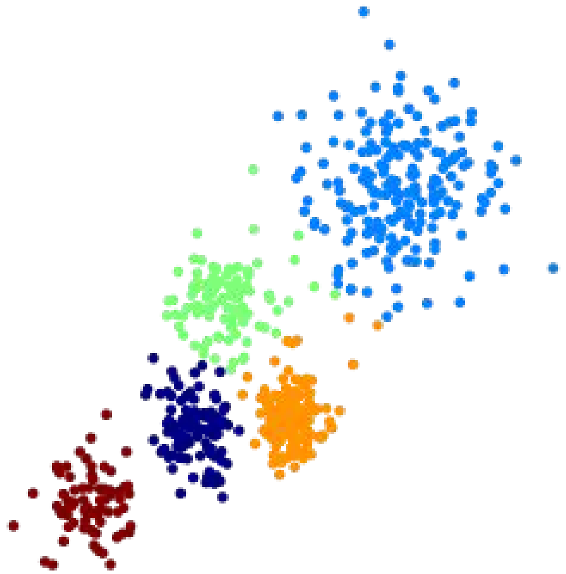
Clustering

K-means
& hierarchical clustering

DSI SEA, jf.omhover



Clustering

K-means
& hierarchical clustering

DSI SEA, jf.omhover

OBJECTIVES

- **Relate** clustering to unsupervised learning
- **Illustrate** the utility of clustering in real-world problems
- **Describe** and **implement** the k-means algorithm
- **Describe** and **implement** the HAC algorithm
- **Compare** purpose and utility of k-means and HAC
- **Discuss** the role of metrics for applying clustering to different problems
- **Analyze** how the (high) dimensionality of data impacts metrics based clustering techniques

Supervised / Unsupervised Learning

Supervised Learning

REALITY

	type	income	education	prestige
accountant	prof	62	86	82
pilot	prof	72	76	83
architect	prof	75	92	90
author	prof	55	90	76
chemist	prof	64	86	90
minister	prof	21	84	87
professor	prof	64	93	93
dentist	prof	80	100	90
reporter	wc	67	87	52
engineer	prof	72	86	88
undertaker	prof	42	74	57
lawyer	prof	76	98	89

data

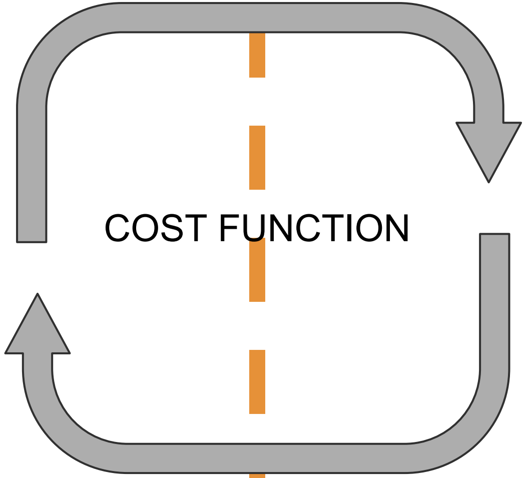
(x_1, y_1)

...

(x_n, y_n)

$x \quad y$

OBJECTIVE:
descriptive
predictive
normative
...



MODEL

$y = f(x) + \epsilon$

take a function as an assumption

$\hat{y} = \hat{f}(x)$

Estimator of the function

Unsupervised Learning

REALITY

	type	income	education	prestige
accountant	prof	62	86	82
pilot	prof	72	76	83
architect	prof	75	92	90
author	prof	55	90	76
chemist	prof	64	86	90
minister	prof	21	84	87
professor	prof	64	93	93
dentist	prof	80	100	90
reporter	wc	67	87	52
engineer	prof	72	86	88
undertaker	prof	42	74	57
lawyer	prof	76	98	89

data

$$(x_1, \cancel{y_1})$$

...

$$(x_n, \cancel{y_n})$$

$$x \quad \cancel{y}$$

OBJECTIVE:
descriptive
predictive
normative
...

COST FUNCTION

MODEL

$$y = f(x) + \epsilon$$

take a function as
an assumption

$$\hat{y} = \hat{f}(x)$$

Estimator
of the function

Machine Learning: different types of learning

[Samuel, 1959] : Machine learning is the “field of study that gives computers the ability to learn without being explicitly programmed”

Supervised learning:

- The model is derived from observations of input/output pairs
- You have data samples with labelled output (quantitative / qualitative)

Unsupervised learning:

- The model is derived from the confrontation of a meta-model with observations
- You have data samples without no output class, and you want to explain or describe them (but you have an idea of what you’re looking for)

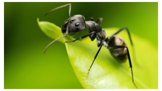
Data Science Tools

Reinforcement learning:

- The model is derived from interactions with an external agent or environment

Unsupervised-type questions





- I have a database of clients with their purchase history and I want to draw profiles
- I have the proceedings of the last 2016 data science conference and I want to see the hot topics
- I have obtained usage traces of users on my GUI (clicks, forward/backward, inputs, time spent on each page etc) and I want to understand what different behavior and trajectories they may have
- I have this dataset of gene expressions and I want to extract groups of genes that have mutual influences
- I have this dataset of tweets on the presidential debate and I want my candidate to know which people were tweeting about what

Clustering

Brainstorm : what's a cluster ?

dataset

[[0.06497338 0.35259884]
 [0.20073913 0.34345291]
 [0.0540259   0.34076791]
 [0.1415917   0.34904249]
 [0.1066884   0.38564734]
 [0.07874009 0.42121996]
 [0.16728357 0.45044774]
 [0.18659554 0.47644782]
 [0.08462494 0.3317815]
 [0.17597371 0.37192779]
 [0.1547712   0.47111988]
 [0.089005   0.53872432]
 [0.11967159 0.3192513]
 [0.12118539 0.21355644]
 [0.17501382 0.36435908]
 [0.12403482 0.30928354]
 [0.12190772 0.40995677]

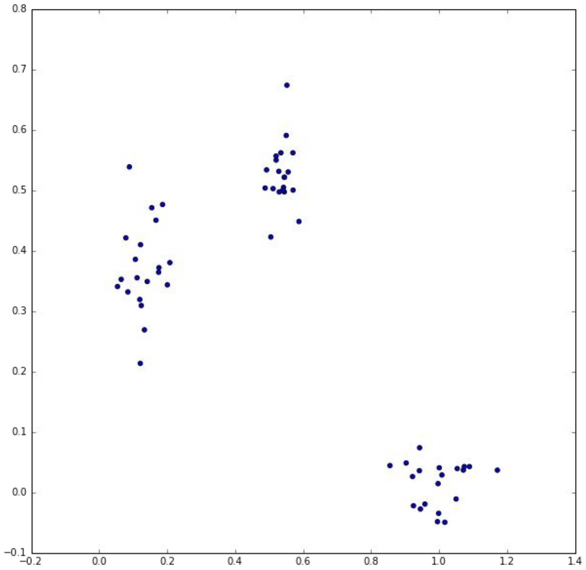
…

How many clusters
do you see ?

Why does it jump out ?

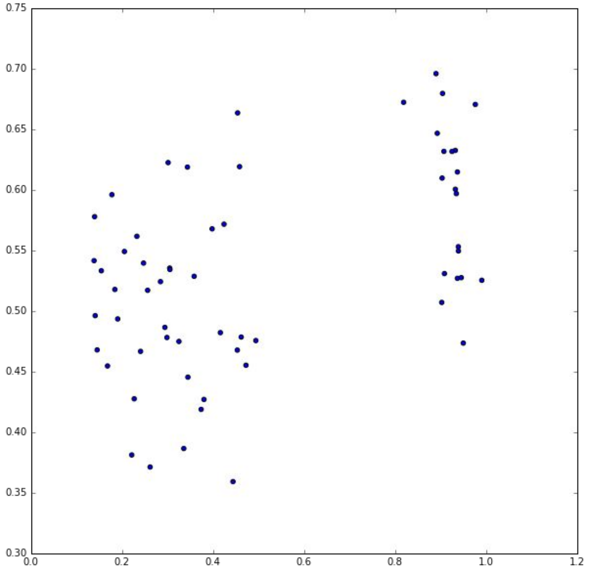
What makes it
a "cluster" ?

scatter plot



Brainstorm : what’s a cluster ?

scatter plot 2



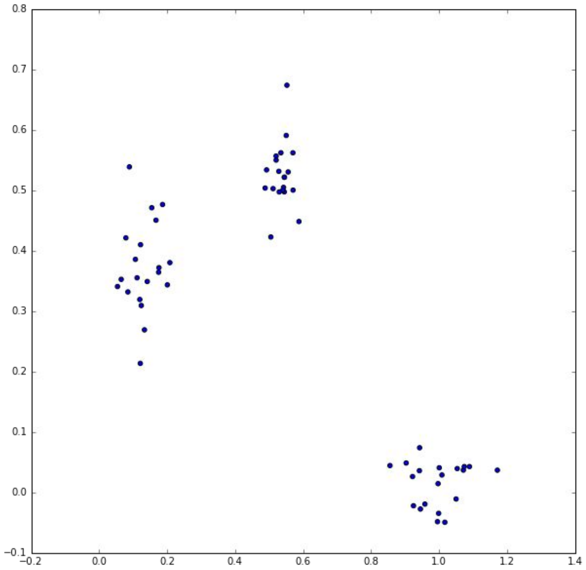
How many clusters
do you see ?

Why does it jump out ?

What makes it
a “cluster” ?

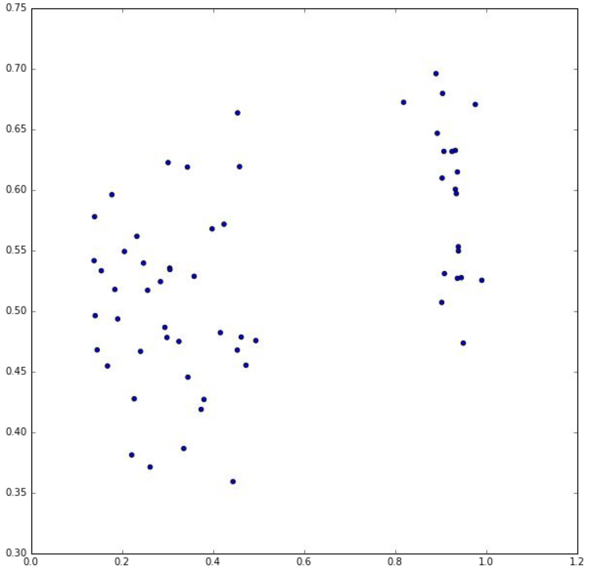
What makes it NOT
a “cluster” ?

scatter plot



Clusters: a cognitive definition

scatter plot 2

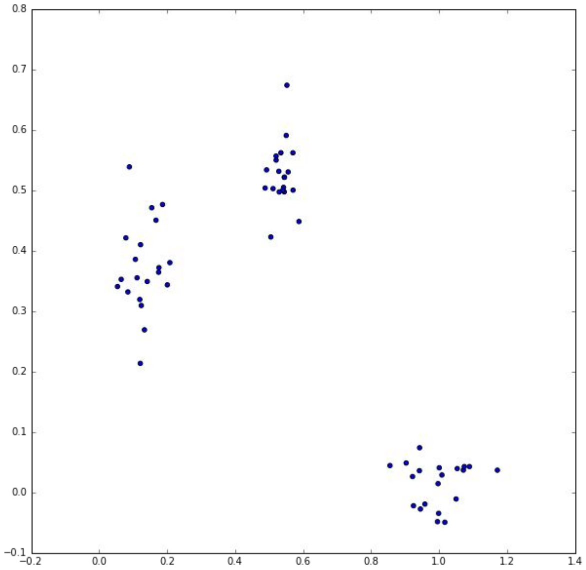


A partition of the dataset
(not necessarily crisp)

A strong <u>internal</u> similarity
(small intra/within
cluster distance)

A strong <u>external</u> dissimilarity
(large extra cluster distance)

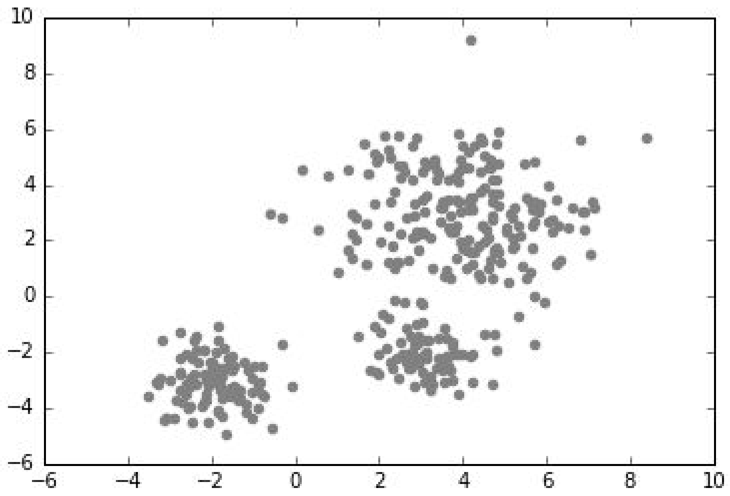
scatter plot



the k-Means algorithm

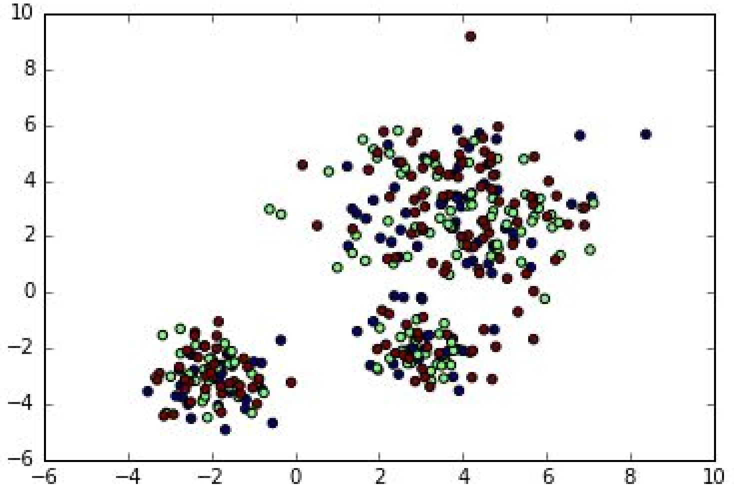
Step by step execution

1. Randomly assign a number, from 1 to K, to each of the observations.
2. **Iterate** until the cluster assignments stop changing:
   a. For each of the K clusters, compute the cluster ***centroid***: the vector of the p features **means** for the observations in the k-th cluster
   b. **Assign** each observation to the cluster whose centroid is **closest** (defined using Euclidian distance)



Step by step execution

1. Randomly assign a number, from 1 to K, to each of the observations.
2. **Iterate** until the cluster assignments stop changing:
    a. For each of the K clusters, compute the cluster ***centroid***: the vector of the *p* features **means** for the observations in the k-th cluster
    b. **Assign** each observation to the cluster whose centroid is **closest** (defined using Euclidian distance)



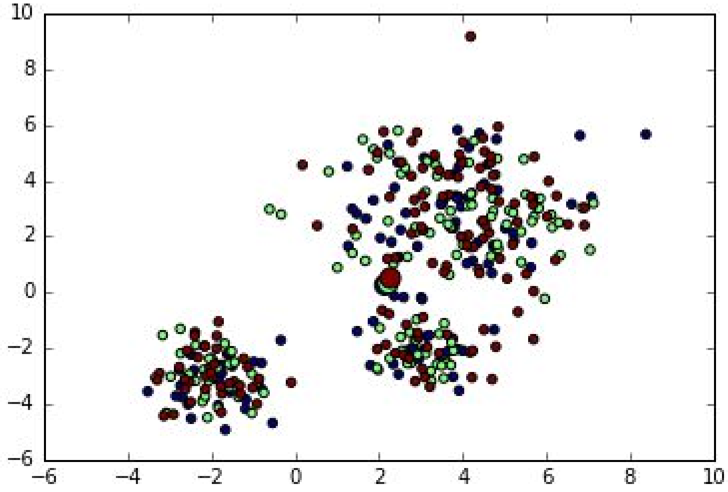
Step by step execution

1. Randomly assign a number, from 1 to K, to each of the observations.

2. **Iterate** until the cluster assignments stop changing:

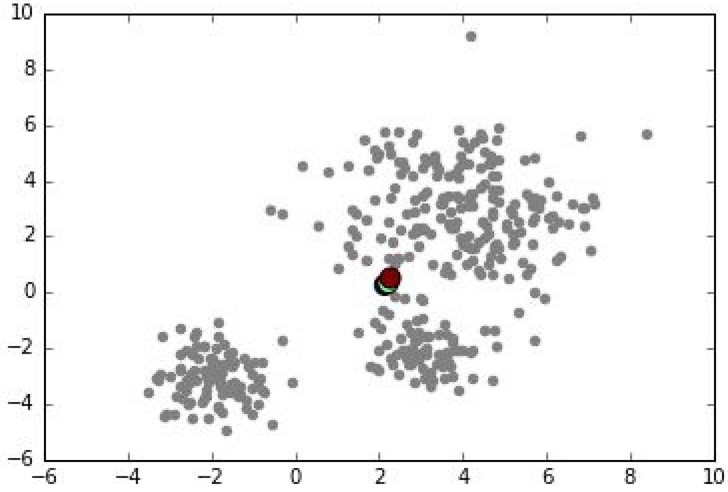
   a. For each of the K clusters, compute the cluster ***centroid***: the vector of the *p* features **means** for the observations in the k-th cluster

   b. **Assign** each observation to the cluster whose centroid is **closest** (defined using Euclidian distance)



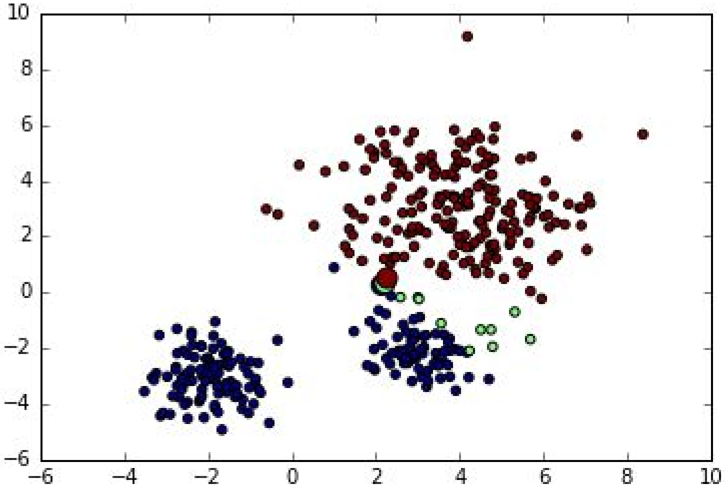
Step by step execution

1. Randomly assign a number, from 1 to K, to each of the observations.
2. **Iterate** until the cluster assignments stop changing:
   a. For each of the K clusters, compute the cluster ***centroid***: the vector of the *p* features **means** for the observations in the k-th cluster
   b. **Assign** each observation to the cluster whose centroid is **closest** (defined using Euclidian distance)



Step by step execution

1. Randomly assign a number, from 1 to K, to each of the observations.
2. **Iterate** until the cluster assignments stop changing:
   a. For each of the K clusters, compute the cluster ***centroid***: the vector of the *p* features **means** for the observations in the k-th cluster
   b. **Assign** each observation to the cluster whose centroid is **closest** (defined using Euclidian distance)



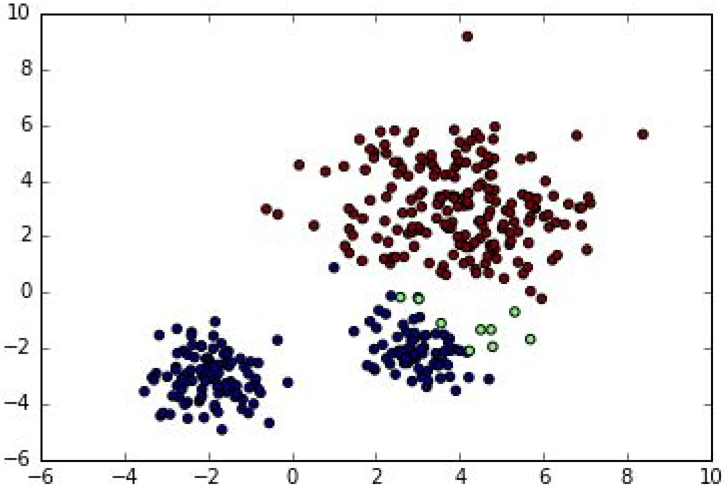
Step by step execution

1. Randomly assign a number, from 1 to K, to each of the observations.

2. **Iterate** until the cluster assignments stop changing:

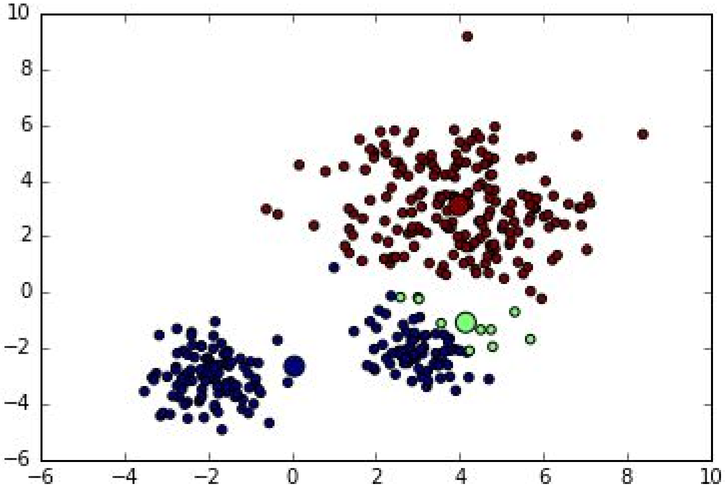
   a. For each of the K clusters, compute the cluster ***centroid***: the vector of the *p* features **means** for the observations in the k-th cluster

   b. **Assign** each observation to the cluster whose centroid is **closest** (defined using Euclidian distance)



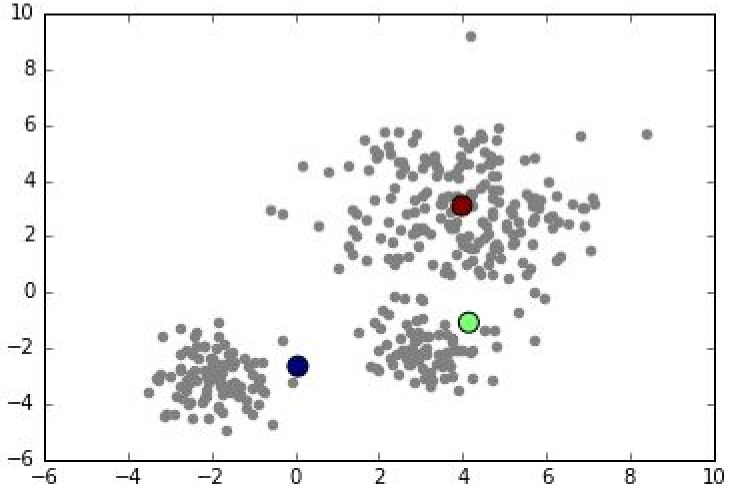
Step by step execution

1. Randomly assign a number, from 1 to K, to each of the observations.
2. **Iterate** until the cluster assignments stop changing:
   a. For each of the K clusters, compute the cluster ***centroid***: the vector of the p features **means** for the observations in the k-th cluster
   b. **Assign** each observation to the cluster whose centroid is **closest** (defined using Euclidian distance)



Step by step execution

1. Randomly assign a number, from 1 to K, to each of the observations.
2. **Iterate** until the cluster assignments stop changing:
   a. For each of the K clusters, compute the cluster ***centroid***: the vector of the *p* features **means** for the observations in the k-th cluster
   b. **Assign** each observation to the cluster whose centroid is **closest** (defined using Euclidian distance)



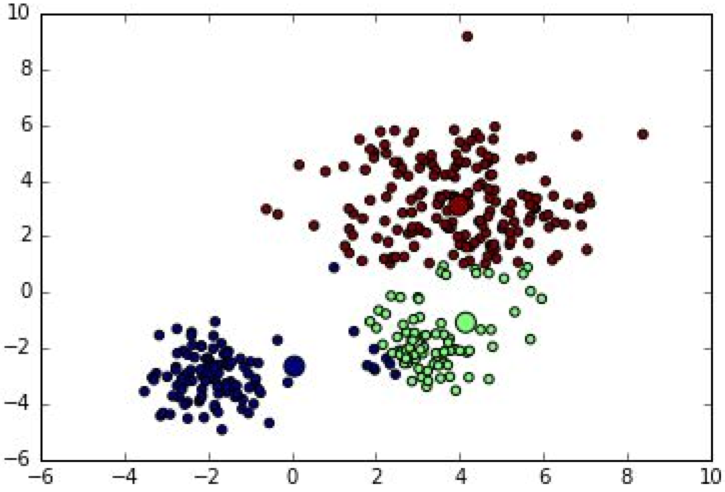
Step by step execution

1. Randomly assign a number, from 1 to K, to each of the observations.

2. **Iterate** until the cluster assignments stop changing:

   a. For each of the K clusters, compute the cluster ***centroid***: the vector of the *p* features **means** for the observations in the k-th cluster

   b. **Assign** each observation to the cluster whose centroid is **closest** (defined using Euclidian distance)



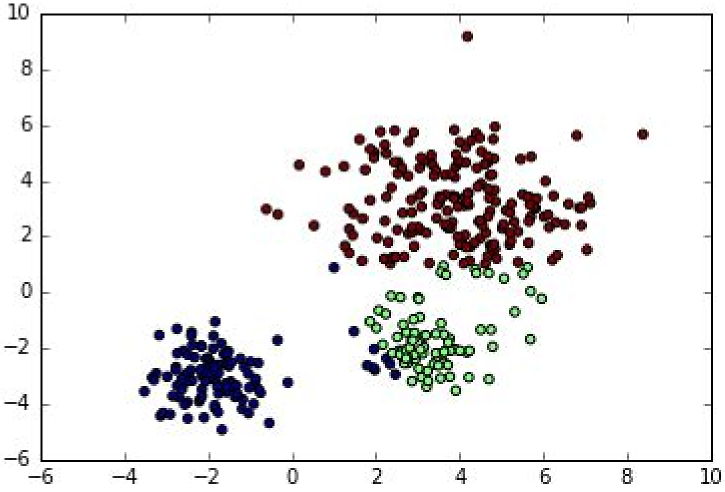
Step by step execution

1. Randomly assign a number, from 1 to K, to each of the observations.

2. **Iterate** until the cluster assignments stop changing:

   a. For each of the K clusters, compute the cluster ***centroid***: the vector of the *p* features **means** for the observations in the k-th cluster

   b. **Assign** each observation to the cluster whose centroid is **closest** (defined using Euclidian distance)



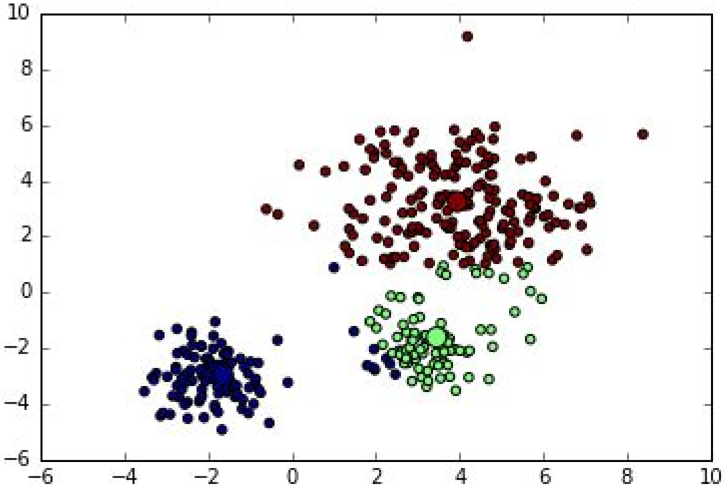
Step by step execution

1. Randomly assign a number, from 1 to K, to each of the observations.

2. **Iterate** until the cluster assignments stop changing:

   a. For each of the K clusters, compute the cluster ***centroid***: the vector of the p features **means** for the observations in the k-th cluster

   b. **Assign** each observation to the cluster whose centroid is **closest** (defined using Euclidian distance)



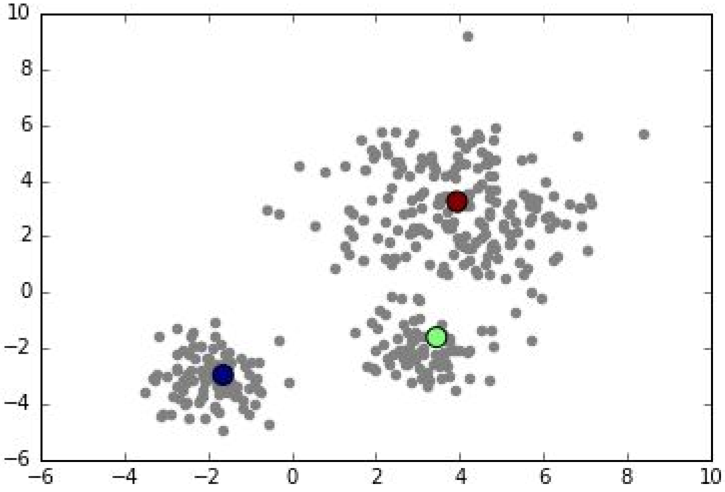
Step by step execution

1. Randomly assign a number, from 1 to K, to each of the observations.

2. **Iterate** until the cluster assignments stop changing:

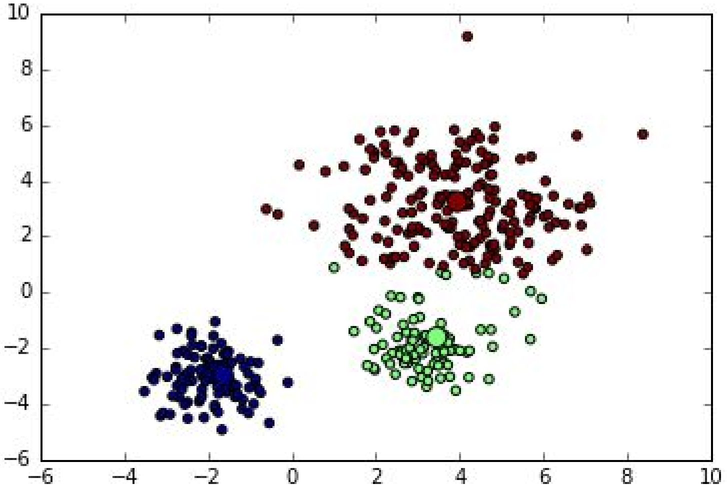
   a. For each of the K clusters, compute the cluster ***centroid***: the vector of the *p* features **means** for the observations in the k-th cluster

   b. **Assign** each observation to the cluster whose centroid is **closest** (defined using Euclidian distance)



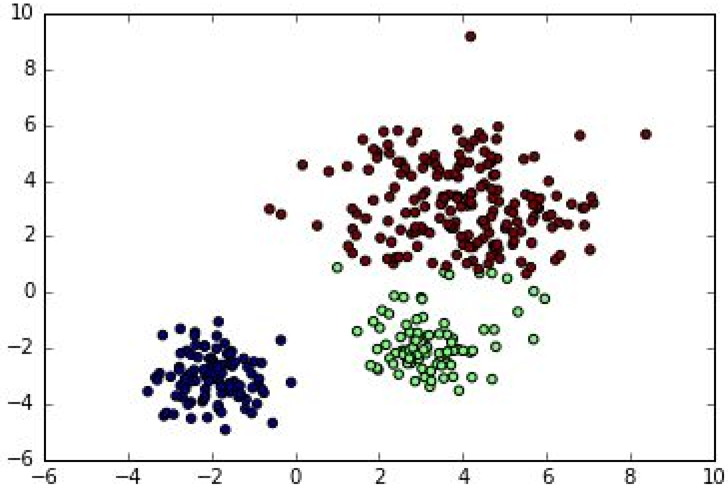
Step by step execution

1. Randomly assign a number, from 1 to K, to each of the observations.
2. **Iterate** until the cluster assignments stop changing:
   a. For each of the K clusters, compute the cluster ***centroid***: the vector of the p features **means** for the observations in the k-th cluster
   b. **Assign** each observation to the cluster whose centroid is **closest** (defined using Euclidian distance)



Step by step execution

1. Randomly assign a number, from 1 to K, to each of the observations.
2. **Iterate** until the cluster assignments stop changing:
   a. For each of the K clusters, compute the cluster ***centroid***: the vector of the *p* features **means** for the observations in the k-th cluster
   b. **Assign** each observation to the cluster whose centroid is **closest** (defined using Euclidian distance)



Convergence of the algorithm

1. Randomly assign a number, from 1 to K, to each of the observations.
2. **Iterate** until the cluster assignments stop changing:
   a. For each of the K clusters, compute the cluster ***centroid***: the vector of the *p* features **means** for the observations in the k-th cluster
   b. **Assign** each observation to the cluster whose centroid is **closest** (defined using Euclidian distance)

Objective: minimize “within cluster similarity”

$$\underset{C_1,\ldots,C_K}{\text{minimize}}\left\{\sum_{k=1}^{K}\frac{1}{|C_k|}\sum_{i,i'\in C_k}\sum_{j=1}^{p}(x_{ij}-x_{i'j})^2\right\}$$

Because of the effect of scale on euclidian distance

Pre-scale is almost mandatory

K-Means concerns

1. Randomly assign a number, from 1 to K, to each of the observations.

2. **Iterate** until the cluster assignments stop changing:

   a. For each of the K clusters, compute the cluster ***centroid***: the vector of the *p* features **means** for the observations in the k-th cluster

   b. **Assign** each observation to the cluster whose centroid is **closest** (defined using Euclidian distance)

PB2 : Robustness to initialization

PB3 : Initialization strategy ?

PB4 : What’s the best k ?

PB1: When to stop ?

PB1: When to stop ?

Convergence is assured, but is not necessarily fast...

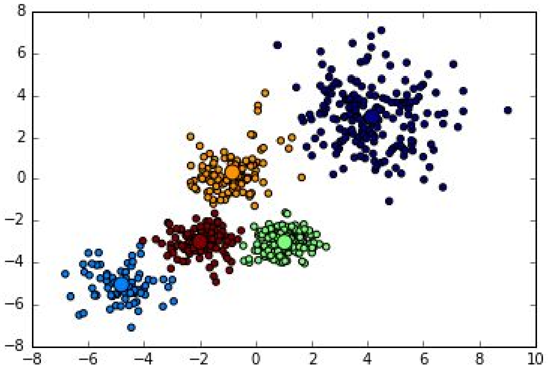
Solution 1 : when the centroids don't change at all (you may wait a long time)

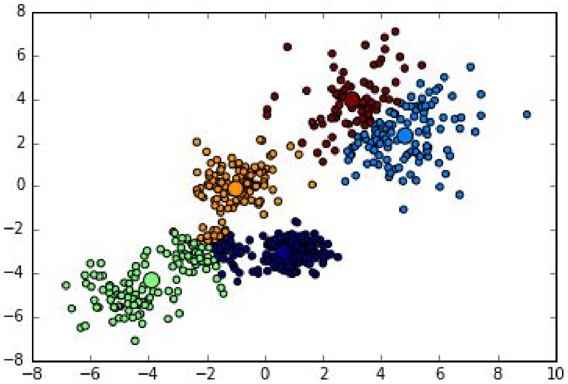
Solution 2 : when the centroids don't change that much (`tol`)

Solution 3 : when we get tired of waiting (`max_iter`)

```
sklearn.cluster.KMeans(    n_clusters=8,
                           init='k-means++',
                           n_init=10,
                           max_iter=300,
                           tol=0.0001        )
```

PB2: Robustness to initialization



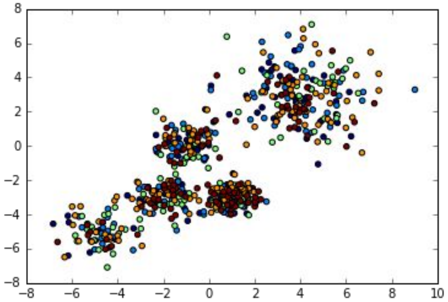


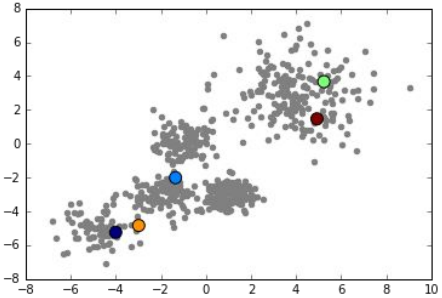
Depending on your initialization, you may (will) have different results.
Solution : try several times and see if the result is stable

```
sklearn.cluster.KMeans(   n_clusters=8,
                          init='k-means++',
                          n_init=10,
                          max_iter=300,
                          tol=0.0001        )
```

PB3: Initialization strategy

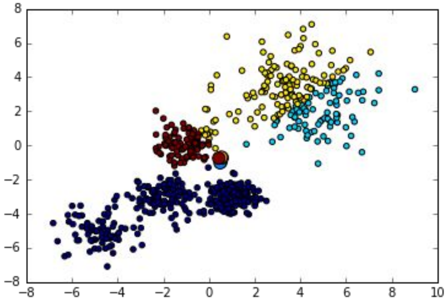
Strategy 1: initialize <u>assignments</u> randomly

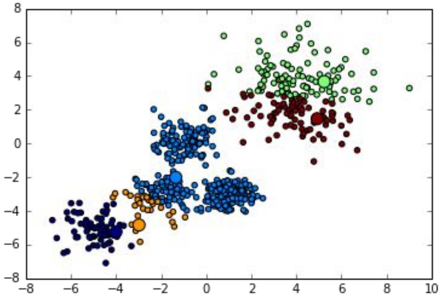




Strategy 2: initialize <u>centroids</u> drawn from observations

First centroid computing



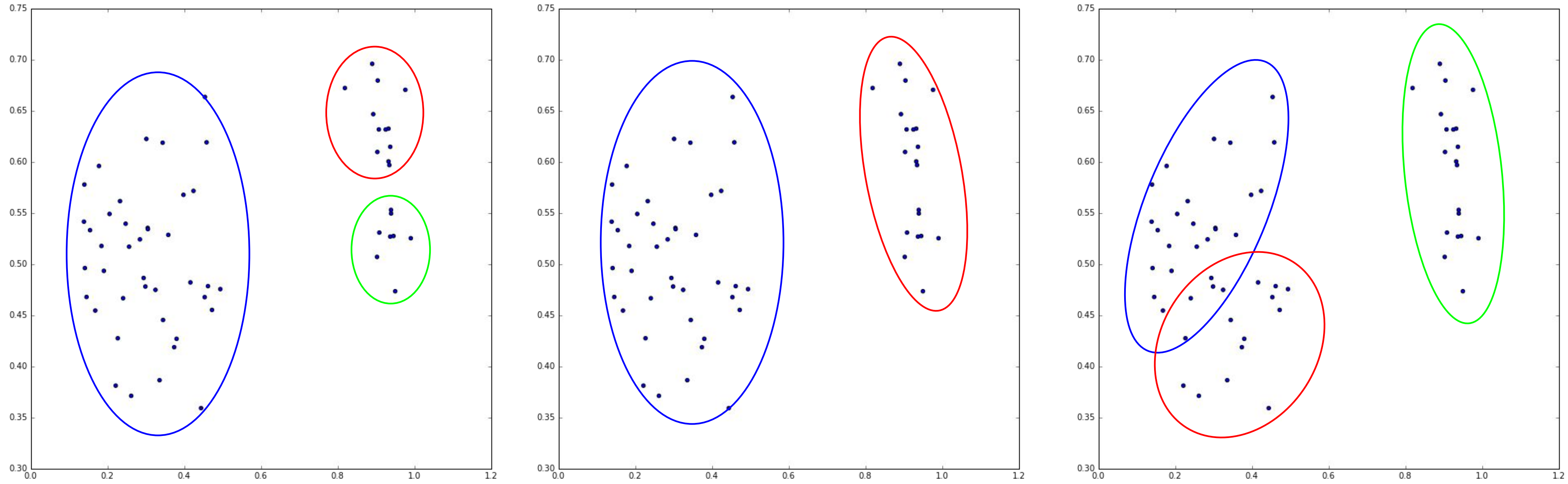


First assignment

```
sklearn.cluster.KMeans(   n_clusters=8,
                          init='k-means++',
                          n_init=10,
                          max_iter=300,
                          tol=0.0001        )
```

PB4: how to choose k ?
(Evaluating clustering)

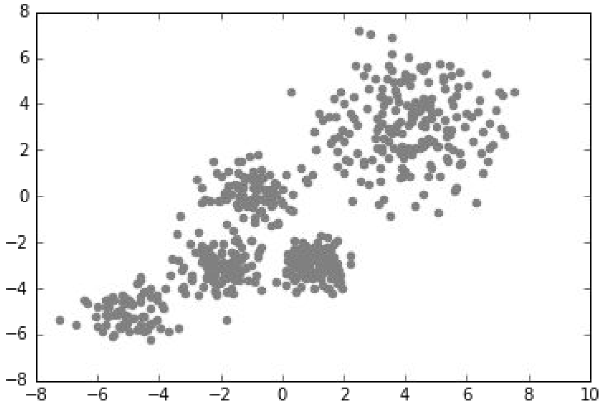
What makes a good clustering ?



Elbow method

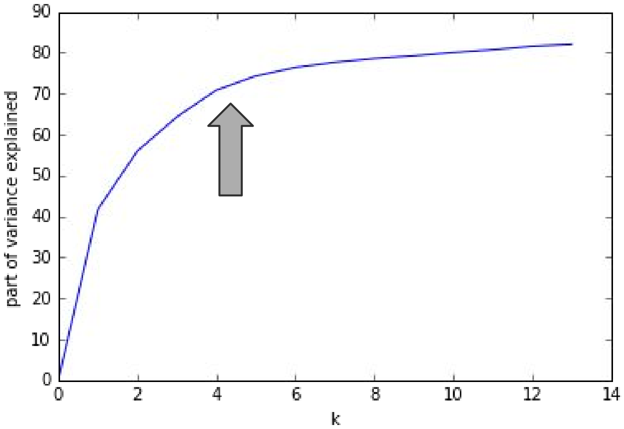
Compute total WCSS (k=1)

Compute WCSS for each iteration of k



Equiv. to a “total variance explained” plot

Observe where does increasing k stop that increase in WCSS ?



Silhouette plot

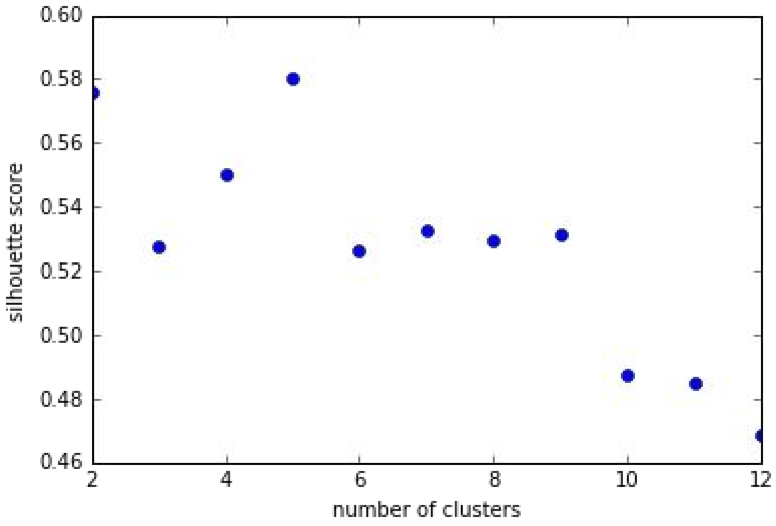
from sklearn.metrics import silhouette_score, silhouette_samples

For each point x_i:

- $a(i)$ average dissimilarity of x_i with points in the same cluster
- $b(i)$ average dissimilarity of x_i with points in the nearest cluster
    - "nearest" means cluster with the smallest $b(i)$

$$\text{silhouette}(i) = \frac{b(i) - a(i)}{max(a(i), b(i))}$$

What's the range of silhouette scores?



GAP Statistic

For each K, compare W_K (within-cluster sum of squares) with that of randomly generated "reference distributions"

Generate B distributions

$$Gap(K) = \frac{1}{B} \sum_{b=1}^{B} \log W_{Kb} - \log W_K$$

Choose smallest K such that $Gap(K) \geq Gap(K+1) - s_{N+1}$

where s_K is the standard error of $Gap(K)$

[paper by Hastie et al]

Individual Assignment

(there’s a hidden gem)