Big Data 2016 - SONY

Håkan Jonsson
Vedran Sekara

Schedule

09:15 - 10:00 Cluster analysis, partition-based clustering

10.00 – 10.15 Break

10:15 – 12:00 Exercise 1: User segmentation based on app usage

12:00 - 13:15 Lunch break

13:15 – 13:45 Density-based clustering

13:45 - 15:00 Exercise 2: Spatial clustering with DBSCAN

15:00 - 16:00 Exercise 3 / Assignment: Your own dataset

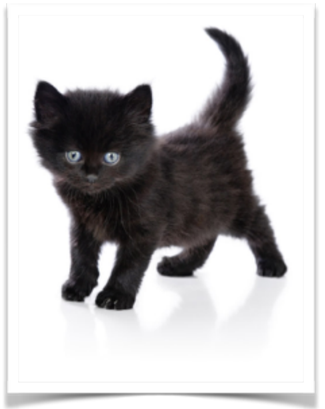
Unsupervised Learning?

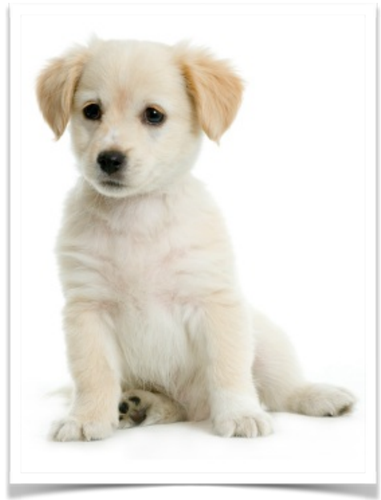
Process of inferring, describing, and uncovering hidden structures in data where for which we have no ground truth

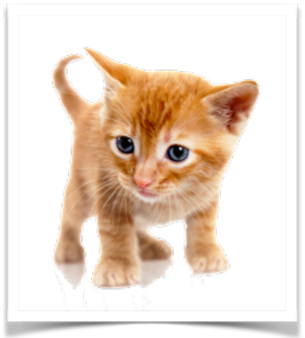
Cluster Analysis
Basic Concepts and Methods

- Cluster Analysis: Basic Concepts
- Partitioning Methods
- Density-Based Methods

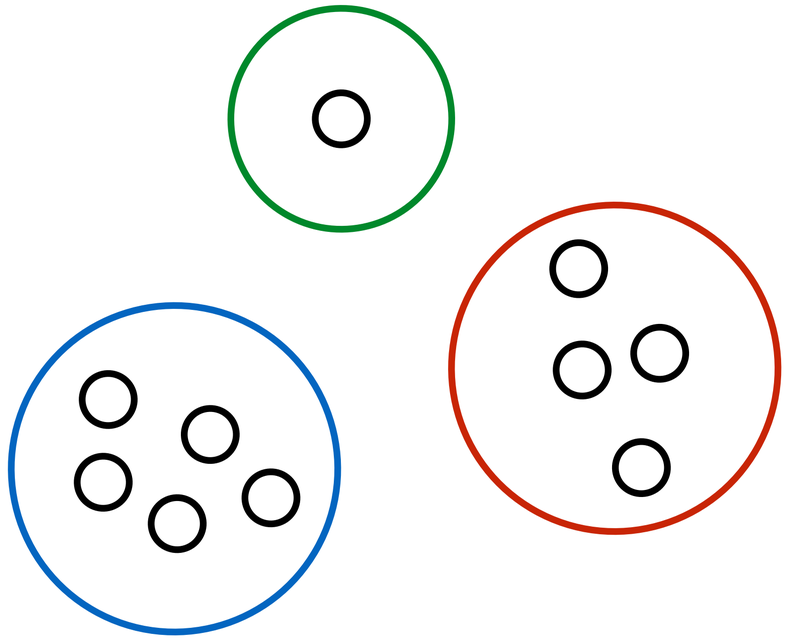
What is
Cluster Analysis?

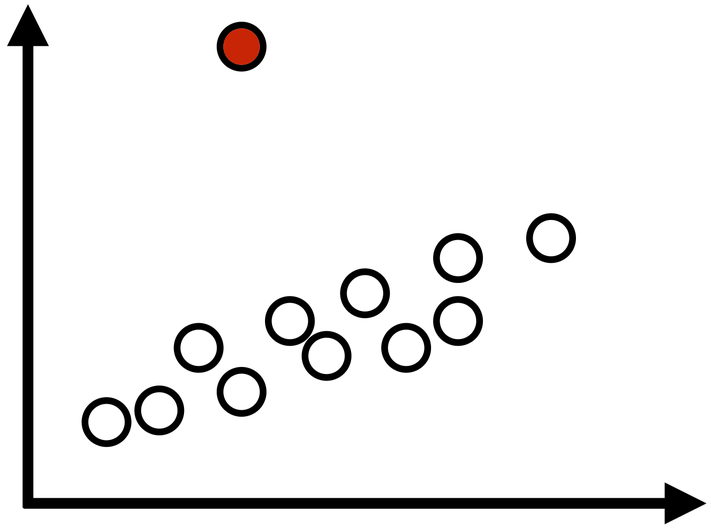






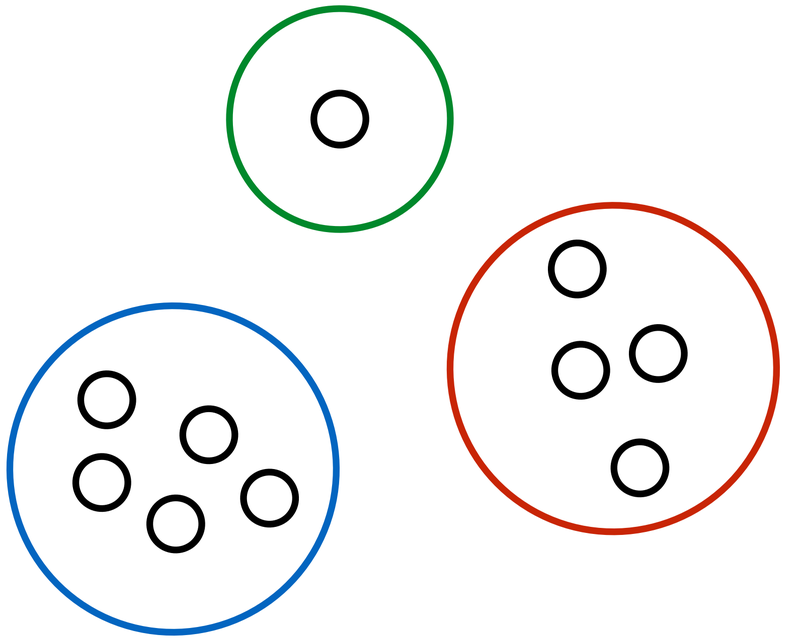
What is
Cluster Analysis?

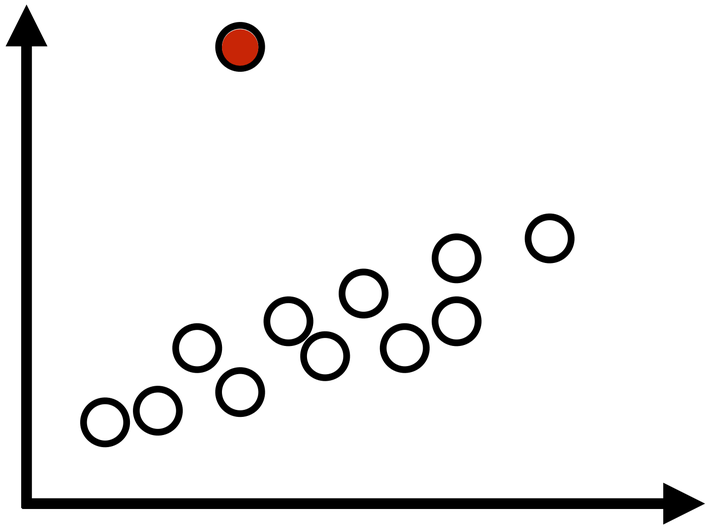




What is
Cluster Analysis?

Humans are skilled at dividing objects into groups.
But how do we train a machine to do the same?





Application

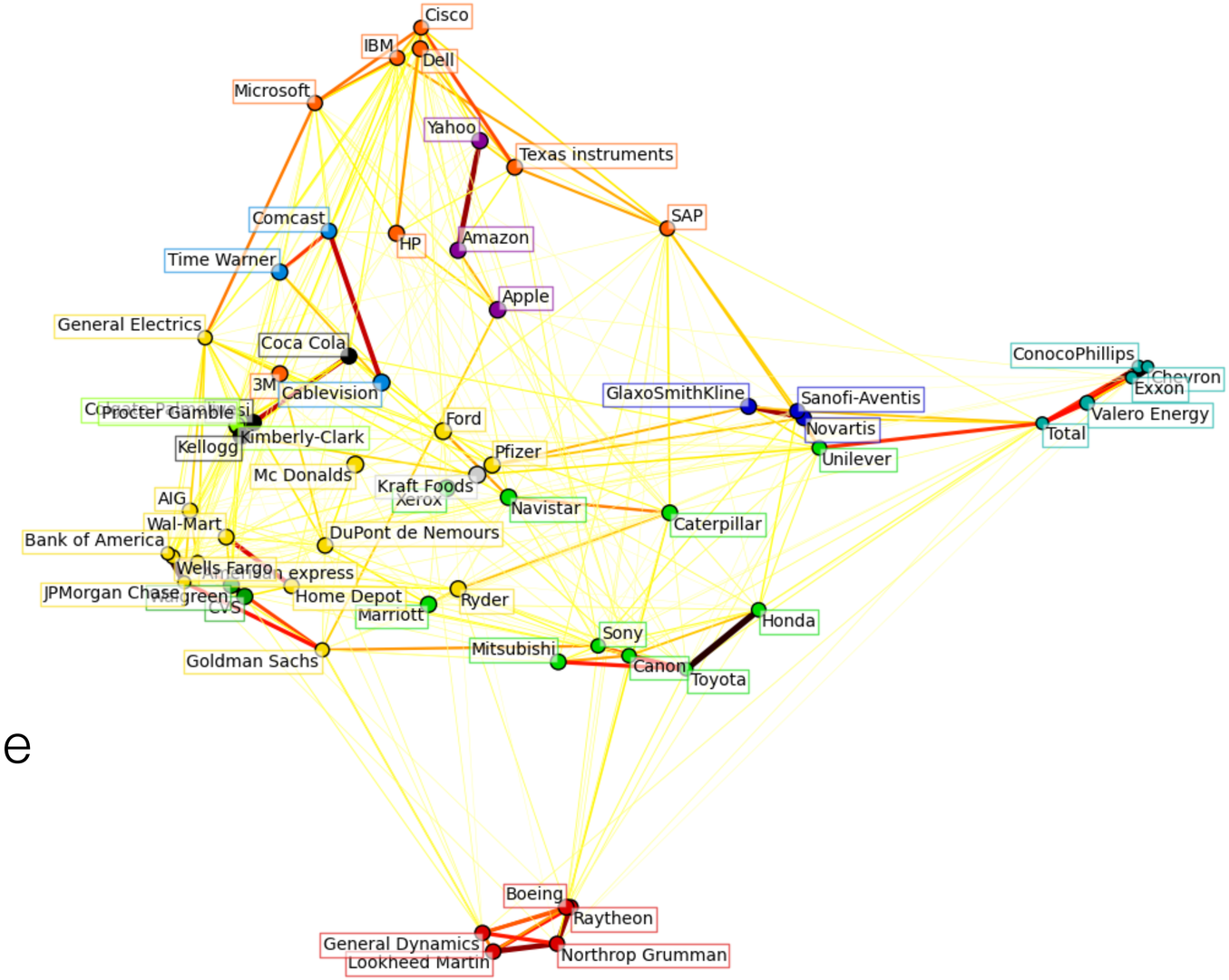
Finance:
Map interdependence of companies

Marketing:
Customer segmentation

Information retrieval:
Document clustering

Land use:
Identification of areas of similar land use

City-planning:
Identifying similar neighbourhoods
according to house type, value, geo-location …



Preprocessing
tool

Summarization

- Preprocessing step for other machine learning algorithms (regression, classification, or association tasks)

Outlier detection

- Outliers are often viewed as those “far away” from any cluster

Quality
What is a good clustering?

A good clustering method will produce high quality clusters

- high intra-class similarity
- low inter-class similarity

The quality of a clustering method depends on

- the applied similarity measure
- the implementation (some are based on randomness)
- its ability to discover specific types of latent structures

Measuring **quality**

Dissimilarity/similarity metrics

- Similarity is expressed in terms of a distance function
- Definition of distance is different for various types of data
  (interval-scaled, boolean, categorical, ordinal, ration, vector, …)

Quality of clustering

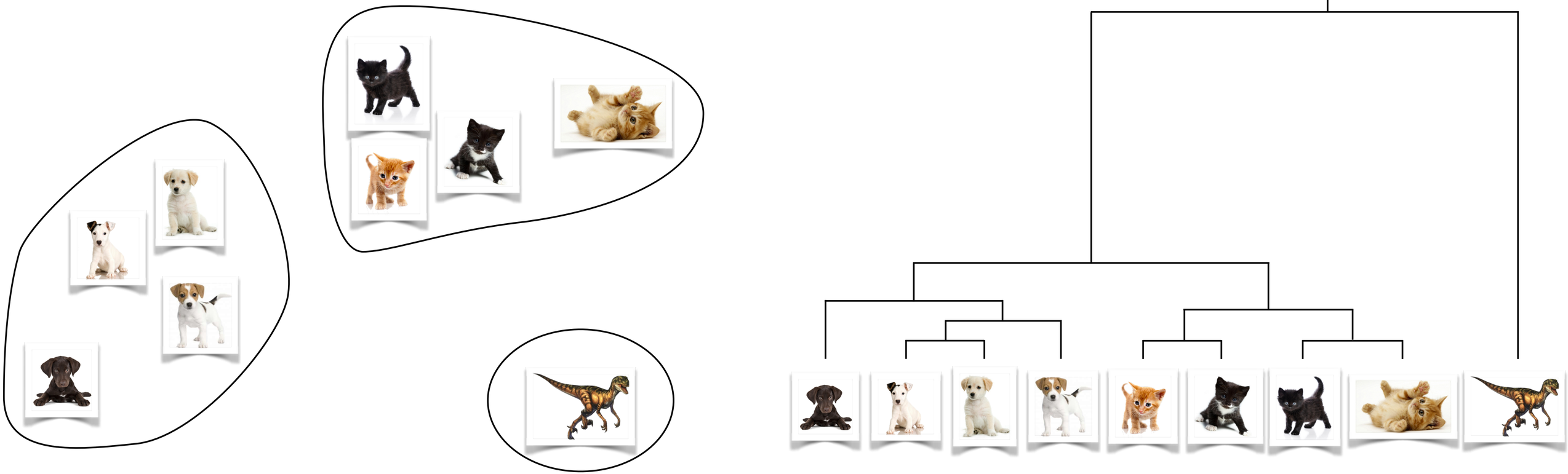
- There is no single universal quality function, depends on the problem at hand and the data
  (gini, entropy, class-error, impurity, squared sum of errors, …)
- It is difficult to define “similar enough” or “good enough” quality
  (depends on problem at hand)

Considerations when **clustering**

Partitioning criteria

- Single level vs. hierarchical partitioning

(often, multi-level hierarchical partitioning is desirable)



Considerations when **clustering**

Partitioning criteria
- Single level vs. hierarchical partitioning
  (often, multi-level hierarchical partitioning is desirable)

Separation of clusters
- overlapping vs. non-overlapping
  (a document can belong to multiple classes)

Similarity measure
- Distance vs. connectivity

Dimensionality
- Look for cluster in full D-dimensional space vs. subspace

Challenges

Scalability
- Sample data or cluster everything?

Ability to deal with different types of attributes
- Numerical, binary, categorical, ordinal, and combinations of these

Constraints on clustering
- User may want to explicitly specify constraints

Interpretability and usability (do clusters look sane, and can we use them?)

Others
- Ability to deal with noisy data
- Ability to infer clusters with arbitrary shapes
- Ability to deal with high-dimensional data
- Incremental clustering and insensitivity to input order

Clustering approaches

Partitioning approach
- Construct partitions and evaluate them by some criterion (e.g. sum of errors squared)
  (typical methods: k-means, k-medoids, CLARANS)

Hierarchical approach
- Create a hierarchical tree of the set of data using some distance criterion
  (typical methods: Diana, Agnes, BIRCH, CAMELEON)

Density-based approach
- Cluster data using to connectivity and density constraints
  (typical methods: DBSCAN, OPTICS, DenClue)

Grid-based approach
- Based on a multiple-level granularity structure
  (typical methods: STING, WaveCluster, CLIQUE)

Clustering approaches (**II**)

Model approach

- A model is hypothesised for each cluster, points that fit the model are assigned to the cluster
  (typical methods: EM, SOM, COBWEB)

Frequent pattern approach

- Based on the analysis of frequent patterns
  (typical methods: p-cluster)

Link-based clustering

- Objects are linked together i various ways (e.g. network)
  (typical methods: LinkClustering, SimRank, Modularity)

Why are there so many methods?

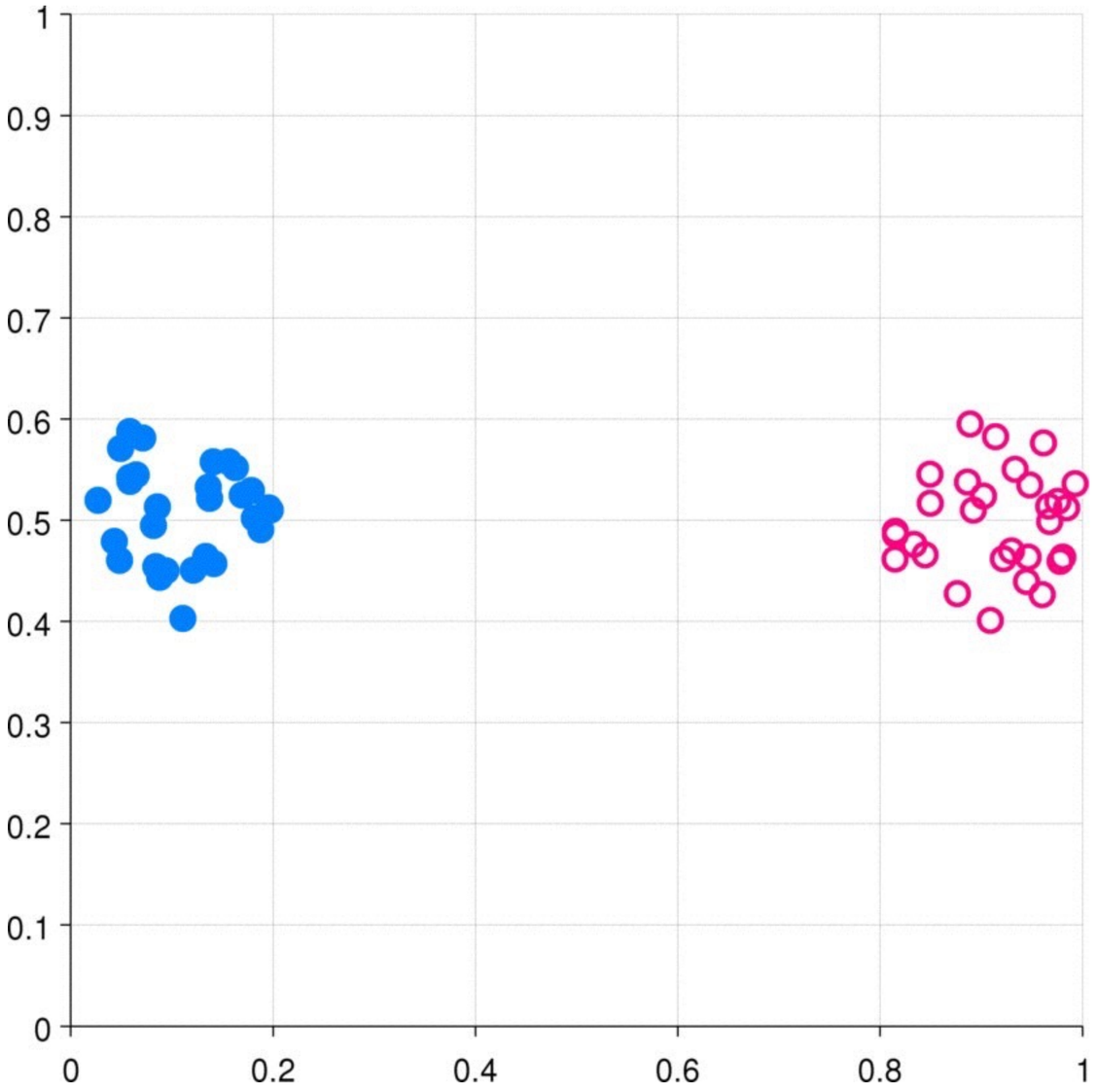
Why are there so many methods?

Data!

Example 1

Well-Separated

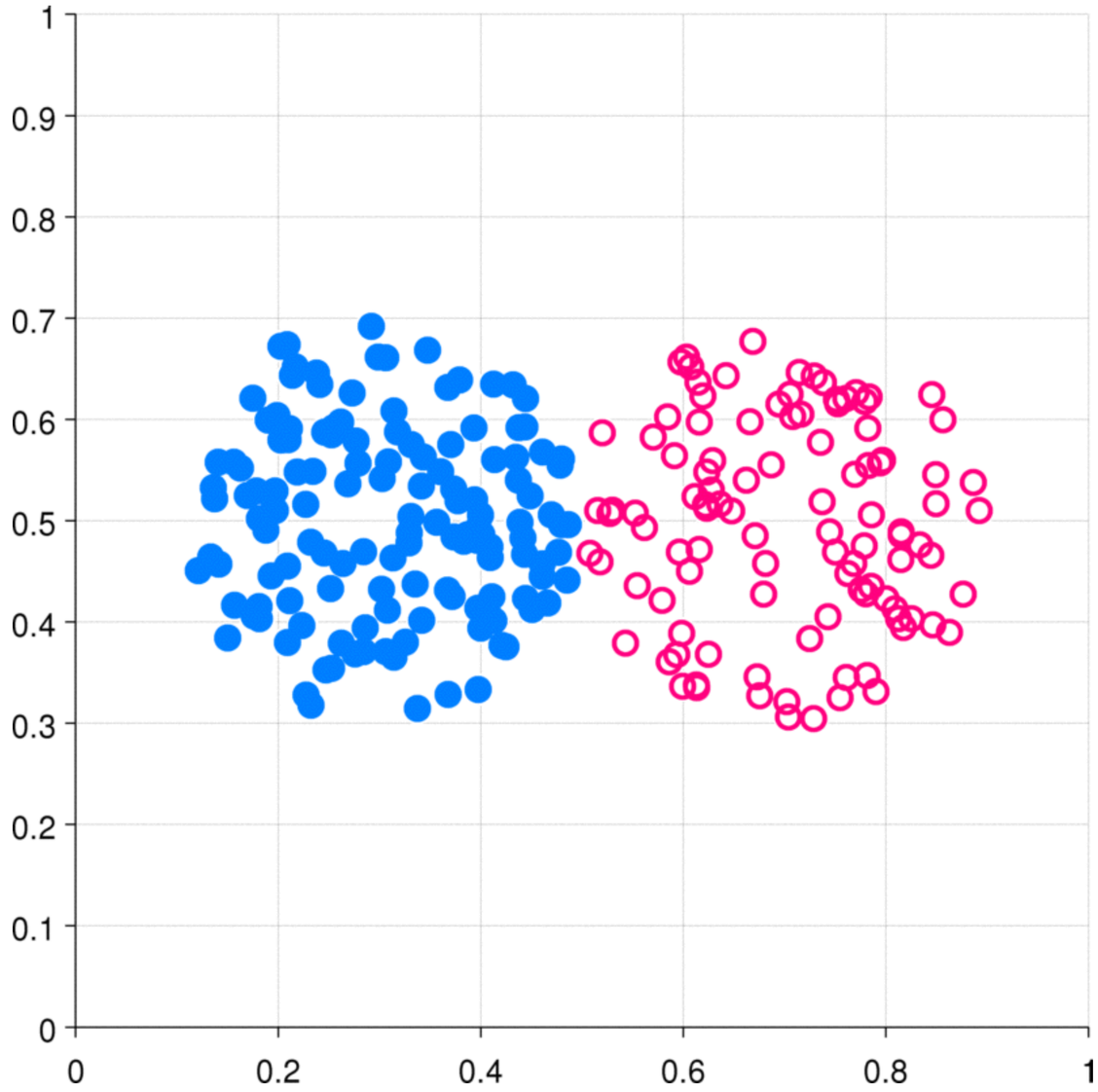
Each point is closer to all points in its cluster than any point in another cluster



Example 2

Center based

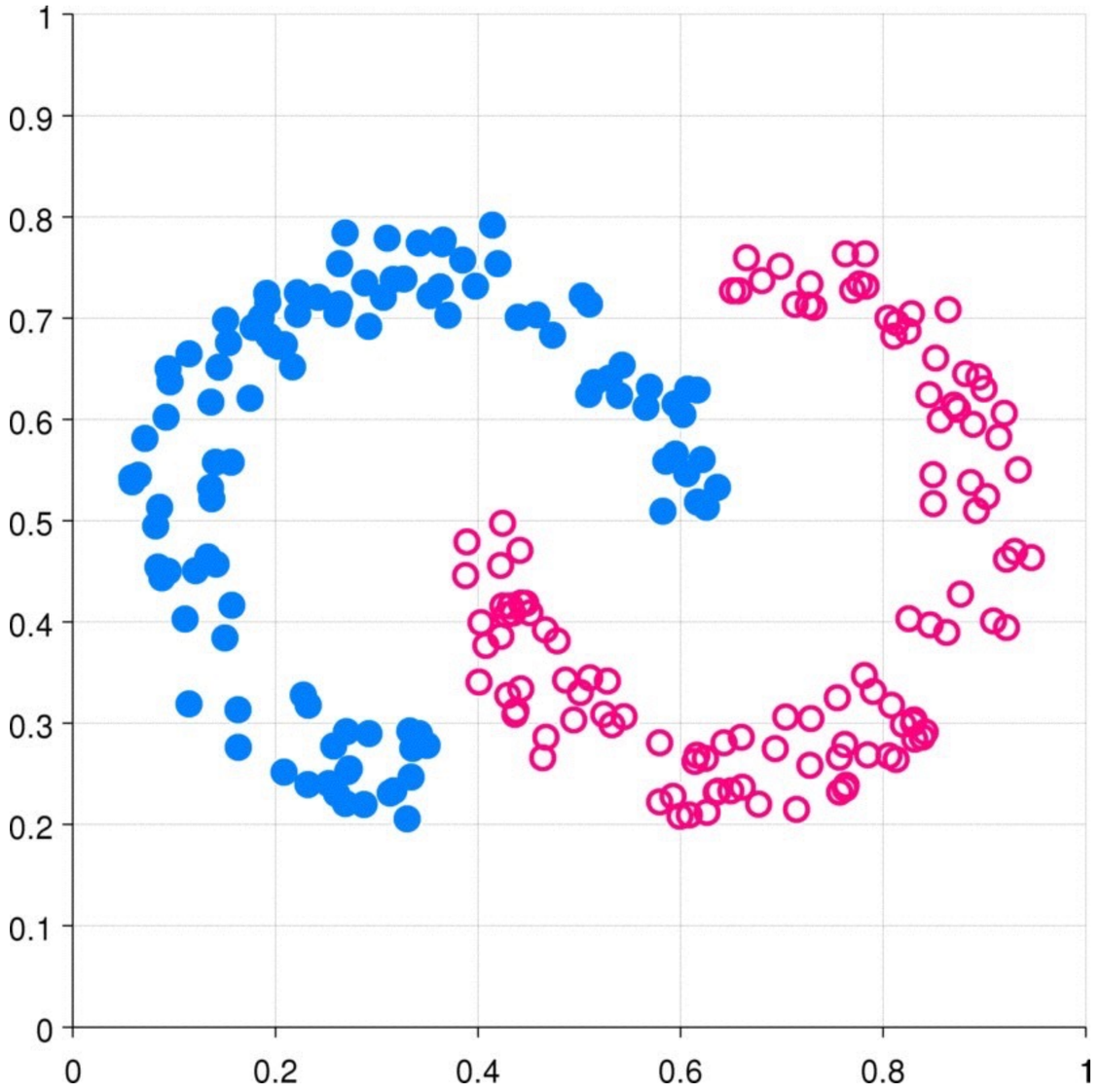
Each point is closer to the center of its cluster than to the center of any other cluster



Example 3

Contiguity based

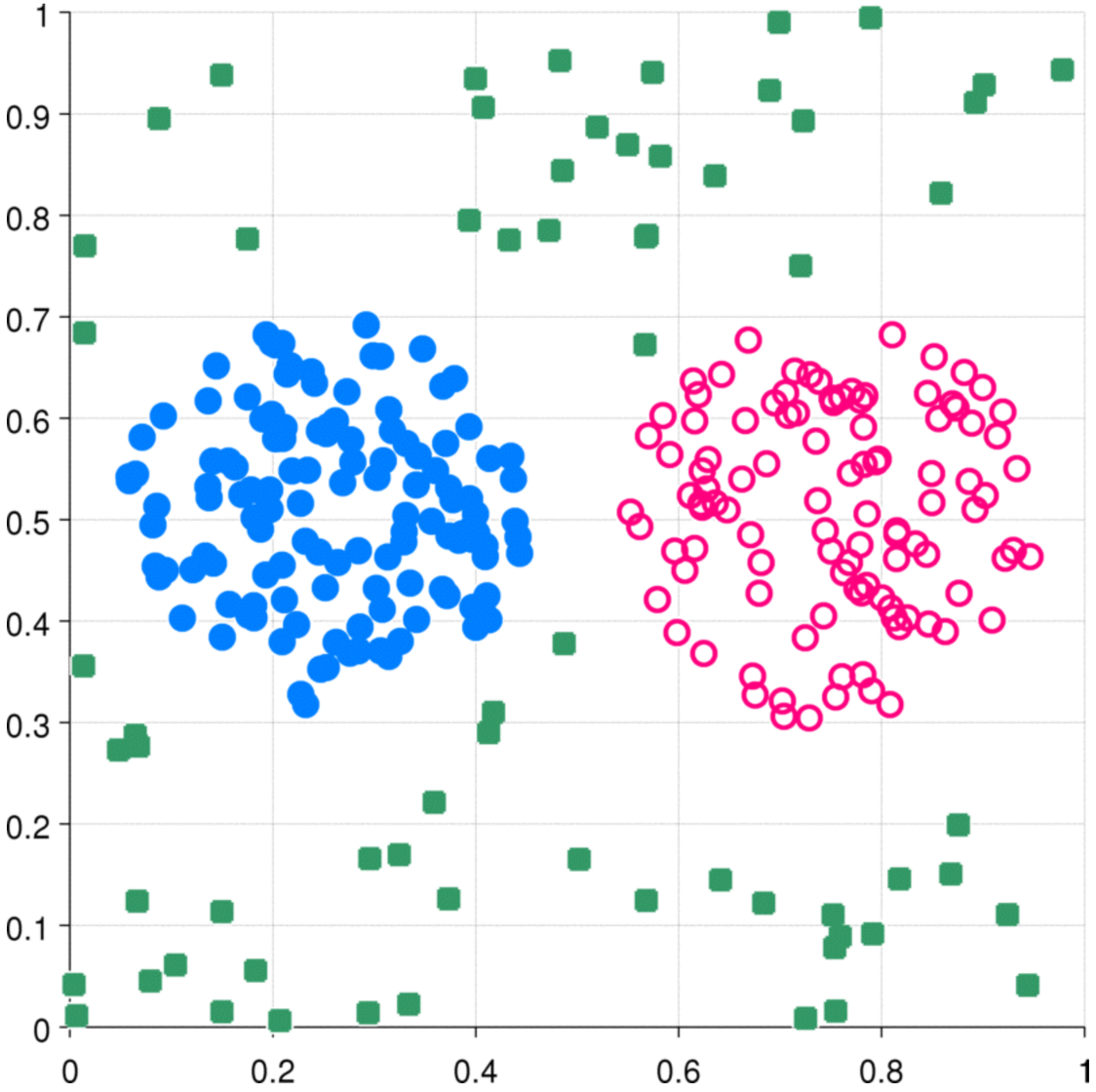
Each point is closer to at least one point in its cluster than to any point in another cluster



Example 4

Density based

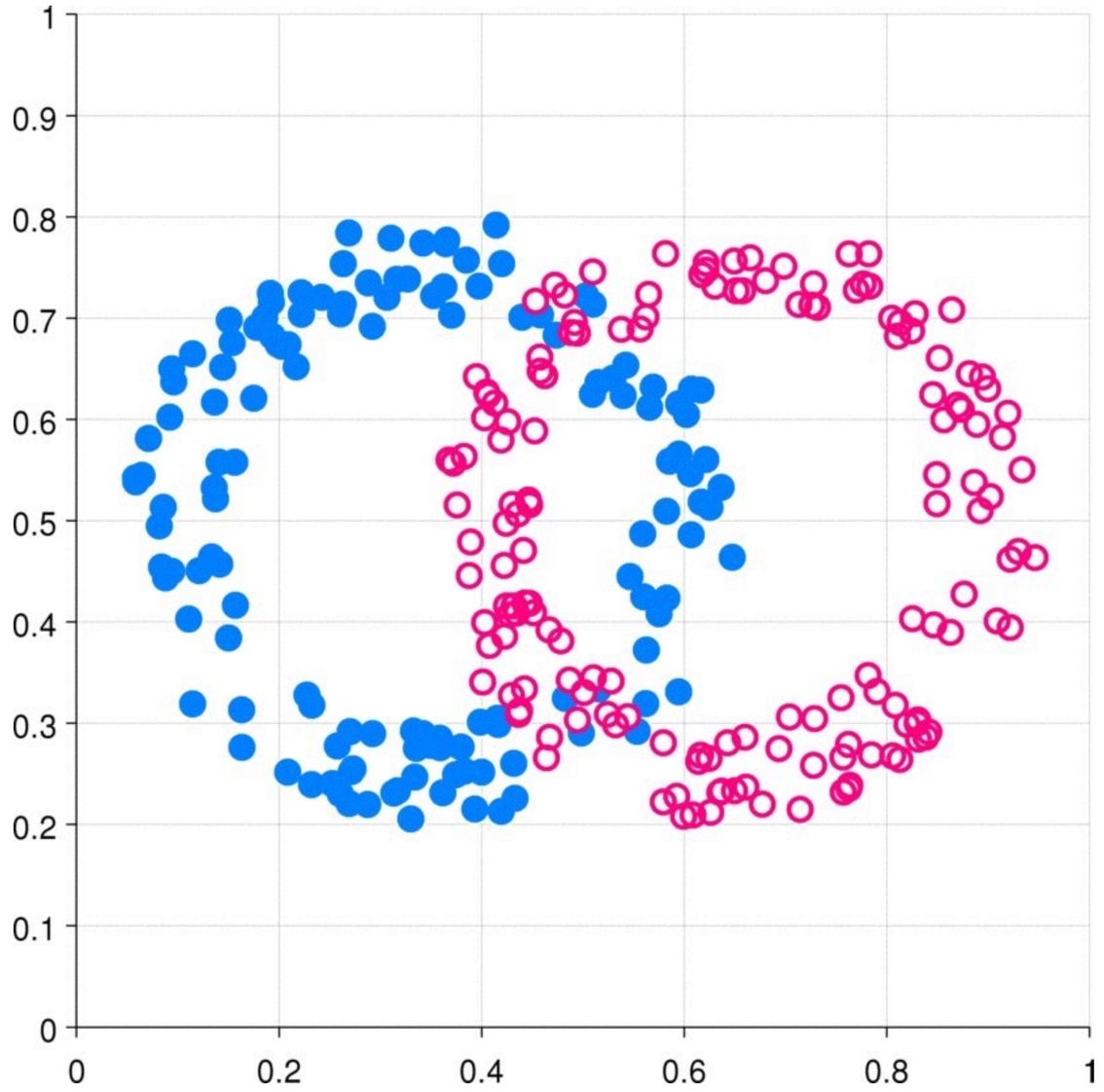
Clusters are regions of high density separated by regions of low density



Example 5

Conceptual

Points in a cluster share some general property that can be derived from the entire set of points



K-means

Objective

Given a dataset **D** with **n** points find a value **k** that separates the data into **k** clusters (by minimizing some error function)

Algorithm

1) Select **k** points as initial centroids

Repeat

2) Form **k** clusters by assigning each points to its closest centroid

3) Recompute the centroids of each cluster

Until centroids do not change

demo

http://www.naftaliharris.com/blog/visualizing-k-means-clustering/

Comments on **k**-means

Strength

Fairly fast [O(ntk), n = # datapoints, k = # clusters, t = # iterations]

Limitations

- Can get stuck at a local minimum
- We need to explicitly specify **k**
- Sensitive to noise, and especially outliers
- Mainly applicable to continuous data
- Cannot detect clusters with strange shapes

Alternative versions addressing some of the issues

- **k**-medoids, **k**-modes

k-means with Spark MLLib

```
from pyspark.mllib.clustering import KMeans, KMeansModel
from numpy import array
from math import sqrt

# Load and parse the data
data = sc.textFile("data/mllib/kmeans_data.txt")
parsedData = data.map(lambda line: array([float(x) for x in line.split(' ')]))

# Build the model (cluster the data)
clusters = KMeans.train(parsedData, k=2, maxIterations=10, runs=10, initializationMode="random")

WSSSE = clusters.computeCost(parsedData)
print("Within Set Sum of Squared Error = " + str(WSSSE))

# Save and load model
clusters.save(sc, "myModelPath")
sameModel = KMeansModel.load(sc, "myModelPath")
```

k-means with Spark MLLib

```
clusters = KMeans.train(parsedData, k=2, maxIterations=10, runs=10, initializationMode="random")
```

Parameters

- k = number of clusters
- maxIterations = max number of iterations to run
- epsilon = threshold within we consider k-means to have converged
- runs = number of k-means runs, from which optimal one is chosen
- InitialModel = how to pick cluster centres when initializing model

Exercise I
User segmentation using **k**-means

Question
Can we segment users into groups based on their application usage?

Representation (Bag-of-Words)
We are going to use a very simple representation of user behaviour,
only taking into account the apps he/she starts and disregard everything else

Procedure
- Every app belongs to a category (Tools, Weather, Music, …)
- Count the number of times a users has started an app within each category
- This creates a vector, one for each user, where each index represents one category

Density based clustering methods

Density based clustering methods

Instead of using on similarity (distance) to cluster datapoints we are going to look at density

Advantages
Can infer clusters of arbitrary shapes
Can handle noise
Are relatively fast (usually require only one scan of data)
We don't have to define how many cluster we want

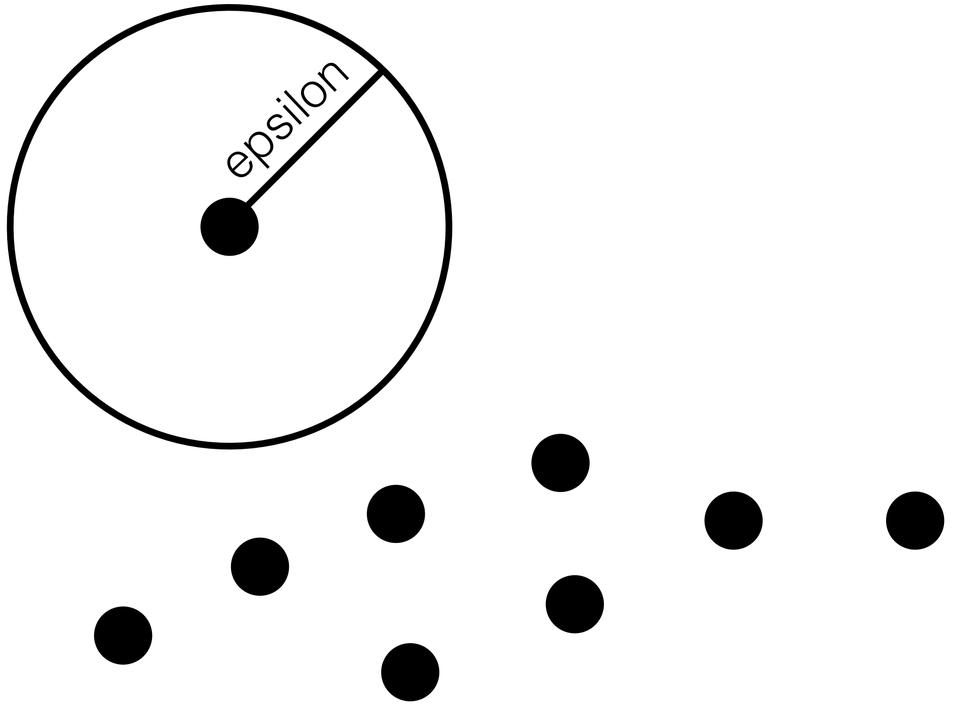
Algorithms
- DBSCAN [Ester, et al. (KDD 1996)]
- CLIQUE [Agrawal, et al. (KDD 1998)]
- OPTICS [Ankerst, et al. (SIGMOD 1999)]

Basic concepts

Instead of using on similarity (distance) to cluster datapoints we are going to look at density

Parameters

- Radius of circle (epsilon)
- Min number of points in that neighbourhood (minPoints)



Basic concepts

minPts = 3

Core points

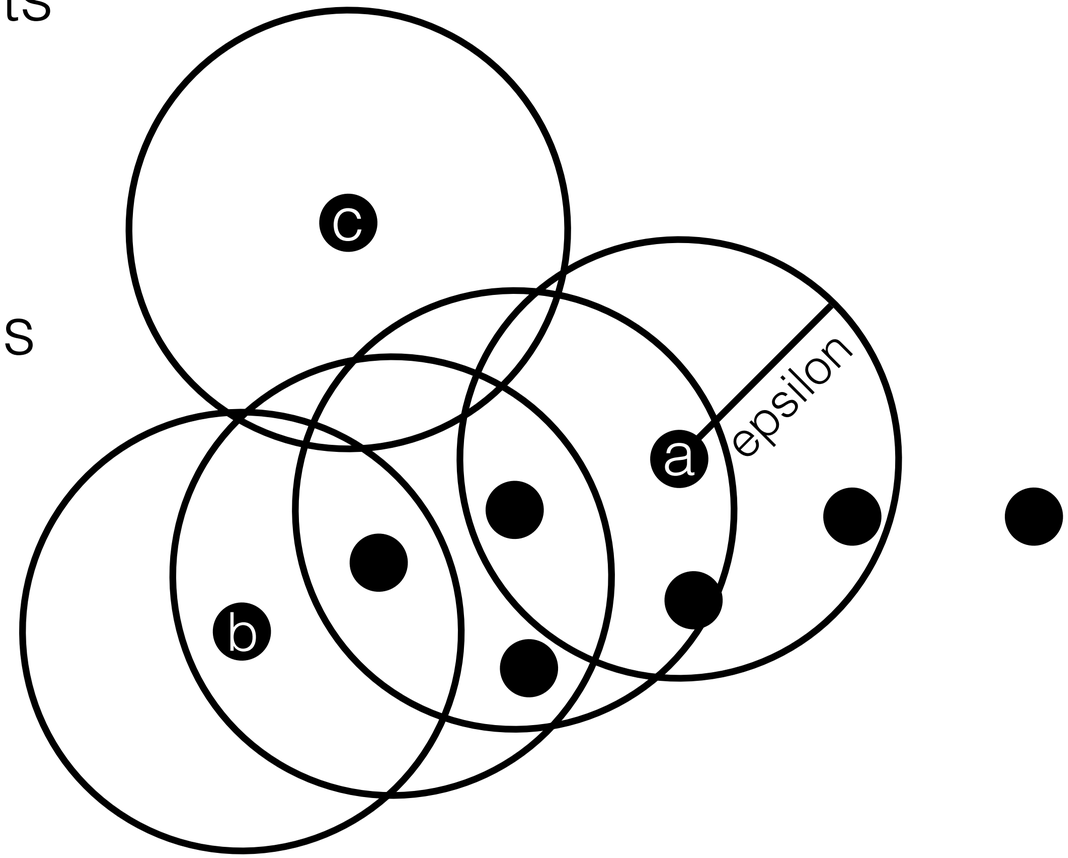
A point (a) is a core point if at least minPts points are within a distance epsilon of it

Reachable points

A point (b) is reachable from (a) if there is a path where all points on the path are core points

Noise

Points not reachable from any other point are outliers (c)



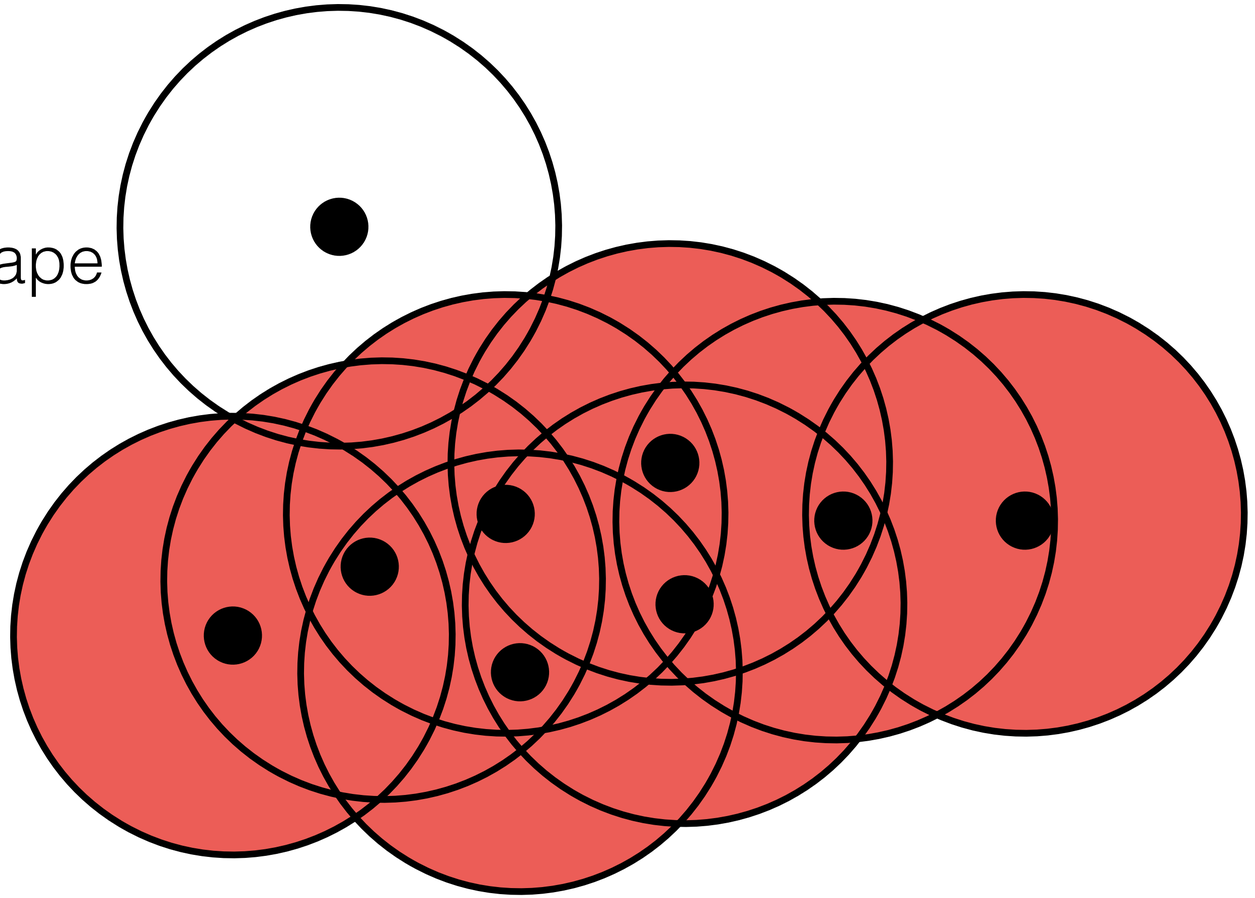
DBSCAN

Density-Based Spatial Clustering of Applications with Noise

- Utilizes the notion of reachability

- Defines a cluster as the maximal set of connected points

- Can discover clusters with arbitrary shape



Any caveats ?

- Assumes that clusters have identical densities

DBSCAN
Algorithm

1) Select an arbitrary data-point, *p*

Repeat

2) Find all points that are reachable (epsilon, minPts)

3) If *p* is a core points, form a cluster, otherwise go to next data-point

Continue until

All points have been visited

demo

http://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

DBSCAN
in Spark

Unfortunately DBSCAN is not implemented in Spark yet, however,
Spark friendly implementations do exist on GitHub

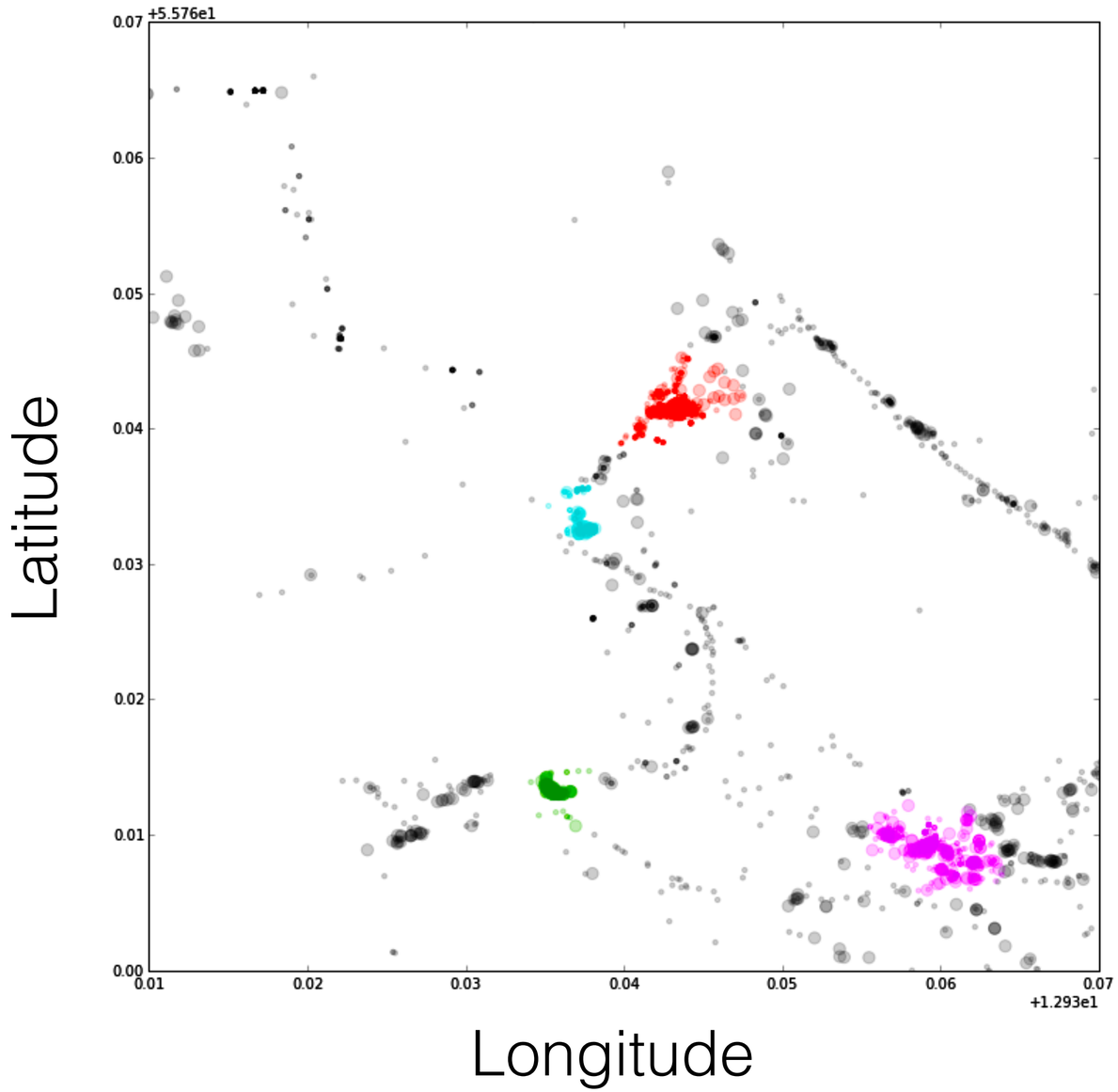
If the number of data-points per key it not excessively large we can cache the data in memory and apply non-distributed versions of DBSCAN

Today we are going to use python's implementation of DBSCAN

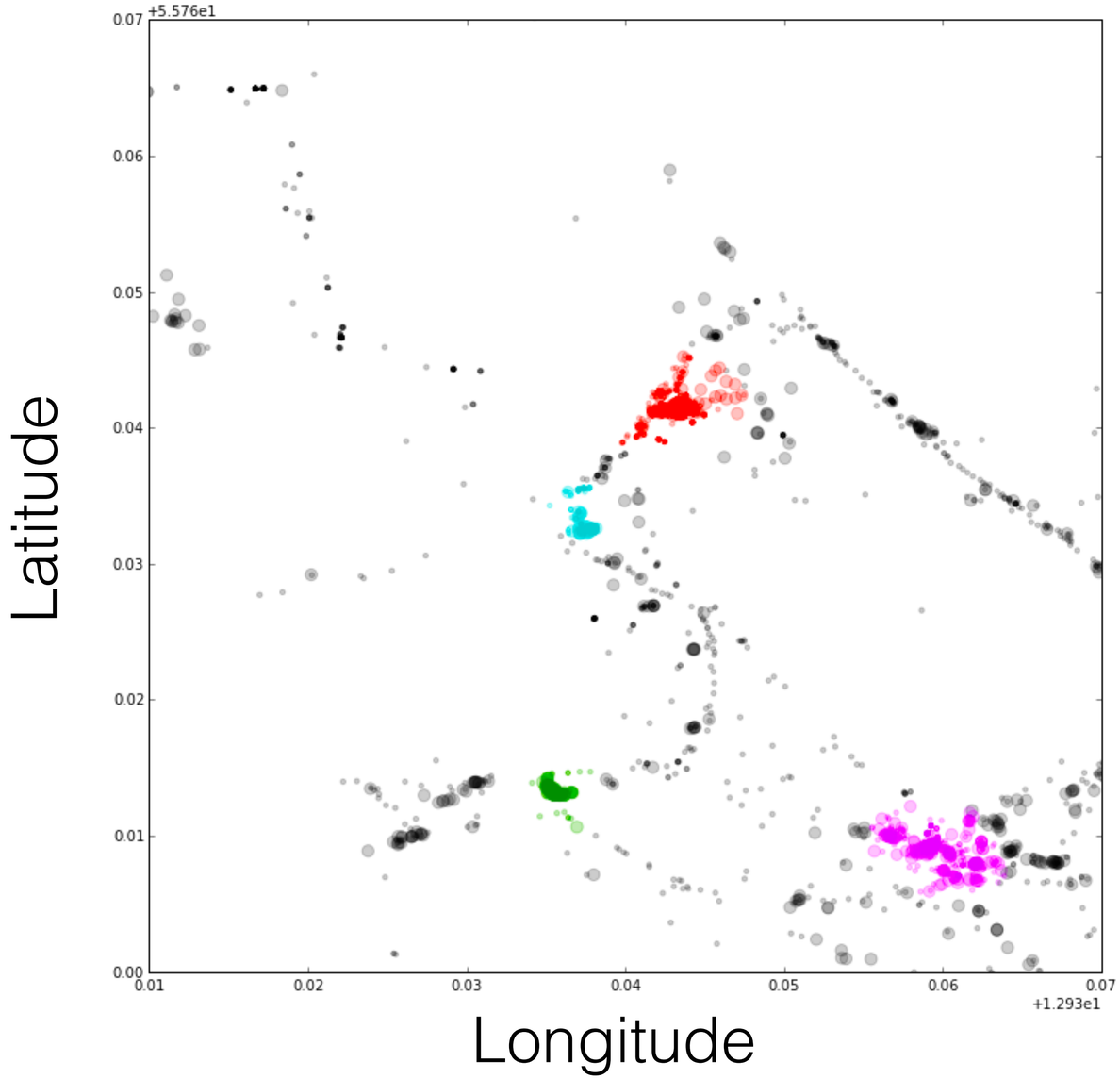
First group values by key, then apply DBSCAN on each key using map()

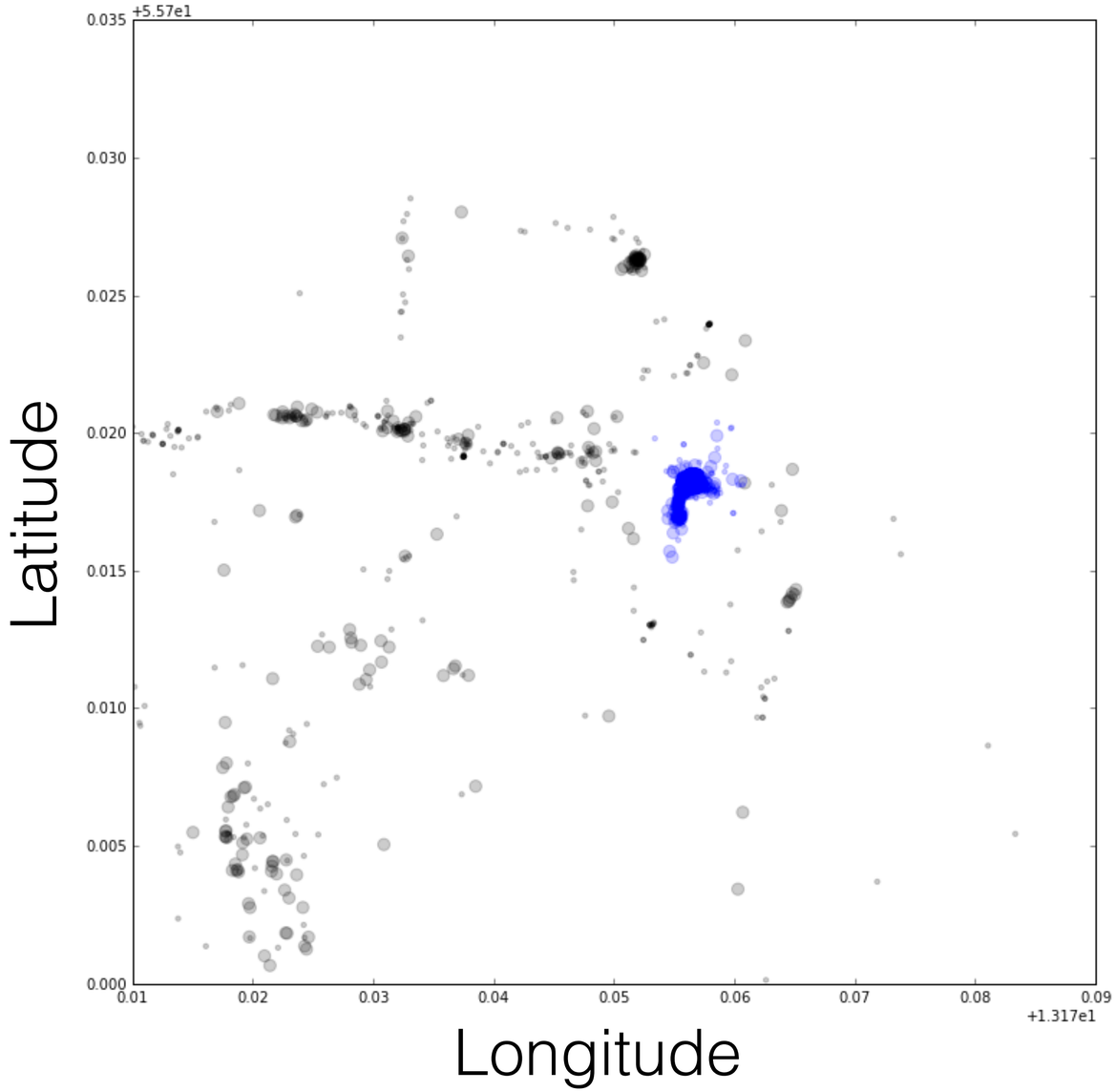
```
from sklearn.cluster import DBSCAN
dbscan = DBSCAN(eps,minPts)
clusterings = dataRdd.map(lambda (k,v): dbscan.fit(v))
```

Points of **interest**



Points of **interest**





Exercise II
Clustering using DBSCAN

Dataset

Geographic traces for multiple individuals

Purpose

Apply DBSCAN to each individuals' traces to discover personal points of interest, i.e. often visited locations (high density of points)

But
first!

Assignment

Option I
Complete optional exercises of Lab4-1 and Lab4-2

Option II
Analysis of your own dataset
- Exploration
- Cleaning & filtering
- Clustering
- Interpretation (do cluster make sense?)

Clustering
methods in Spark

- k-means
  streaming k-means (dynamical clustering)

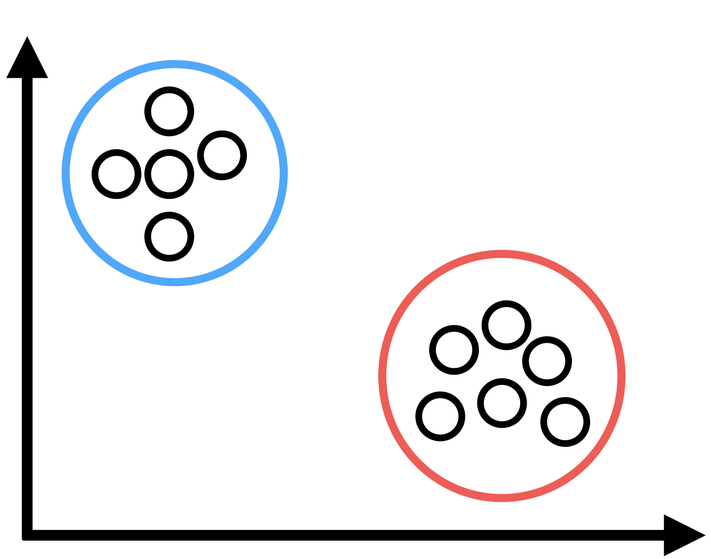
- Gaussian mixture models

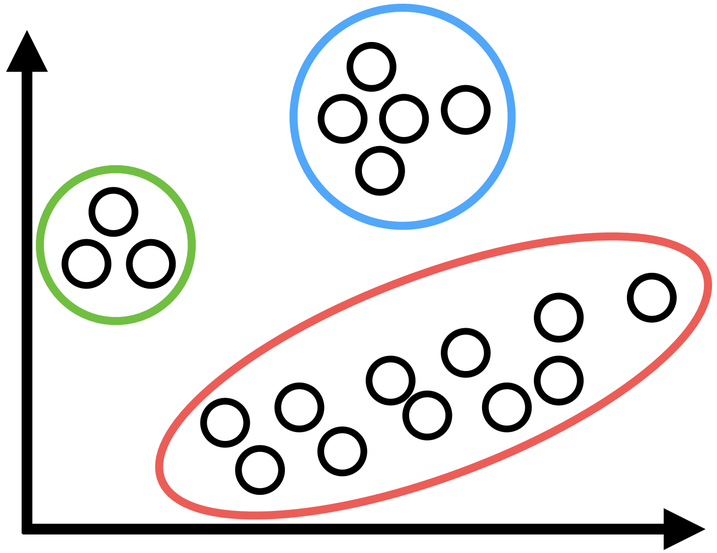
- Latent Dirichlet Allocation (LDA)

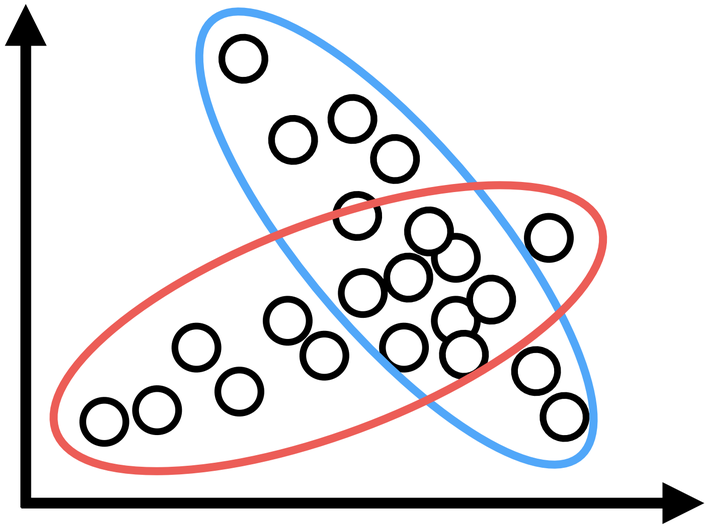
- Power Iteration Clustering (PIC)

Gaussian
mixture models

A Gaussian mixture model is a probabilistic model that assumes all data-points are generated from a mixture of a finite number of Gaussian distributions

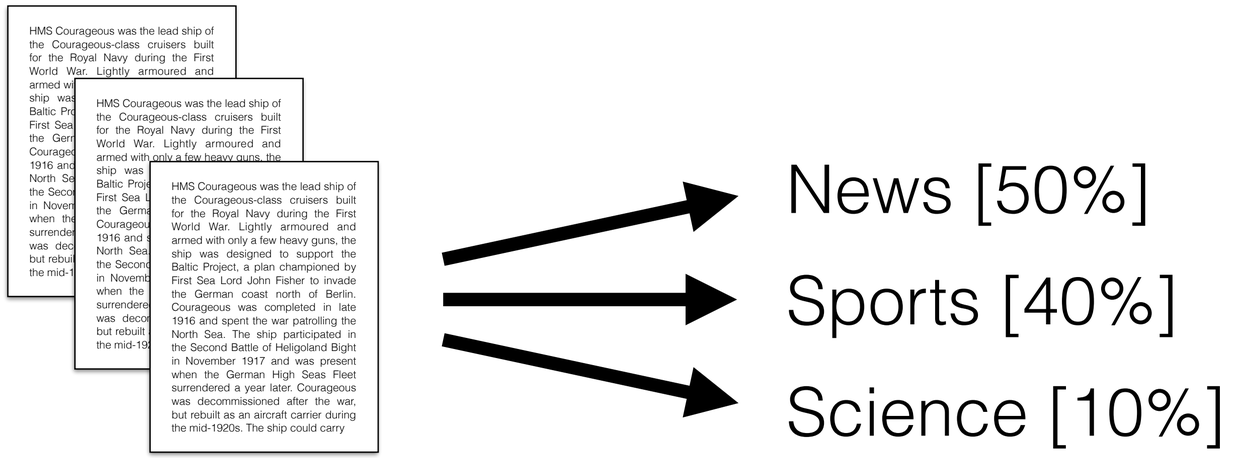






Latent
Dirichlet Allocation

LDA is a generative model that allows sets of observations to be described by unknown groups or topics

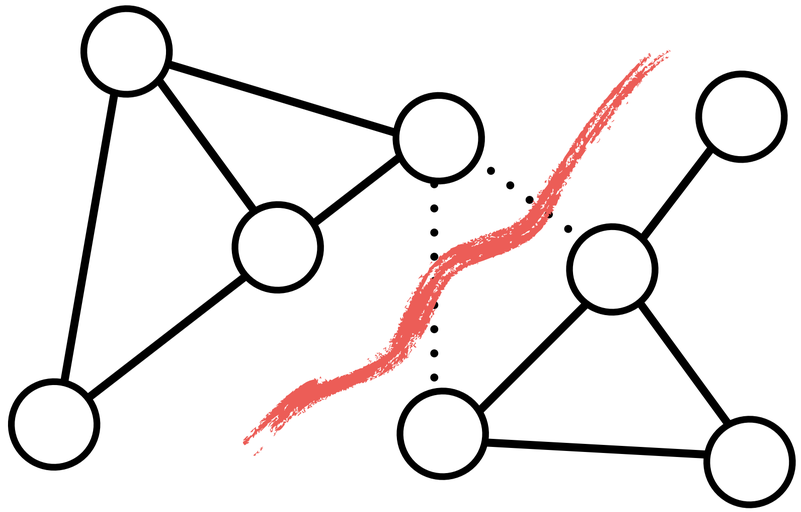


Example

LDA can look at words written in documents and determine the mixture of topics

Power
Iteration Clustering

Power Iteration Clustering uses the spectrum (eigenvalues) of the similarity matrix of the data to perform dimensionality reduction before clustering in fewer dimensions



Treats clustering problem as a graph partitioning problem.

Assignment
summary

Option I
Complete optional exercises of Lab4-1 and Lab4-2

Option II
Analysis of your own dataset
- Exploration
- Cleaning & filtering
- Clustering
- Interpretation (do cluster make sense?)

Data
If you don't have a dataset you can use
nips12raw_str602.tgz (LDA)
http://cs.nyu.edu/~roweis/data/nips12raw_str602.tgz

Quiz
Will be mailed to you