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

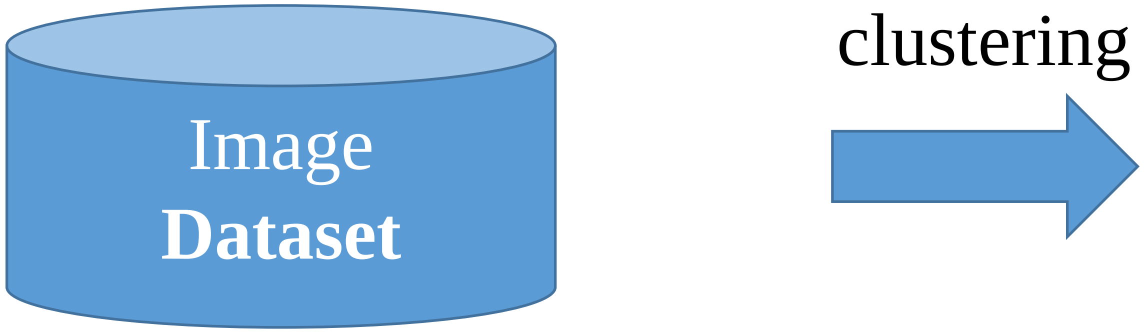
- Unsupervised Learning

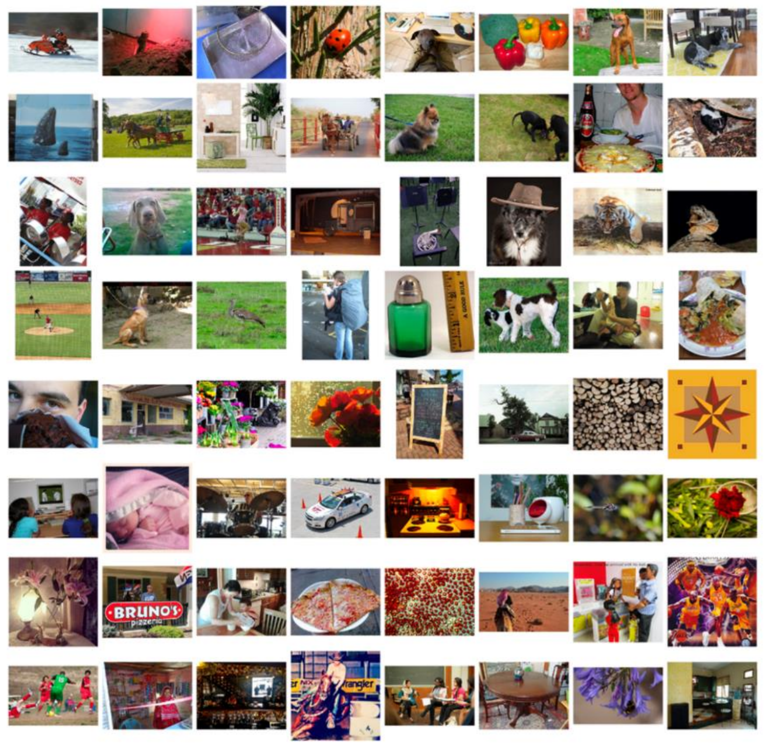
to model the features of a dataset without reference to any label

- Reinforcement Learning

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

Cluster images

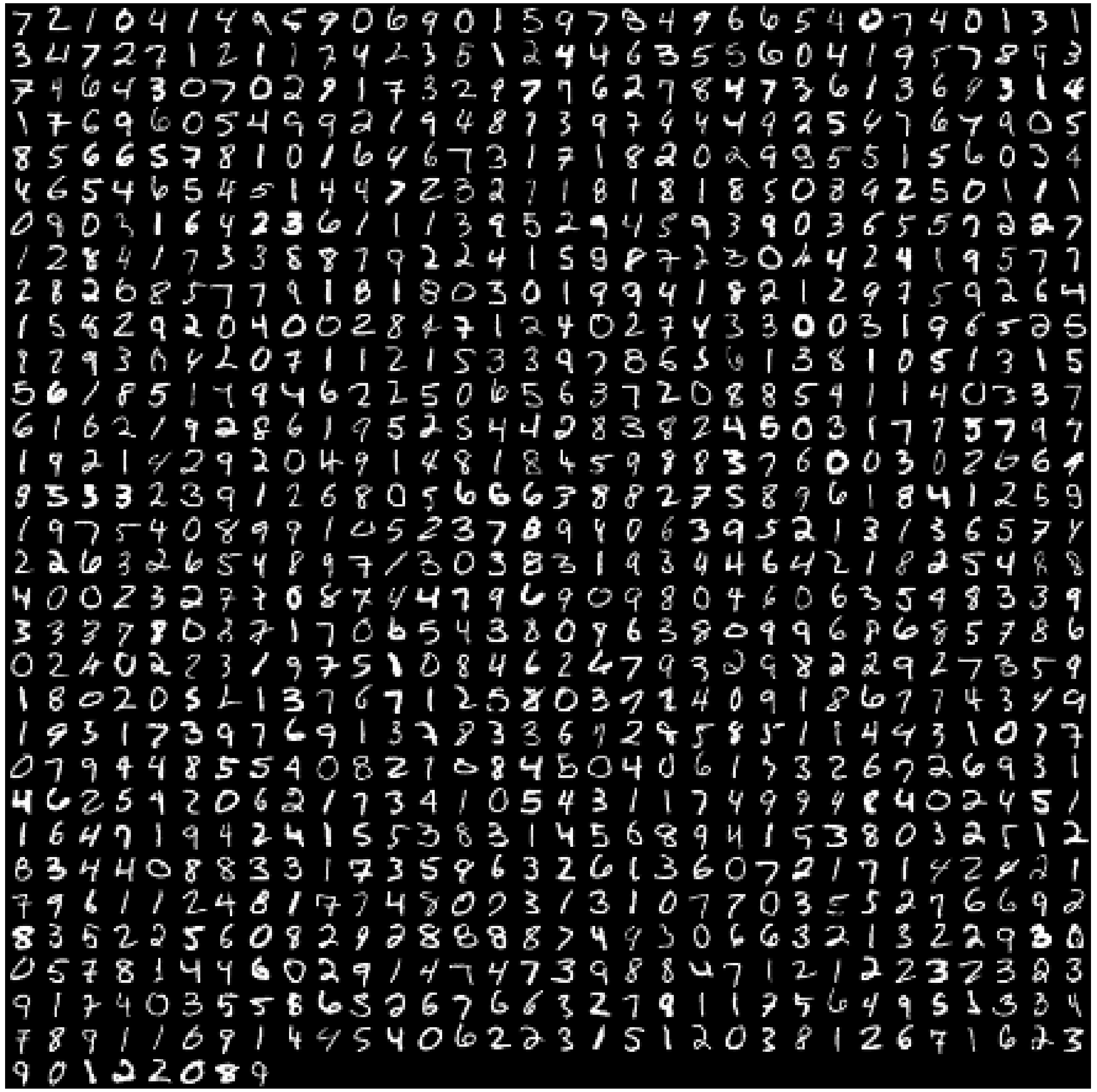




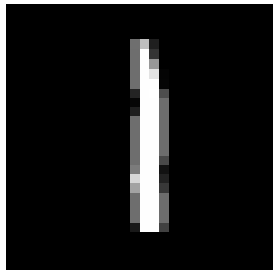
Goal of clustering:
Divide objects into groups, and objects within a group are more similar than those outside the group

unsupervised learning

Clustering handwritten digit images



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
  you 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

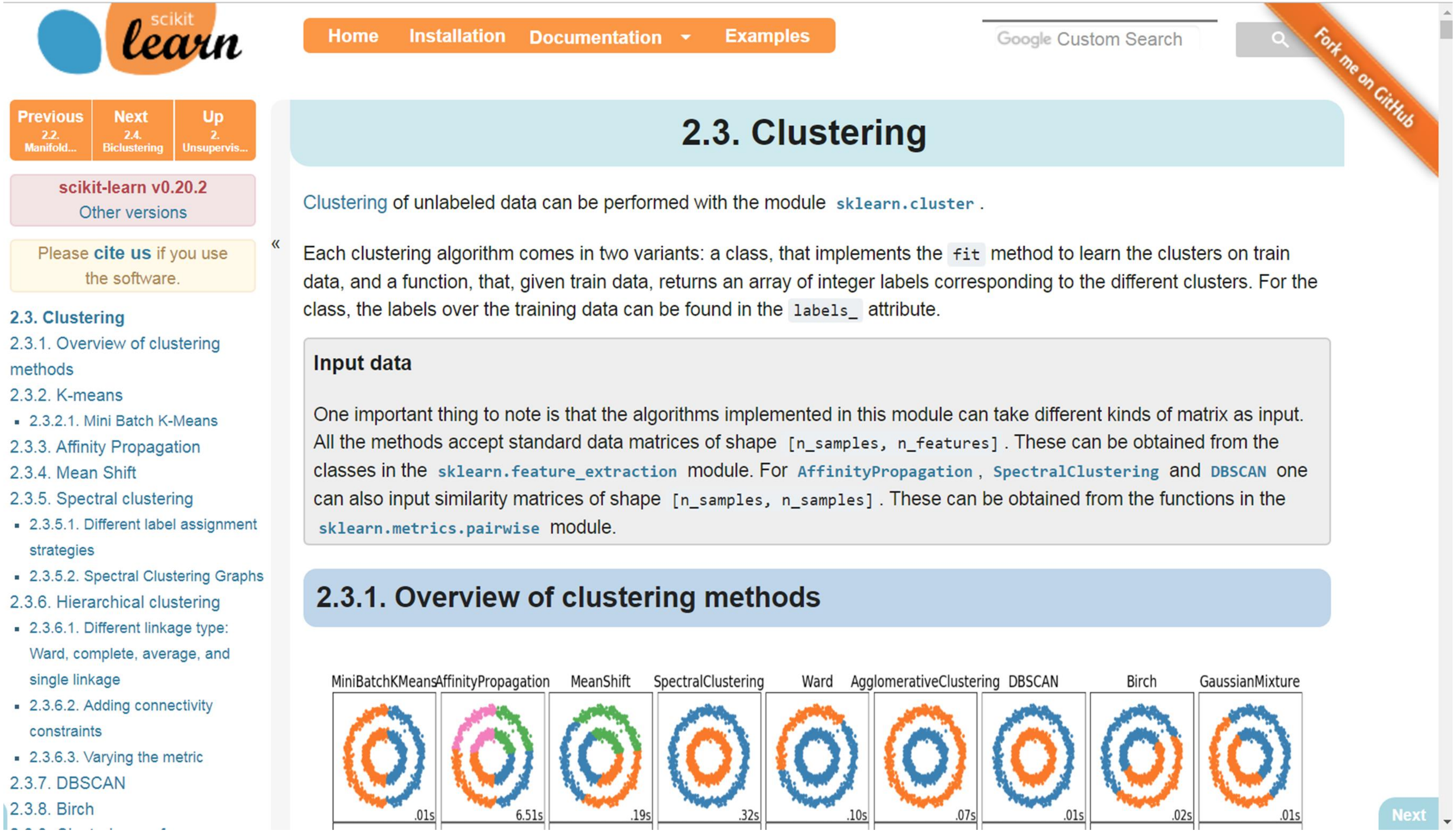
https://arxiv.org/pdf/1803.00919.pdf

- 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 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).



Please **cite us** if you use the software.

2.3. Clustering

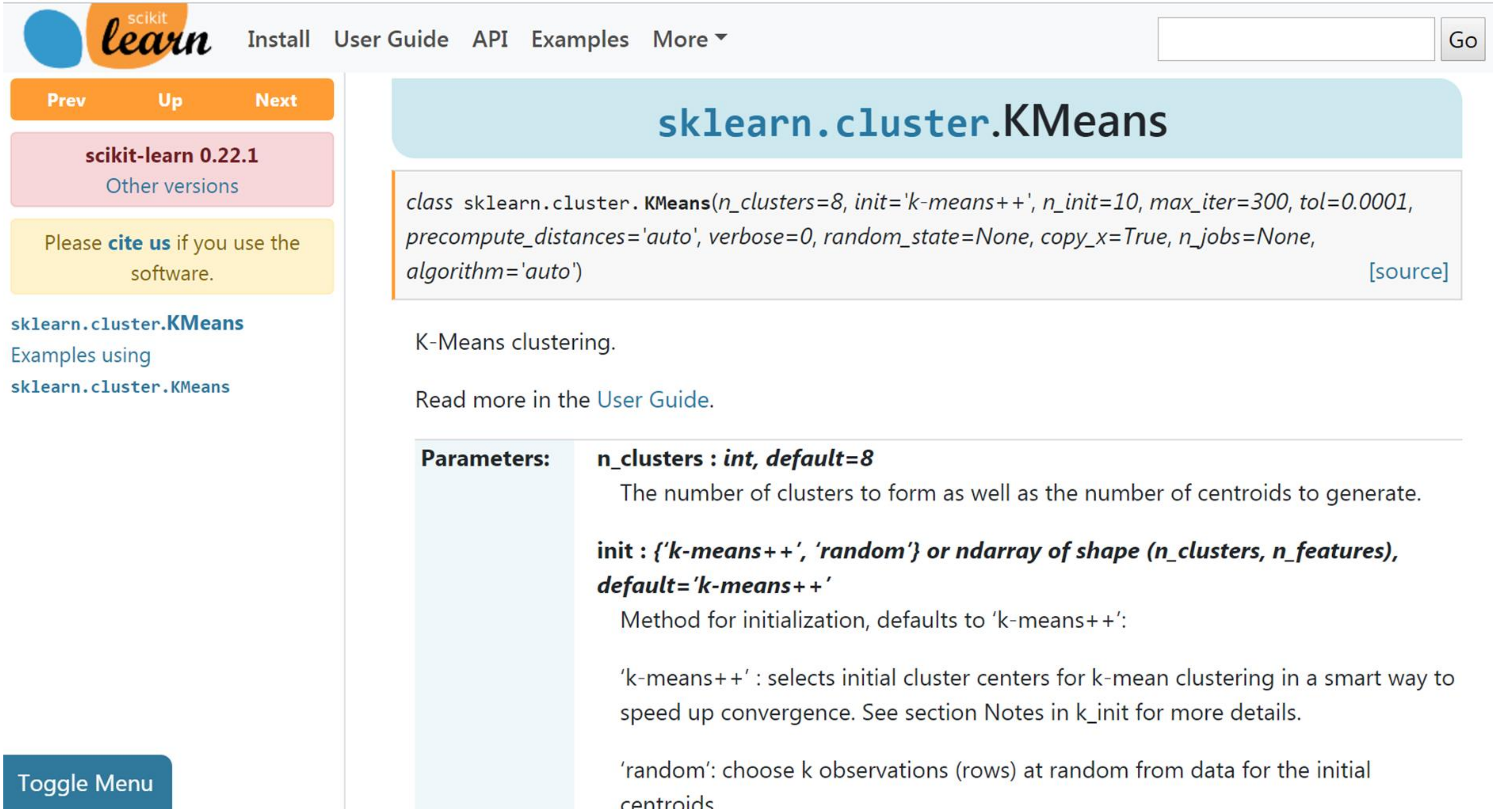
Clustering of unlabeled data can be performed with the module `sklearn.cluster` .

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



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*

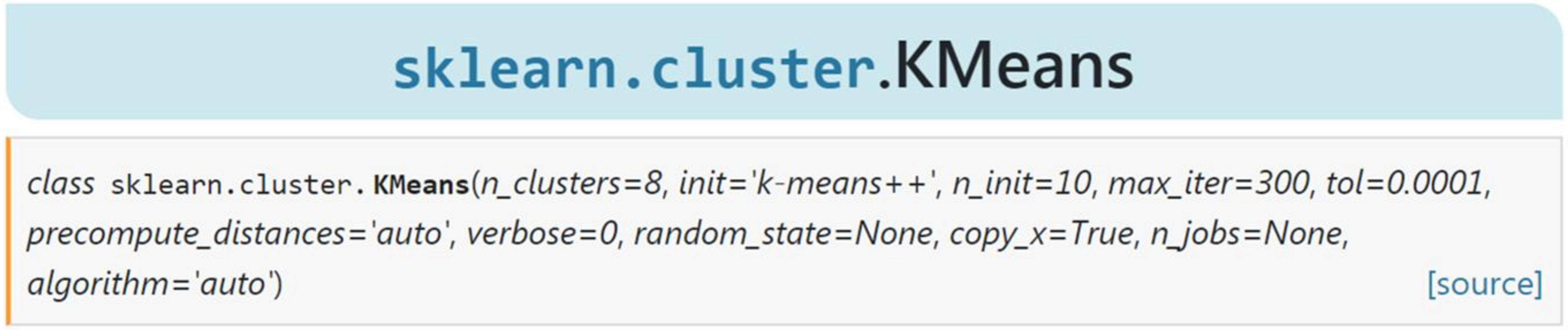
The number of clusters to form as well as the number of centroids to generate.

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


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

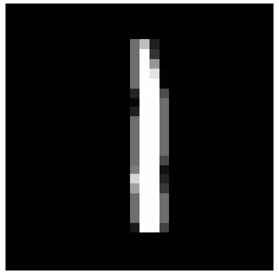
- Represent each object by a numerical vector

- 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 28×28 pixels.

It is a matrix/ 2D array $A \in \mathbb{R}^{28\times28}$

$$A=\begin{bmatrix} A_{0,0} & \cdots & A_{0,27} \\ \cdots & \cdots & \cdots \\ A_{27,0} & \cdots & A_{27,27} \end{bmatrix} \begin{array}{l} \text{row-0} \\ \\ \text{row-27} \end{array}$$

$$\boldsymbol{x} = \begin{bmatrix} A_{0,0} \\ A_{0,1} \\ A_{0,2} \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ \cdots \\ A_{27,27} \end{bmatrix}$$

the first row

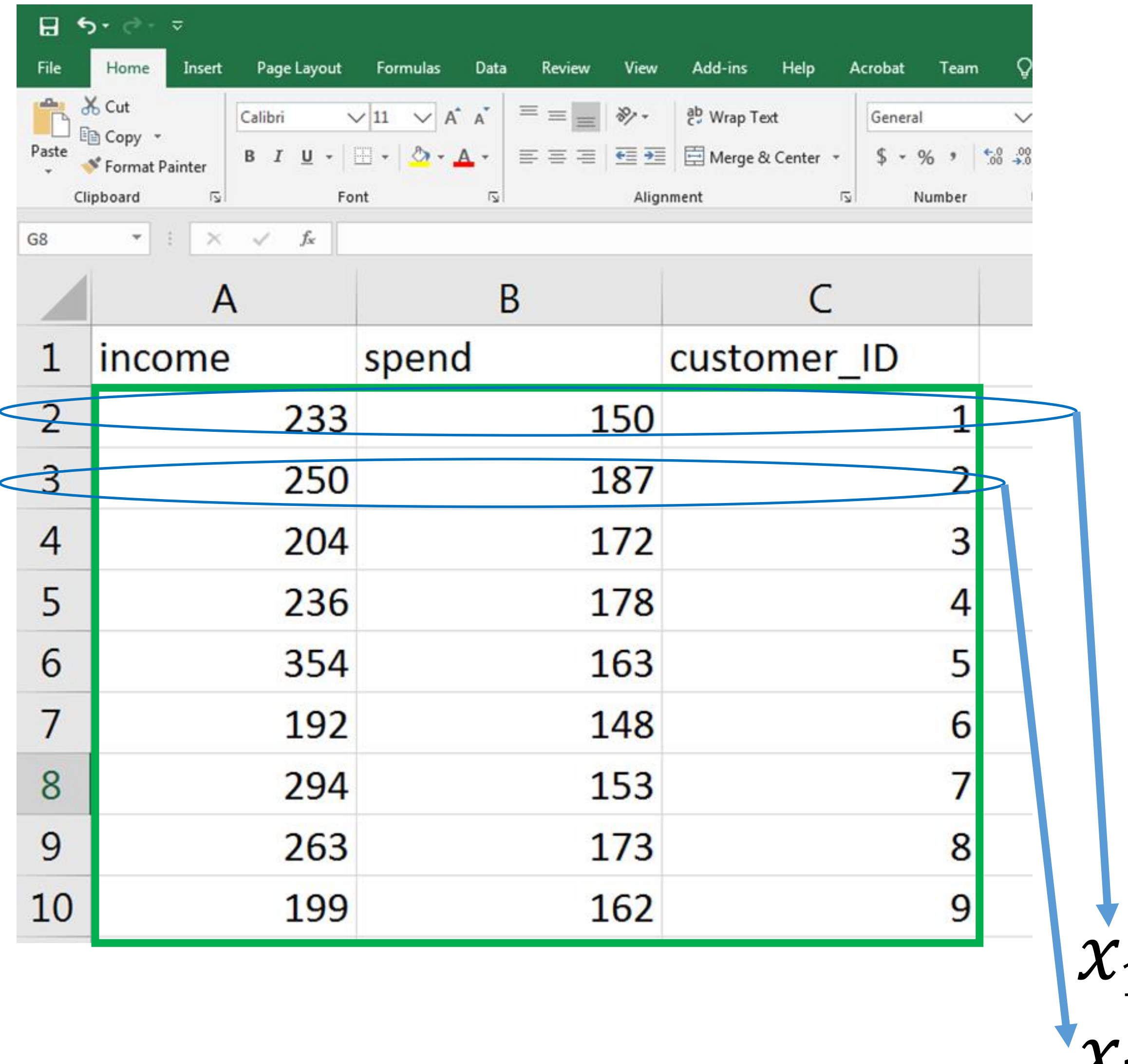
the second row

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

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 1st customer (1st row in the table)

x_2 : the 2nd customer (2nd row in the table)

Vector Norm and Distance Measure

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

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

It measures the "length" of the vector

$\left|x_{[1]}\right|$ is the absolute value of $x_{[1]}$

$$\sum_{m=1}^{M} \left|x_{[m]}\right|^p = \left|x_{[1]}\right|^p + \left|x_{[2]}\right|^p + \left|x_{[3]}\right|^p + \ldots + \left|x_{[M]}\right|^p$$

Vector Norm and Distance Measure

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

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

It measures the length of the vector

- ℓ_2 norm $\|\boldsymbol{x}\|_2 = \sqrt{\sum_{m=1}^{M} x_{[m]}^2} = \sqrt{\boldsymbol{x}^T \boldsymbol{x}}$ (Euclidean norm)

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

$$\boldsymbol{x}^T = [0.1, 1.2]$$

Vector Norm and Distance Measure

- two vectors/points $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^M$, and the norm of $\boldsymbol{x}$ is ℓ_p norm $||\boldsymbol{x}||_p$
- $\boldsymbol{x}$ is the feature vector of object-A
- $\boldsymbol{y}$ is the feature vector of object-B
- Then the distance between $\boldsymbol{x}$ and $\boldsymbol{y}$ is $||\boldsymbol{x} - \boldsymbol{y}||_p$
- ℓ_2 norm is used in k-means algorithm to measure distance

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

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

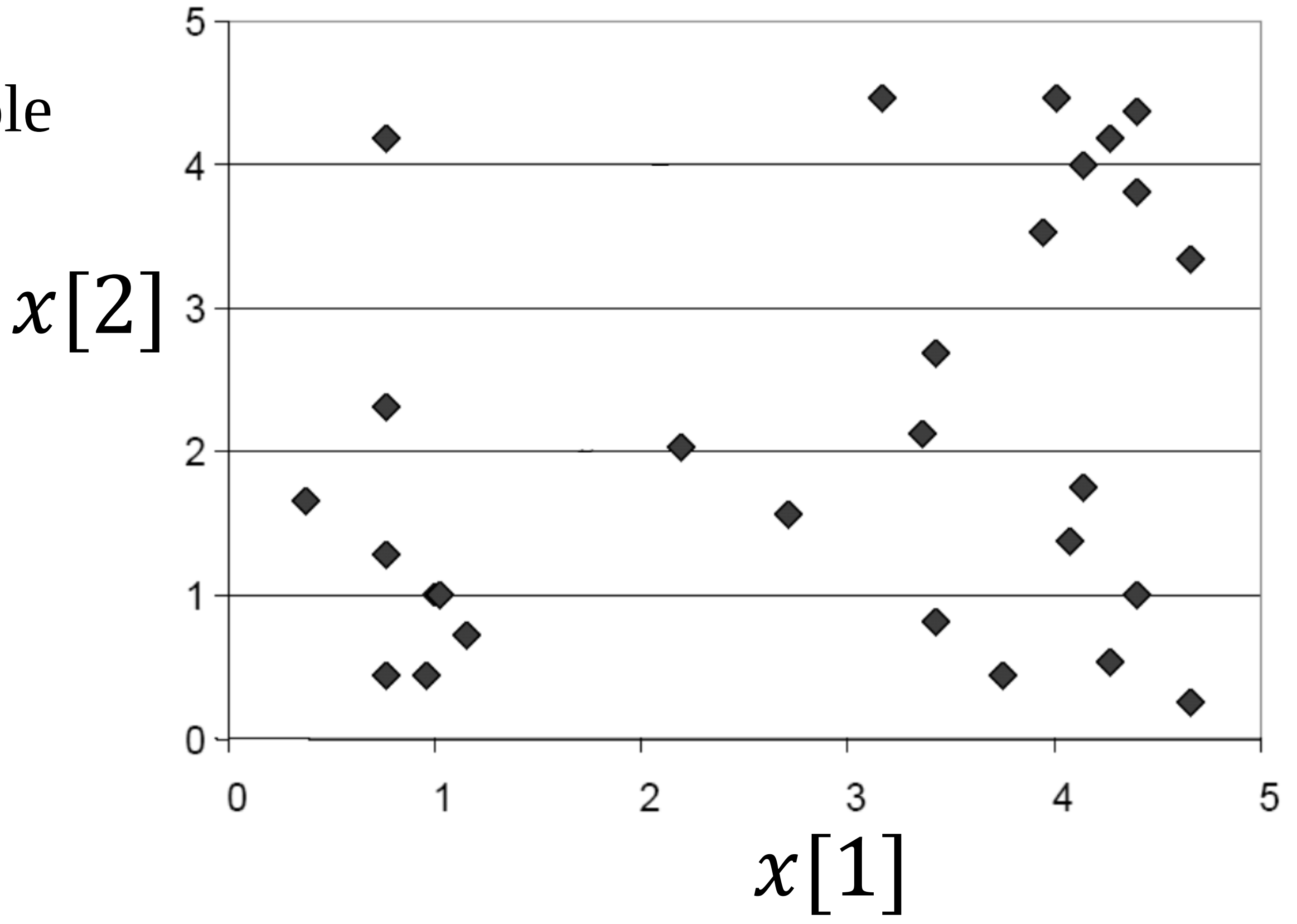
Run kmeans_cust_seg.ipynb

Before clustering, a dataset of vectors/samples

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

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

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



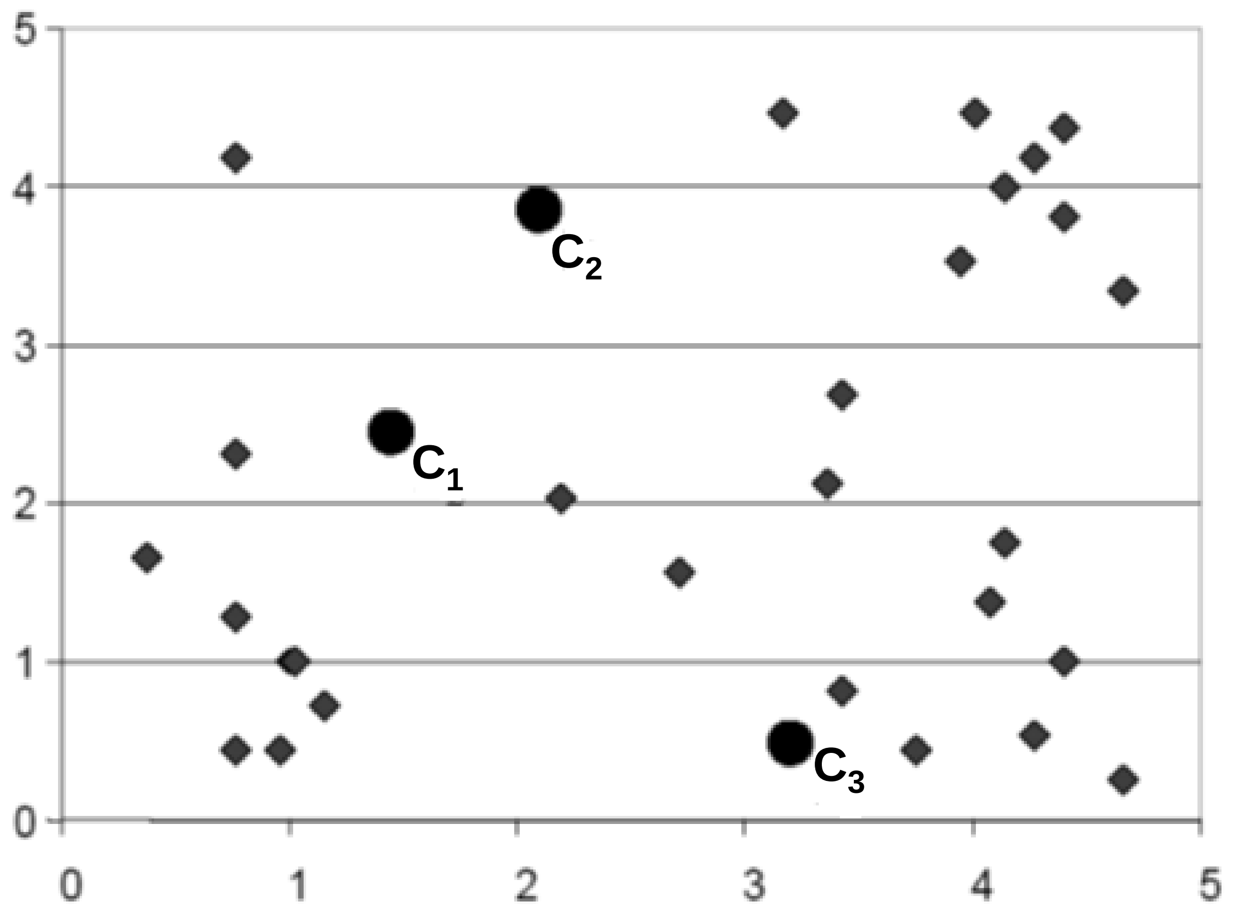
Apply k-means algorithm: Initialization

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

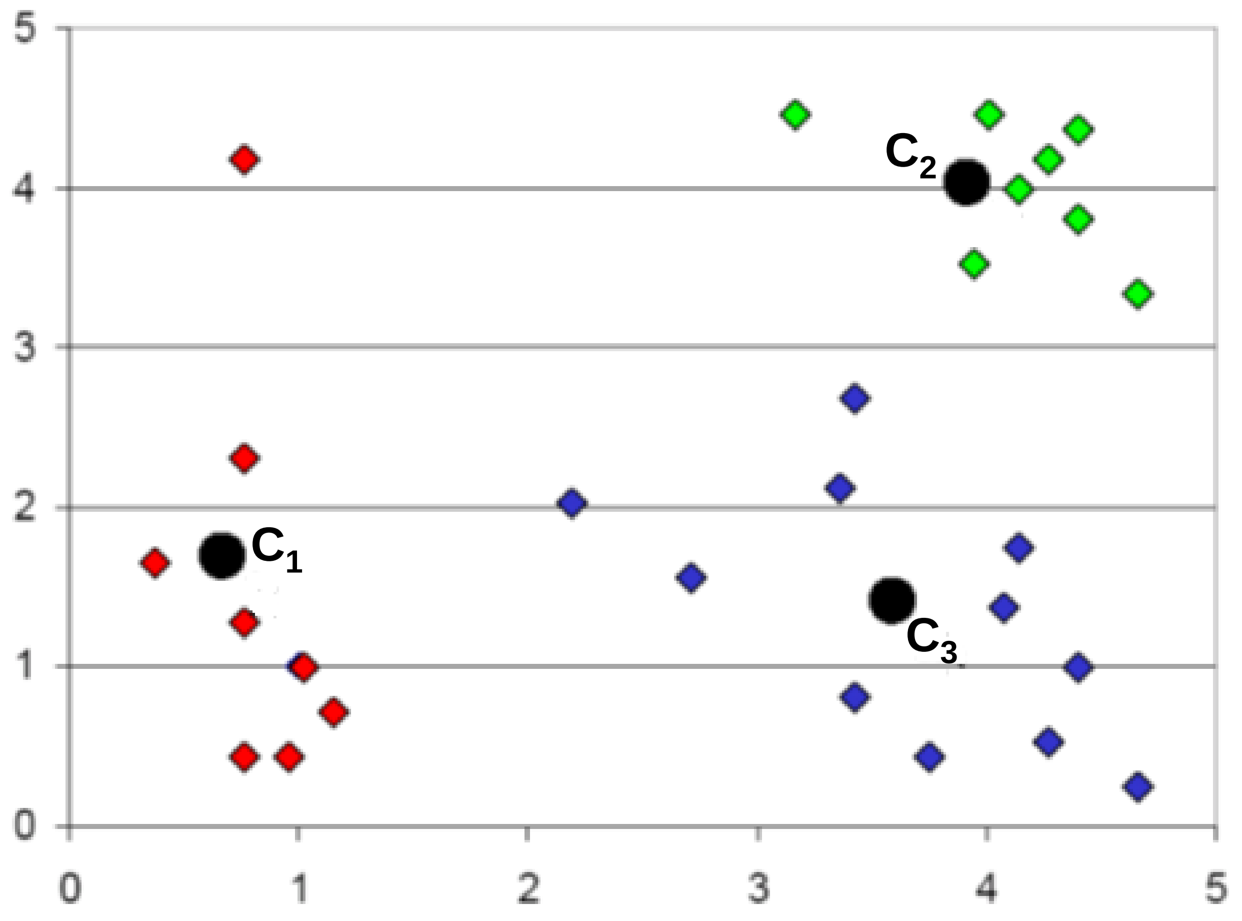
After k-means clustering, clusters/groups are formed

After k-means clustering:

- The data points are assigned to the three clusters
  red-cluster
  green-cluster
  blue-cluster

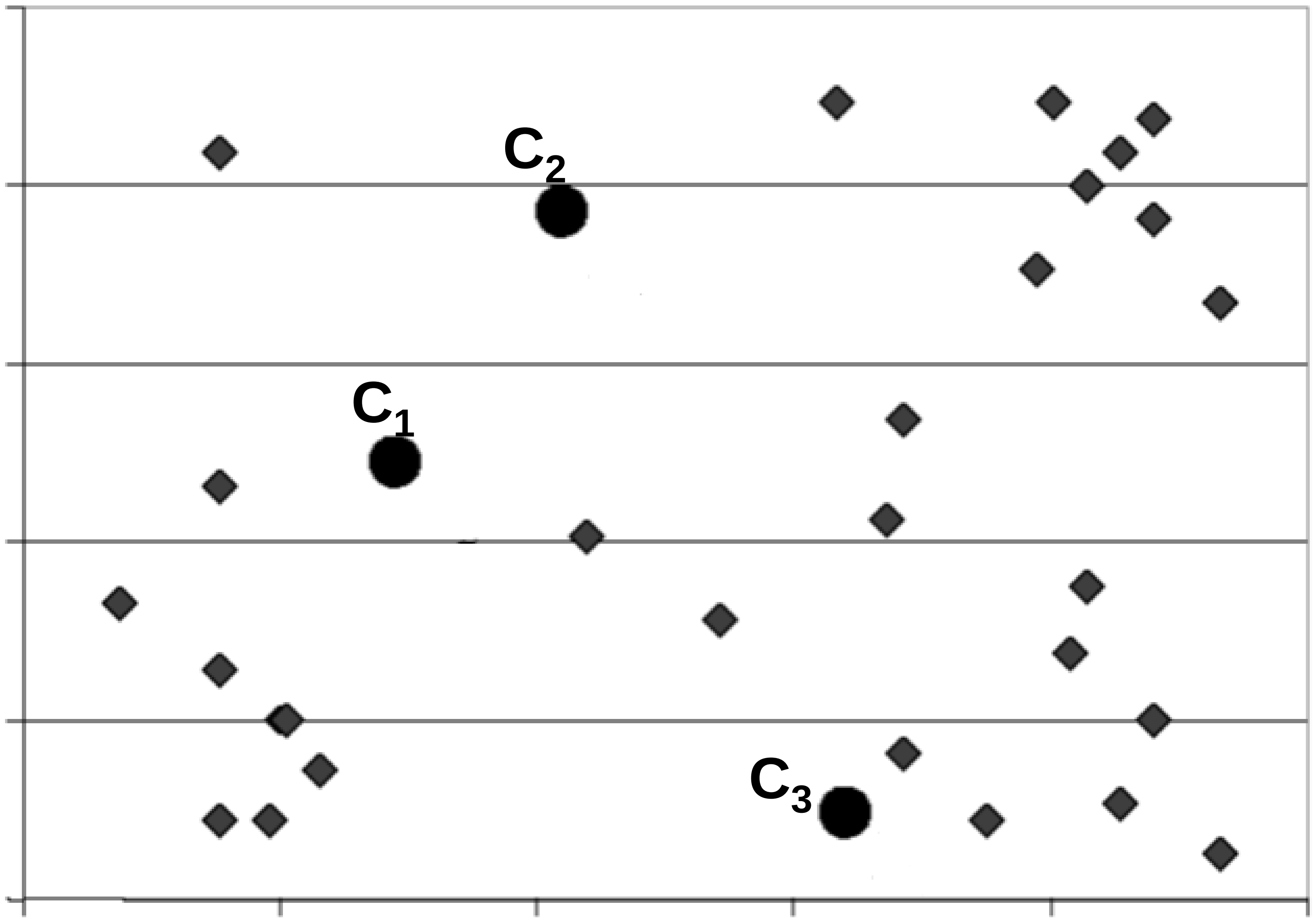
- Every data point has a cluster label that could be 1, 2, or 3

- The final cluster centers are different from the initial centers

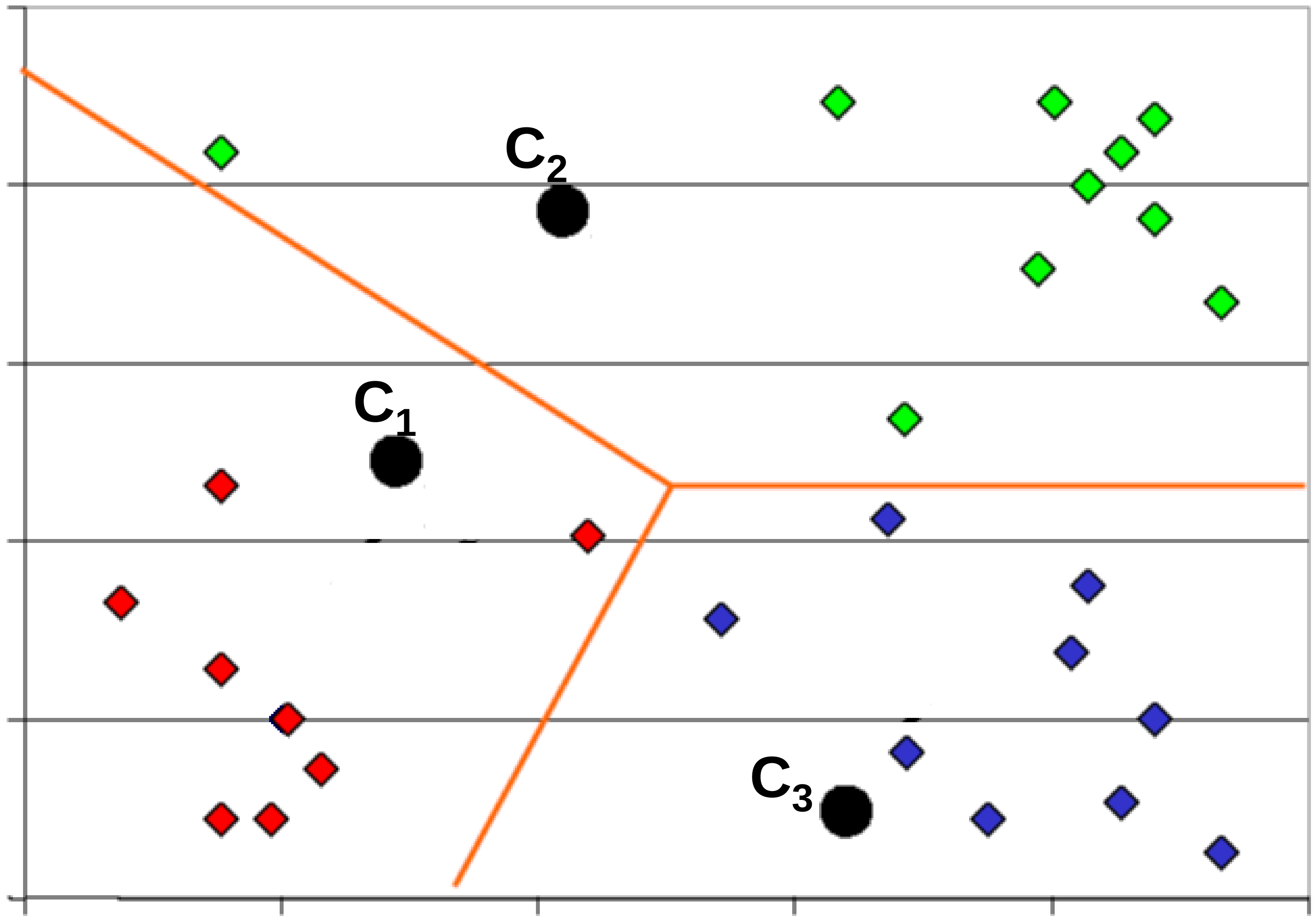


c_1, c_2, c_3 are the cluster centers

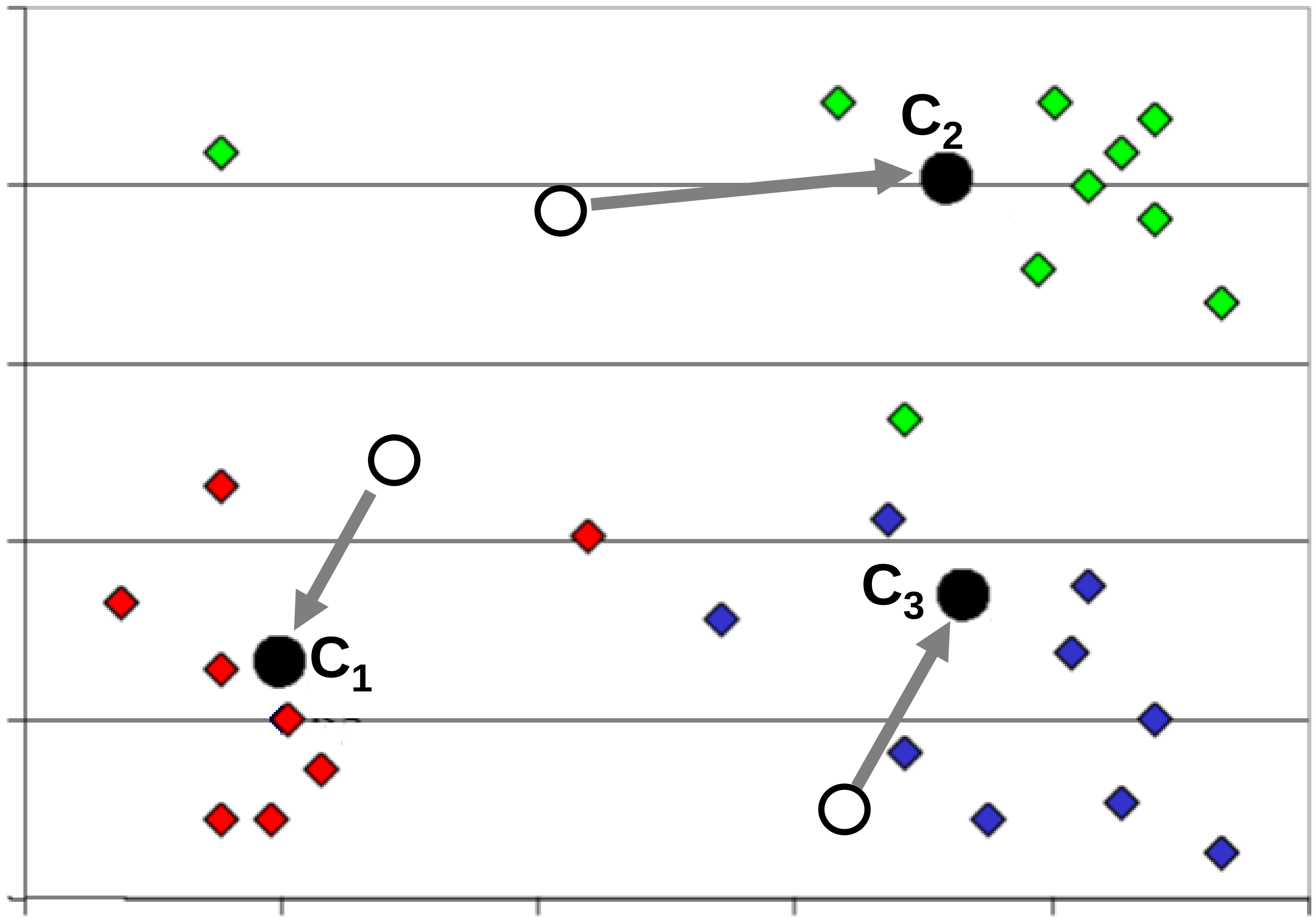
Initialization: the number of clusters and random locations of the cluster centers



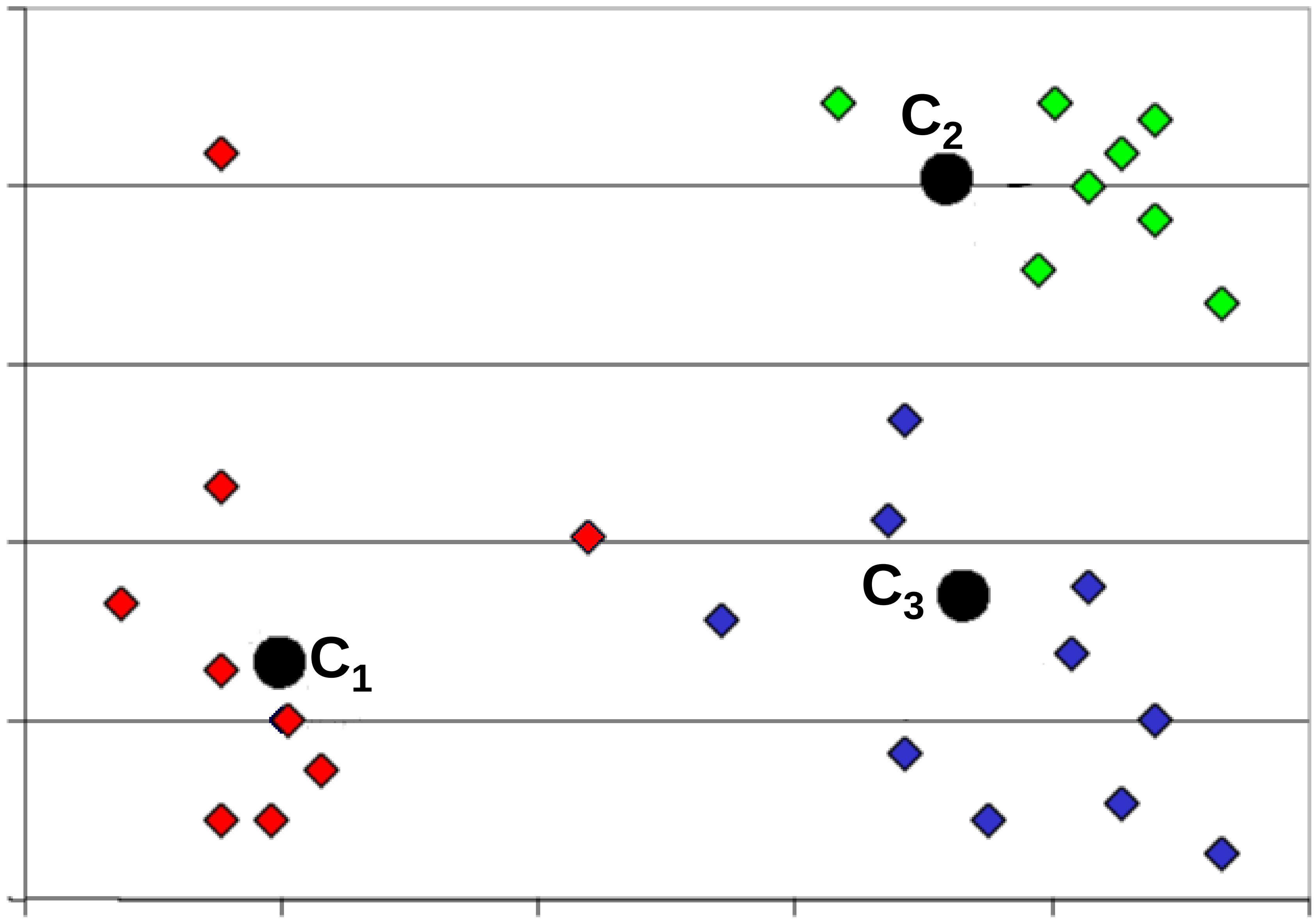
Assign Labels: assign each data point to the nearest cluster center



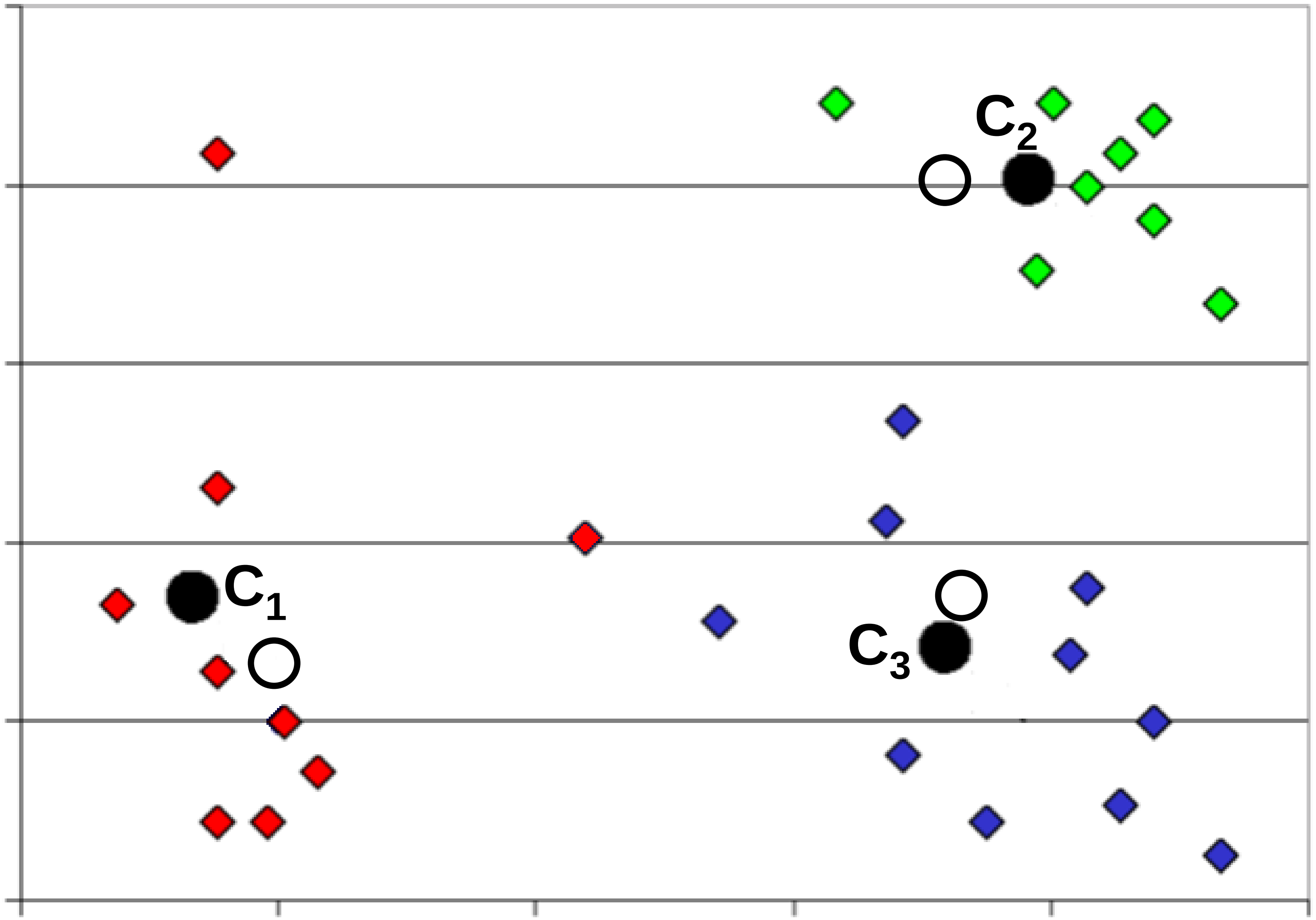
Update Centers: re-compute the center of each cluster



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:

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

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

Clustering is difficult in general

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

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

$\alpha(n) = k$ 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 $\{\alpha(1), \alpha(2), \ldots, \alpha(N)\}$

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*

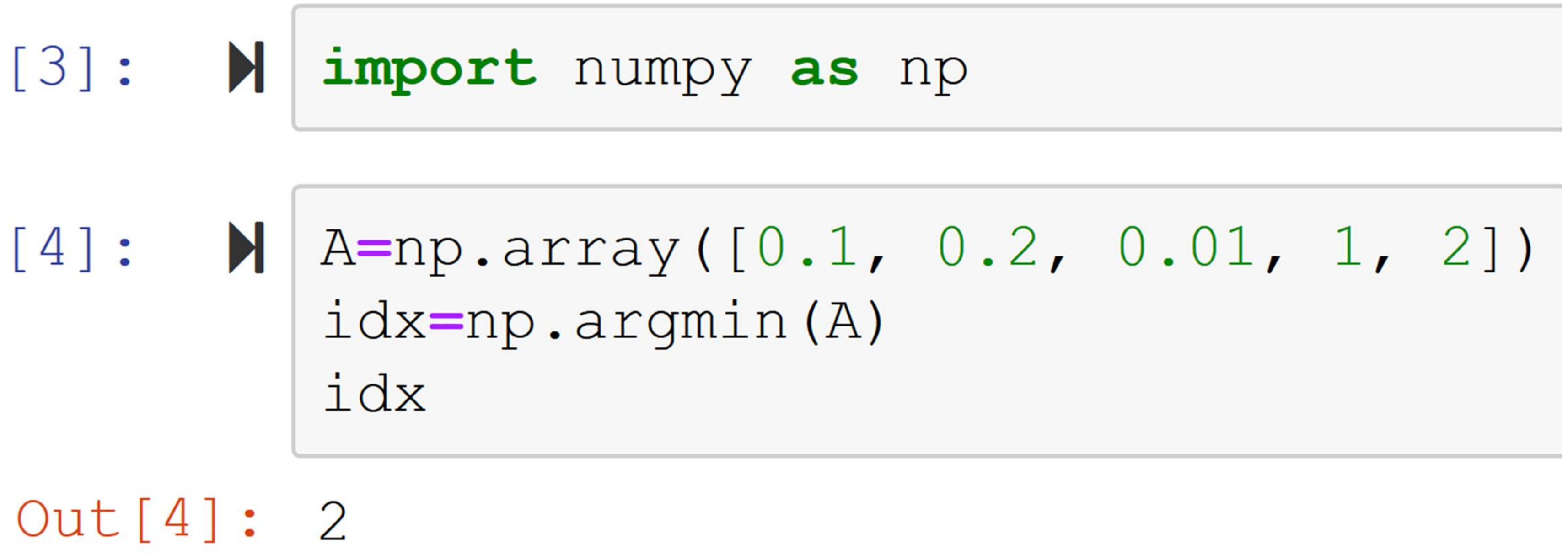
$$c_k = \frac{1}{N_k} \sum_{n:\, \alpha(n)=k} x_n$$

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

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

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

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



```
[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
```

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

the k-means algorithm

- Loss: $L = \frac{1}{N}\sum_{n=1}^{N} \left\| x_n - c_{\alpha(n)} \right\|^2$
- 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 $\begin{bmatrix} \alpha_{(1,1)} & \cdots & \alpha_{(1,K)} \\ \cdots & \cdots & \cdots \\ \alpha_{(N,1)} & \cdots & \alpha_{(N,K)} \end{bmatrix}$, $\alpha_{(n,k)} = 0 \; or \; 1$

$\alpha_{(n,k)} = 1$ 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

an assignment matrix

Data Point	*Cluster 1*	*Cluster 2*	*Cluster 3*
x_1	$\alpha_{(1,1)} = 1$	$\alpha_{(1,2)} = 0$	$\alpha_{(1,3)} = 0$
x_2	$\alpha_{(2,1)} = 0$	$\alpha_{(2,2)} = 1$	$\alpha_{(2,3)} = 0$
x_3	$\alpha_{(3,1)} = 0$	$\alpha_{(3,2)} = 1$	$\alpha_{(3,3)} = 0$

x_1 is assigned to cluster 1
x_2 and x_3 are assigned to cluster 2

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

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

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

$L = \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} \alpha_{(n,k)} \|x_n - c_k\|^2$, k-means uses ℓ_2 norm

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

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

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

$L = \frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K}\alpha_{(n,k)}\|x_n - c_k\|^2$, k-means uses ℓ_2 norm

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

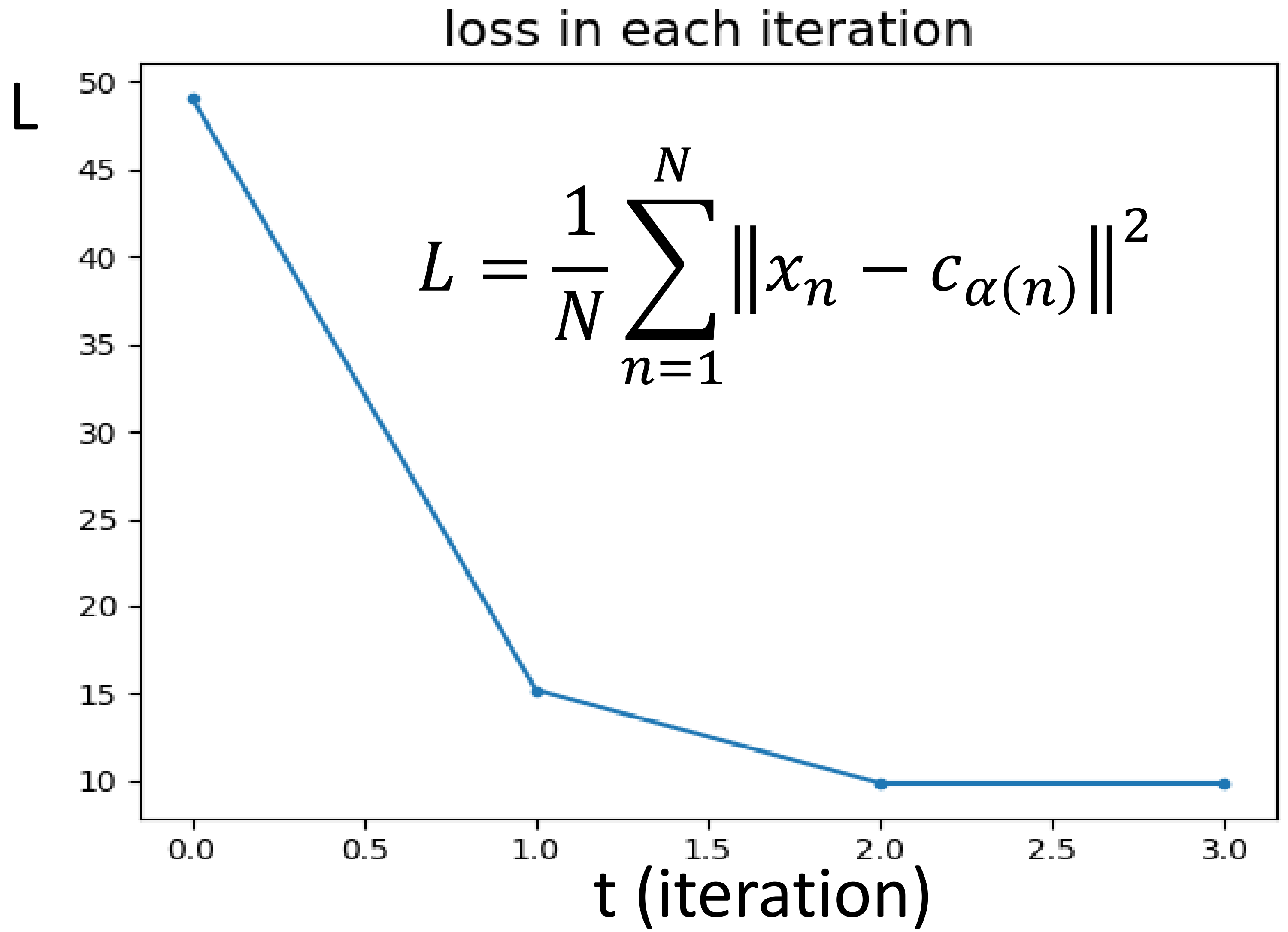
$\frac{\partial L}{\partial c_k} = \frac{1}{N}\sum_{n=1}^{N} 2\alpha_{(n,k)}(c_k - x_n)$

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

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

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

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



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

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

- $loss: L = \frac{1}{N}\sum_{n=1}^{N}\left\|x_n - c_{\alpha(n)}\right\|^2$
- 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: $\alpha(n) = arg \min_{k\in\{1,\ldots,K\}} \|x_n - c_k\|^2$
  - update centers: $c_k = arg \min_{c} \sum_{n:\,\alpha(n)=k} \|x_n - c\|^2$

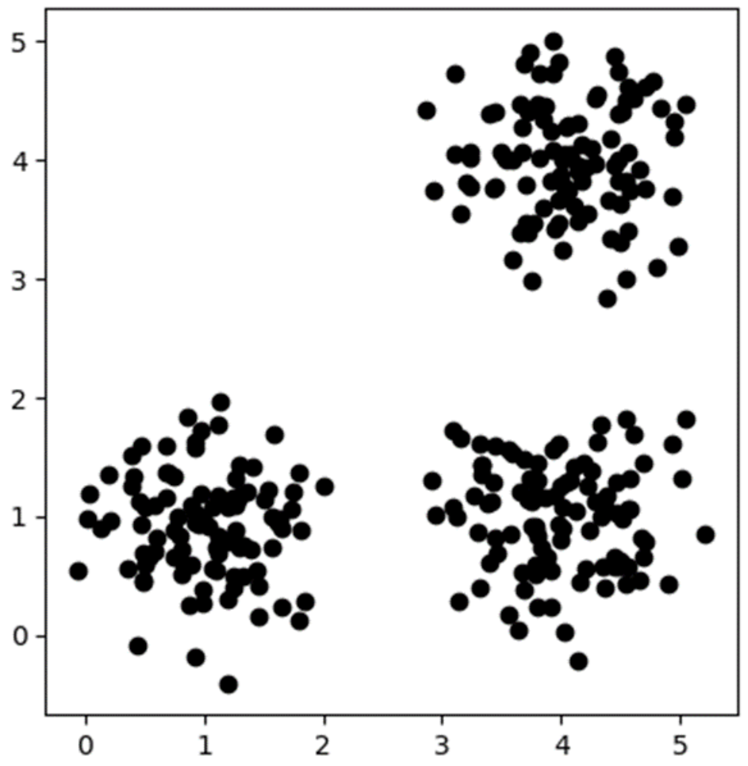
Question:

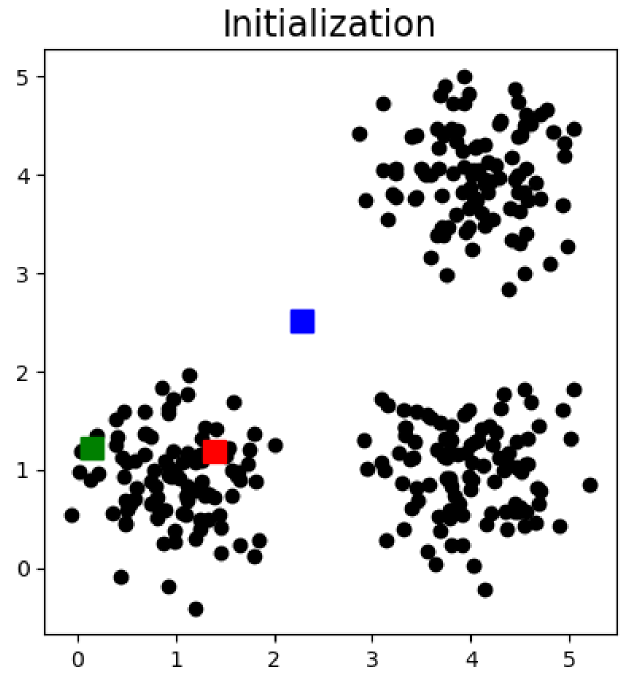
- Will different initialization lead to different clustering results?

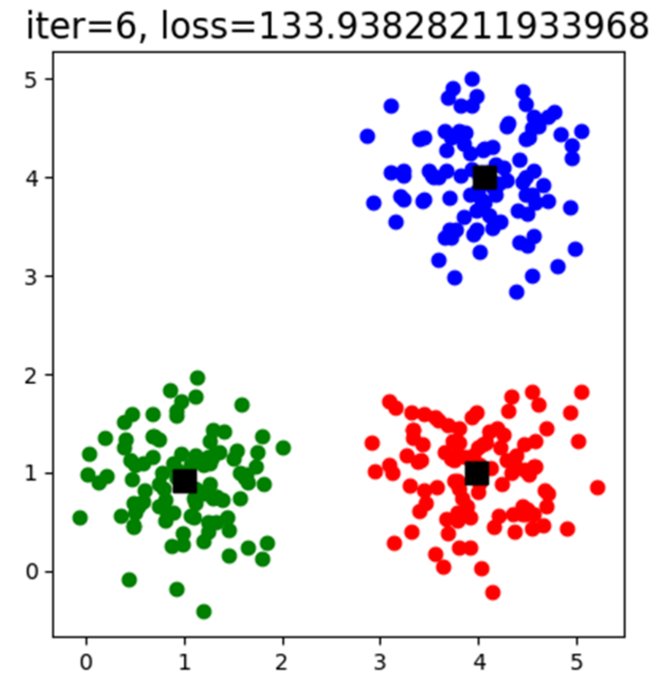
Yes ? No ? Maybe ?

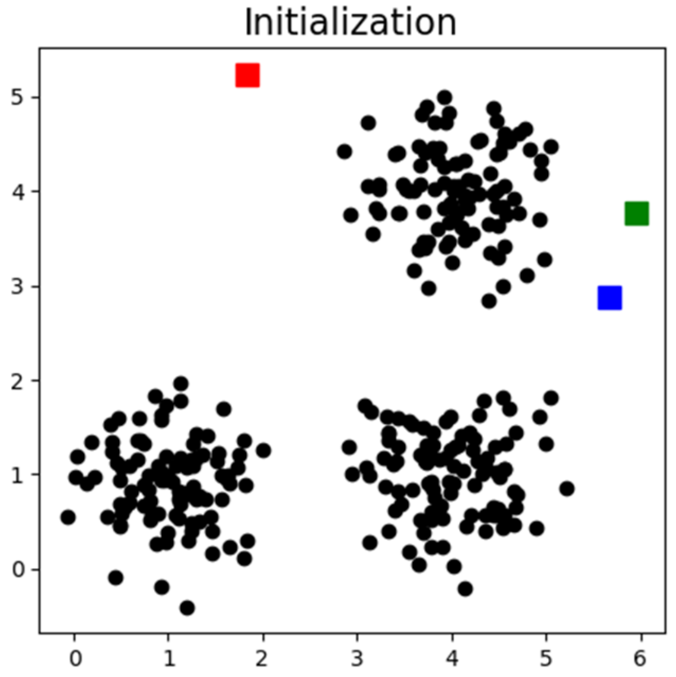
Run kmeans_raw.ipynb

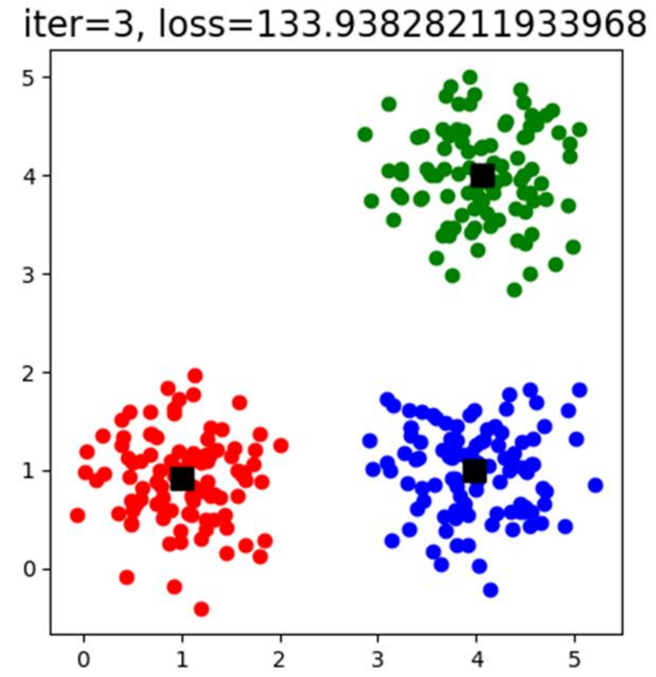
Centers/centroids are randomly initialized



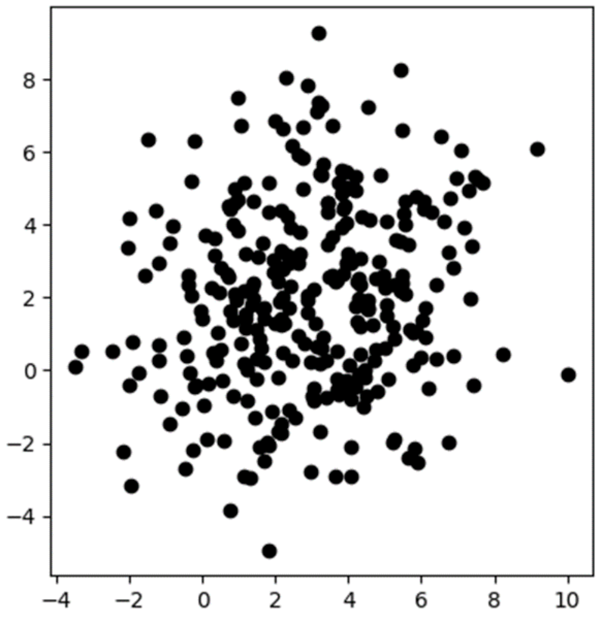


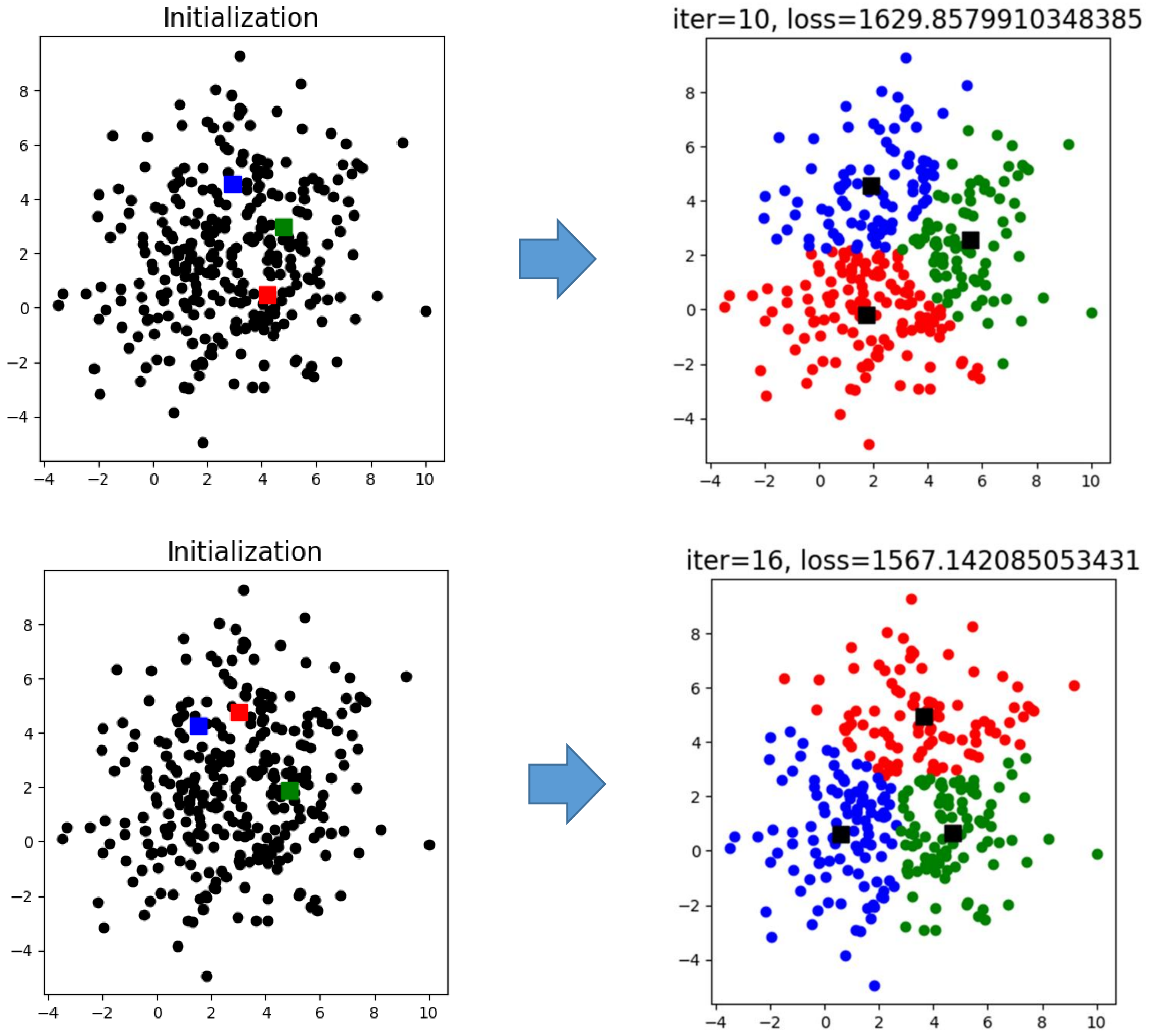




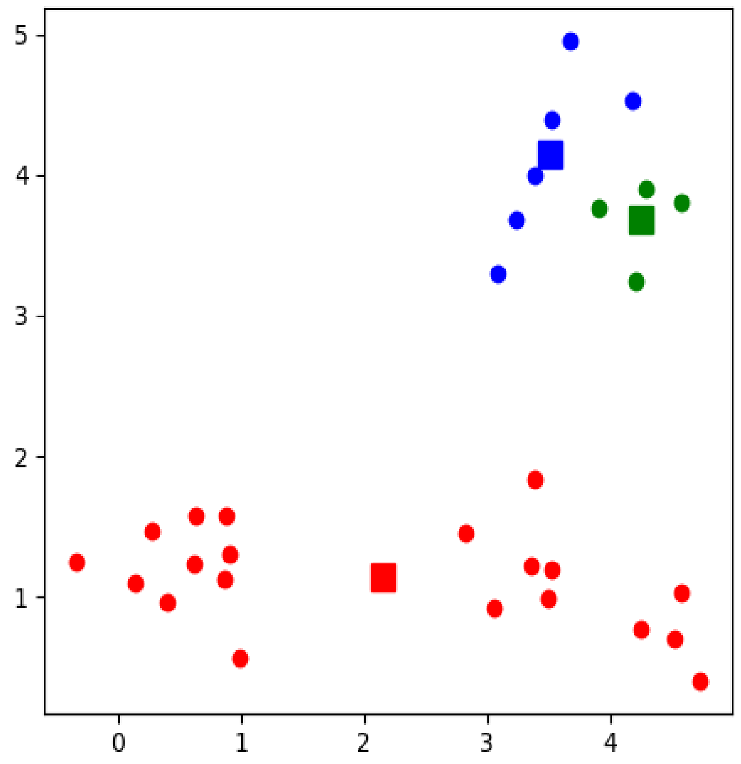


Centers are randomly initialized

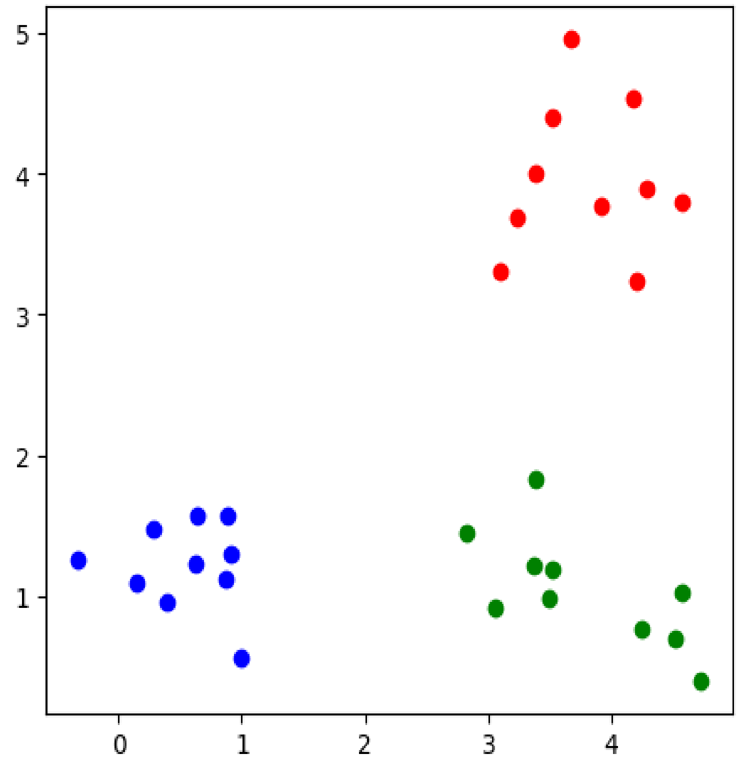




a bad result from k-means



the best clustering result



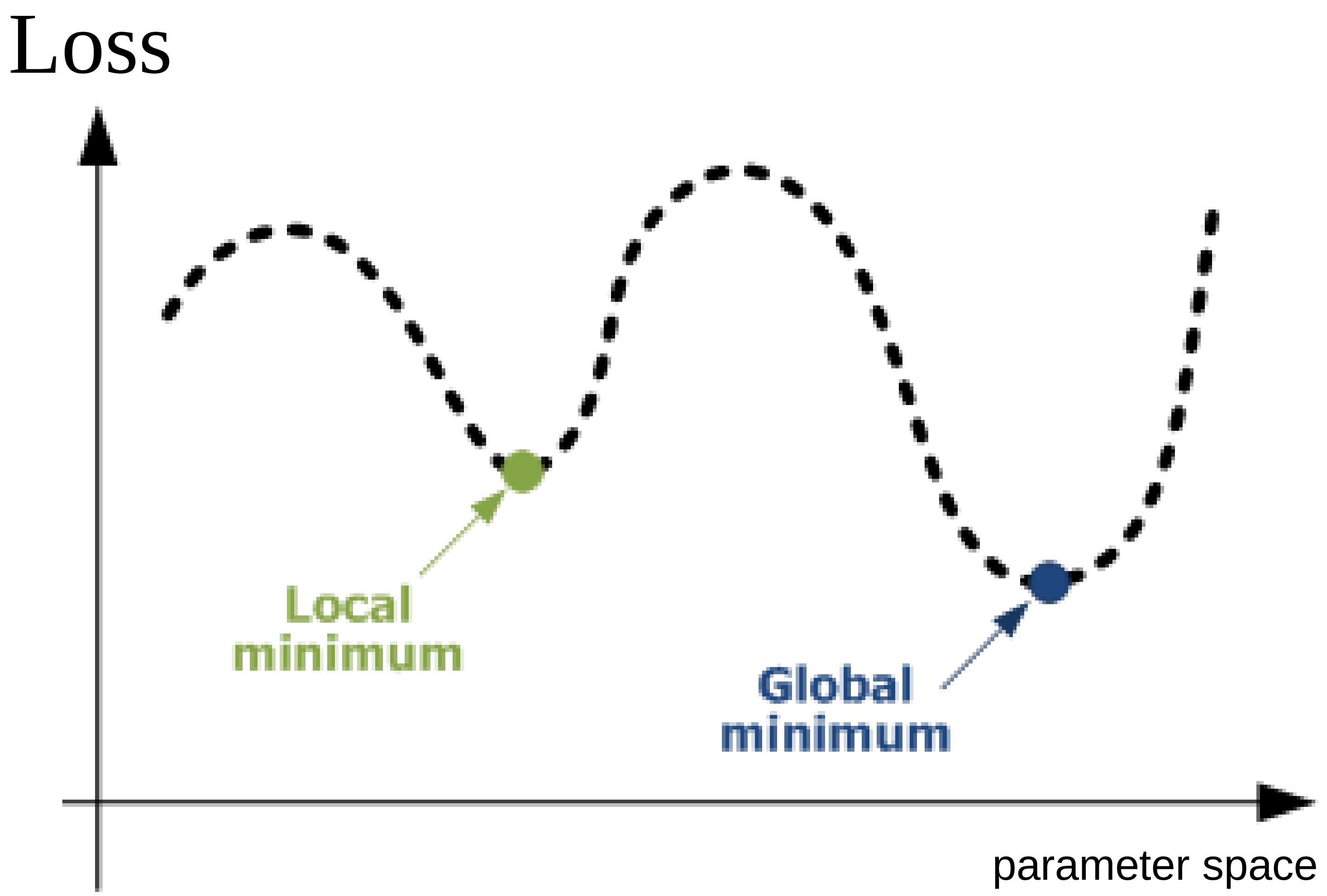
Clustering result of k-means algorithm could be random

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

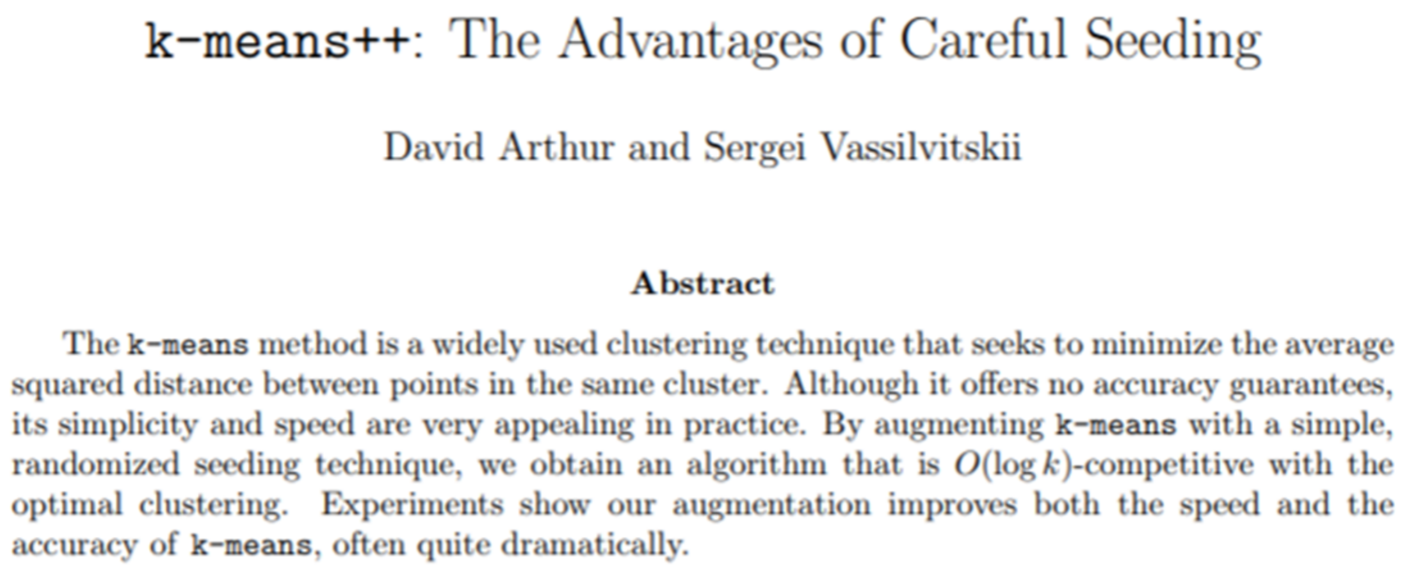
Initialization of the centers is random.

Different initializations could lead to different clustering results.

k-means may just find a local optimal solution



Better Initialization for k-means?



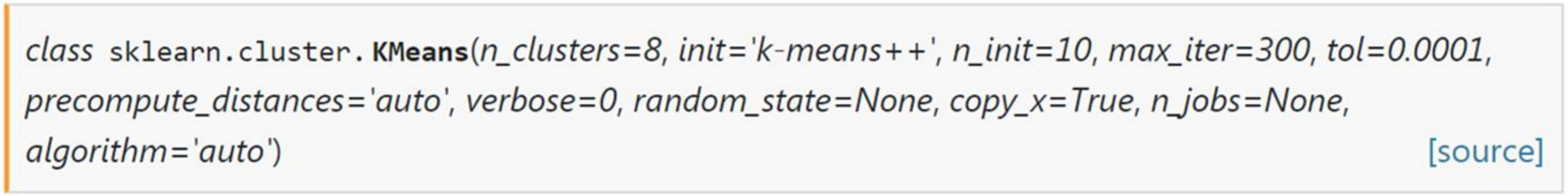
k-means++: The Advantages of Careful Seeding

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.

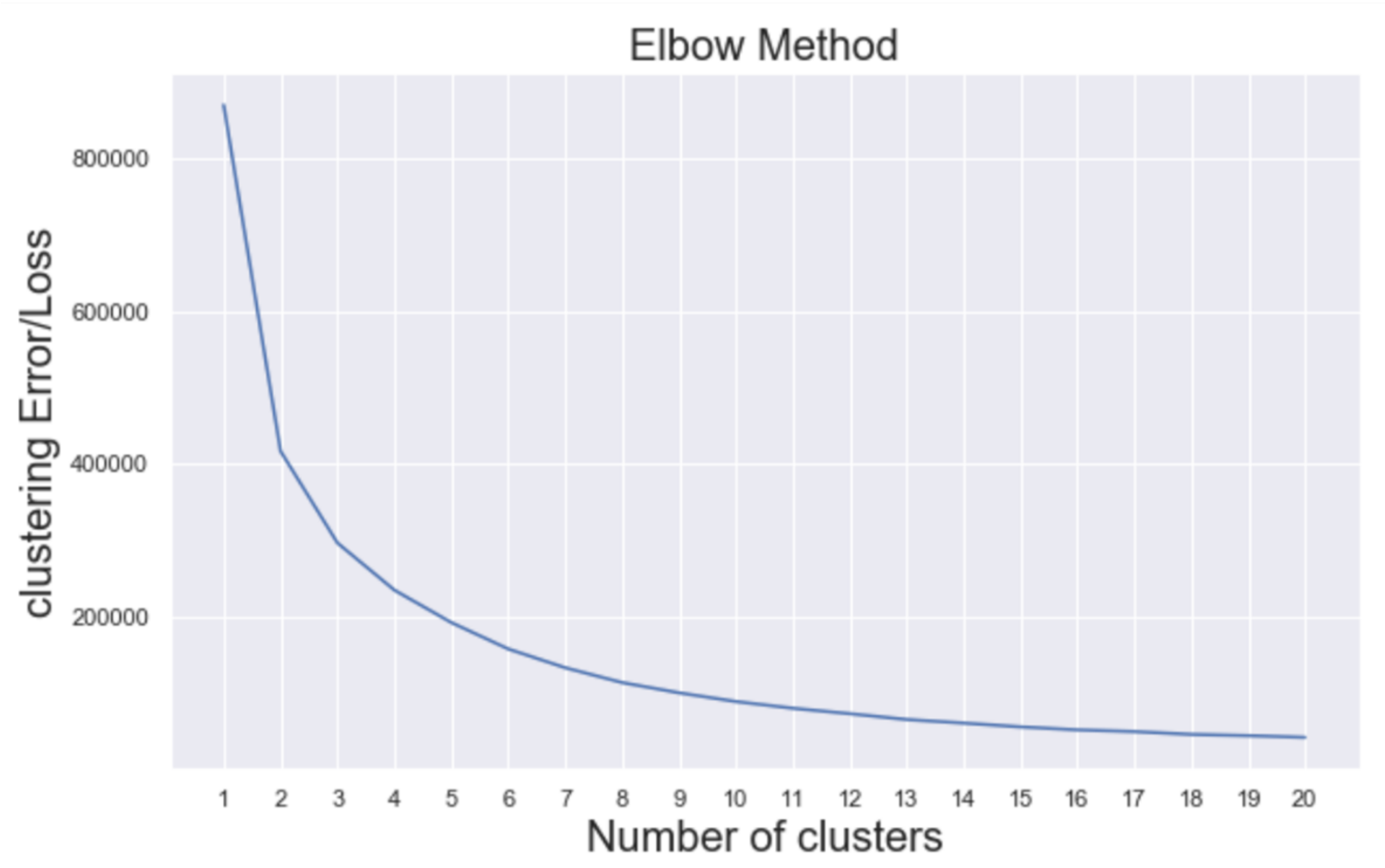
k-means++ use a special algorithm to initialize centers from the data points.
The initial centers will be far away to each other
It helps to find the best solution



```
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, …. 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

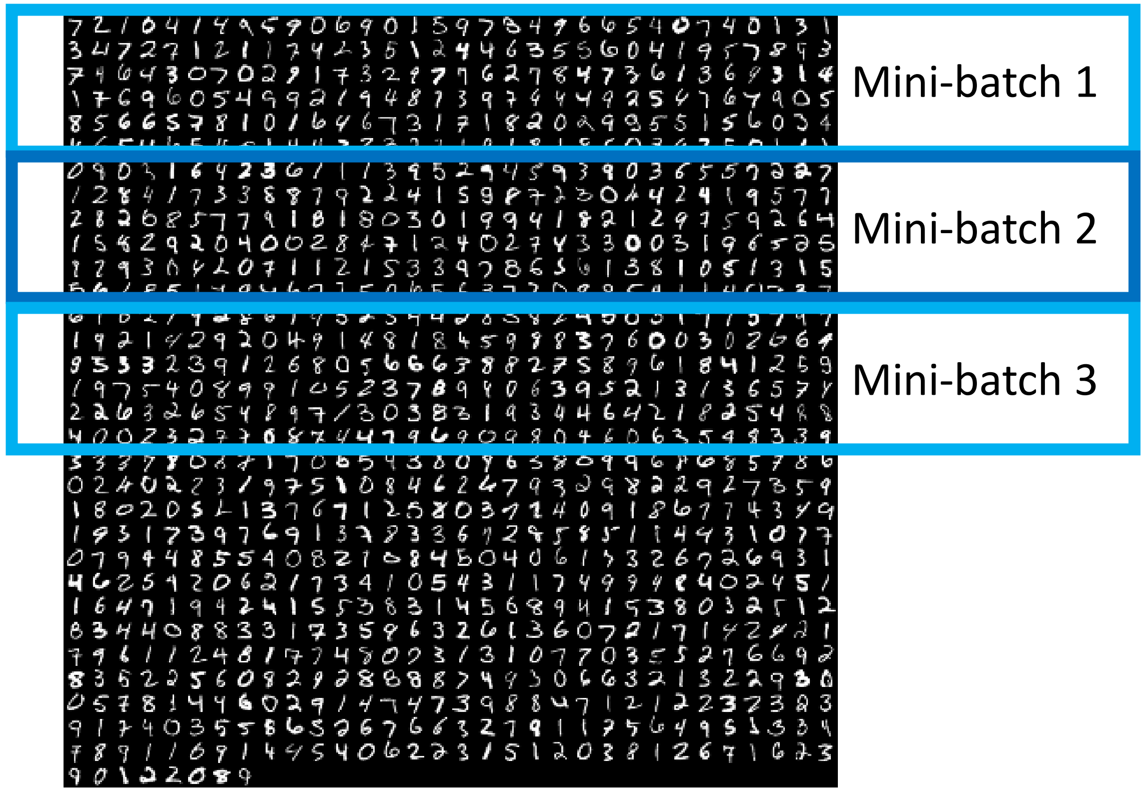
Faster ?

sklearn.cluster.MiniBatchKMeans

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

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

- Mini-batch k-means



from the previous mini-batch

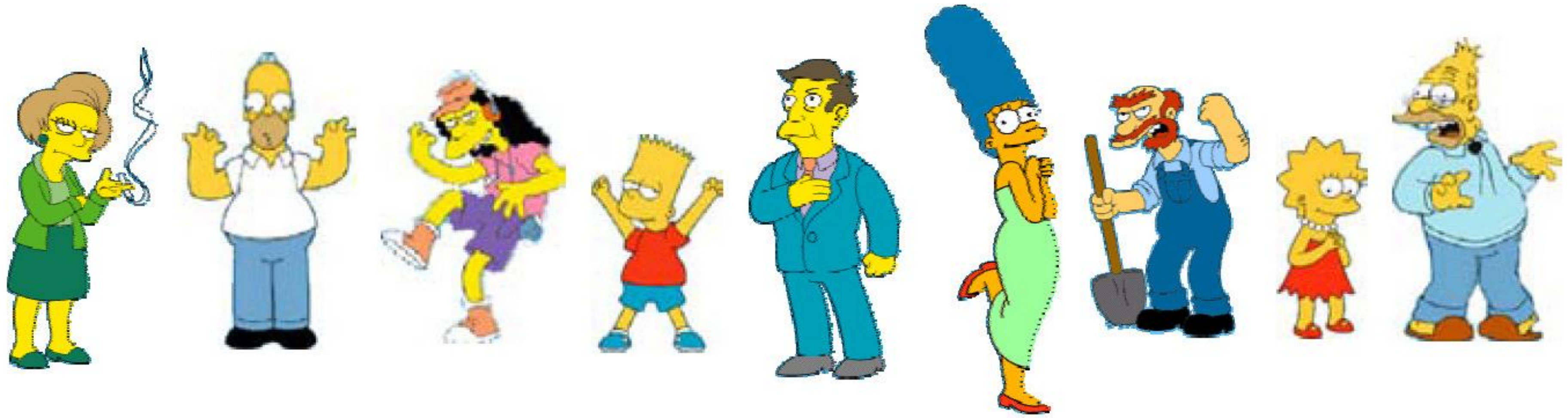
new center

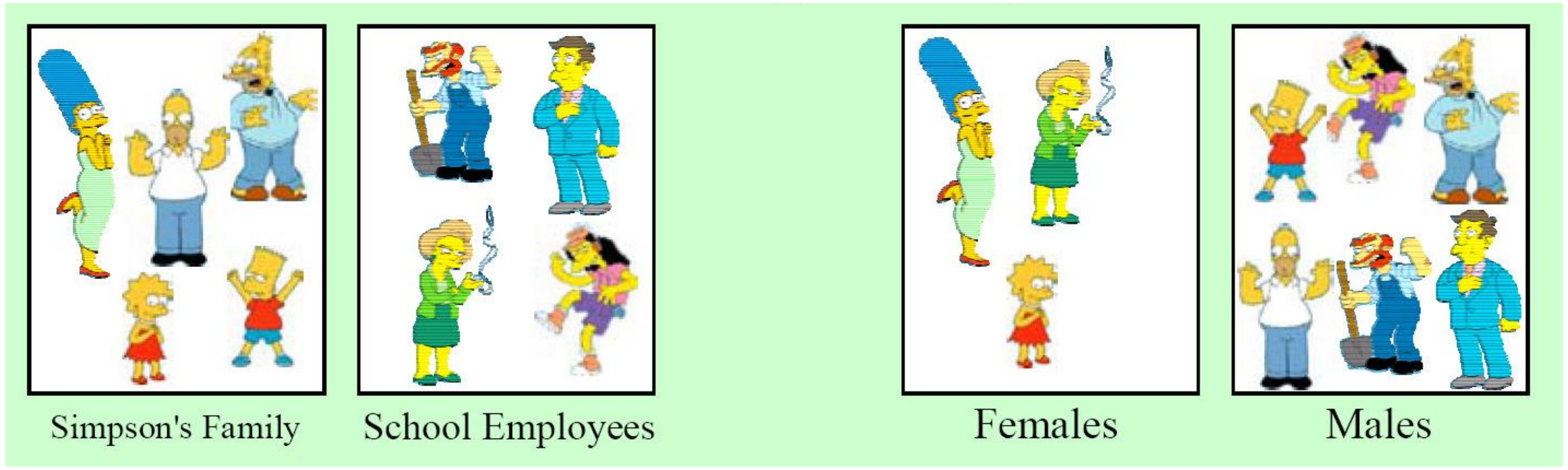
$$\mathbf{c} \leftarrow (1-\eta)\mathbf{c} + \eta\mathbf{x}$$

learning rate

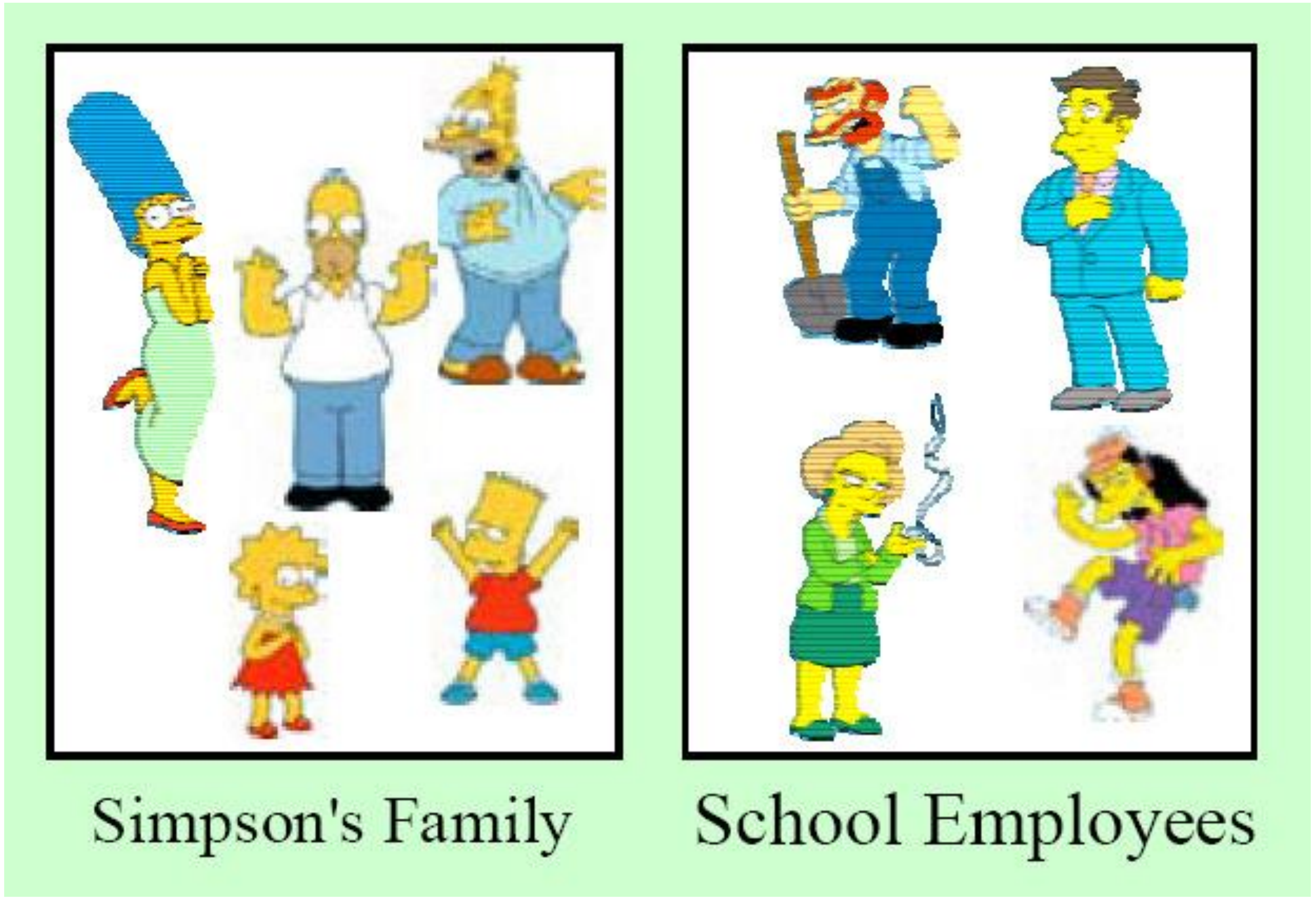
c is the center nearest to x

Clustering is based on distance measure and feature vector





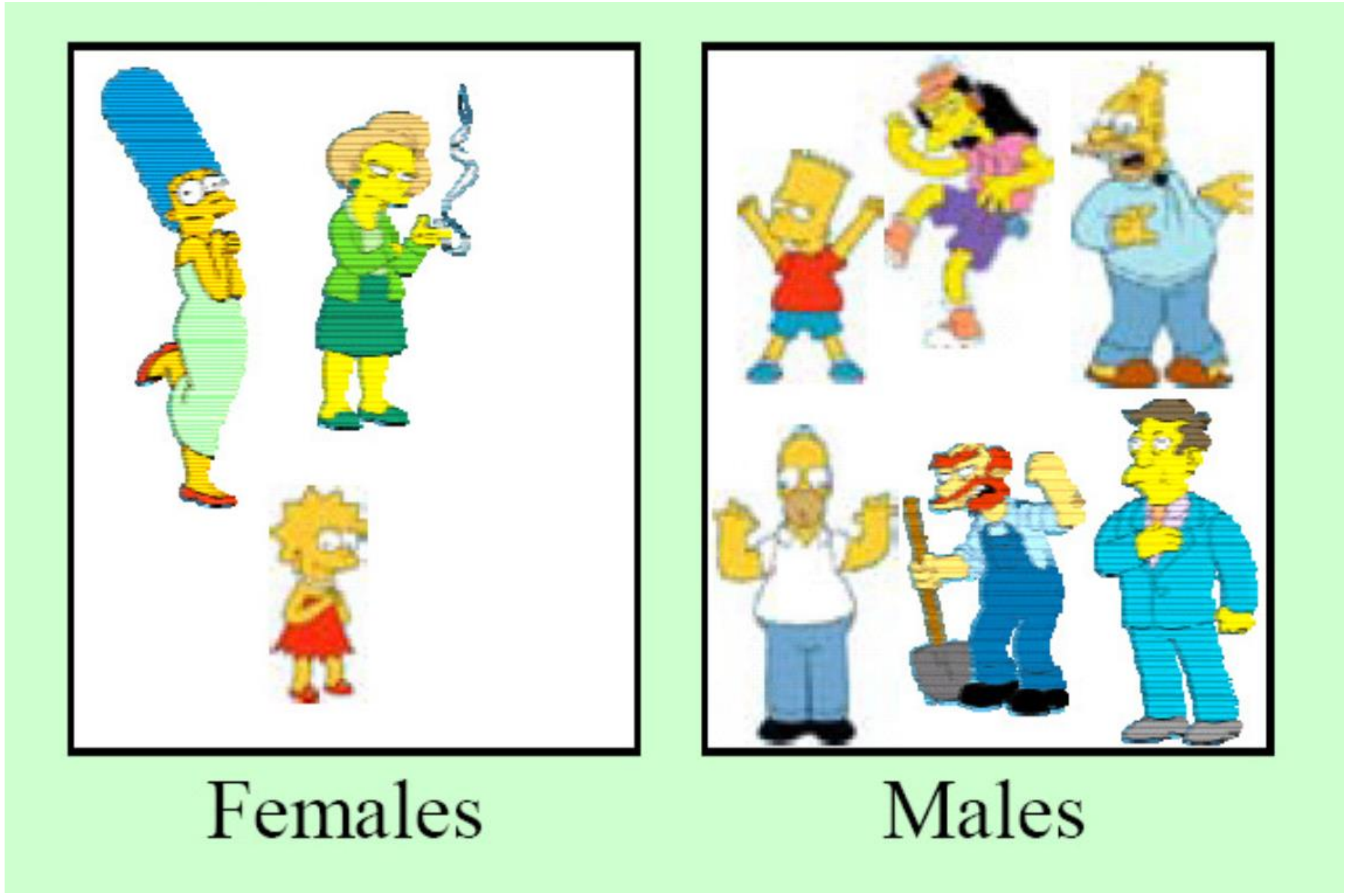
Clustering is based on distance measure and feature vector



Feature Vector

$$\boldsymbol{x} = [last\_name]$$

Clustering is based on distance measure and feature vector



Feature Vector

$$\boldsymbol{x} = [gender]$$

many distance/dissimilarity measures

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

So what is clustering in general?

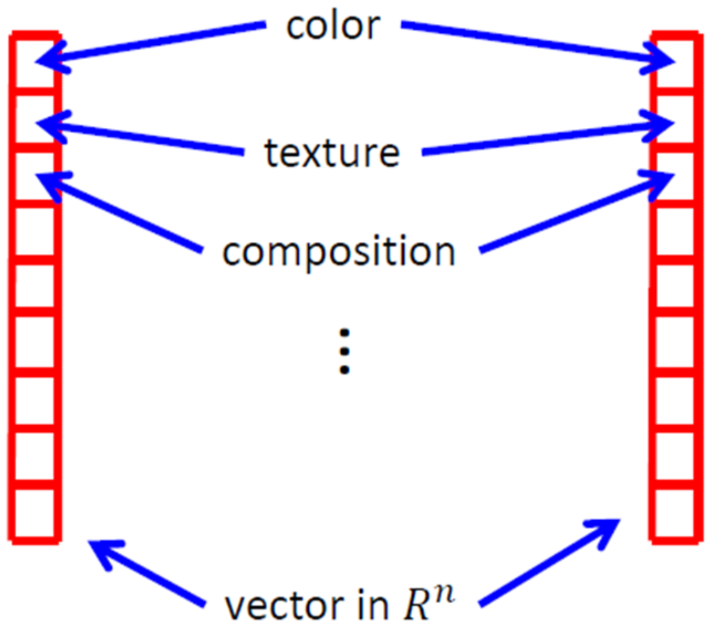
- You choose a distance/dissimilarity function
- The algorithm figures out the grouping of objects based on the distance function: distance(vectorA, vectorB)
- Data points within a cluster are similar
- Data points across clusters are not so similar

Feature Extraction Before Clustering

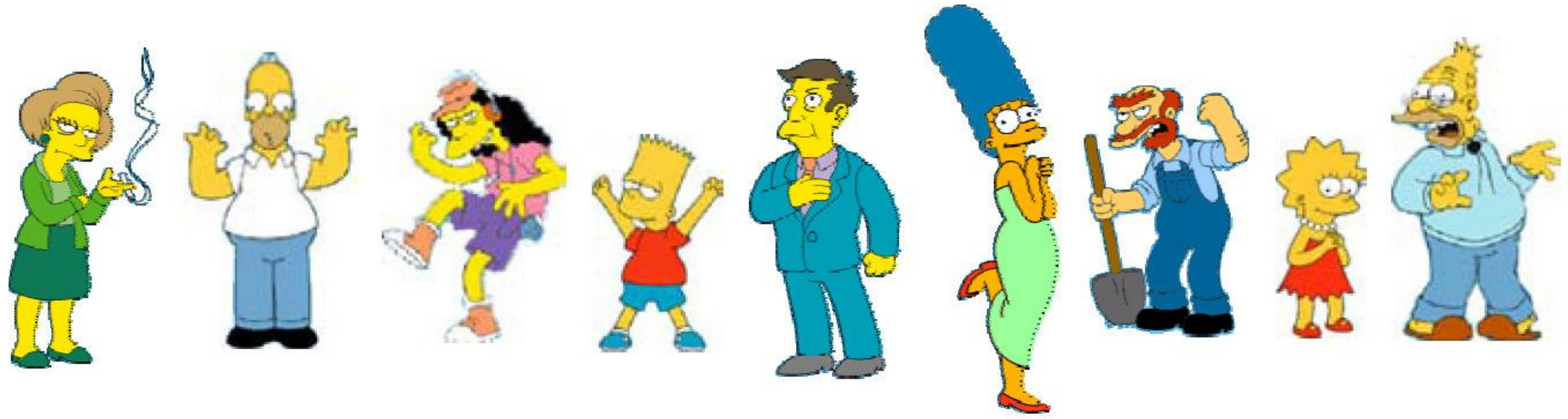
- Images of different sizes

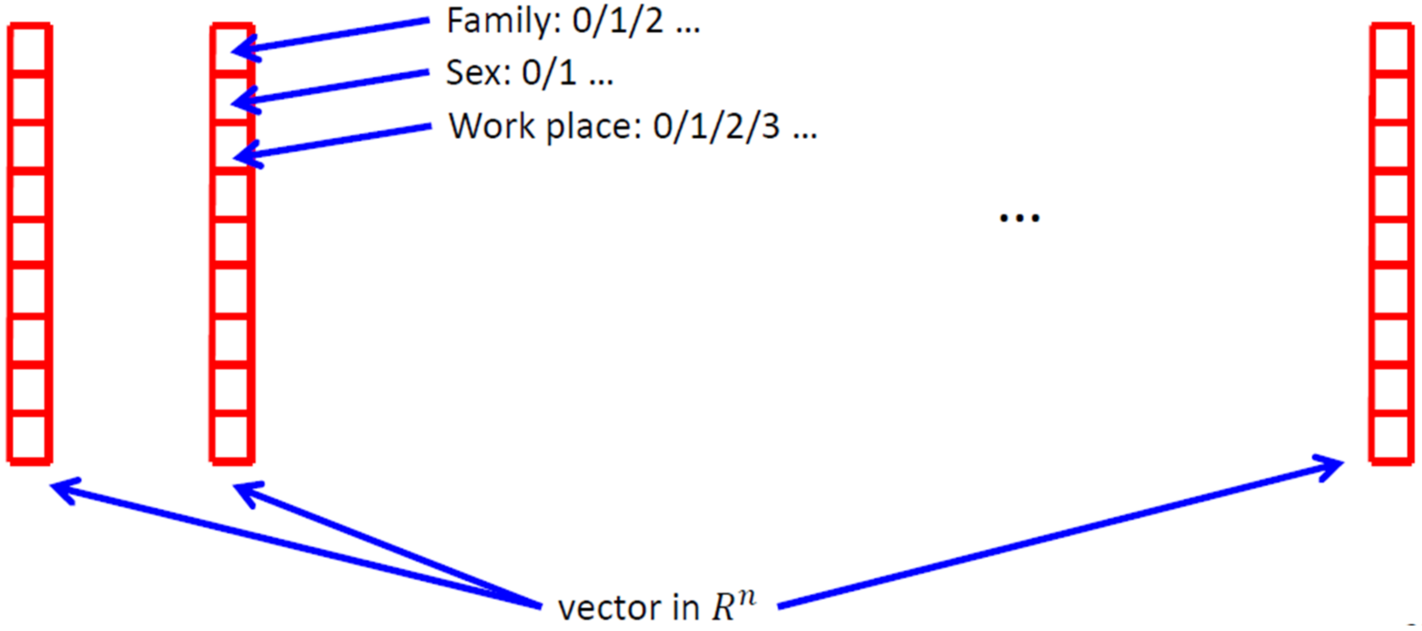
Can not directly compare the two images
because they have different number of pixels
resize the images, or extract some features

small image

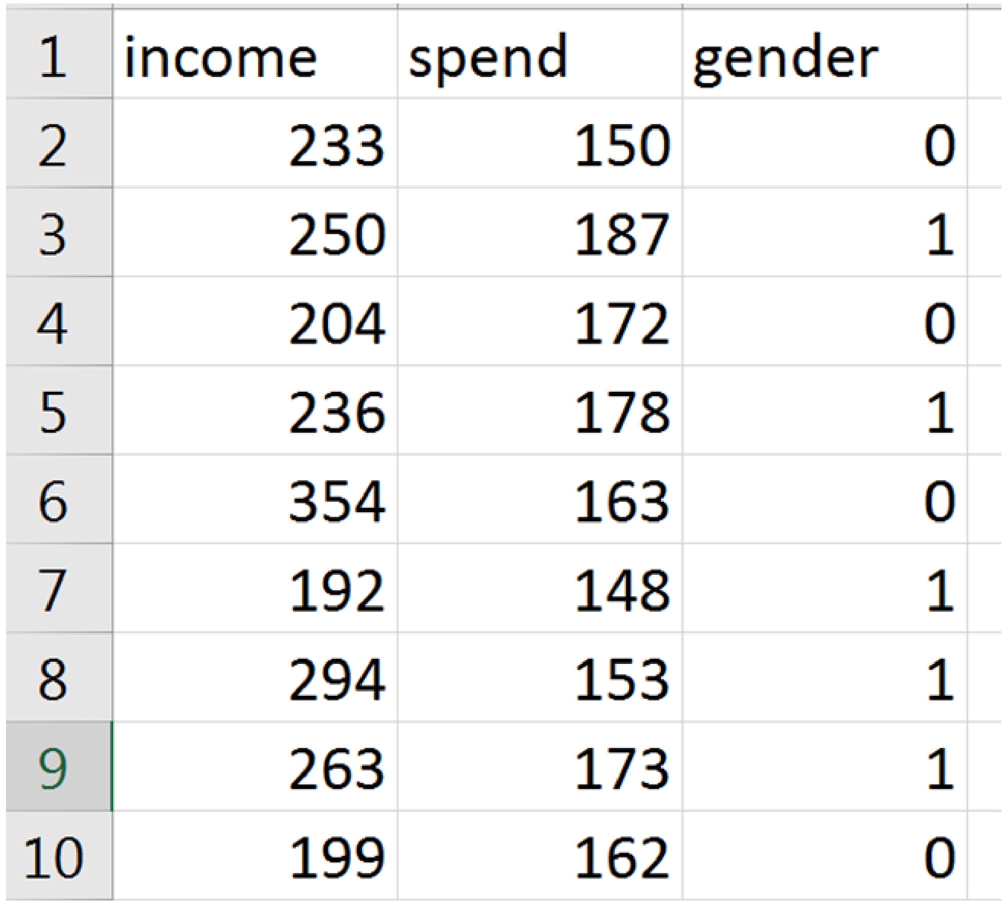


Objects in real life





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

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

the distance is $||\boldsymbol{x} - \boldsymbol{y}||_2$

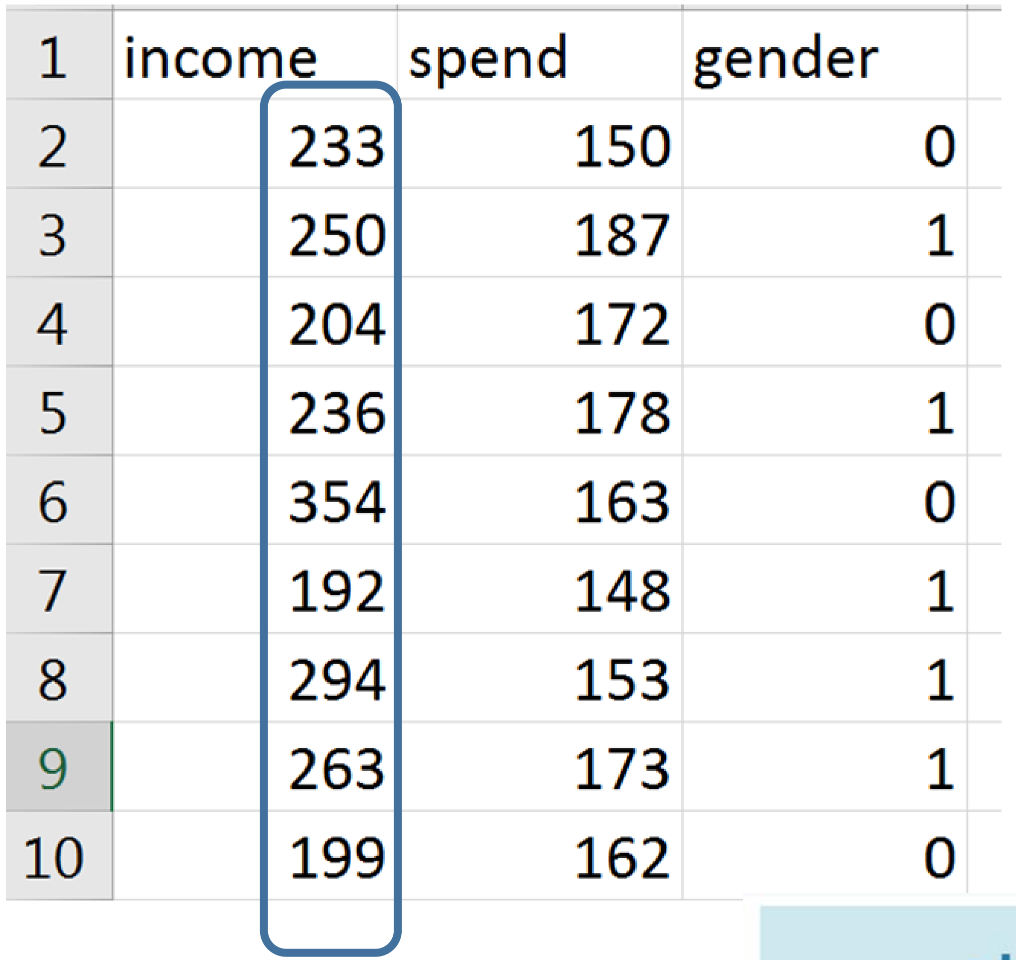
$$= \sqrt{(233-250)^2 + (150-187)^2 + (0-1)^2}$$

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

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

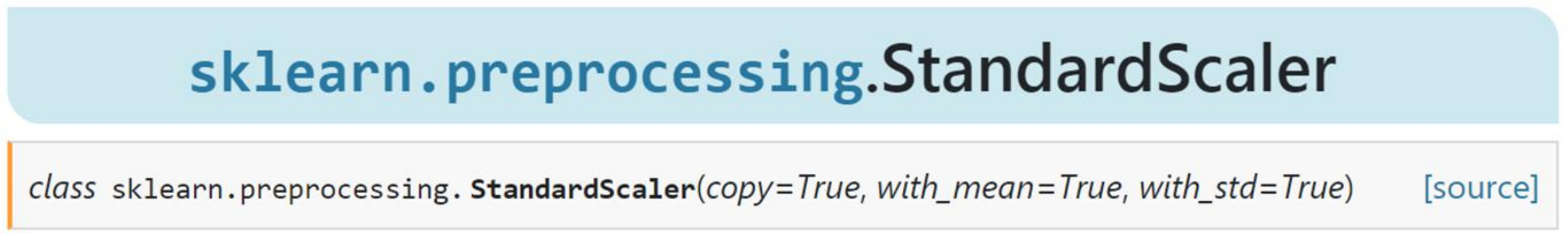


1	income	spend	gender
2	233	150	0
3	250	187	1
4	204	172	0
5	236	178	1
6	354	163	0
7	192	148	1
8	294	153	1
9	263	173	1
10	199	162	0

Method-1:

calculate mean of income (1st column), **m**
calculate standard deviation (std) of income, **s**
normalize income by mean and std
income <= (income - m)/s

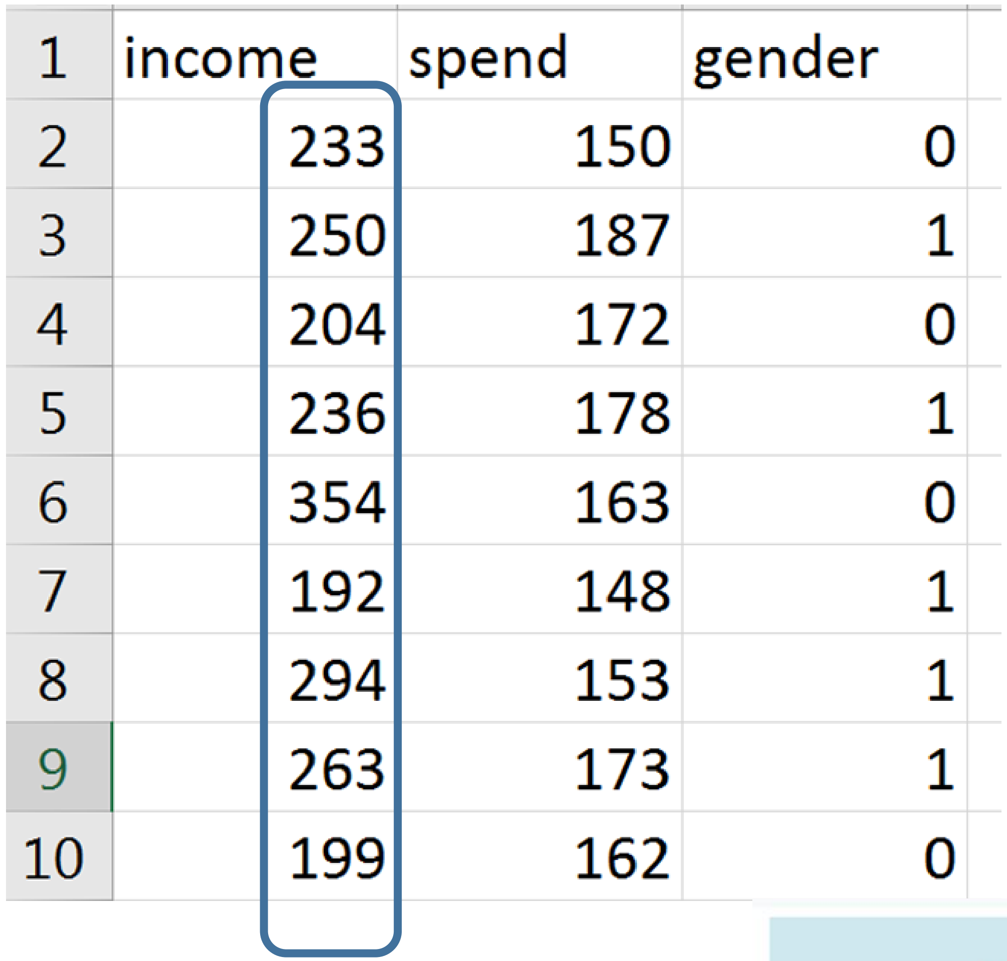
do the same thing for the other two features



sklearn.preprocessing.StandardScaler

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

Normalize features: weight the features equally

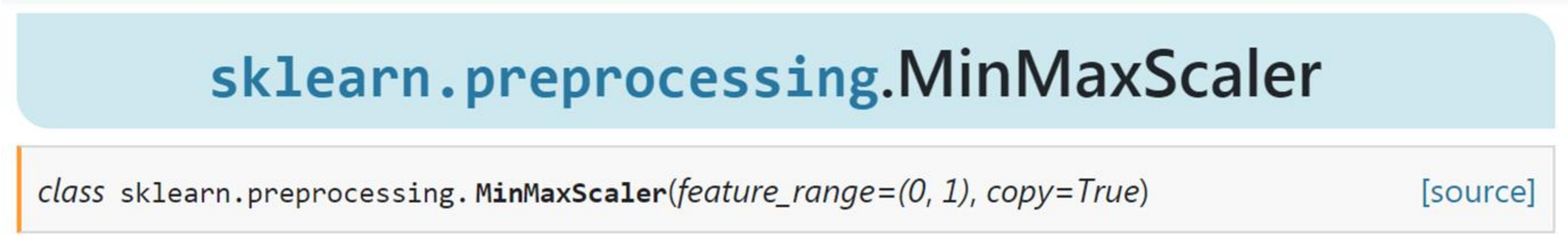


1	income	spend	gender
2	233	150	0
3	250	187	1
4	204	172	0
5	236	178	1
6	354	163	0
7	192	148	1
8	294	153	1
9	263	173	1
10	199	162	0

Method-2:

calculate the max of income (1st column), **a**
calculate the min of income, **b**
normalize income into the range of 0 to1 by
income <= (income - b)/(a-b)

do the same thing for the other two features



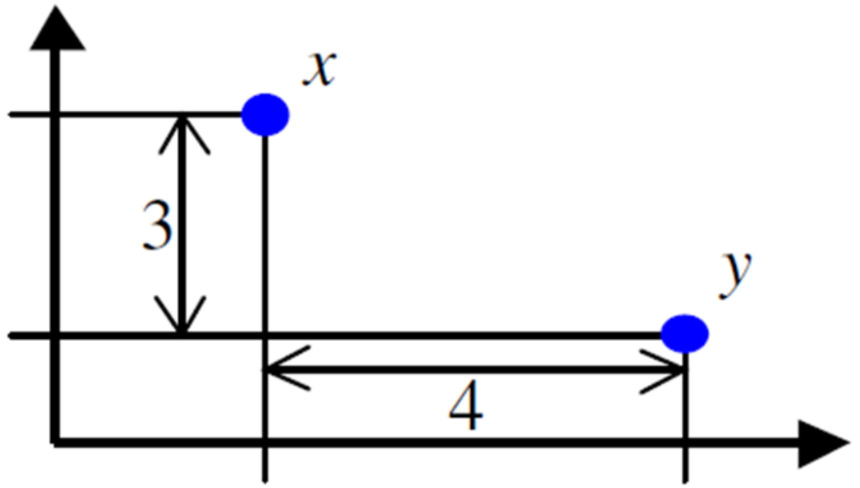
sklearn.preprocessing.MinMaxScaler

class sklearn.preprocessing.**MinMaxScaler**(*feature_range=(0, 1), copy=True*) [source]

Vector ℓ_p Norm of $\boldsymbol{x} \in \mathbb{R}^M$

- Common norms used in machine learning are
  - ℓ_1 norm $\|\boldsymbol{x}\|_1 = \sum_{m=1}^{M} \left| x_{[m]} \right|$
    (sum of the absolute values of the elements)
  - ℓ_2 norm $\|\boldsymbol{x}\|_2 = \sqrt{\sum_{m=1}^{M} x_{[m]}^2} = \sqrt{\boldsymbol{x}^T \boldsymbol{x}}$
    (Euclidean norm)
  - ℓ_∞ norm $\|\boldsymbol{x}\|_\infty = \max\{\left|x_{[1]}\right|, \dots, \left|x_{[M]}\right|\}$
    (max of the absolute values)

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: $d(x, y) \leq d(x, z) + d(y, z)$

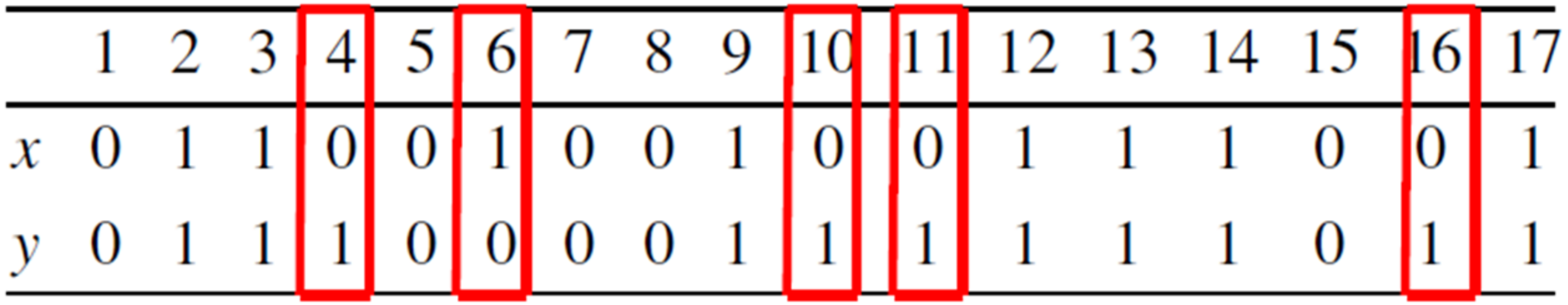
    if $d(x, z)$ is small and $d(y, z)$ is small, then $d(x, y)$ is small

    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	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)

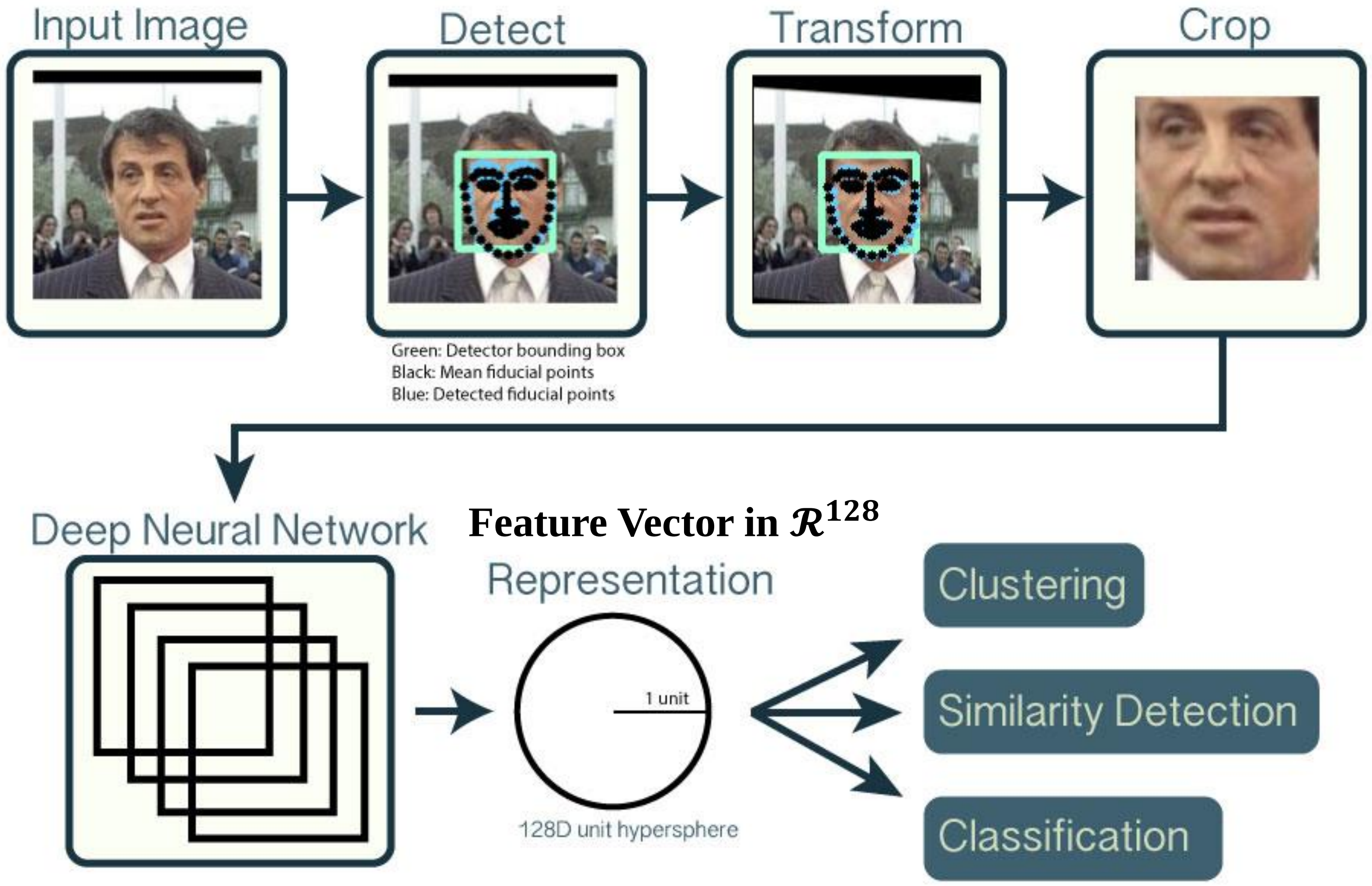
- Given N data points, $\{x_1, \dots, x_N\}$, $x_n \in \mathcal{R}^M$
- Find K cluster centers, $\{c_1, \dots, c_K\}$, $c_k \in \mathcal{R}^M$, $K \ll N$
- Assign each data point x_n to one cluster: label $\alpha(n) \in \{1, \dots, K\}$
- The optimal clustering result is obtained by minimizing the loss

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

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



https://cmusatyalab.github.io/openface/#openface