GENERAL ASSEMBLY

UNSUPERVISED LEARNING

Sri Kanajan

LEARNING OBJECTIVES

- Supervised vs unsupervised algorithms
- Understand and apply k-means clustering
- Density-based clustering: DBSCAN
- Silhouette Metric

OPENING

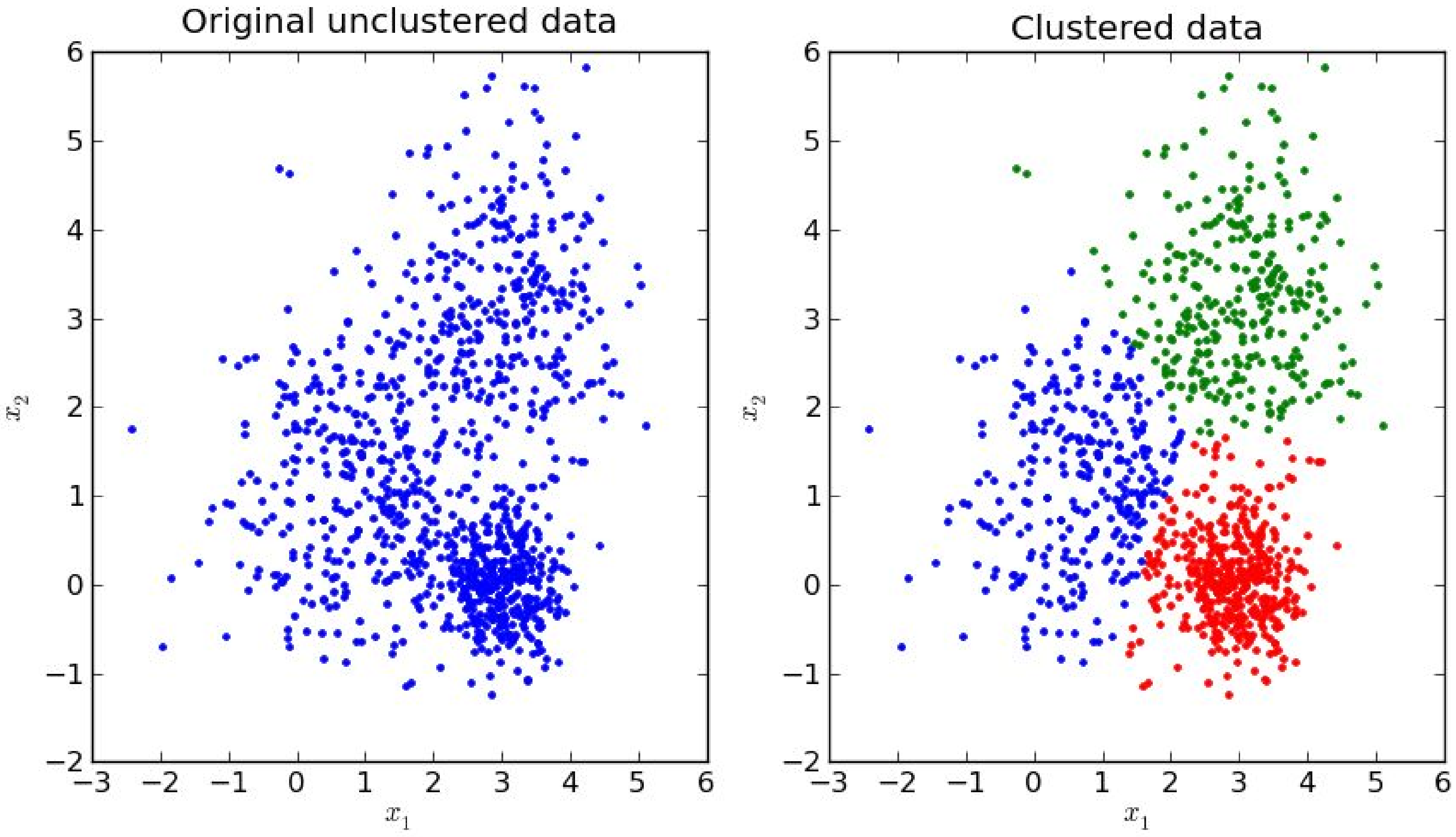
UNSUPERVISED LEARNING

UNSUPERVISED LEARNING

- So far all the algorithms we have used are *supervised*: each observation (row of data) came with one or more *labels*, either *categorical variables* (classes) or *measurements* (regression)
- **Unsupervised learning** has a different goal: **feature discovery**
- **Clustering** is a common and fundamental example of unsupervised learning
- **Clustering** algorithms try to find meaningful groups within data

CLUSTERING

CLUSTERING: Centroids



Source: http://stackoverflow.com/questions/24645068/k-means-clustering-major-understanding-issue

ACTIVITY: KNOWLEDGE CHECK

EXERCISE

ANSWER THE FOLLOWING QUESTIONS

1. How is unsupervised learning different from classification?
2. Can you think of a real-world clustering application?

DELIVERABLE

Answers to the above questions

K-MEANS: CENTRIOD CLUSTERING

Q: How does the algorithm work?

THE BASIC K-MEANS ALGORITHM

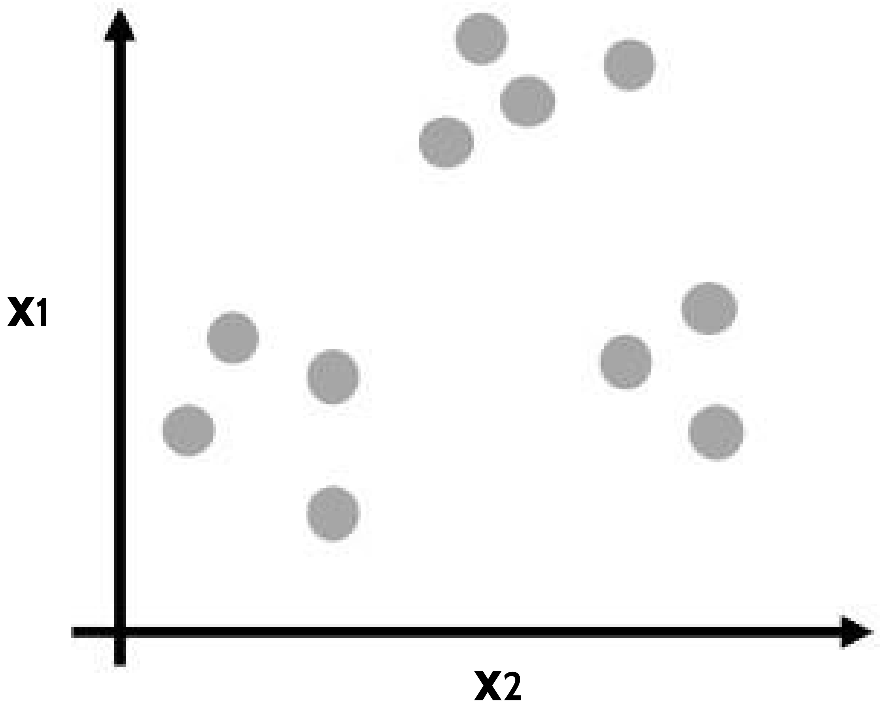
1) choose k initial centroids (note that k is an input)

2) for each point:
    - assign point to nearest centroid

3) recalculate centroid positions
4) repeat steps 2-3 until stopping criteria met

1) choose k initial centroids (note that k is an input)

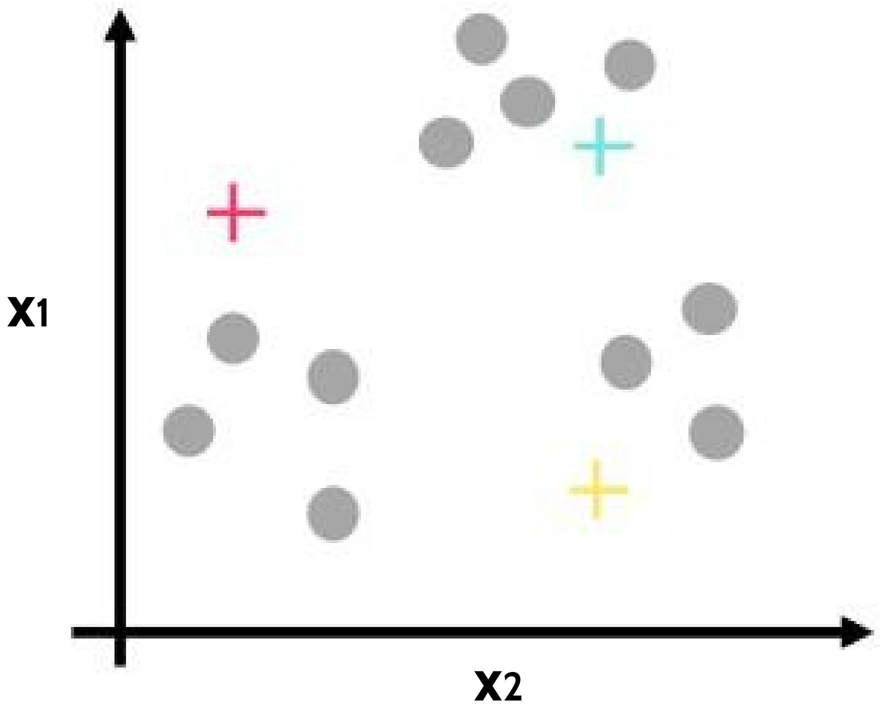
2) for each point:
   - find distance to each centroid
   - assign point to nearest centroid



3) recalculate centroid positions
4) repeat steps 2-3 until stopping criteria met

1) choose k initial centroids (note that k is an input)

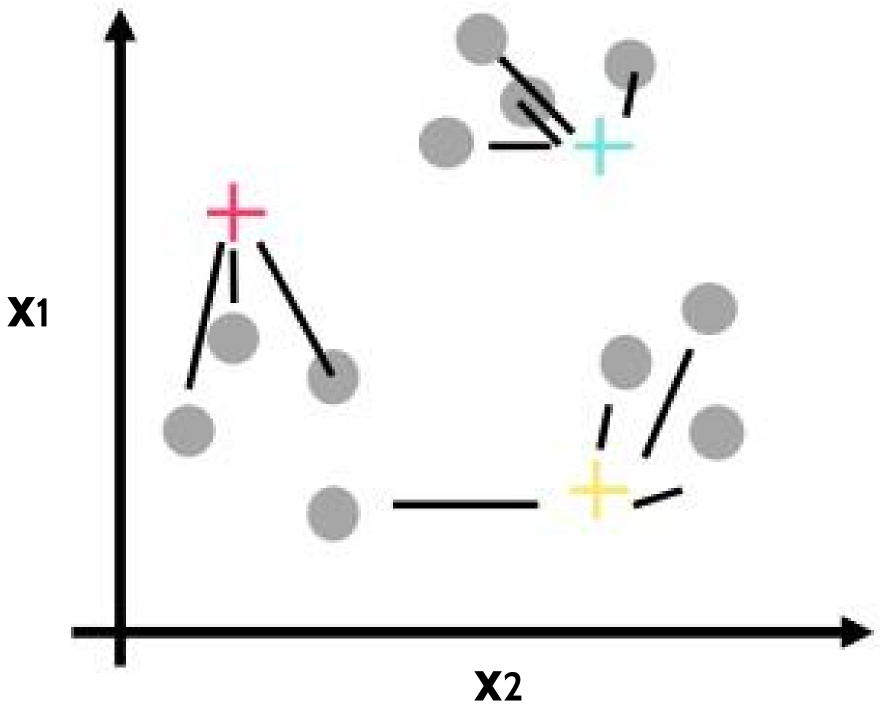
2) for each point:
   - find distance to each centroid
   - assign point to nearest centroid



3) recalculate centroid positions
4) repeat steps 2-3 until stopping criteria met

1) choose k initial centroids (note that k is an input)

2) for each point:
   - find distance to each centroid
   - assign point to nearest centroid

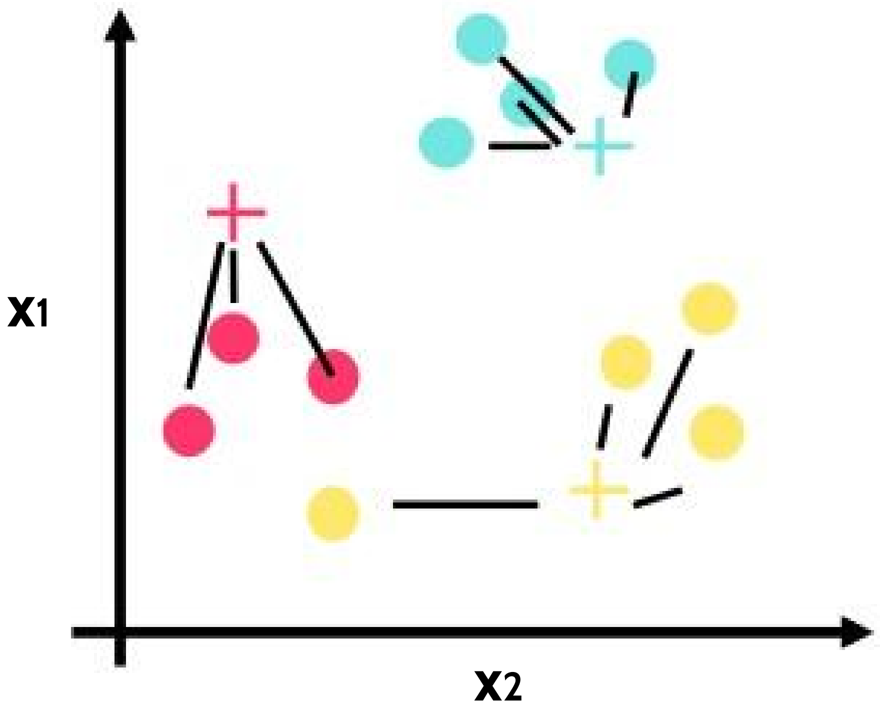


3) recalculate centroid positions
4) repeat steps 2-3 until stopping criteria met

1) choose k initial centroids (note that k is an input)

2) for each point:
 - find distance to each centroid
 - assign point to nearest centroid

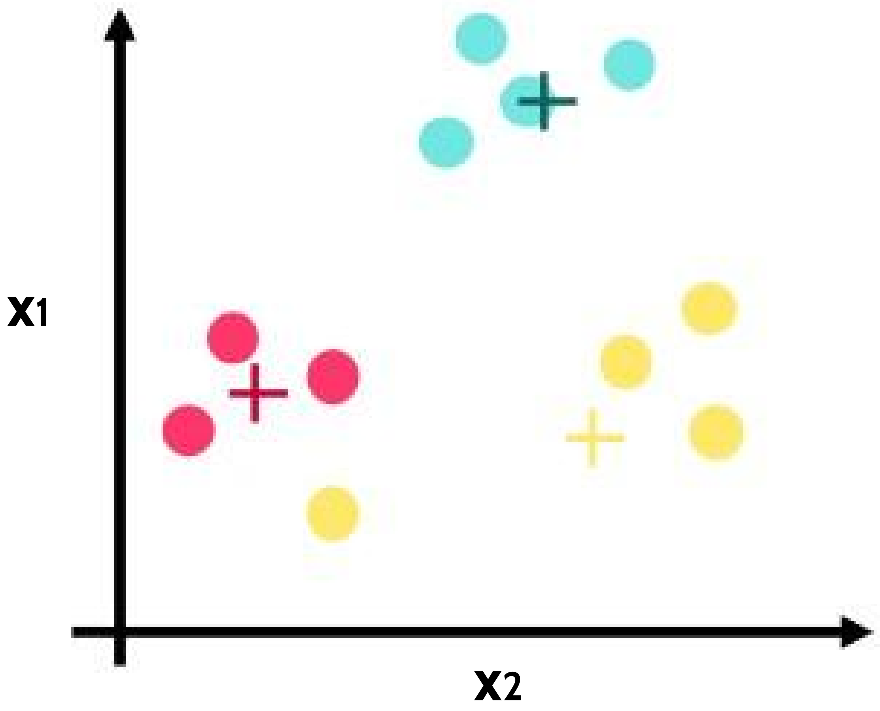
3) recalculate centroid positions
4) repeat steps 2-3 until stopping criteria met



1) choose k initial centroids (note that k is an input)

2) for each point:
   - find distance to each centroid
   - assign point to nearest centroid

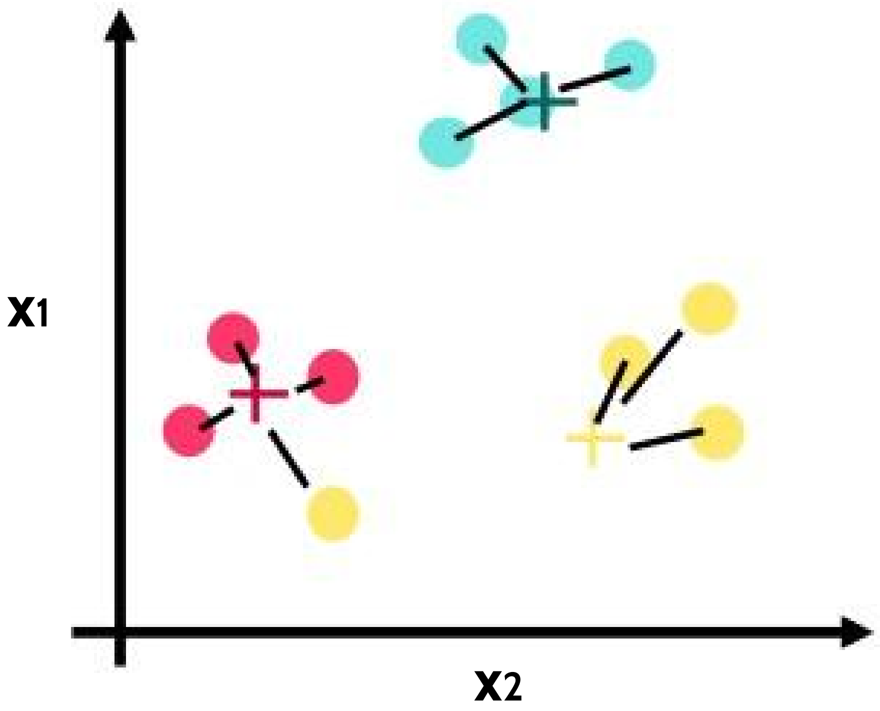
3) recalculate centroid positions
4) repeat steps 2-3 until stopping criteria met



1) choose k initial centroids (note that k is an input)

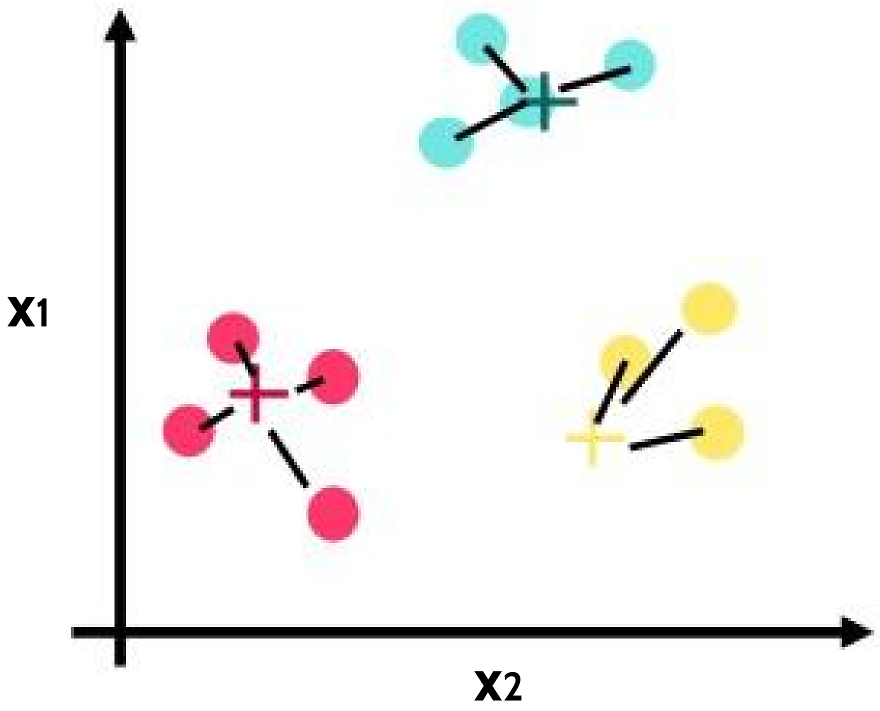
2) for each point:
   - find distance to each centroid
   - assign point to nearest centroid

3) recalculate centroid positions
4) repeat steps 2-3 until stopping criteria met



1) choose k initial centroids (note that k is an input)

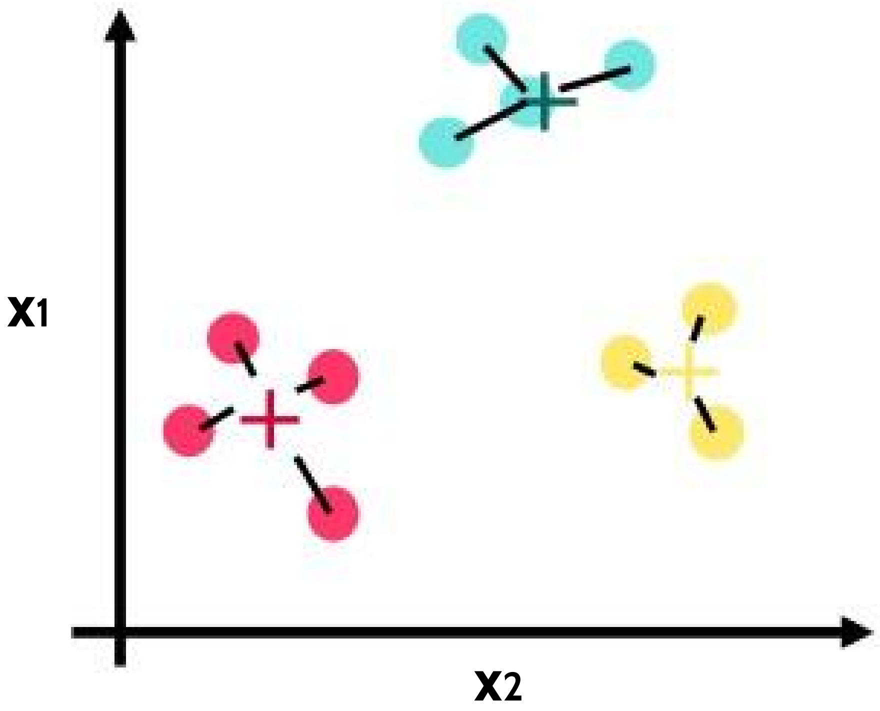
2) for each point:
   - find distance to each centroid
   - assign point to nearest centroid



3) recalculate centroid positions
4) repeat steps 2-3 until stopping criteria met

1) choose k initial centroids (note that k is an input)

2) for each point:
   - find distance to each centroid
   - assign point to nearest centroid

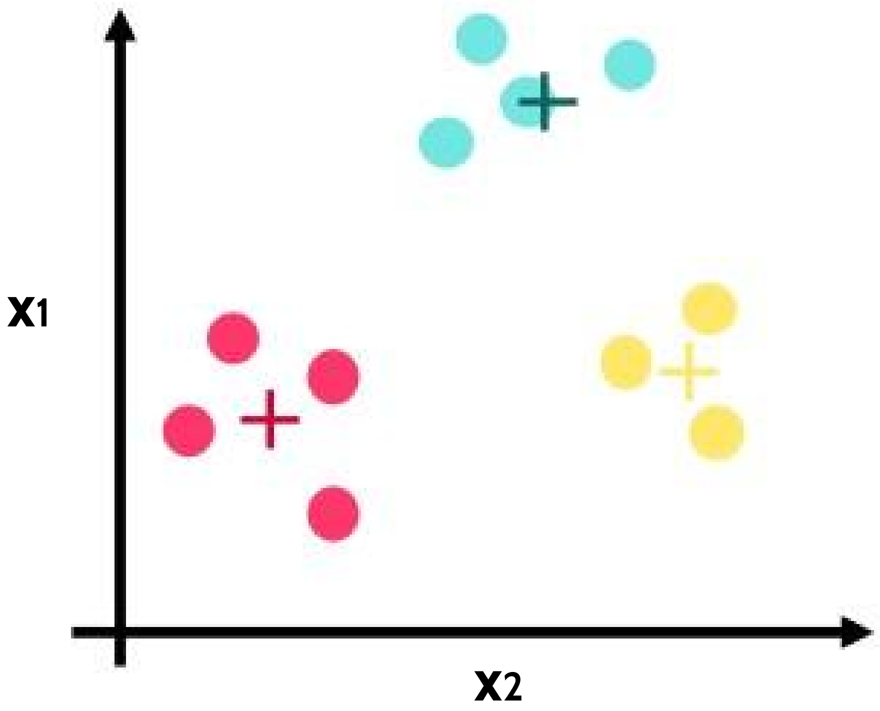


3) recalculate centroid positions

4) repeat steps 2-3 until stopping criteria met

1) choose k initial centroids (note that k is an input)

2) for each point:
   - find distance to each centroid
   - assign point to nearest centroid



3) recalculate centroid positions
4) repeat steps 2-3 until stopping criteria met

Q: How do you determine which centroid a given point is most similar to?

Q: How do you determine which centroid a given point is most similar to?

The similarity criterion is determined by the measure we choose.

In the case of k-means clustering, one similarity metric is the Euclidian distance:

Q: How do we re-compute the positions of the centers at each iteration of the algorithm?

A: By calculating the centroid (i.e., the geometric center)

This is done by taking the average of each index of vectors

Centroid of [1, 4, 2] and [6, 4, 2] is

[(1 + 6) / 2, (4 + 4) / 2, (2 + 2) / 2] == [3.5, 4, 2]

STEP 4 - CONVERGENCE

We iterate until some stopping criteria are met; in general, suitable convergence is achieved in a small number of steps.

K-MEANS CLUSTERING

- k-Means clustering is a popular centroid-based clustering algorithm
- Basic idea: find *k* clusters in the data centrally located around various mean points
- Awesome Demo

K-MEANS CLUSTERING

- from sklearn.cluster import KMeans
- est = KMeans(n_clusters=3)
- est.fit(X)
- labels = est.labels_

Let's try it out!

ACTIVITY: KNOWLEDGE CHECK

EXERCISE

ANSWER THE FOLLOWING QUESTIONS

1. How do we assign meaning to the clusters we find?
2. Do clusters always have meaning?

DELIVERABLE

Answers to the above questions

K-MEANS CLUSTERING

- Assumptions are important! k-Means assumes:
    - k is the correct number of clusters
    - the data is isotropically distributed (circular/spherical distribution)
    - the variance is the same for each variable
    - clusters are roughly the same size

Nice counterexamples / cases where assumptions are not met:
- http://varianceexplained.org/r/kmeans-free-lunch/
- Scikit-Learn Examples

CLUSTERING METRICS

CLUSTERING METRICS

- As usual we need a metric to evaluate model fit

- For clustering we use a metric called the Silhouette Coefficient
  - **a** is the mean distance between a sample and all other points in the cluster
  - **b** is the mean distance between a sample and all other points in the *nearest* cluster

- The Silhouette Coefficient is:

$$\frac{b - a}{\max(a, b)}$$

- Ranges between 1 and -1
- Average over all points to judge the cluster algorithm

CLUSTERING METRICS

- from sklearn import metrics
- from sklearn.cluster import KMeans
- kmeans_model = KMeans(n_clusters=3, random_state=1).fit(X)
- labels = kmeans_model.labels_
- metrics.silhouette_score(X, labels, metric='euclidean')

DBSCAN: DENSITY BASED CLUSTERING

DBSCAN CLUSTERING

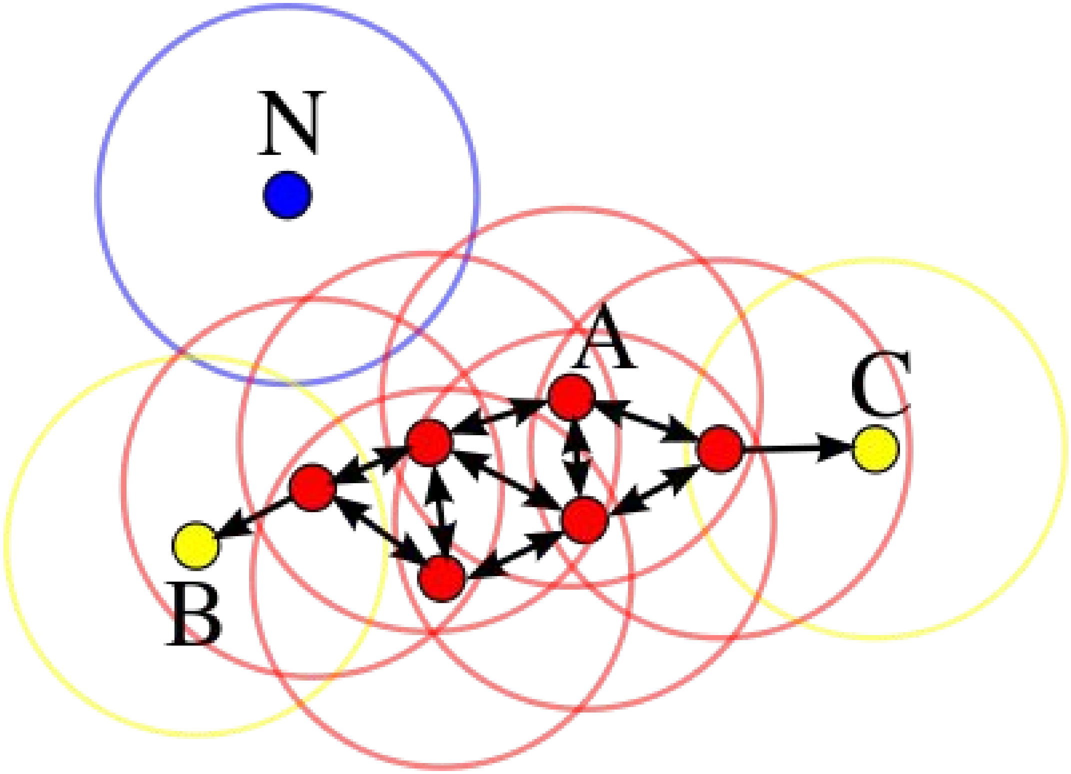
- DBSCAN: Density-based spatial clustering of applications with noise (1996)

- Main idea: Group together closely-packed points by identifying
  - Core points
  - Reachable points
  - Outliers (not reachable)

- Two parameters:
  - min_samples
  - eps

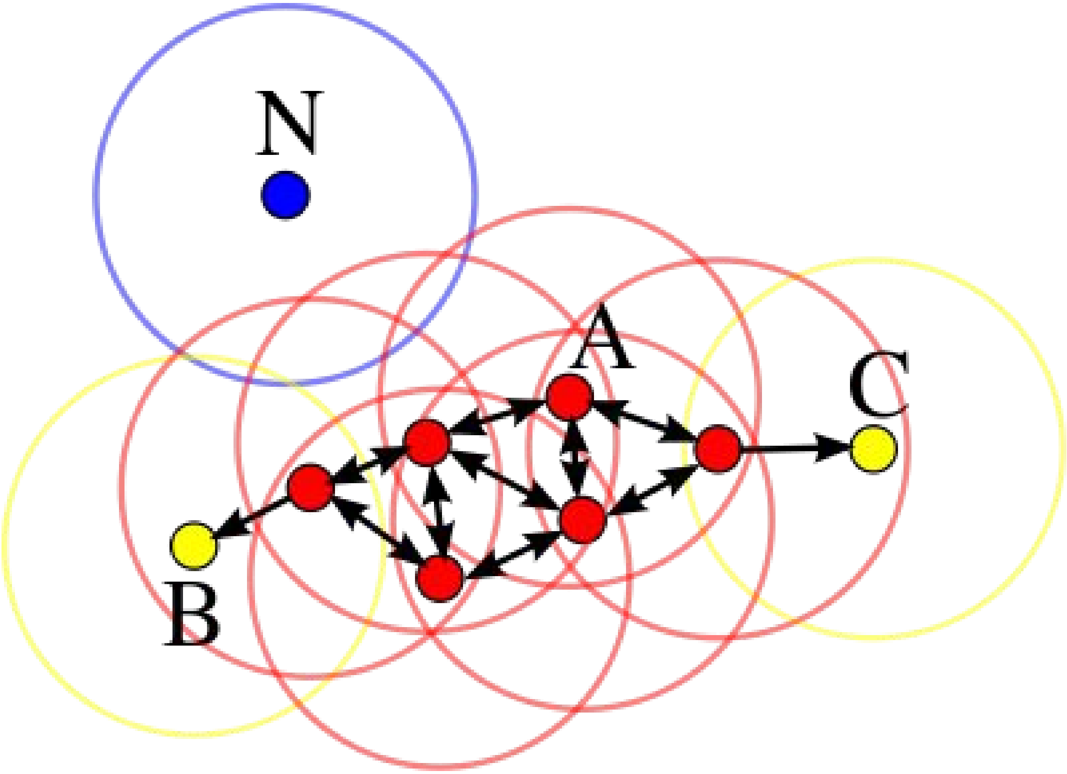
DBSCAN CLUSTERING

- Core points: at least **min_samples** points within **eps** of the core point
  - Such points are *directly reachable* from the core point
- Reachable: point *q* is reachable from *p* if there is a path of core points from *p* to *q*
- Outlier: not reachable



DBSCAN CLUSTERING

- A cluster is a collection of connected core and reachable points



CLUSTERING: Density-Based

- Another example: Page 6
- Awesome Demo

ACTIVITY: KNOWLEDGE CHECK

EXERCISE

ANSWER THE FOLLOWING QUESTIONS

1. How does DBSCAN differ from k-means?

DELIVERABLE

Answers to the above questions

DBSCAN CLUSTERING

- from sklearn.cluster import DBSCAN
- est = DBSCAN(eps=0.5, min_samples=10)
- est.fit(X)
- labels = est.labels_

Let’s try it out!

DBSCAN CLUSTERING

- DBSCAN advantages:
  - Can find arbitrarily-shaped clusters
  - Don’t have to specify number of clusters
  - Robust to outliers

- DBSCAN disadvantages:
  - Doesn’t work well when clusters are of varying densities
    - hard to chose parameters that work for all clusters
  - Can be hard to chose correct parameters regardless

ACTIVITY: CLUSTERING USERS

EXERCISE

ANSWER THE FOLLOWING QUESTIONS

1. How does DBSCAN differ from k-means?

DELIVERABLE

Answers to the above questions

TOPIC REVIEW

REVIEW AND NEXT STEPS

- Clustering is used to discover features, e.g. segment users or assign labels (such as species)

- Clustering may be the goal (user marketing) or a step in a data science pipeline

COURSE

BEFORE NEXT CLASS

LESSON

EXIT TICKET

DON'T FORGET TO FILL OUT YOUR EXIT TICKET