CS 60050
Machine Learning

Clustering

Some material borrowed from course materials of Andrew Ng and Jing Gao

Unsupervised learning

- Given a set of unlabeled data points / items
- Find patterns or structure in the data
- Clustering: automatically group the data points / items into groups of ‘similar’ or ‘related’ points
- Main challenges
  - How to measure similarity?
  - What is the ideal number of clusters? Few larger clusters, or more number of smaller clusters?

Motivations for Clustering

- Understanding the data better
  - Grouping Web search results into clusters, each of which captures a particular aspect of the query
  - Segment the market or customers of a service
- As precursor for some other application
  - Summarization and data compression
  - Recommendation

Different types of clustering

- Partitional
  - Divide set of items into non-overlapping subsets
  - Each item will be member of one subset

- Overlapping
  - Divide set of items into potentially overlapping subsets
  - Each item can simultaneously belong to multiple subsets

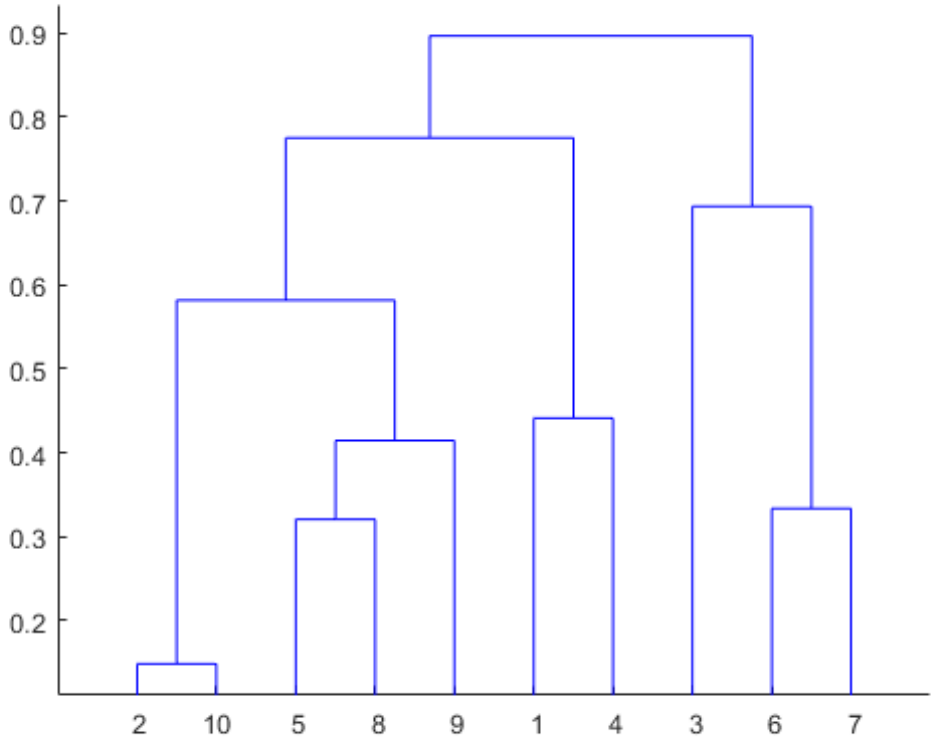
Different types of clustering

- Fuzzy
  - Every item belongs to every cluster with a membership weight between 0 (absolutely does not belong) and 1 (absolutely belongs)
  - Usual constraint: sum of weights for each individual item should be 1
  - Convert to partitional clustering: assign every item to that cluster for which its membership weight is highest

Different types of clustering

- Hierarchical
  - Set of nested clusters, where one larger cluster can contain smaller clusters
  - Organized as a tree (dendrogram): leaf nodes are singleton clusters containing individual items, each intermediate node is union of its children sub-clusters
  - A sequence of partitional clusterings – cut the dendrogram at a certain level to get a partitional clustering

An example dendrogram



Different types of clustering

- Complete vs. partial
  - A complete clustering assigns every item to one or more clusters
  - A partial clustering may not assign some items to any cluster (e.g., outliers, items that are not sufficiently similar to any other item)

Types of clustering methods

- Prototype-based
  - Each cluster defined by a prototype (centroid or medoid), i.e., the most representative point in the cluster
  - A cluster is the set of items in which each item is closer (more similar) to the prototype of this cluster, than to the prototype of any other cluster
  - Example method: K-means

Types of clustering methods

- Density-based
  - Assumes items distributed in a space where ‘similar’ items are placed close to each other (e.g., feature space)
  - A cluster is a dense region of items, that is surrounded by a region of low density
  - Example method: DBSCAN

Types of clustering methods

- Graph-based
  - Assumes items represented as a graph/network where items are nodes, and ‘similar’ items are linked via edges
  - A cluster is a group of nodes having more and / or better connections among its members, than between its members and the rest of the network
  - Also called ‘community structure’ in networks
  - Example method: Algorithm by Girvan and Newman

We are applying clustering
in this lecture itself.

How?

K-means clustering

K-means

- Prototype-based, partitioning technique
- Finds a user-specified number of clusters (K)
- Each cluster represented by its centroid item

- There have been extensions where number of clusters is not needed as input

K-means algorithm

Randomly initialize K cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$

Repeat {

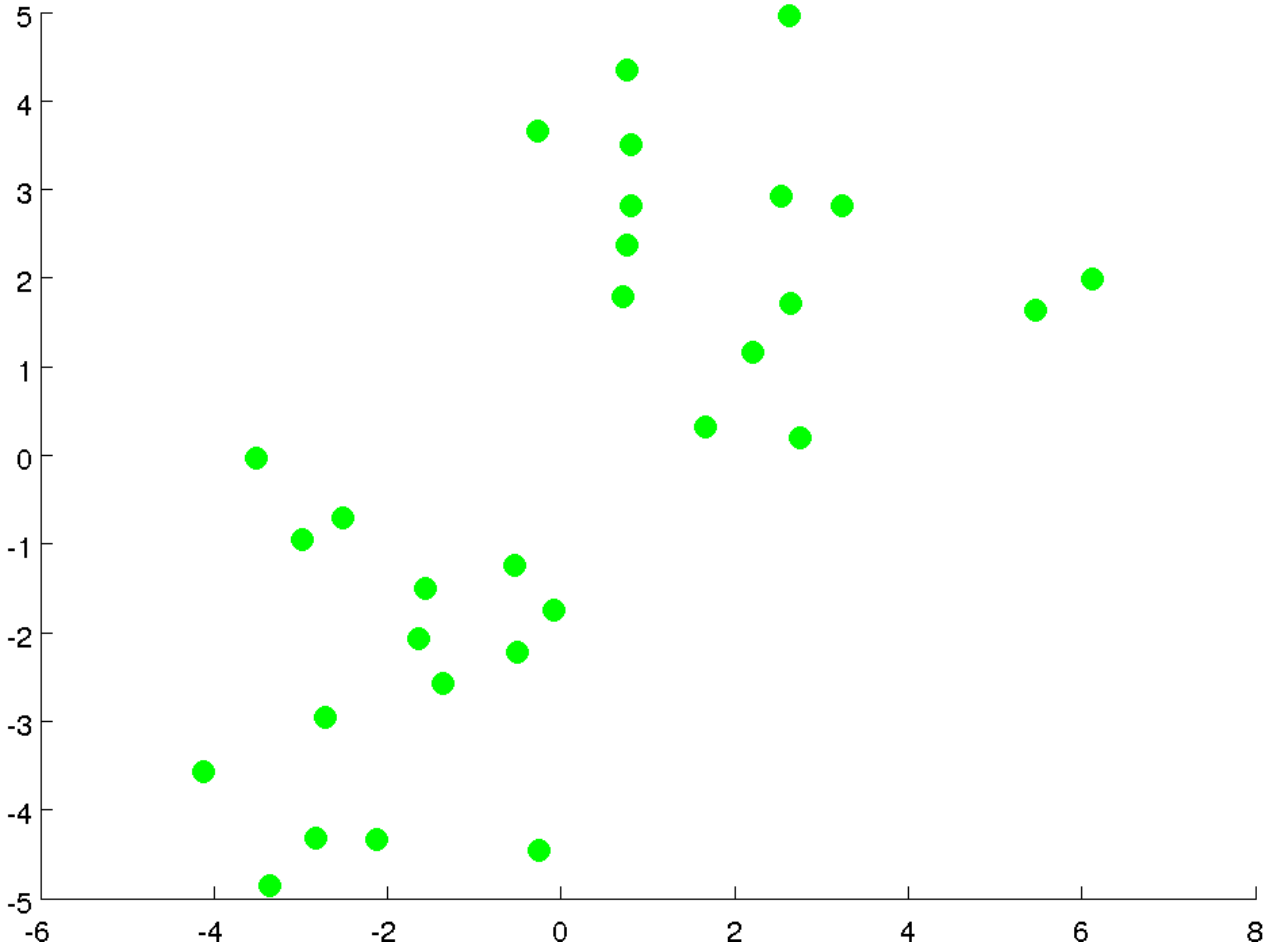
for i = 1 to m

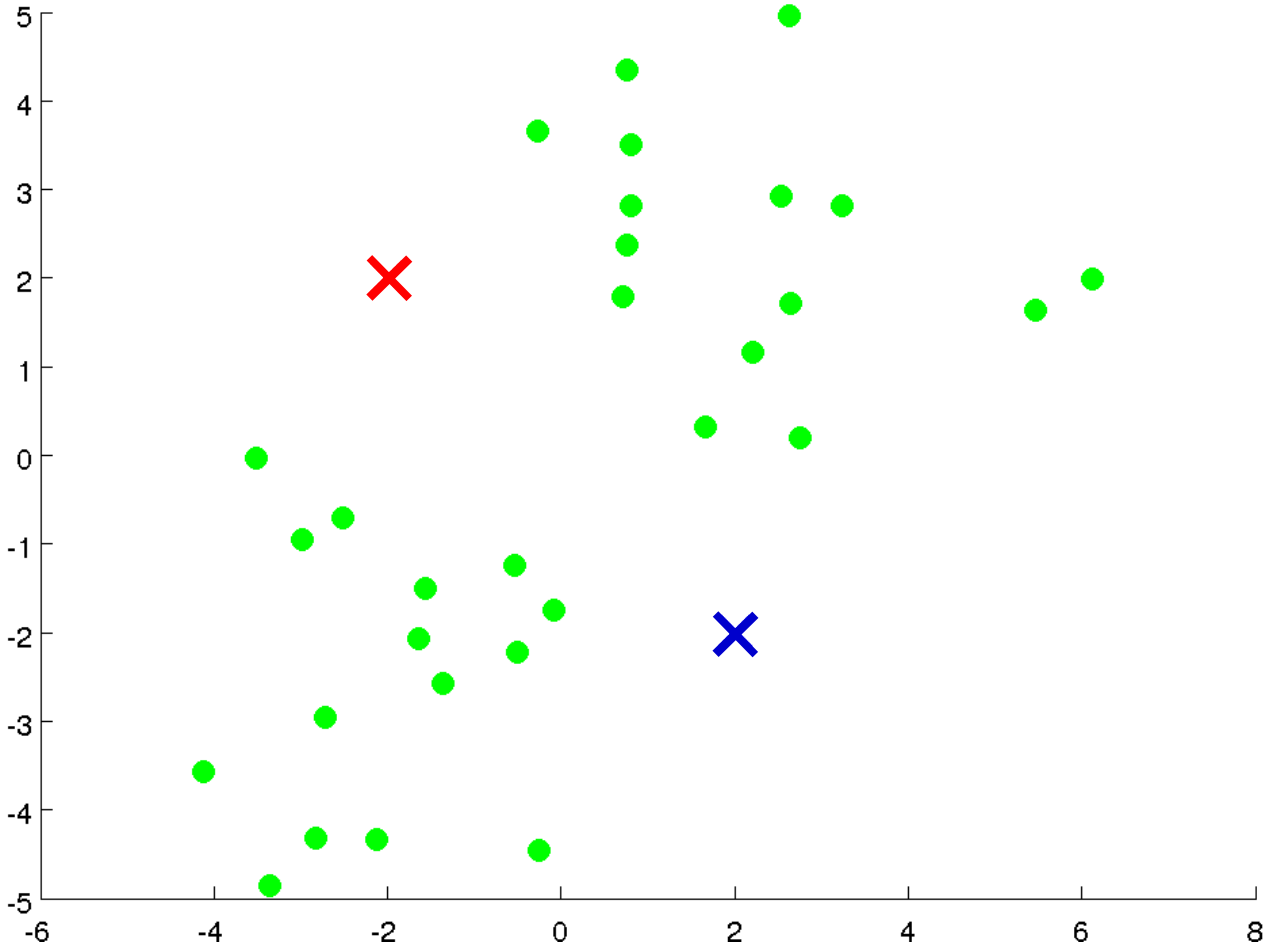
Cluster assignment: $c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$

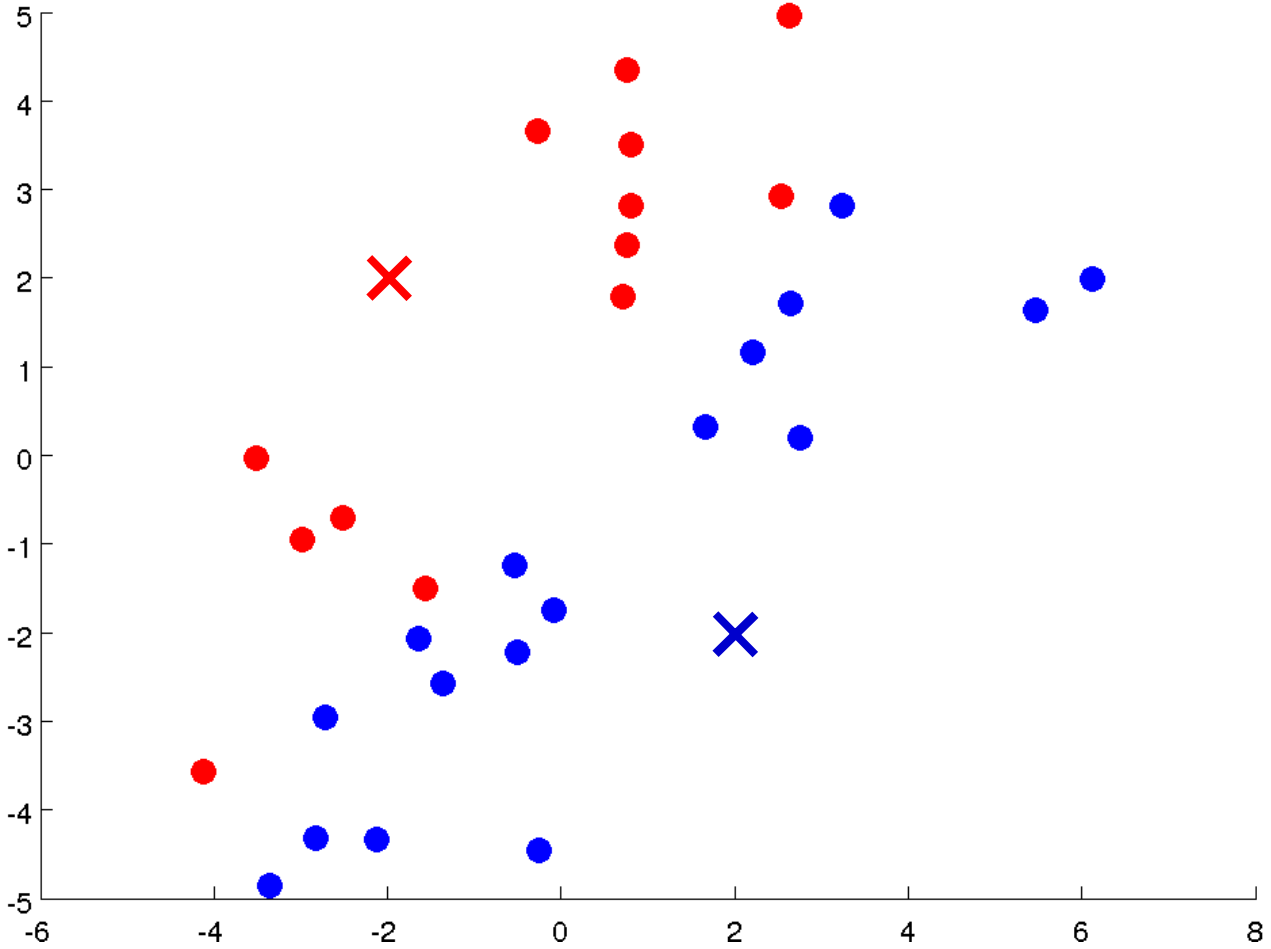
for k = 1 to K

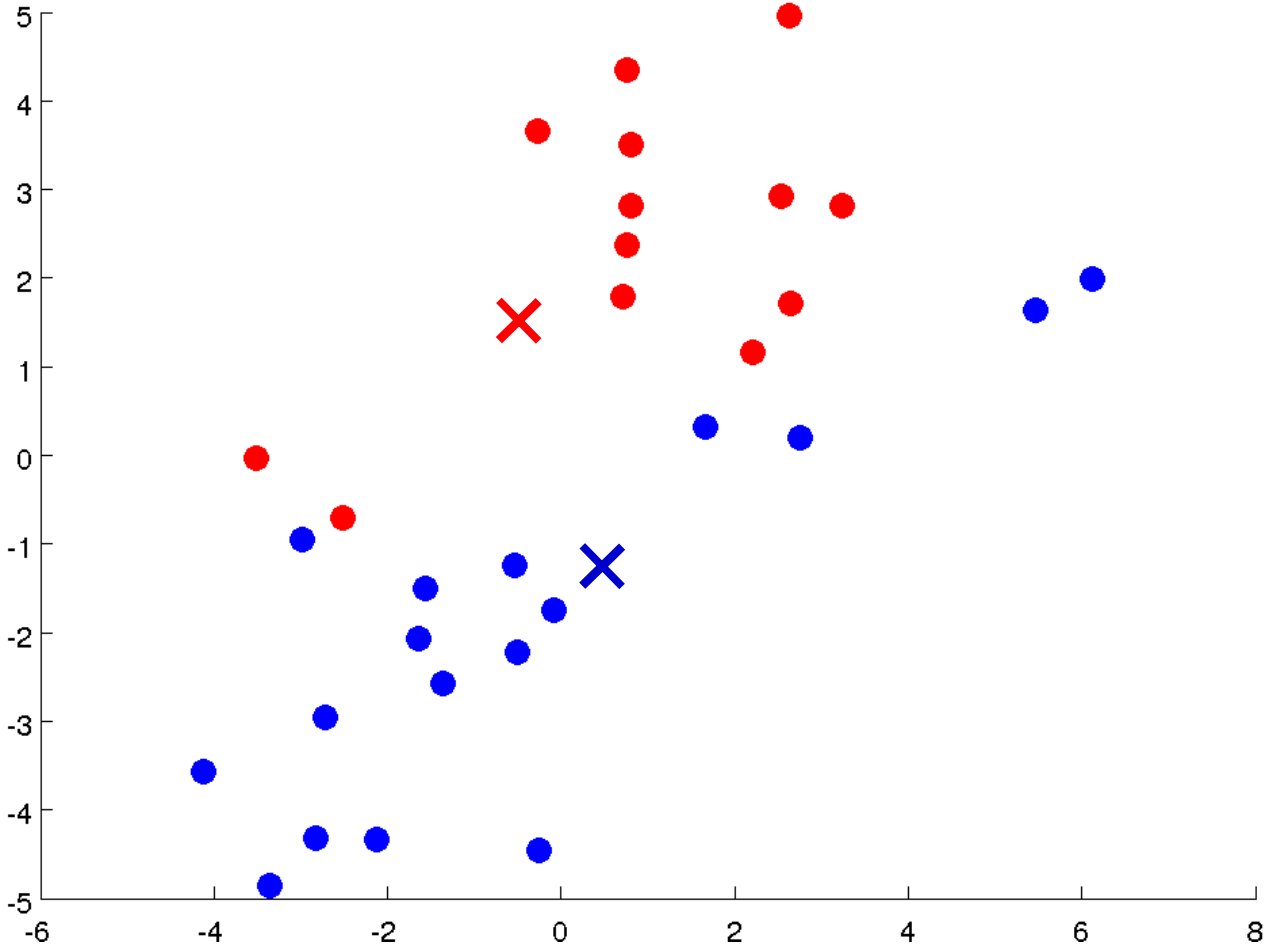
Move centroid: μ_k := average (mean) of points assigned to cluster k

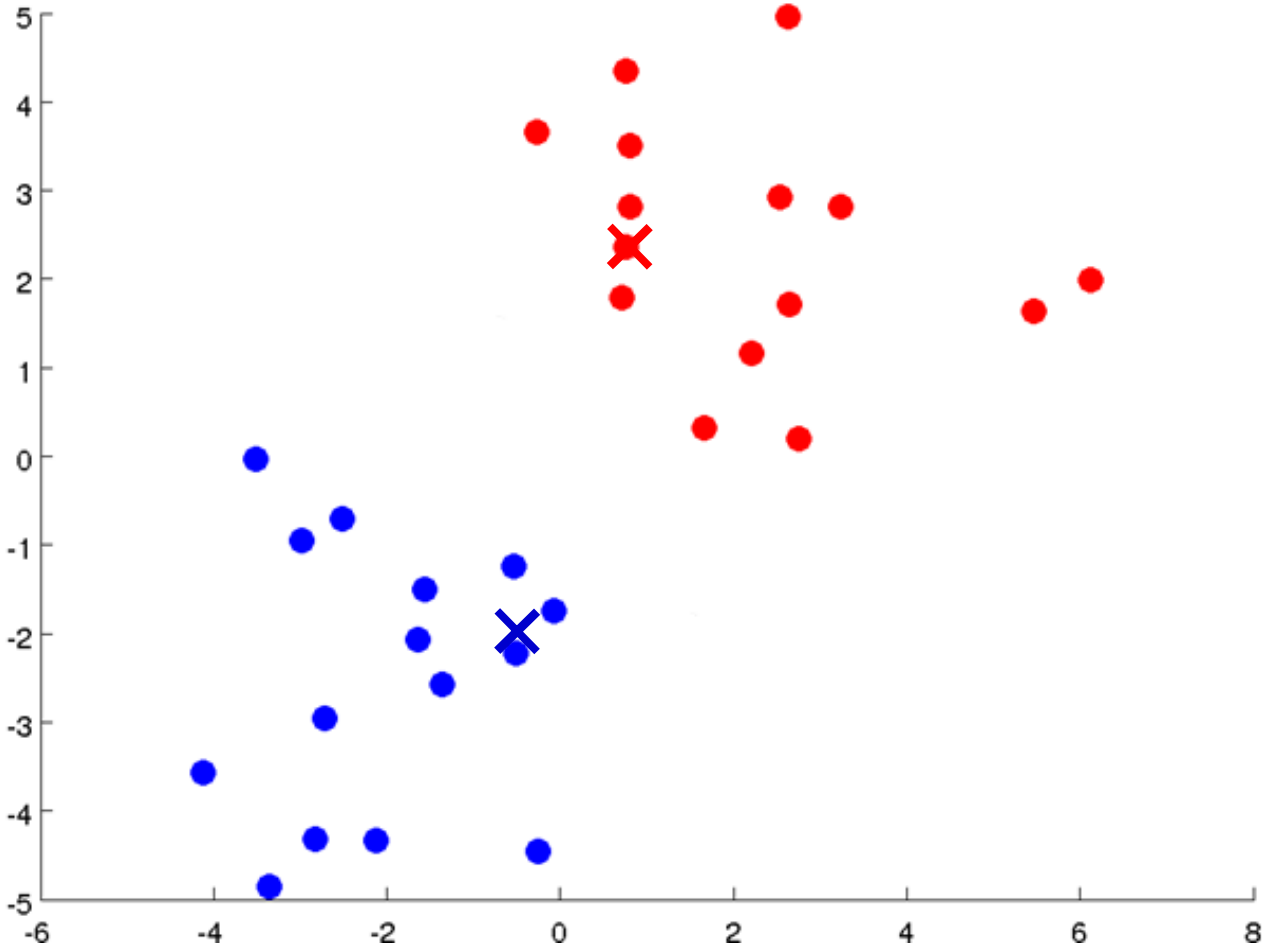
}

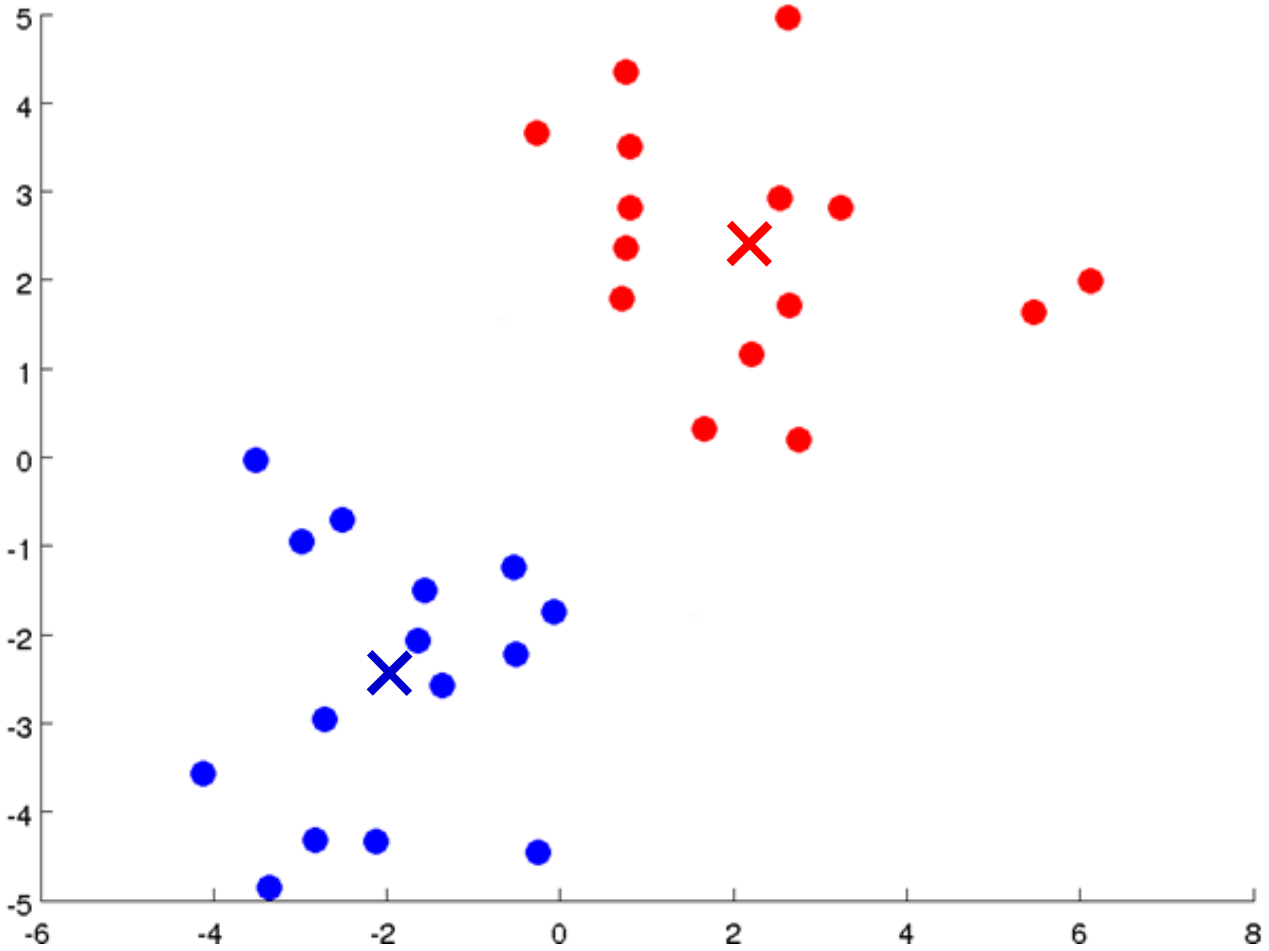

5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8


5
4
3
2
1
0
-1
-2
-3
-4
-5
-6
-4
-2
0
2
4
6
8

K-means algorithm

Randomly initialize K cluster centroids $\mu_1, \mu_2, \ldots, \mu_K \in \mathbb{R}^n$

Repeat {

for i = 1 to m

Cluster assignment: $c^{(i)}$:= index (from 1 to K) of cluster centroid closest to $x^{(i)}$

for k = 1 to K

Move centroid: μ_k := average (mean) of points assigned to cluster k

}

Optimization in K-means

- Consider data points in Euclidean space
- A measure of cluster quality: Sum of Squared Error (SSE)
  - Error of each data point: Euclidean distance of the point to its closest centroid
  - SSE: total sum of the squared error for each point
  - Will be minimized if the centroid of a cluster is the mean of all data points in that cluster
- Steps of K-means minimizes SSE (finds a local minima)

Choosing value of K

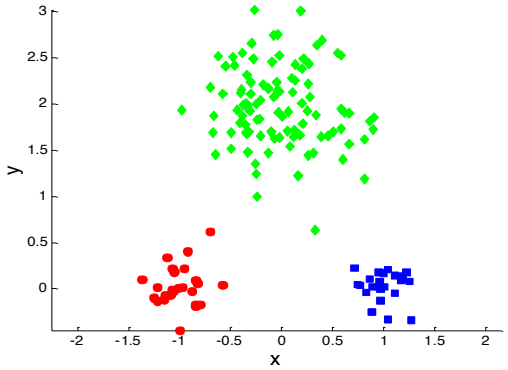
- Based on domain knowledge about suitable number of clusters for a particular problem domain

- Alternatively, based on some measure of cluster quality, e.g., try for different values of K and choose that value for which SSE is minimum

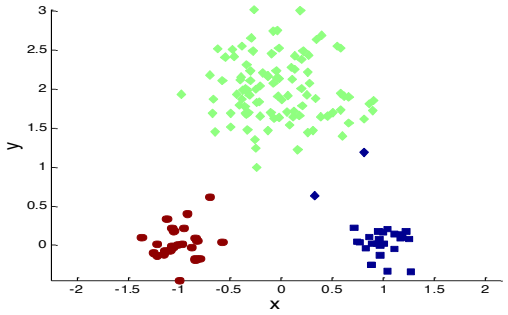
Choosing initial centroids

- Can be selected randomly, but can lead to poor clustering
- Perform multiple runs, each with a different set of randomly chosen initial centroids, and select that configuration that yields minimum SSE
- Use domain knowledge to choose centroids, e.g., while clustering search results, select one search result relevant to each aspect of the query

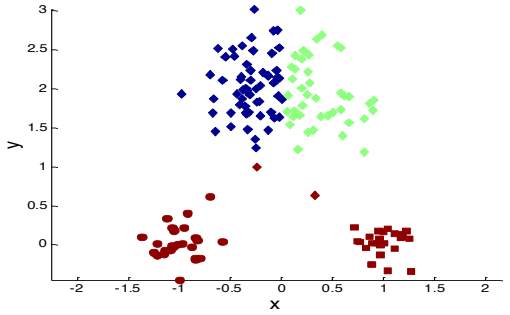
Importance of choosing initial centroids well



Original Points



Optimal Clustering



Sub-optimal Clustering

Similarity/closeness between items

- Measure of similarity/closeness between items depends on the problem domain
- Will be performed many times over the course of the algorithm, hence needs to be efficient
- Examples
  - Points in Euclidean space → Euclidean distance
  - Text documents → cosine similarity between term-vectors

Reducing SSE with post-processing

- Finding more clusters will reduce SSE, but sometimes we want to improve SSE without increasing clusters

- K-means has found a local minima; find another "nearby" clustering with lower SSE (if exists)

Reducing SSE with post-processing

- Techniques used
  - Splitting a cluster, e.g., the cluster with highest SSE, or the cluster with highest standard deviation of a chosen feature
  - Merging two clusters, e.g., the clusters with the closest centroids

Known problem of K-means

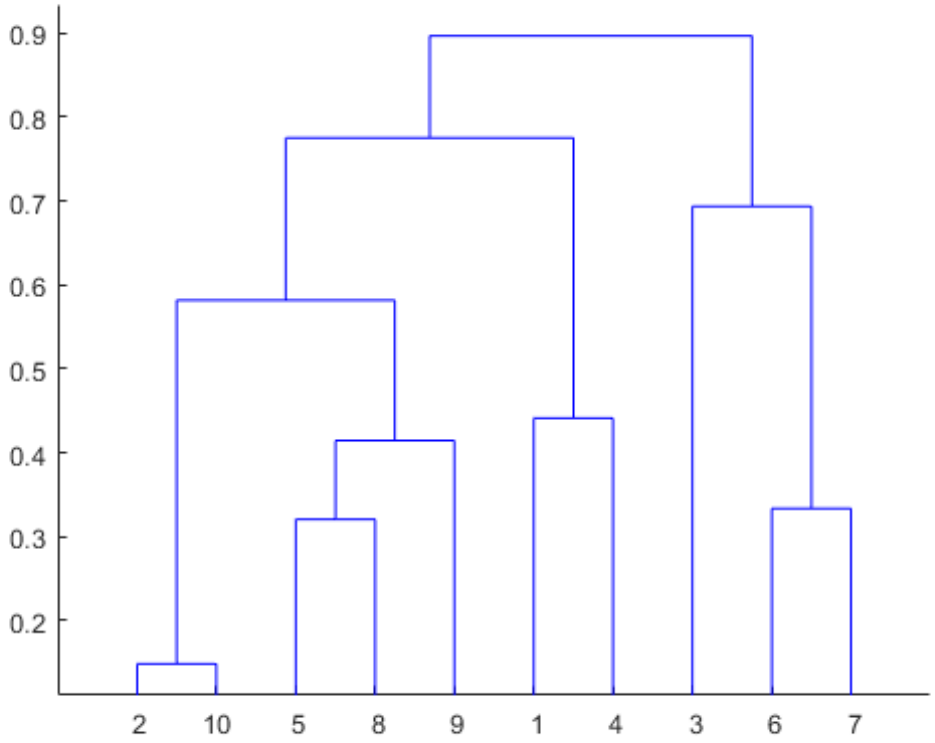
- Sensitive to outliers that can change the distribution of the clusters
  - A solution: K-Mediods: instead of taking the mean value of the points in a cluster, use the medoid that is the most centrally located point in the cluster
- Detected clusters are usually globular (spherical) in shape; problems in detecting arbitrary-shaped clusters

Hierarchical clustering

Hierarchical clustering

- Bottom-up or Agglomerative clustering
  - Start considering each data point as a singleton cluster
  - Successively merge clusters if similarity is sufficiently high
  - Until all points have been merged into a single cluster
- Top-down or Divisive clustering
  - Start with all data points in a single cluster
  - Iteratively split clusters into smaller sub-clusters if the similarity between two sub-parts is low

Both Divisive and Agglomerative clustering can be represented as a Dendrogram



Basic agglomerative hierarchical clustering algorithm

- Start with each item in a singleton cluster
- Compute the proximity/similarity matrix between clusters
- Repeat
  - Merge the closest/most similar two clusters
  - Update the proximity matrix to reflect proximity between the new cluster and the other clusters
- Until only one cluster remains

Proximity/similarity between clusters

- MIN similarity between two clusters: Proximity (similarity) between the closest (most similar) two points, one from each cluster (minimum pairwise distance)
- MAX similarity between two clusters: Proximity between the farthest two points, one from each cluster (maximum pairwise distance)
- Group average similarity: average pairwise proximity of all pairs of points, one from each cluster

Types of hierarchical clustering

- Complete linkage
  - Merge in each step the two clusters with the smallest **maximum** similarity
- Single linkage
  - Merge in each step the two clusters with the smallest **minimum** similarity

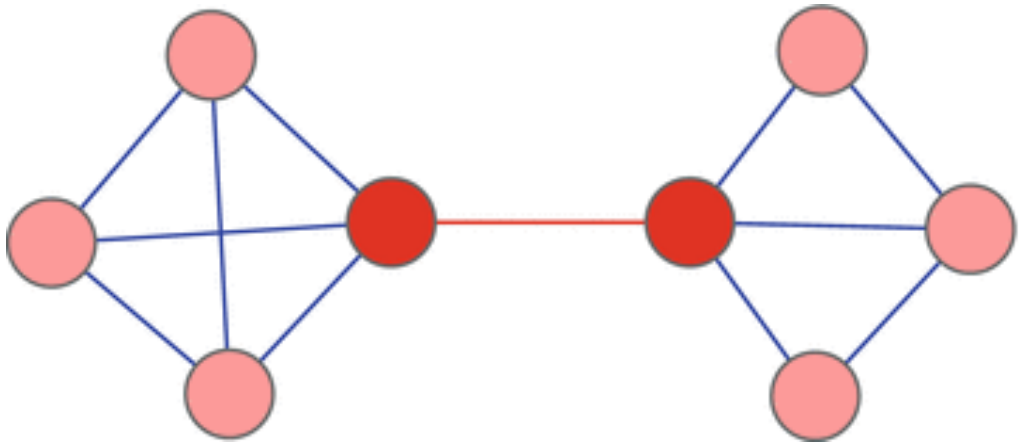
A divisive graph-based clustering algorithm

A graph-based hierarchical clustering algorithm

- A cluster is a group of nodes having more and / or better connections among its members, than between its members and the rest of the network
- Cluster in graphs/networks: also called community structure
- Algorithm by Girvan and Newman: *Community structure in social and biological networks*, PNAS 2002

Girvan-Newman algorithm

- Focus on edges / links that are most 'between' clusters
- Edge betweenness of an edge *e* : fraction of shortest paths between all pairs of nodes, which pass through *e*



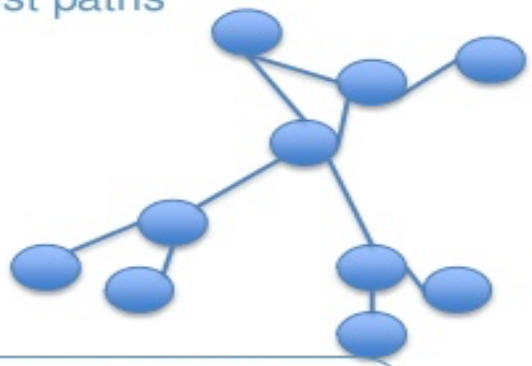
Edge betweenness centrality

Definition

Locates structurally the "well-connected" edges

If it is located on many shortest paths

$$BC(e) = \sum_{v,w \in V} \frac{b_{vw}(e)}{b_{vw}}$$



B_{vw} (e) = the number of shortest paths from V to W through e
B_{vw} = the total number of shortest paths from V to W

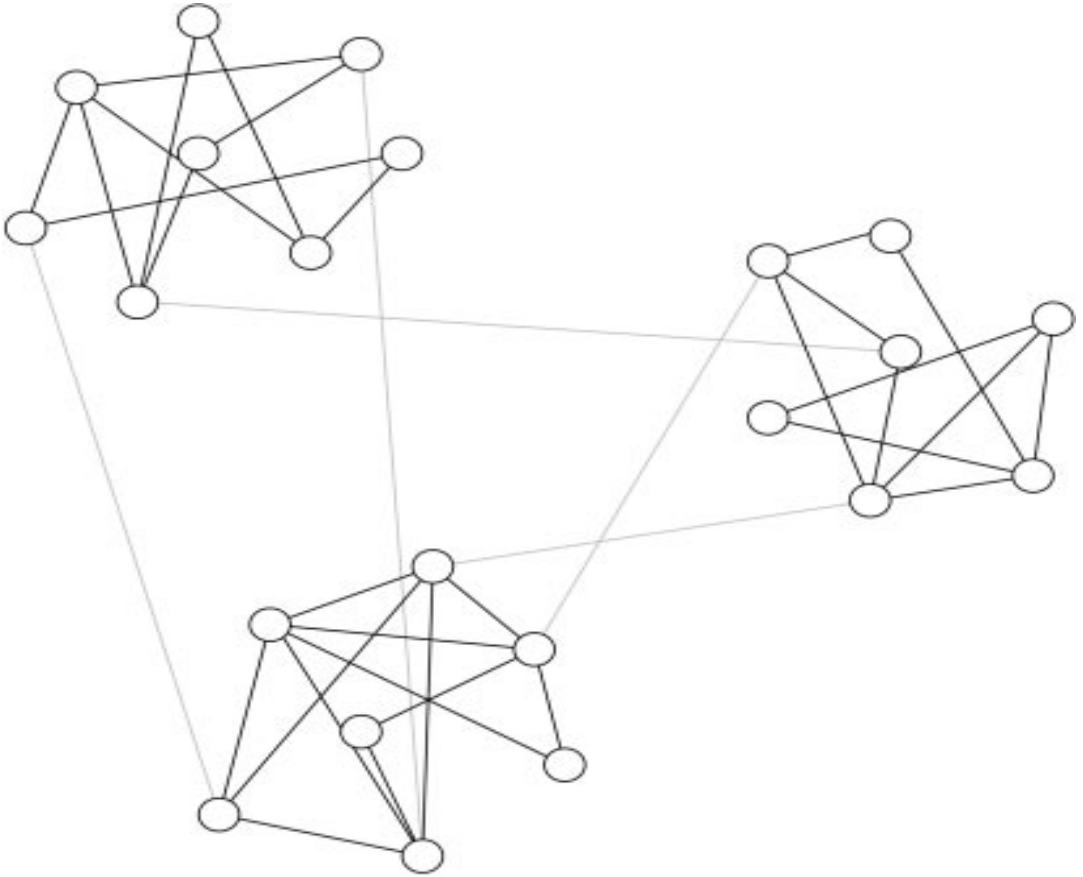
EURA NOVA

S. Wasserman and K. Faust. Social Network Analysis: Methods and Applications.Number 8 in Structural analysis in the social sciences. Cambridge University Press, 1 edition, 1994.

Girvan-Newman algorithm

- Edges between clusters/communities are likely to have high betweenness centrality
- Progressively remove edges having high betweenness centrality, to separate clusters from one another

Girvan-Newman algorithm



Girvan-Newman algorithm

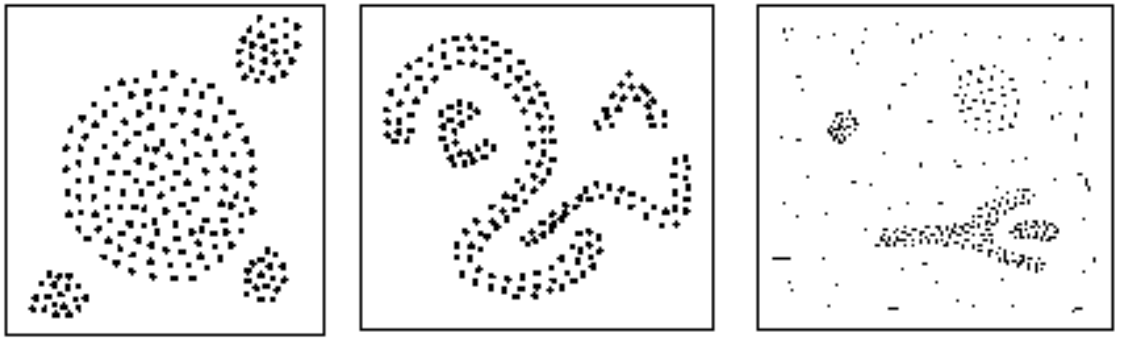
1. Compute betweenness centrality for all edges
2. Remove the edge with highest betweenness centrality
3. Re-compute betweenness centrality for all edges affected by the removal
4. Repeat steps 2 and 3 until no edges remain

Results in a hierarchical clustering tree (dendrogram)

Density-based clustering

Density based clustering methods

- Locates regions of high density, that are separated from one another by regions of low density



DBSCAN

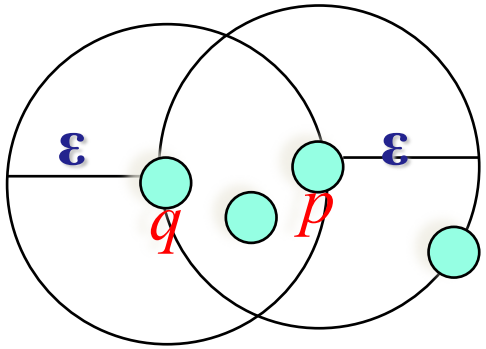
- DBSCAN: Density Based Spatial Clustering of Applications with Noise
  - Proposed by Ester et al. in SIGKDD 1996
  - First algorithm for detecting density-based clusters
- Advantages (e.g., over K-means)
  - Can detect clusters of arbitrary shapes (while clusters detected by K-means are usually globular)
  - Robust to outliers

DBSCAN: intuition

- For any point in a cluster, the local point density around that point has to exceed some threshold
- The set of points in one cluster is spatially connected
- Local point density at a point *p* defined by two parameters
  - ε : radius for the neighborhood of point p: $N_\varepsilon(p) := \{q \text{ in data set } | \ dist(p, q) \leq \varepsilon\}$
  - *MinPts* : minimum number of points in the given neighborhood $N_\varepsilon(p)$

Neighborhood of a point

- ε-Neighborhood of a point *p :* Points within a radius of ε from the point *p*
- "High density": if ε-Neighborhood of a point contains at least *MinPts* number of points



ε-Neighborhood of *p*

ε-Neighborhood of *q*

Density of p is "high" (MinPts = 4)

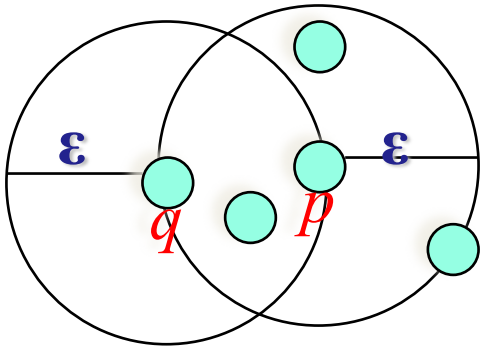
Density of q is "low" (MinPts = 4)

Divide points into three types

- Core point: A point that has more than a specified number of points (MinPts) within its ε-Neighborhood (points that are at the interior of a cluster)
- Border point: has fewer than MinPts points within its ε-Neighborhood (not a core point), but falls within the ε-Neighborhood of a core point
- Outlier point: any point that is not a core point nor a border point

Density-Reachability

- Directly density-reachable: A point *q* is directly density-reachable from object *p* if *p* is a core point and *q* is in *p*'s ε-neighborhood.



MinPts = 4

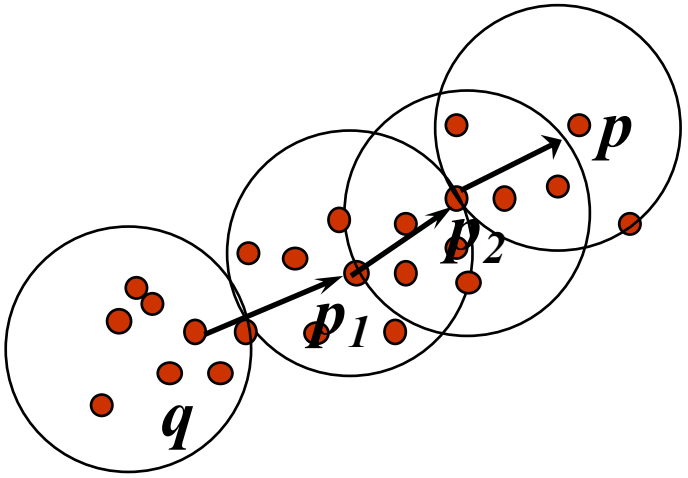
q is directly density-reachable from p

p is not directly density-reachable from q

Density-reachability is not symmetric

Density-Reachability

- Density-reachability can be direct or indirect
  - Point p is directly density-reachable from p2
  - p2 is directly density-reachable from p1
  - p1 is directly density-reachable from q
  - p←p2←p1←q form a chain



MinPts = 7

p is (indirectly) density-reachable from q

q is not density-reachable from p

DBSCAN algorithm

Input: The data set D

Parameters: ε, MinPts

for each point *p* in *D*
    if *p* is a core point and not processed then
        C = {all points density-reachable from *p*}
        mark all points in *C* as processed
        report *C* as a cluster
    else
        mark *p* as outlier
    end if

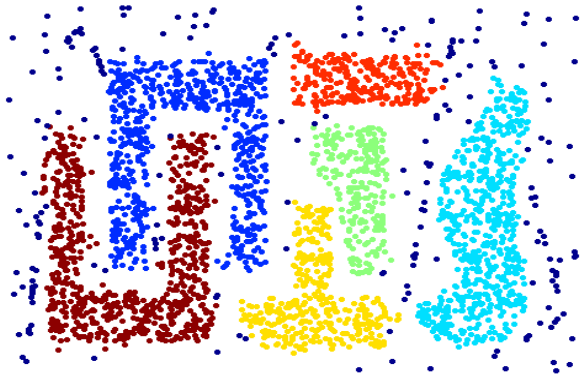
end for

Understanding the algorithm

- Arbitrary select a point *p*
- Retrieve all points density-reachable from *p* w.r.t. ε and *MinPts*
- If *p* is a core point, a cluster is formed
- If *p* is a border point, no points are density-reachable from *p* and DBSCAN visits the next point of the database
- Continue the process until all of the points have been processed (each point marked as either core or border or outlier)

When DBSCAN works well

Original Points



Clusters

- Resistant to noise / outliers (note: partial clustering)
- Can handle clusters of different shapes and sizes
- Number of clusters identified automatically

When DBSCAN does not work well

- Cannot identify clusters of varying densities
- Sensitive to parameters

Resources on Clustering (free on web)

- Data Clustering: Algorithms and Applications
  - Book by Charu Aggarwal, Chandan Reddy

- Community Detection in graphs
  - Survey paper by Santo Fortunato