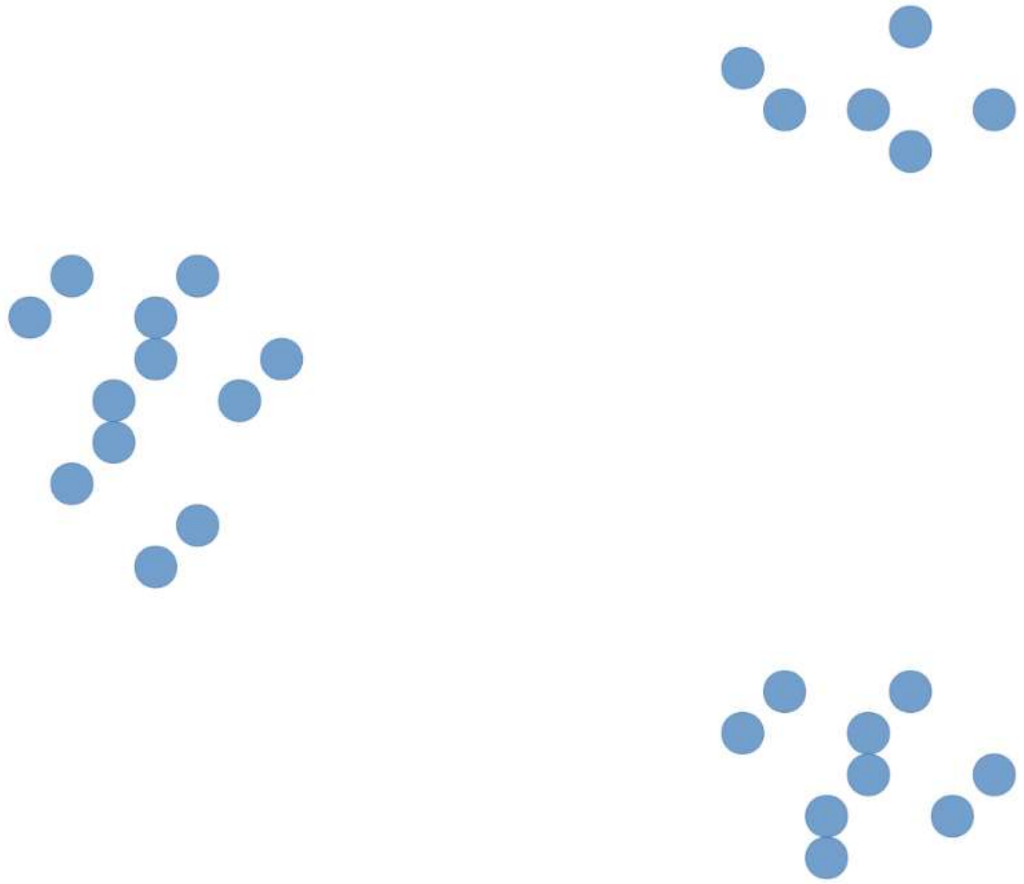
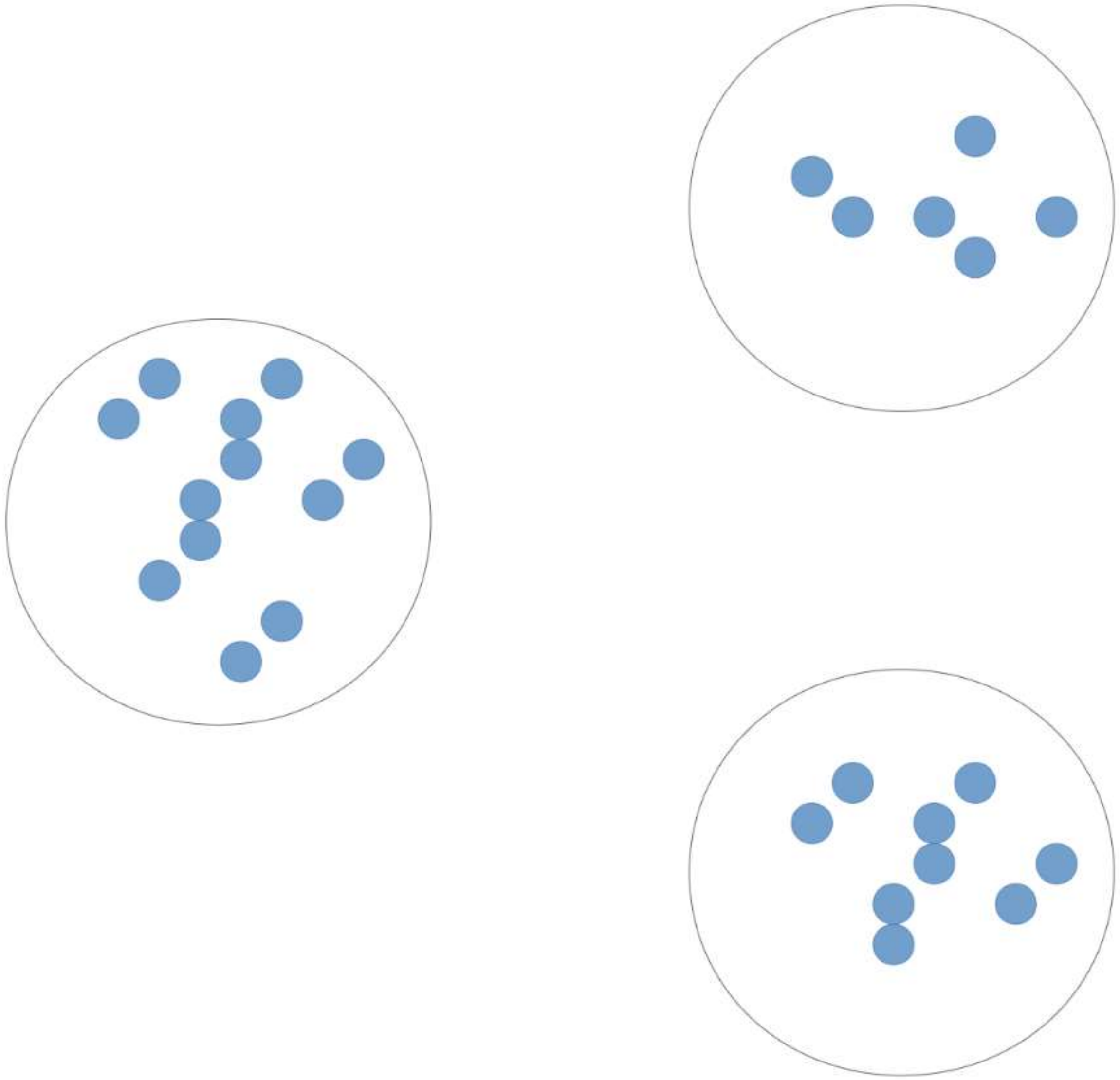
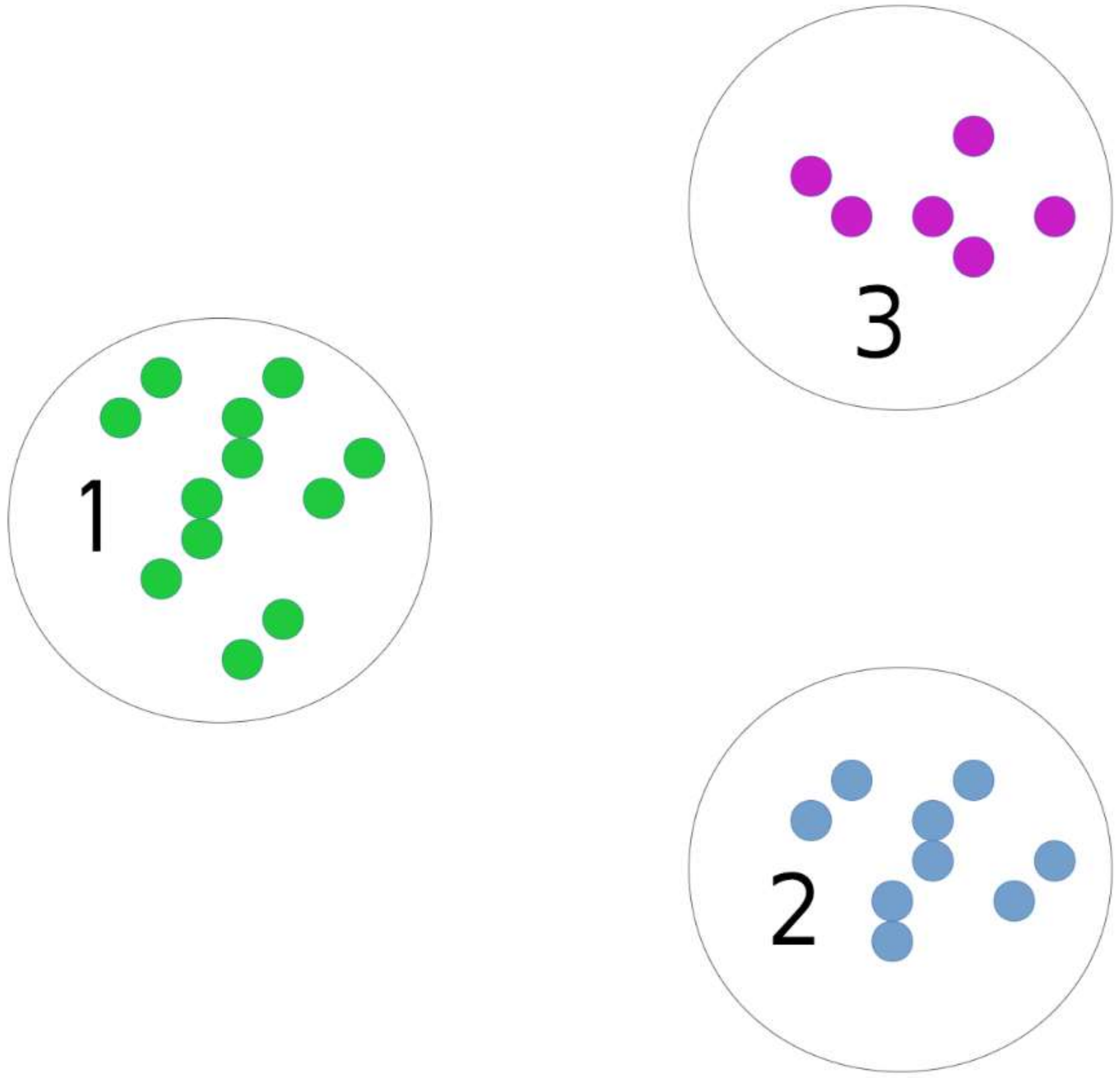
POWERING
POTENTIAL

Clustering





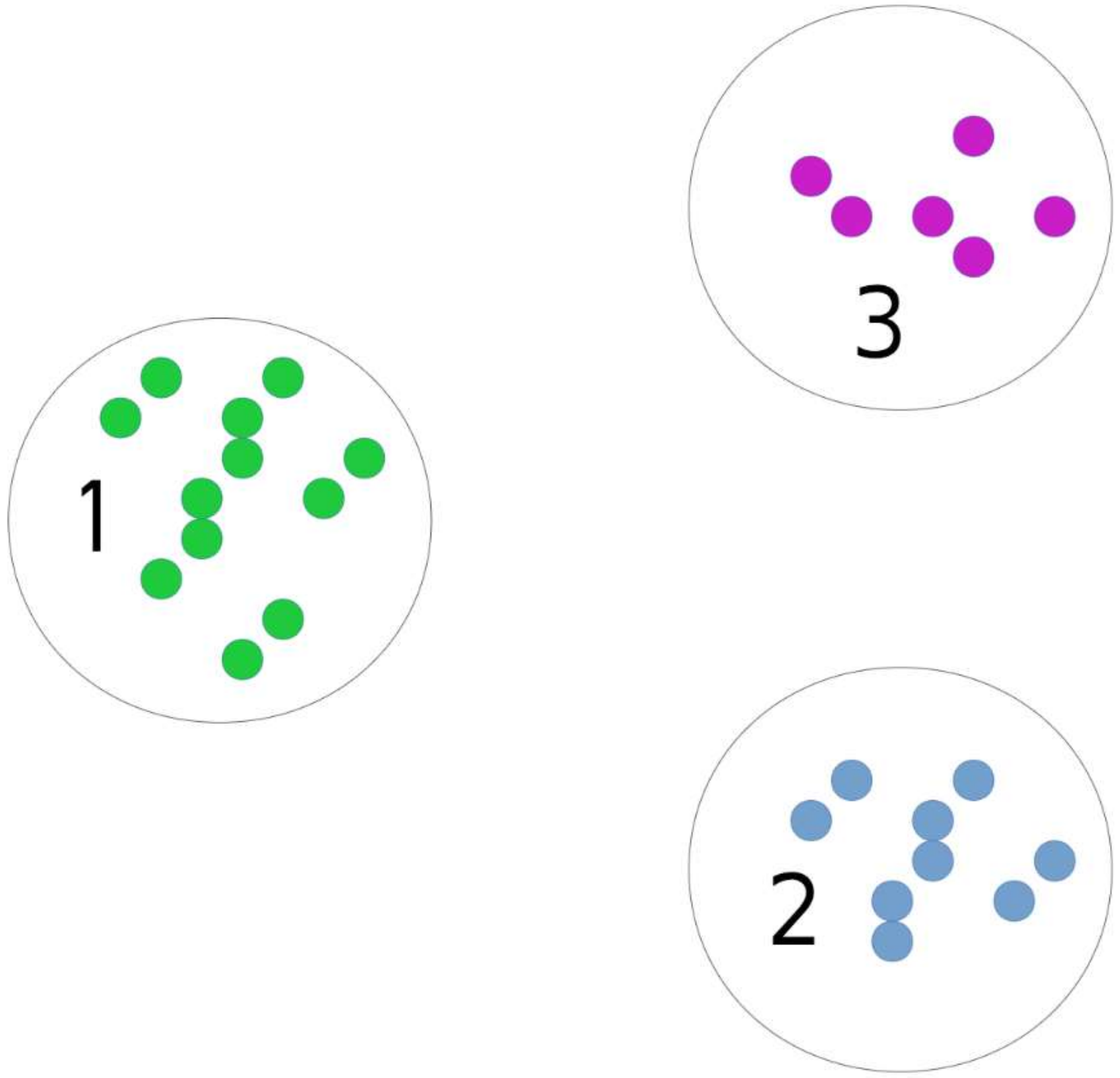

1
2
3

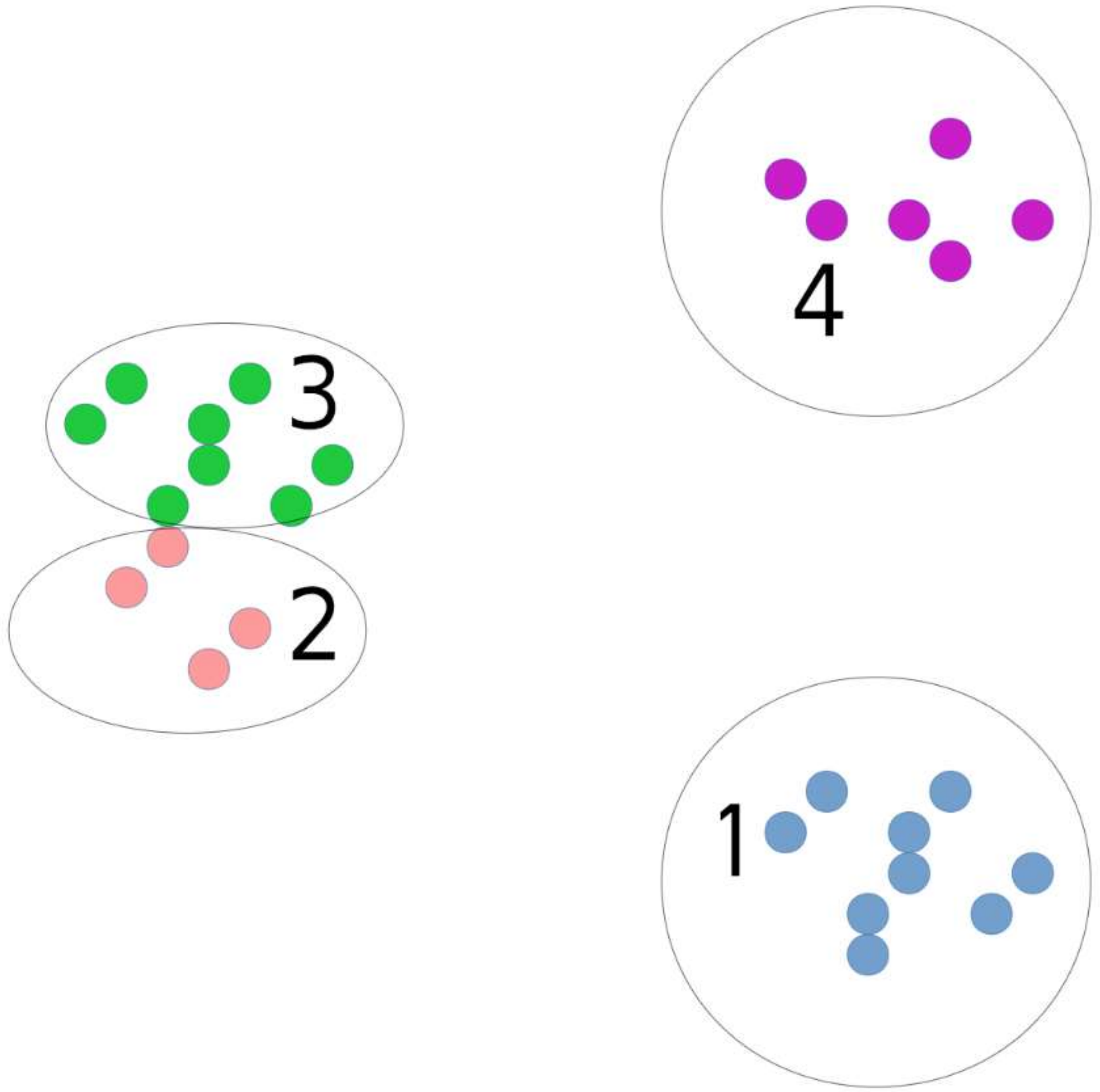
CLUSTERING

Partition data into groups (clusters)

Points within a cluster should be “similar”.

Points in different cluster should be ”different”


1
2
3


4
3
2
1

WHAT IS SIMILARITY?

Difficult to define! -- we know it when we see it ☺

pragmatic approach:

consider **distance** (rather than similarity)

between random variables

GOALS OF CLUSTERING

Data Exploration

- Understand the structure of the your data, (qualitative analysis):
- **Are there coherent groups ?**
- **How many groups are there ?**

Data Partitioning

- Divide data by group before further processing

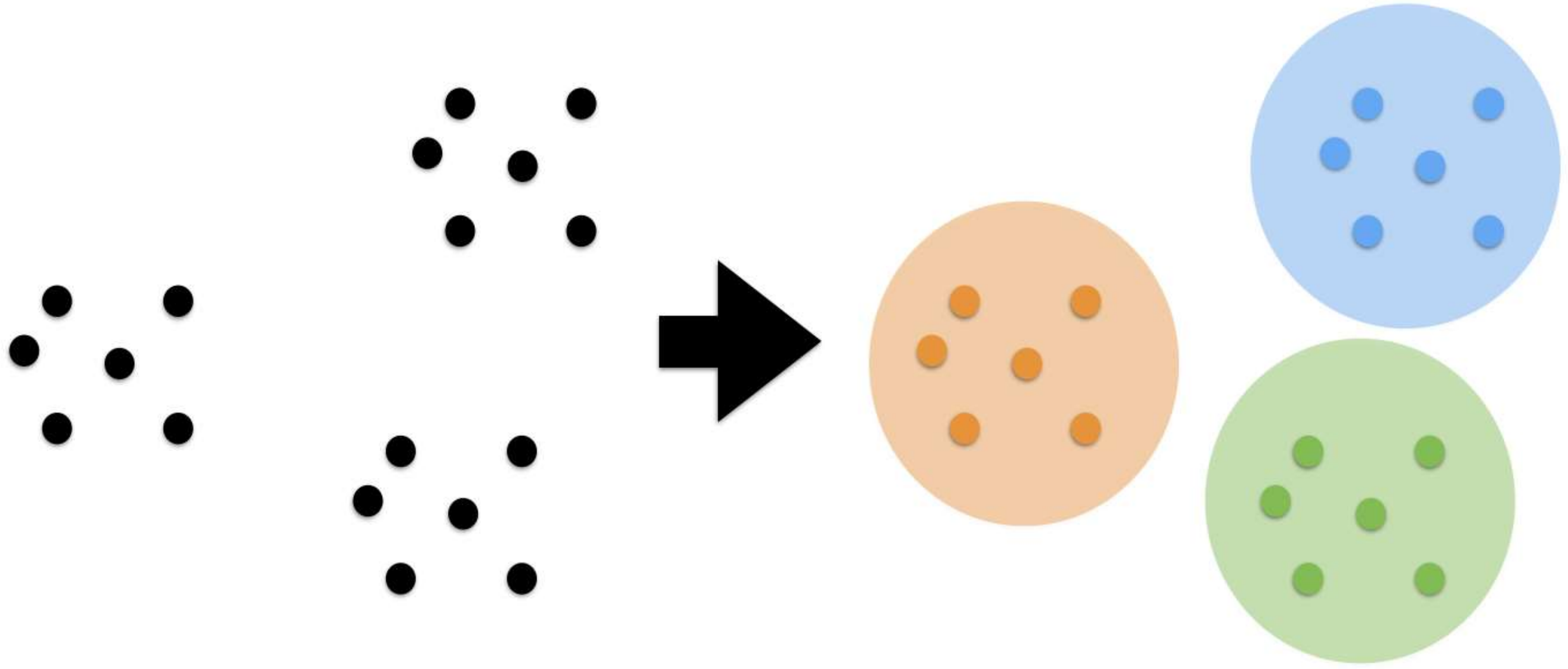
Compress Data

- Represent each group by a single prototype, say the center

Unsupervised feature extraction

- Derive features from clusters or cluster distances for downstream tasks
- Downstream tasks can help evaluate if the choice of clusters was appropriate

Clustering = task of gathering samples into groups of similar samples according to some predefined similarity or distance (dissimilarity) measure



Clustering: Methods

- **Partition** based methods
- **Probabilistic model** based methods
- **Density** based methods

K MEANS: BUSINESS USES

Behavioral segmentation:

- Segment by purchase history
- Segment by activities on application, website, or platform
- Define personas based on interests
- Create profiles based on activity monitoring

Inventory categorization:

- Group inventory by sales activity
- Group inventory by manufacturing metrics

Sorting sensor measurements:

- Detect activity types in motion sensors
- Group images
- Separate audio
- Identify groups in health monitoring

Detecting bots or anomalies:

- Separate valid activity groups from bots
- Group valid activity to clean up outlier detection

K MEANS: OBJECTIVE FUNCTION

What is k?

k stands for the number of clusters you're trying to find.

Goal of k means?

To find a set of k clusters, which minimise the Euclidean distance from every point to the closest cluster

k means algorithm:

Partition the observations $(x_1, x_2, \ldots, x_n)$ into k sets:

$\boldsymbol{S} = \{S_1, S_2, \ldots, S_k\}$

$(k \leq n)$

such that the sets minimize:

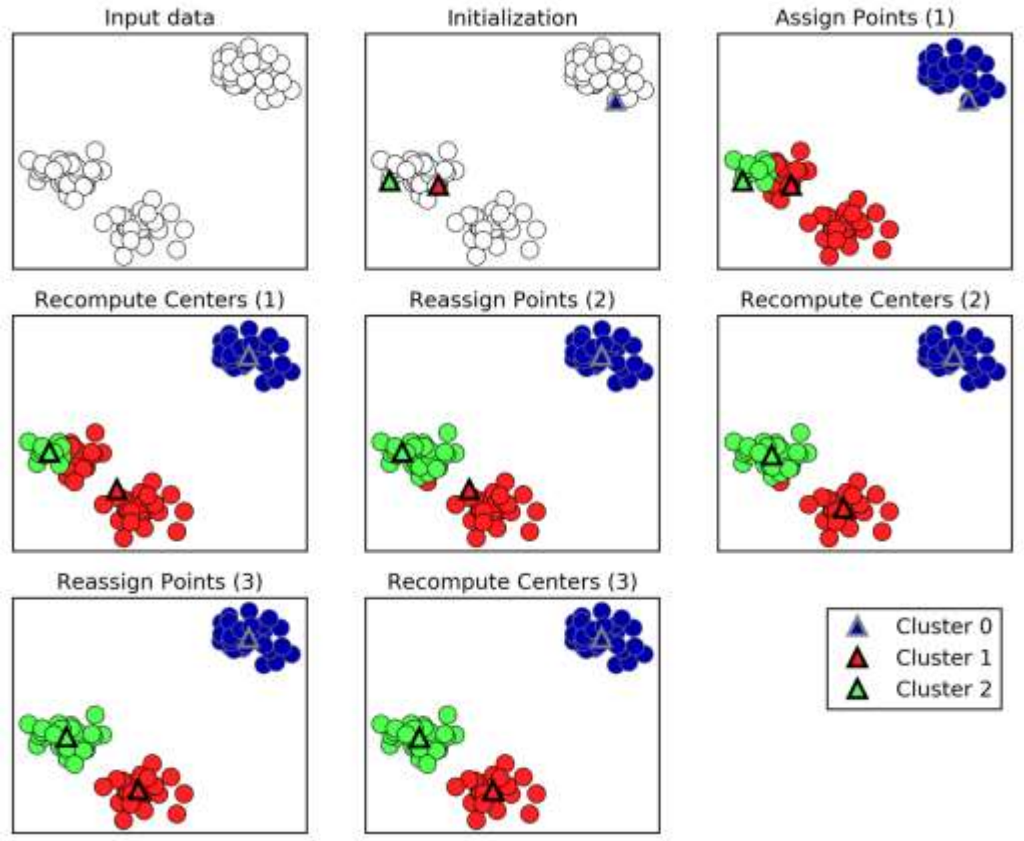
within-cluster sum of squares

$$\underset{\boldsymbol{S}}{argmin} \sum_{i=1}^{k} \sum_{x_j \in S_i} \left\| x_j - \mu_i \right\|^2$$

μ_i = mean of the points in set S_i

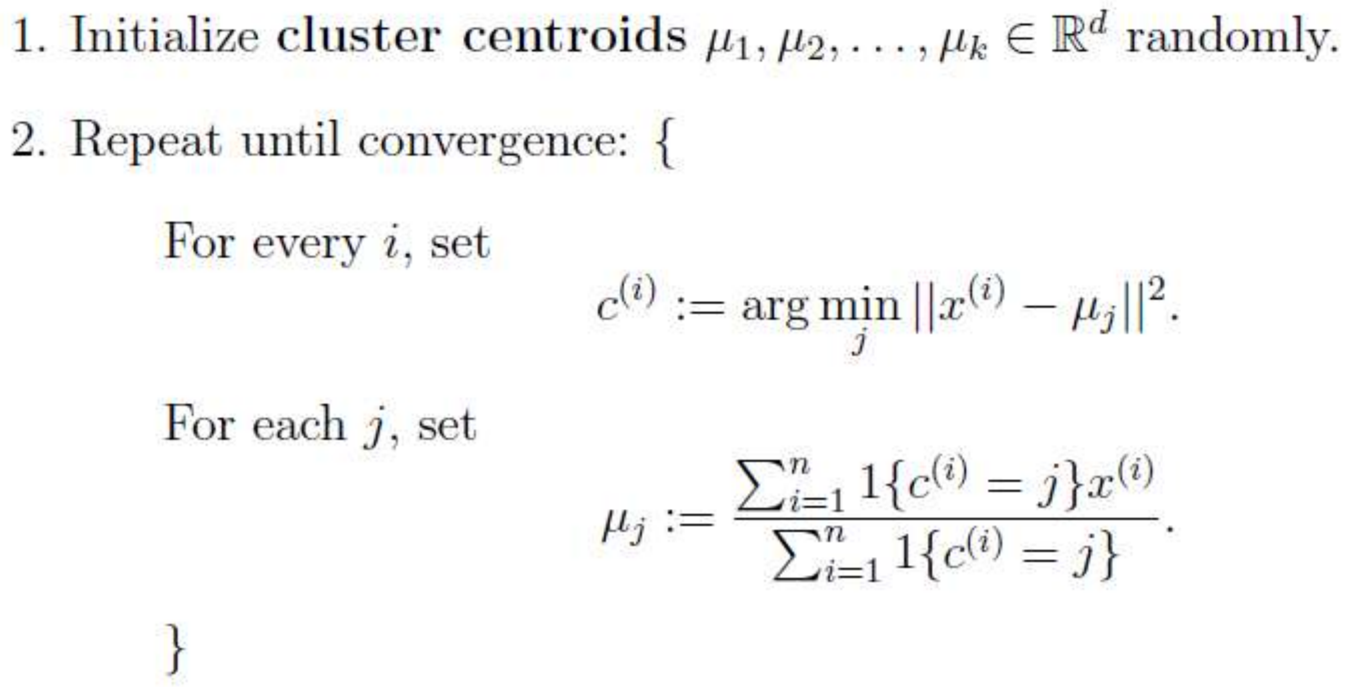
K MEANS: ALGORITHM

- Random Initialisation:
  1) Pick number of clusters k.
  2) Pick k random points as “cluster center”
- Iterate over the following steps:
  1) Assign each data point to its closest cluster center
  2) Recompute cluster centers as the mean of the assigned points.



K MEANS: ALGORITHM

- Random Initialisation:
  1) Pick number of clusters k.
  2) Pick k random points as “cluster center”
- Iterate over the following steps:
  1) Assign each data point to it’s closest cluster center
  2) Recompute cluster centers as the mean of the assigned points.



```
1. Initialize **cluster centroids** $\mu_1, \mu_2, \ldots, \mu_k \in \mathbb{R}^d$ randomly.
2. Repeat until convergence: {
        For every $i$, set
            $c^{(i)} := \arg\min_j ||x^{(i)} - \mu_j||^2.$
        For each $j$, set
            $\mu_j := \frac{\sum_{i=1}^n 1\{c^{(i)} = j\} x^{(i)}}{\sum_{i=1}^n 1\{c^{(i)} = j\}}.$
   }
```

K MEANS: API

K MEANS: CHOOSING K

To find the number of clusters in the data, the user needs to run the k means clustering algorithm for a range of k values and compare the results

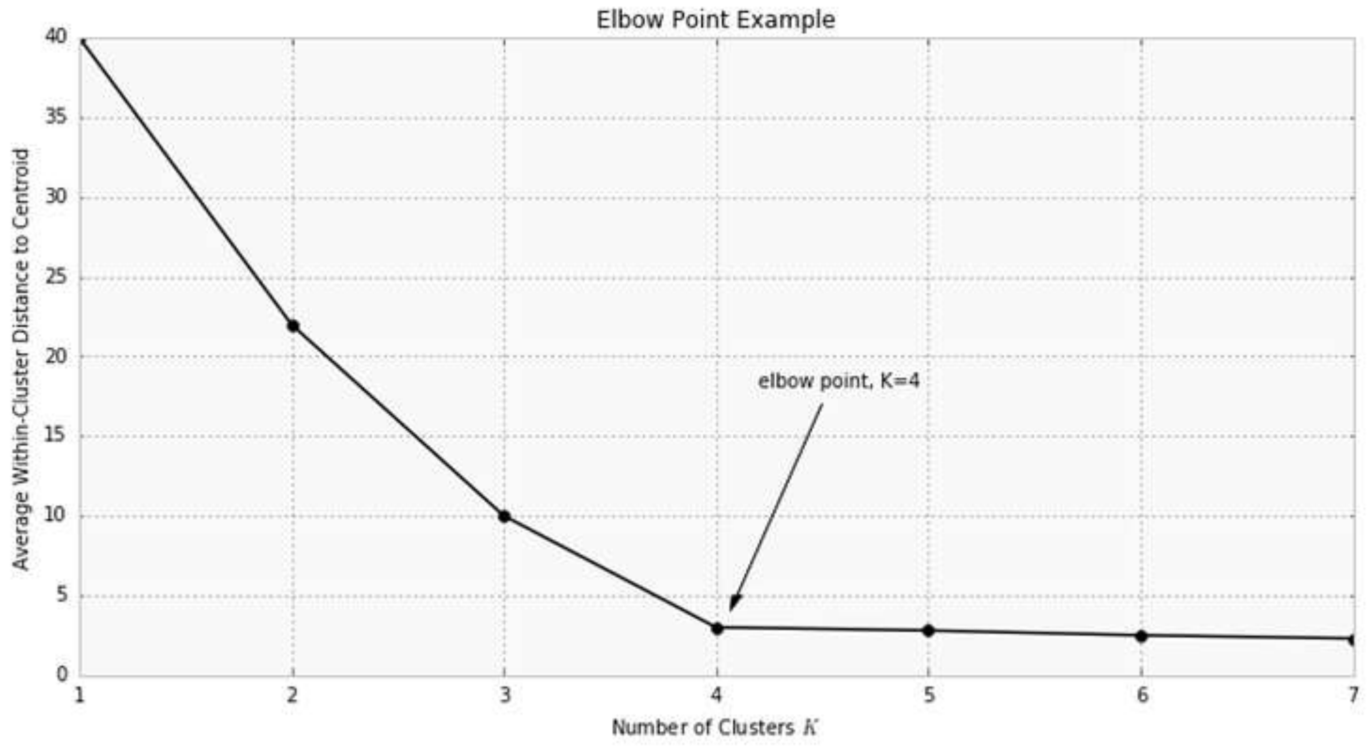
mean distance between data points and their cluster centroid

↑ k => ↓ distance to data points

(reaching zero when k is the same as the number of data points)

The "elbow point" is where the rate of decrease sharply shifts

is used to roughly determine k.



RESTRICTION OF CLUSTER SHAPES

Clusters = Voronoi-diagram of centers

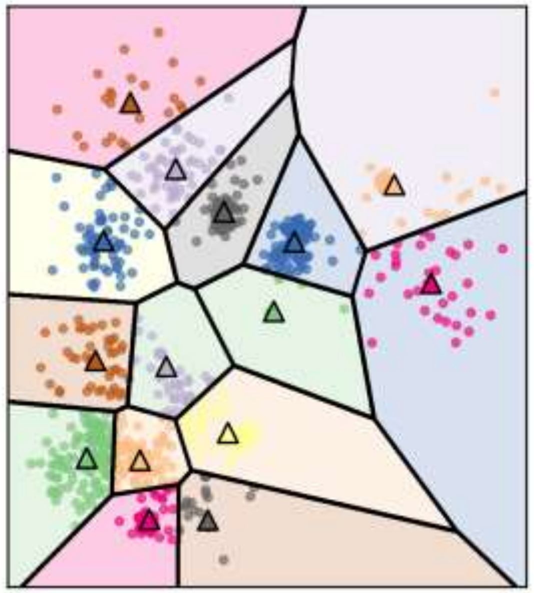
boundary between two neighbouring clusters

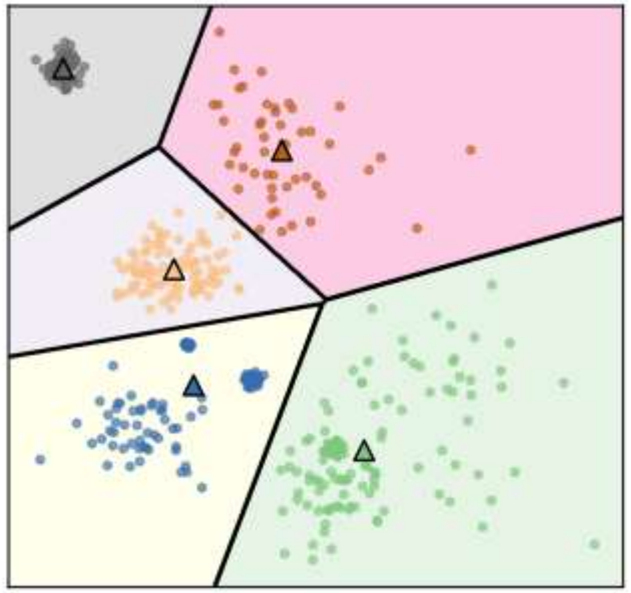
= line along the center of the two cluster centers

The cluster centres completely define the partitions

This restricts the shape of the clusters to convex

(shapes such as "banana" will not occur)



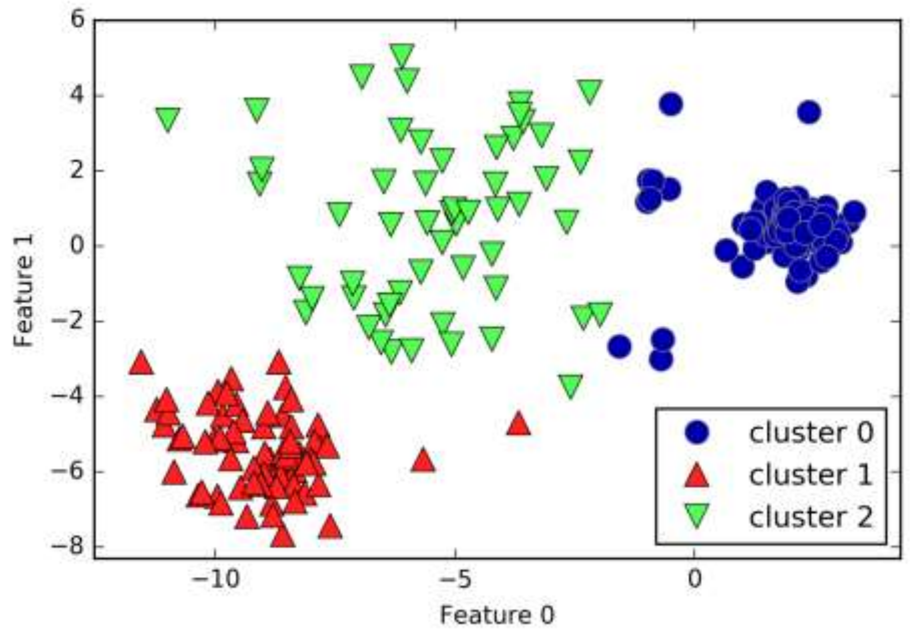


LIMITATIONS OF K-MEANS

Cluster boundaries equidistant to centers

Difficulty in modelling the size of clusters

- dataset created from three Gaussians;
- k means algorithm resulted in three clusters
- boundary between clusters = middle between the clusters



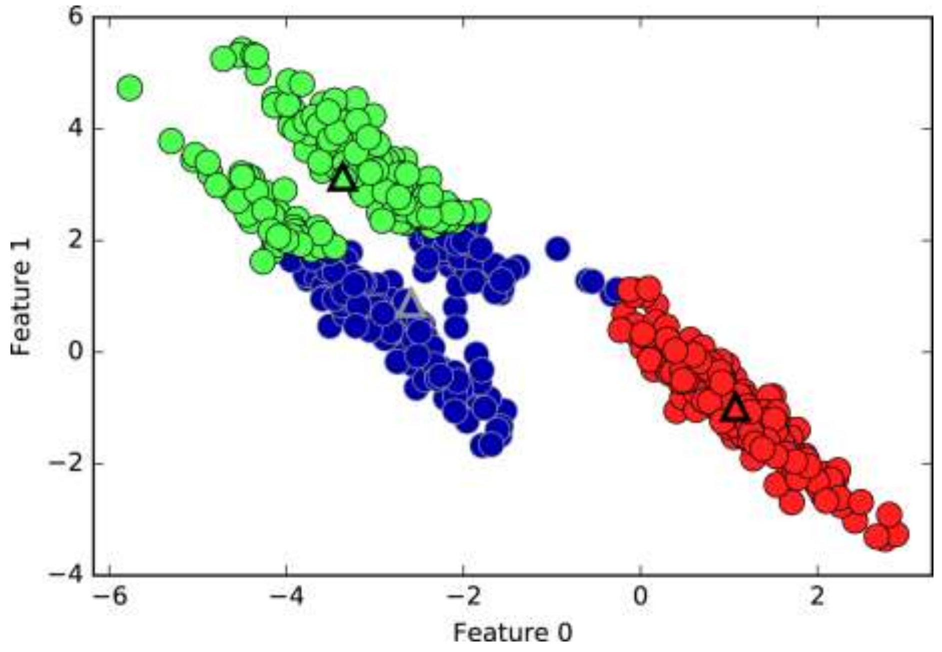
A human eye may view points on the boundaries between the clusters as green

LIMITATIONS OF K-MEANS

Can't model covariances well

dataset = three groups, with strong covariance in one direction

k means fails to pick up these shapes (due to Euclidean distances)

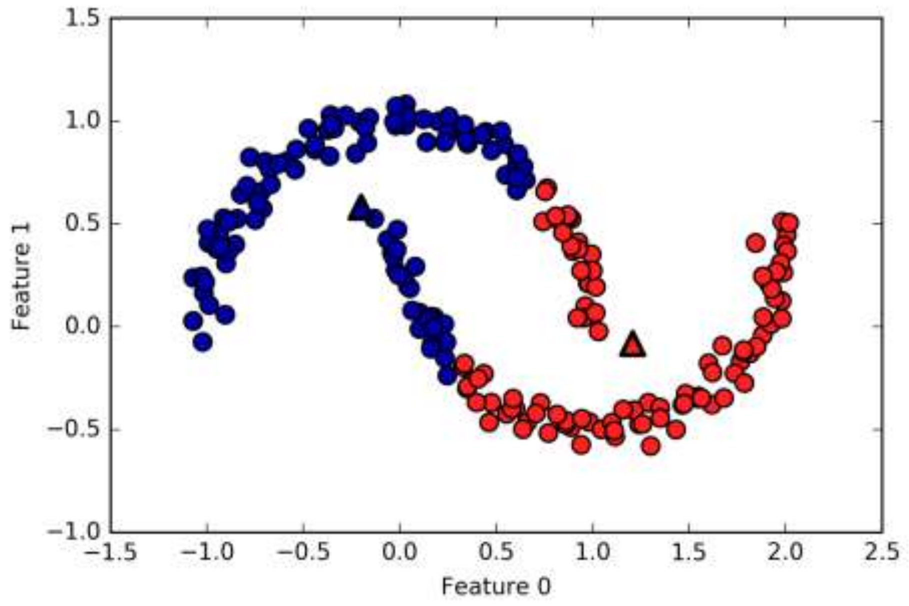


LIMITATIONS OF K-MEANS

Only simple cluster shapes

dataset = two clusters (two moon shapes)

k means uses straight lines for two clusters, as clusters are Voronoi-diagram of centers, i.e. convex, thus fails to capture the two moon shapes



COMPUTATIONAL PROPERTIES

Naive implementation (Lloyd's):

- compute the distance from all data points to all the cluster centers
- n_cluster * n_samples distance calculations per iteration

Fast algorithms:

- Elkan's
- Ying-Yang

Approximate algorithms:

- minibatch K-Means

INITIALISATION

compute random centers → fast k means algorithm

KMeans in **sklearn.cluster**

processing time: initialisation vs clustering

✓ 10 random restarts with different initializations

✓ "k means++" = Greedily add "furthest way" point

picks a point randomly

adds the most furthest away point in the data to it

adds the most further away point to both of these two points

initial cluster center = far away as possible

better optima

speed up the optimization, converges quick

compute a lot of distances

take more time than basic clustering

e.g. big data set - might set init to one (not 10)

to reduce processing time

K MEANS = EM APPLIED TO A NAIVE BAYES MODEL

naive bayes model

class variable which can take on discrete values

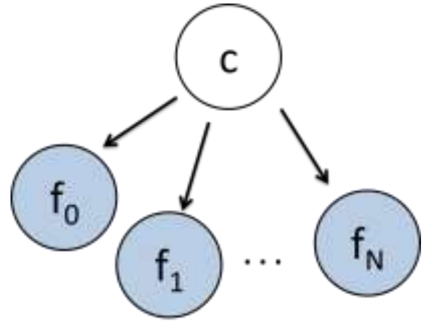
(with domain size "k")

set of feature variables,

each of which can take on a continuous value

conditional probability distributions

$P(f_i = x \mid C = c) \sim N(\mu_{c,i}, 1)$



learning $\mu_{c,i}$ given a dataset with assigned values to features

$\equiv$

running k-means on that dataset

K MEANS = EM APPLIED TO A NAIVE BAYES MODEL

EM, randomly initialize model parameters,

Alternate between:

(E) assigning values to hidden variables, based on parameters

choosing the best values for the class variable based on the features of a given piece of data in your data set

"for each data-point, chose the centroid that it is closest to, by Euclidean distance, and assign that centroid's label."

(M) computing parameters based on fully observed data.

choosing the best parameter values based on the features of a given piece of data in your data set

"take the mean of all the data-points that were labelled as c."

K MEANS = EM APPLIED TO A NAIVE BAYES MODEL

Both k means and EM:

- find a local optimum
- not necessarily going to find a global optimum (random initial values do matter)

Clustering: Methods

- **Partition** based methods
- **Probabilistic model** based methods
- **Density** based methods

MIXTURE MODELS

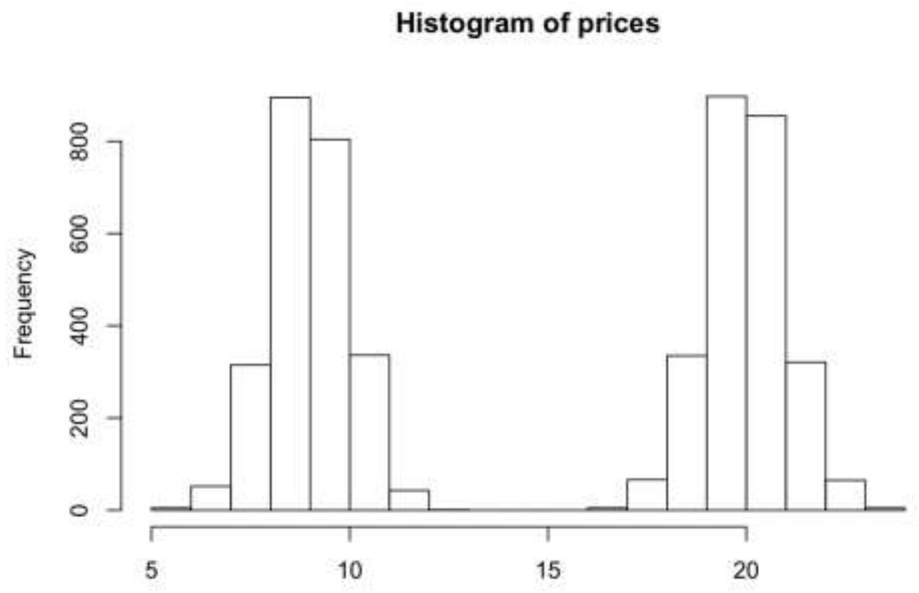
assuming
each sample
comes from
the same
unimodal distribution
is INCORRECT

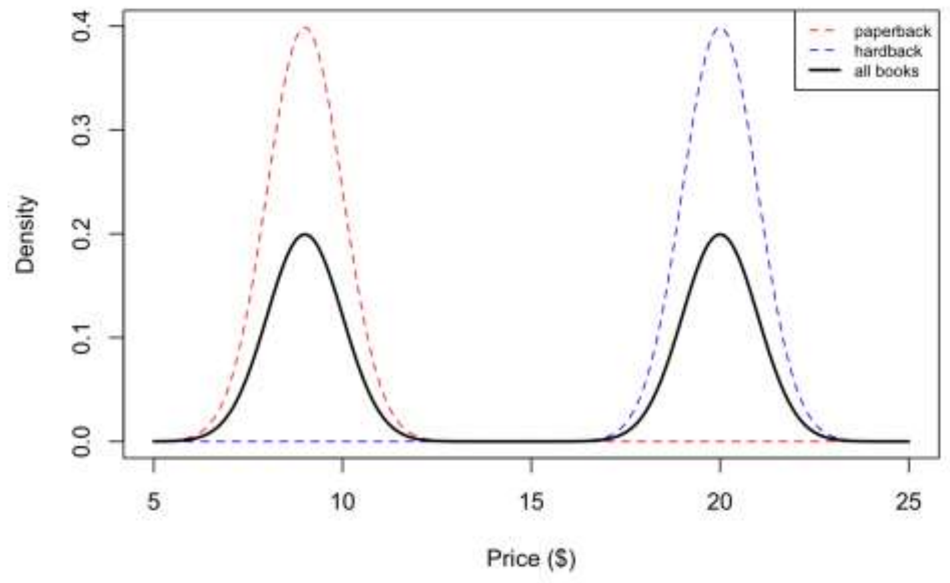
e.g. price of a book,
our model is a mixture of
paperbacks and hardbacks:

histogram = bimodal

mixture components:
N(mean = 9 ,   sd = 1)
N(mean = 20 , sd = 2)

distribution of a randomly
chosen book is not ~***N***(,)
not symmetric

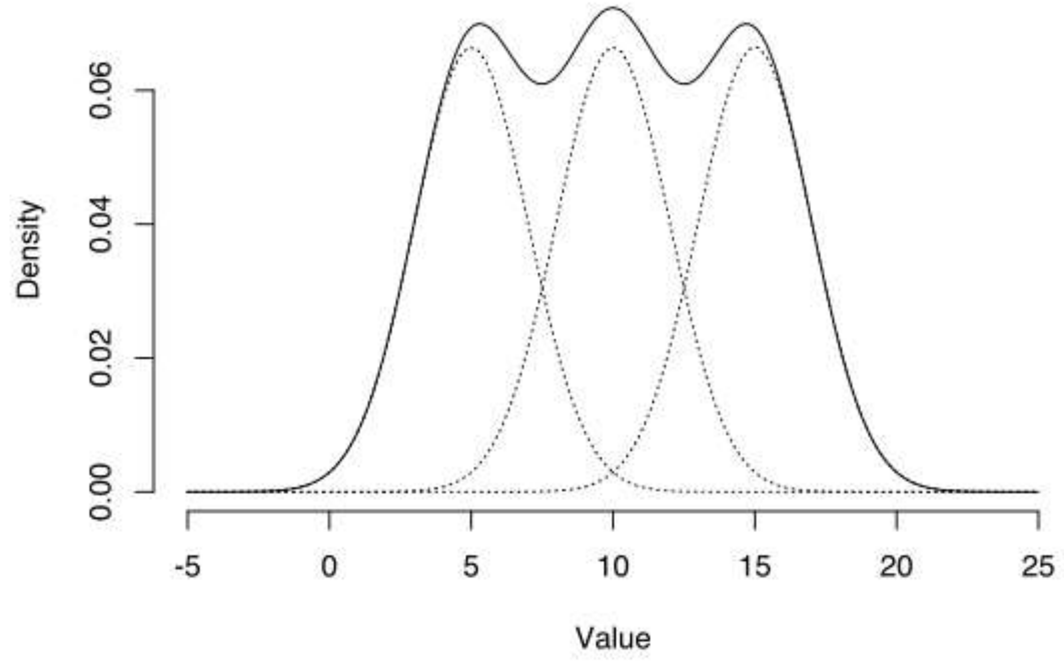




MIXTURE MODELS

UL => subpopulation distribution **un**known

Mixture Model => "learns" subpopulation distribution



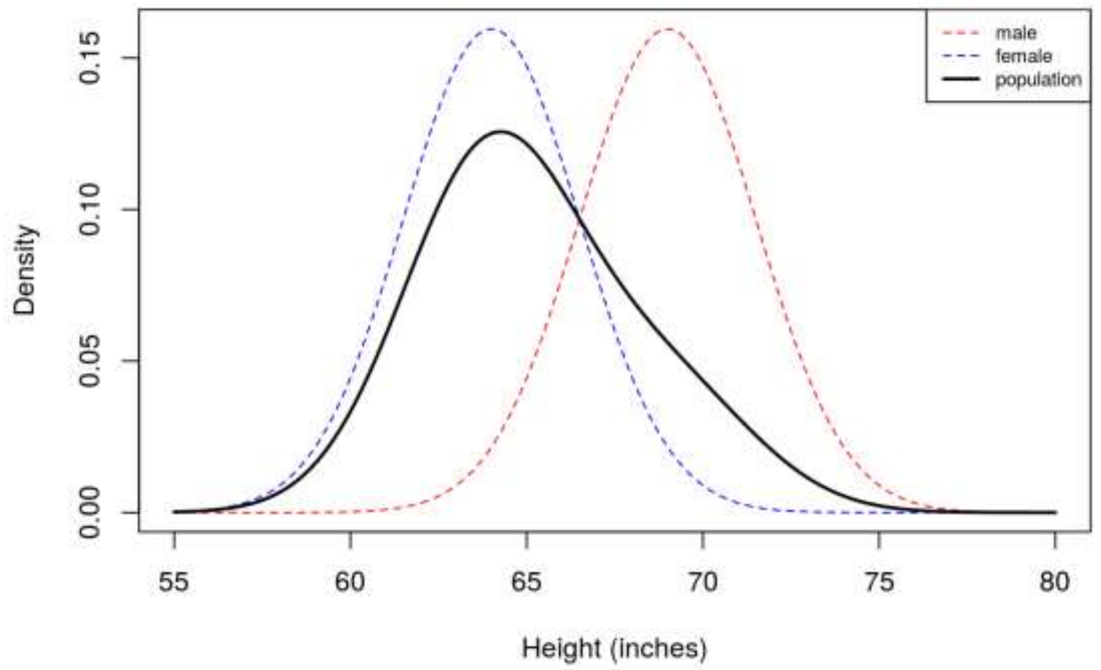
When to use MM?

- data appear multimodal
- Knowledge of data indicates multimodal

MIXTURE MODELS

UL => subpopulation distribution **un**known

Mixture Model => “learns” subpopulation distribution



MIXTURE MODELS

$$p(\mathbf{x}) = \sum_{j=1}^{k} \pi_k p_k(\mathbf{x}|\theta)$$

Mixture models are unsupervised models

Goal: fit a distribution to your data (i.e. a mixture of distributions)

i.e. a weighted sum p_{k},

where p_{k}, is a distribution

& weight is π_k (proportion, must sum to ONE)

Mixture model assumption:

- Data is mixture of small number of known distributions.
- Each mixture component distribution can be learned "simply"
- Each point comes from one particular component

Goal: to learn the component parameters and weights of components

MIXTURE MODELS

Gaussian Mixture Models:

$$p(\mathbf{x}) = \sum_{j=1}^{k} \pi_k \mathcal{N}(\mathbf{x}, \mu_k, \Sigma_k)$$

$$k \sim \mathrm{Mult}(\pi), x \sim \mathcal{N}(\mathbf{x}, \mu_k, \Sigma_k)$$

Similar iterative process to “k means”:

1. assign each data point a probability of being generated from a component “k” Gaussian distribution with weight π_k [distribution is normalized, so the weights sum to one]
2. recompute mean and standard deviation or covariance matrix for each of the component “k” Gaussian distribution based on the points that are assigned to it.

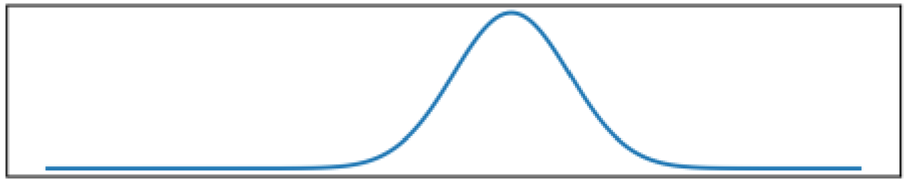
Initialisation differs to “k means”:

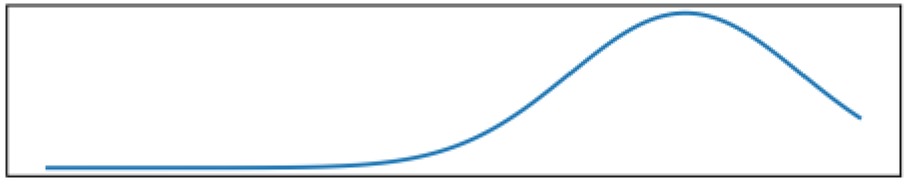
- “k means”: hard assignment: EACH data point belongs to ONE cluster “k”
- “GMM”: soft assignment: Probability that EACH data point to belongs to a component “k” Gaussian distribution

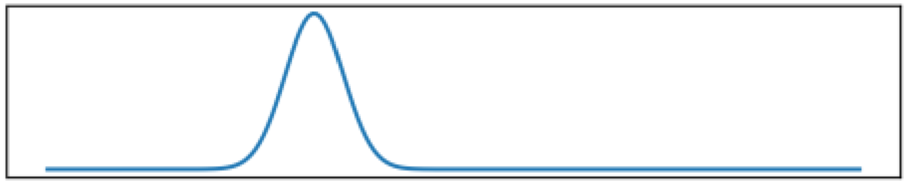
Similar to “k means”: choose “k” components *before* fitting model

MIXTURE MODELS

component gaussian distribuions







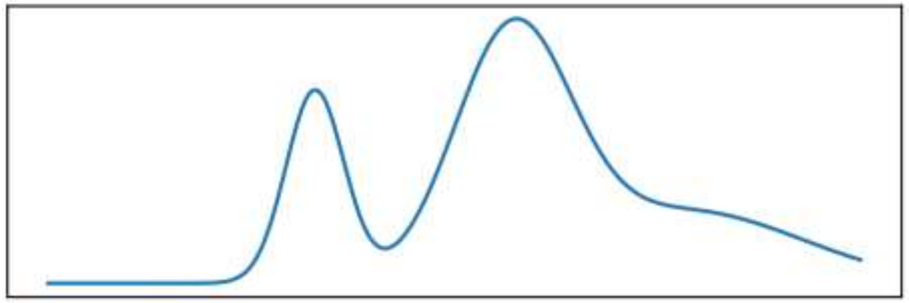
```
from scipy import stats
line = np.linspace(-8, 6, 200)
norm1 = stats.norm(0, 1)
axes[0].plot(line, norm1.pdf(line))
norm2 = stats.norm(3, 2)
axes[1].plot(line, norm2.pdf(line))
norm3 = stats.norm(-3.4, .5)
axes[2].plot(line, norm3.pdf(line))
```

mixture =

weighted sum of top three plots

$$p(\mathbf{x}) = \sum_{j=1}^{k} \pi_k \mathcal{N}(\mathbf{x}, \mu_k, \Sigma_k)$$

```
plt.plot(line,
          .5 * norm1.pdf(line)
        + .3 * norm2.pdf(line)
        + .2 * norm3.pdf(line)
        )
```

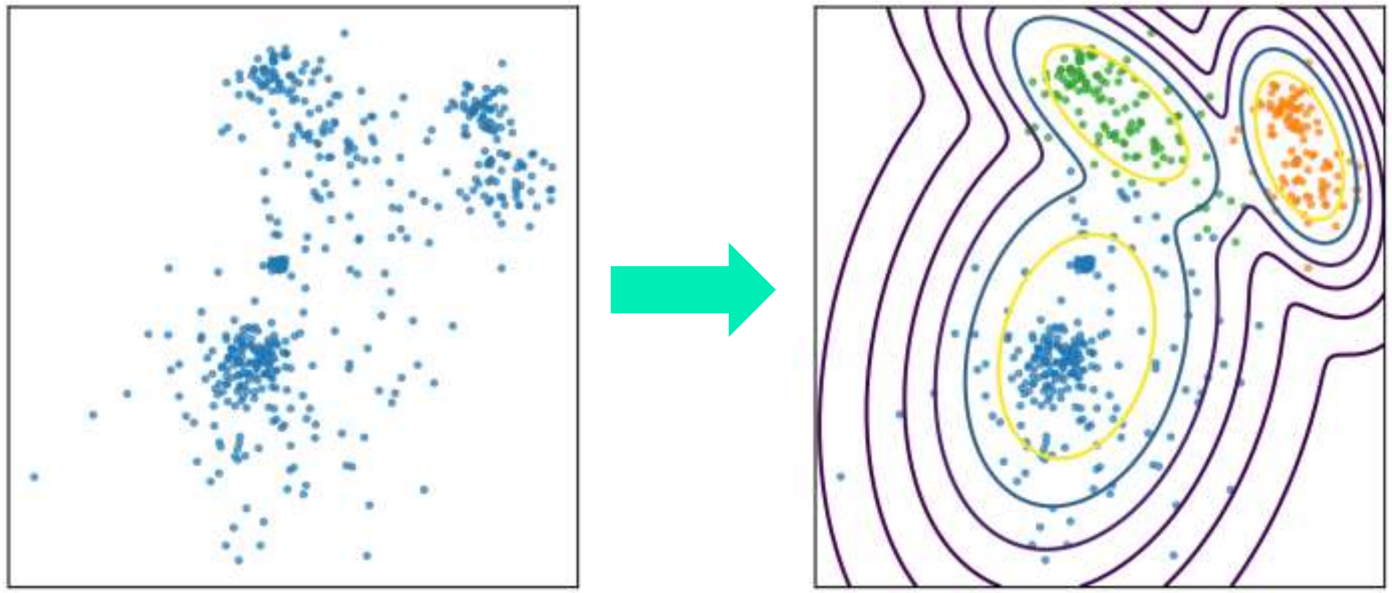


MIXTURE MODELS

Why?

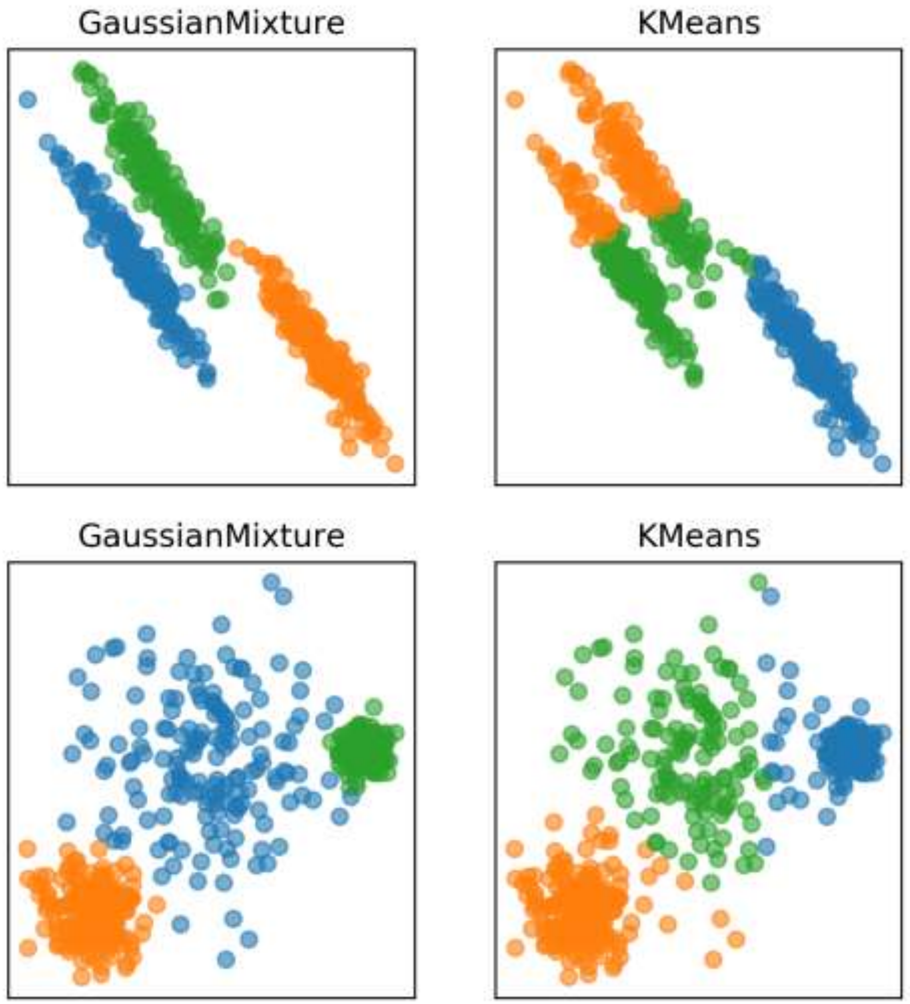
- Clustering (components are clusters)
  - Assign every data point to a “k” component Gaussian, such that x has the highest probability of belonging to the “k” component Gaussian
- Parametric density model
  - Probability distribution that models data, i.e. p(x)
- Outlier Detection
  - Given a new data point, how likely is it to observe this data point, given the fitted model

MIXTURE MODELS



GMM VS KMEANS

Gaussian mixture model can learn the covariance structure of the components whilst “k means” failed

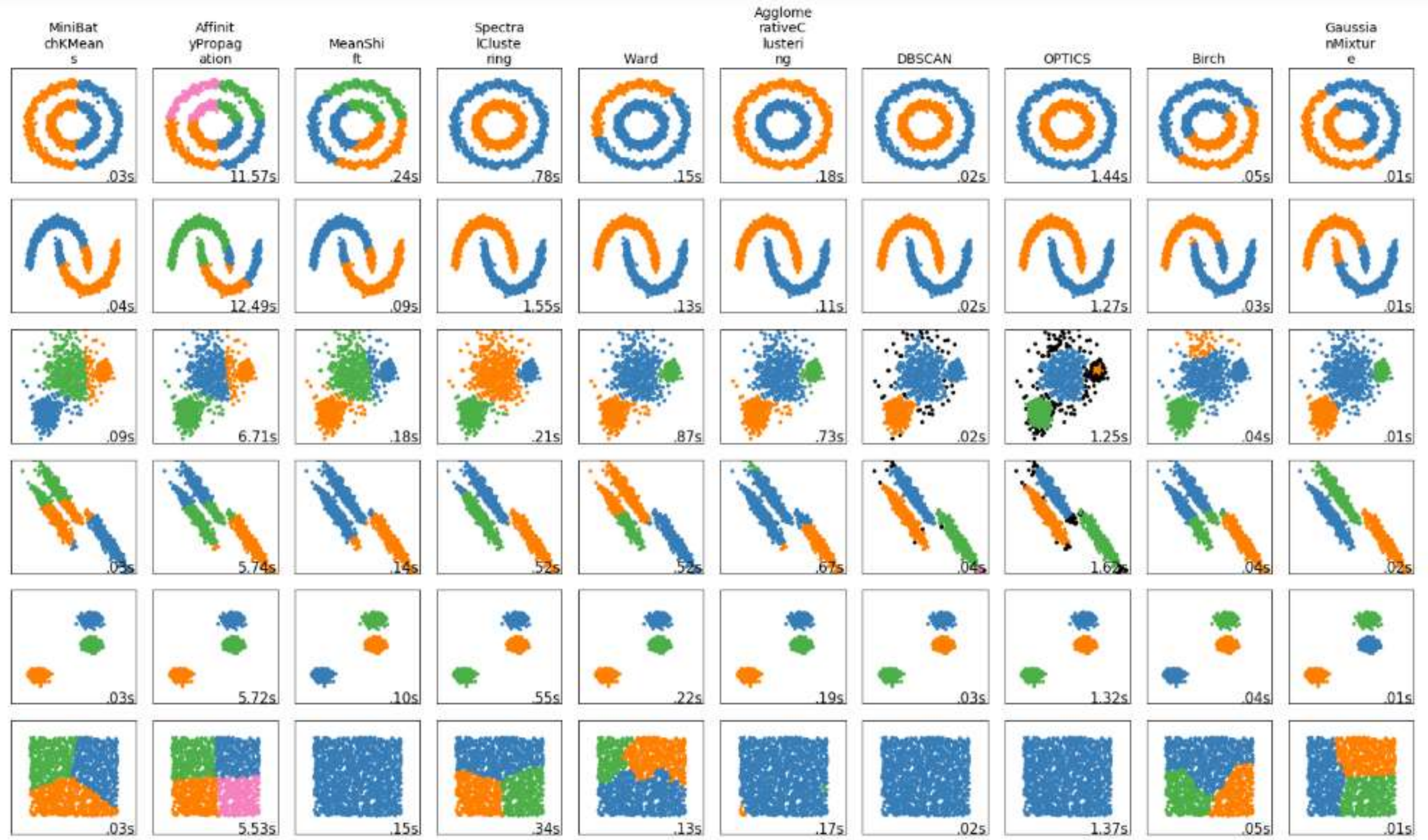


In the real world, your data will not be generated by Gaussian and you won’t know how many components there might be.

GMM can be more unstable, is more expensive to compute.

Clustering:

comparing algorithms


MiniBatchKMeans
AffinityPropagation
MeanShift
SpectralClustering
Ward
AgglomerativeClustering
DBSCAN
OPTICS
Birch
GaussianMixture
.03s 11.57s .24s .78s .15s .18s .02s 1.44s .05s .01s
.04s 12.49s .09s 1.55s .13s .11s .02s 1.27s .03s .01s
.09s 6.71s .18s .21s .87s .73s .02s 1.25s .04s .01s
.03s 5.74s .14s .52s .52s .67s .04s 1.62s .04s .02s
.03s 5.72s .10s .55s .22s .19s .03s 1.32s .04s .01s
.03s 5.53s .15s .34s .13s .17s .02s 1.37s .05s .01s