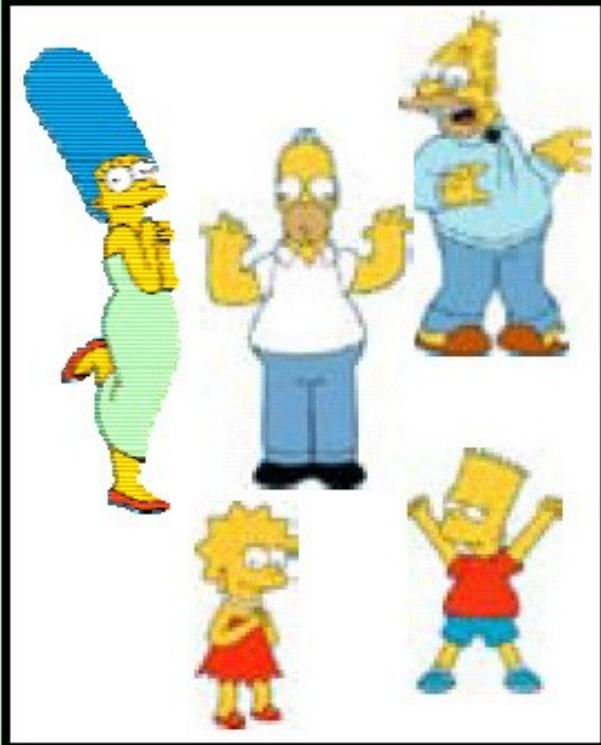


Females



Males

# Clustering is based on distance measure and feature vector



Simpson's Family

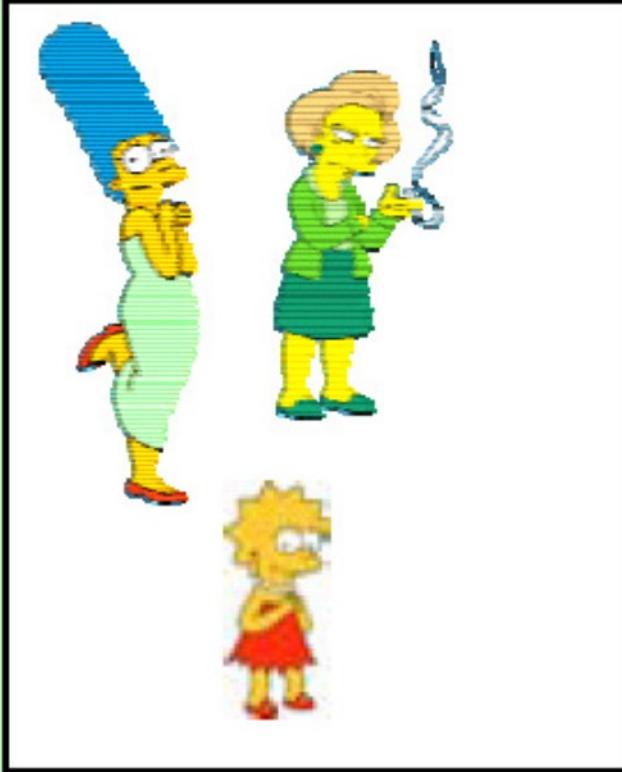


School Employees

Feature Vector

$$x = [last\_name]$$

# Clustering is based on distance measure and feature vector



Females



Males

Feature Vector  
 $x = [gender]$

# many distance/dissimilarity measures



<https://www.psychologytoday.com/us/blog/canine-corner/201308/do-dogs-look-their-owners>

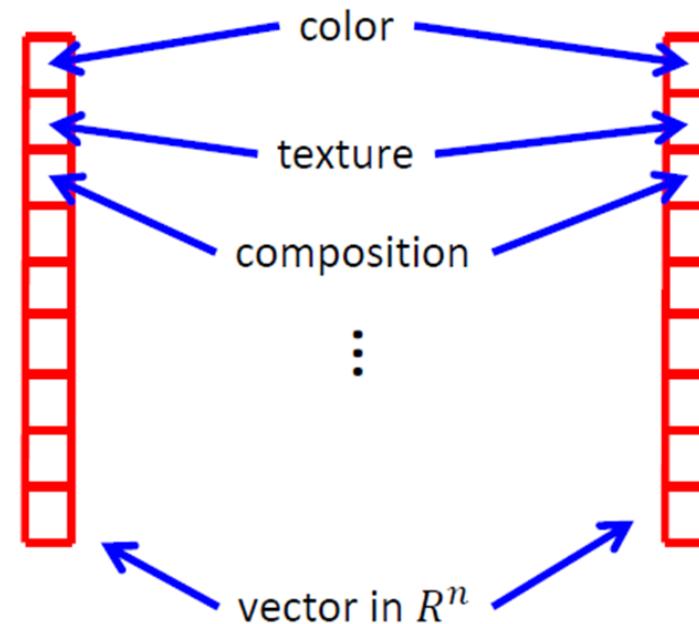
# So what is clustering in general?

- You choose a distance/dissimilarity function
- The algorithm figures out the grouping of objects based on the distance function:  $\text{distance}(\text{vectorA}, \text{vectorB})$
- Data points within a cluster are similar
- Data points across clusters are not so similar

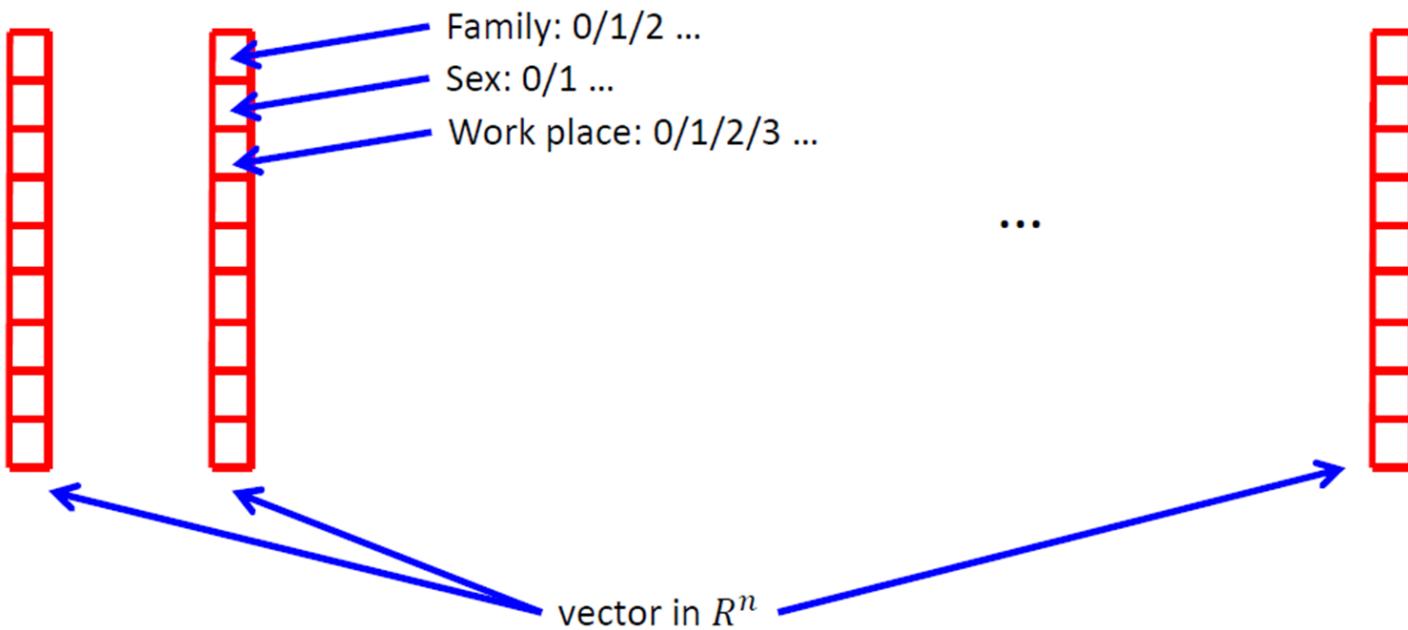
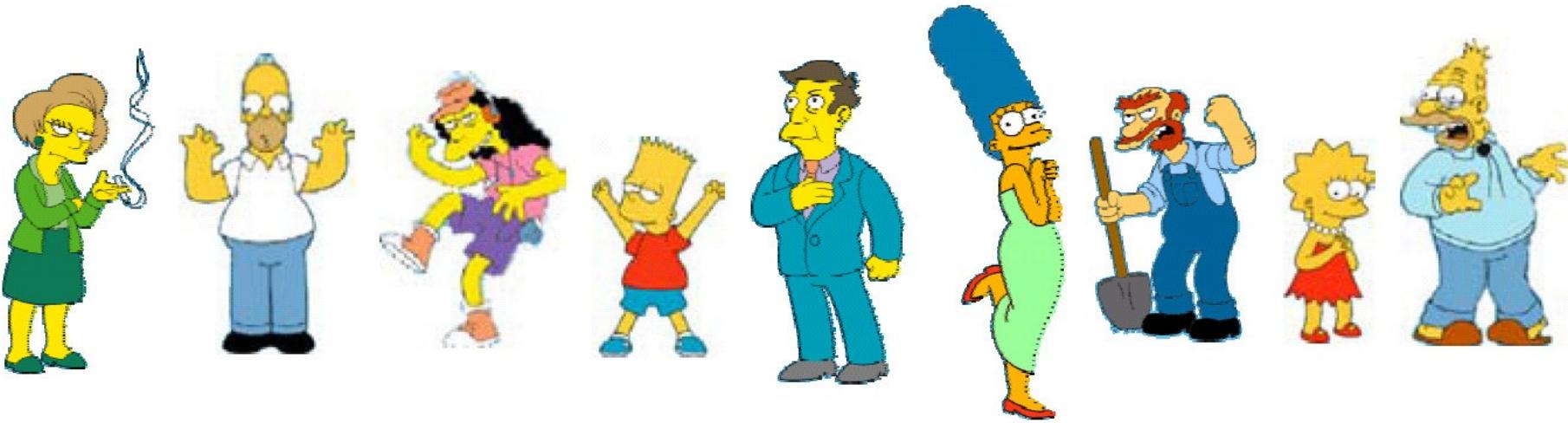
# Feature Extraction Before Clustering

- Images of different sizes

Can not directly compare the two images because they have different number of pixels  
resize the images, or extract some features



## Objects in real life



## A Potential Problem in features and distance

1	income	spend	gender
2	233	150	0
3	250	187	1
4	204	172	0
5	236	178	1
6	354	163	0
7	192	148	1
8	294	153	1
9	263	173	1
10	199	162	0

$$x = \begin{bmatrix} 233 \\ 150 \\ 0 \end{bmatrix}, \quad y = \begin{bmatrix} 250 \\ 187 \\ 1 \end{bmatrix}$$

the distance is  $\|x - y\|_2$

$$= \sqrt{(233 - 250)^2 + (150 - 187)^2 + (0 - 1)^2}$$

$$\approx \sqrt{(233 - 250)^2 + (150 - 187)^2}$$

The distance is dominated by the differences in income and spend; gender is almost "ignored"

This is bad because gender information is very useful for clustering: male and female customers have different spending patterns

# Normalize features: weight the features equally

1	income	spend	gender
2	233	150	0
3	250	187	1
4	204	172	0
5	236	178	1
6	354	163	0
7	192	148	1
8	294	153	1
9	263	173	1
10	199	162	0

## Method-1:

calculate mean of income (1<sup>st</sup> column), **m**

calculate standard deviation (std) of income, **s**

**normalize income by mean and std**

**income <= (income - m)/s**

**do the same thing for the other two features**

**sklearn.preprocessing.StandardScaler**

`class sklearn.preprocessing.StandardScaler(copy=True, with_mean=True, with_std=True)`

[\[source\]](#)

# Normalize features: weight the features equally

1	income	spend	gender
2	233	150	0
3	250	187	1
4	204	172	0
5	236	178	1
6	354	163	0
7	192	148	1
8	294	153	1
9	263	173	1
10	199	162	0

## Method-2:

calculate the max of income (1<sup>st</sup> column), a  
calculate the min of income, b  
**normalize income into the range of 0 to 1 by**  
**income <= (income - b)/(a-b)**

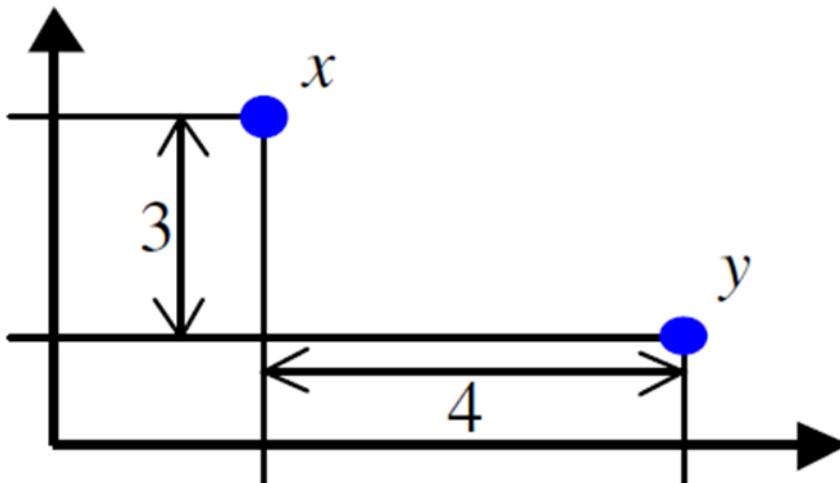
**do the same thing for the other two features**

`sklearn.preprocessing.MinMaxScaler`

`class sklearn.preprocessing.MinMaxScaler(feature_range=(0, 1), copy=True)`

[\[source\]](#)

two data points  $x$  and  $y$  in  $\mathcal{R}^2$



- Euclidian distance:  $\sqrt{4^2 + 3^2} = 5$
- Manhattan distance:  $4 + 3 = 7$
- “inf”-distance:  $\max\{4,3\} = 4$

# distance/dissimilarity functions

- desired properties of a distance function:  $d(x, y)$ 
  - Symmetry:  $d(x, y) = d(y, x)$   
if  $x$  looks like  $y$ , then  $y$  looks like  $x$
  - Positive separability:  $d(x, y) = 0$  if and only if  $x=y$   
if  $x \neq y$ , then  $d(x, y) > 0$
  - Triangle inequality:  $d(x, y) \leq d(x, z) + d(y, z)$   
if  $d(x, z)$  is small and  $d(y, z)$  is small, then  $d(x, y)$  is small  
if  $x$  looks like  $z$  and  $y$  looks like  $z$ , then  $x$  and  $y$  are similar

# Hamming Distance for Binary Vectors

- Manhattan distance is also called *Hamming distance* when all features are binary
- count the number of different binary digits between two vectors

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	
$x$	0	1	1	0	0	1	0	0	1	0	0	1	1	1	1	0	0	1
$y$	0	1	1	1	0	0	0	0	1	1	1	1	1	1	0	1	1	

$$d(x, y) = 5$$

# K-means with a general distance function (not implemented in sk-learn)

- Given  $N$  data points,  $\{x_1, \dots, x_N\}$ ,  $x_n \in \mathcal{R}^M$
- Find  $K$  cluster centers,  $\{c_1, \dots, c_K\}$ ,  $c_k \in \mathcal{R}^M$ ,  $K \ll N$
- Assign each data point  $x_n$  to one cluster: label  $\alpha(n) \in \{1, \dots, K\}$
- The optimal clustering result is obtained by minimizing the loss

$$L = \frac{1}{N} \sum_{n=1}^N d(x_n, c_{\alpha(n)})^2$$

the average distance (squared) from data points to corresponding centers

K-means is old (1957) but not obsolete

# Maybe we only need Euclidian distance if we train a deep neural network for feature extraction

