Clustering in KDD

Javier Béjar

UL - 2026 Spring Term

CS - MIA

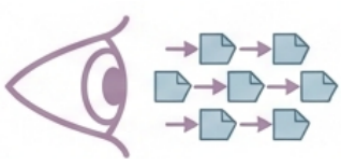
Introduction

- One of the main tasks in the KDD process is the analysis of data when we do not know its structure
- This task is very different from the task of prediction where we know the goal, and we try to approximate it
- A great part of the KDD tasks are non supervised problems (KDNuggets poll, 2-3 most frequent task)
- Problems: Scalability, arbitrary cluster shapes, limited types of data, finding the correct parameters...
- There are some new algorithms that deal with these kinds of problems

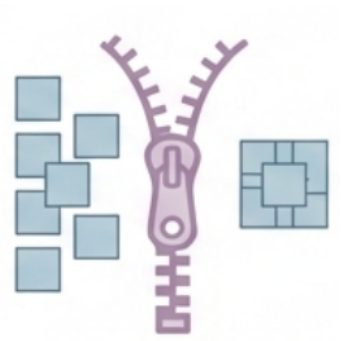
Scalability Strategies

- One-pass
  - Process data as a stream
- Summarization/Data compression
  - Compress examples to fit more data in memory
- Sampling/Batch algorithms
  - Process a subset of the data and maintain/compute a global model
- Approximation
  - Avoid expensive computations by approximate estimation
- Paralelization/Distribution
  - Divide the task into several parts and merge models

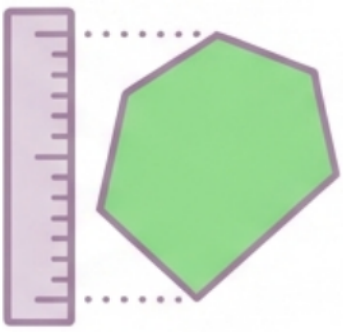
- This strategy is based on incremental clustering algorithms
- They are cheap, but order of processing affects greatly their quality
- Although they can be used as a preprocessing step
- Two steps algorithms
  1. Many clusters are generated using the one-pass algorithm
  2. A more accurate algorithm clusters the preprocessed data



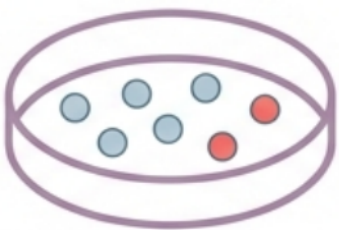
- Not all the data is necessary to discover the clusters
- Discard sets of examples and summarize by:
  - Sufficient statistics
  - Density approximations
- Discard data irrelevant for the model (do not affect the result)



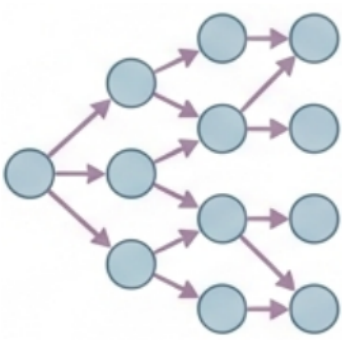
- Not using all the information available to make decisions
  - Using K-neighbours (data structures for computing k-neighbours)
- Preprocessing the data using a cheaper algorithm
  - Generate batches using approximate distances (eg: canopy clustering)
- Use approximate data structures
  - Use of hashing or approximate counts for distances and frequency computation



- Process only data that fits in memory
- Obtain from the data set:
  - Samples (process only a subset of the dataset)
    - Determine the size of the sample so all the clusters are represented
  - Batches (process all the dataset)



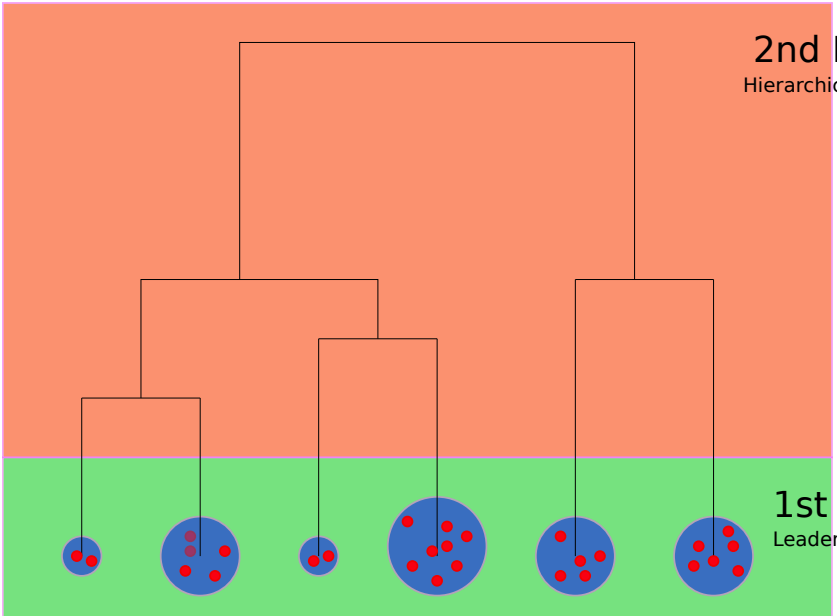
- Paralelization usually depends on the algorithm
- Some not easy to parallelize (eg: hierarchical clustering)
- Some have specific parts that can be solved in parallel or by Divide&Conquer
  - Distance computations in k-means
  - Parameter estimation in EM algorithms
  - Grid density estimations
  - Space partitioning
- Batches and sampling are more general approaches
  - The problem is how to merge all the partitions



Scalable Algorithms

Patra, Nandi, Viswanath **Distance based clustering method for arbitrary shaped clusters in large datasets** Pattern Recognition, 2011, 44, 2862-2870

- Strategy: One pass + Summarization
- The leader algorithm is used as a one pass summarization using Leader algorithm (many clusters)
- Single link is used to cluster the summaries
- Guarantees the equivalence to SL at top levels
- Summarization makes the algorithm independent of the dataset size (depends on the radius used on the leader algorithm and the volume of the data)
- Complexity $O(c^2)$


2nd Phase
Hierarchical Clustering
1st Phase
Leader Algorithm

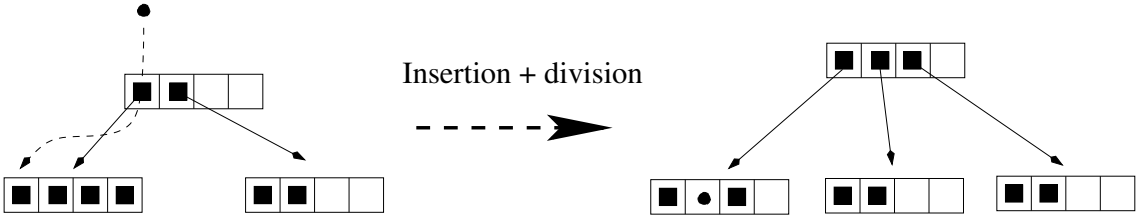
Zhang, Ramakrishnan, Livny **BIRCH: An Efficient Data Clustering Method for Very Large Databases** (1996)

- Strategy: One-pass + Summarization
- Hierarchical clustering with limited memory
- Incremental algorithm
- Based on probabilistic prototypes and distances
- We need two passes from the database
- Based on a specialized data structure named CF-tree (Clustering Feature Tree)

- Balanced n-ary tree containing clusters represented by probabilistic prototypes
- Leaves have a capacity of L prototypes and the clusters radius can not be more than T
- Non-terminal nodes have a fixed branching factor (B), each element summarizes its subtree
- Choice of parameters is crucial because available space could be filled during the process
- This is solved by changing the parameters (basically T), and recompressing the tree (T determines the granularity of the final groups)

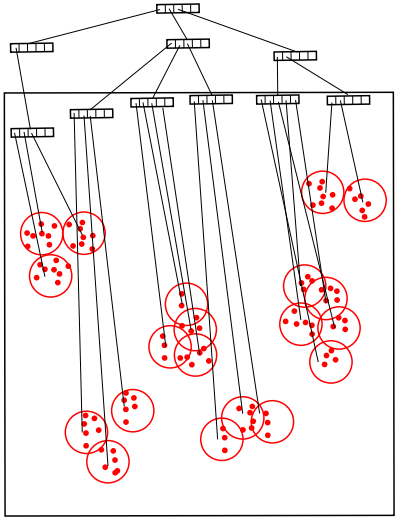
1. Traverse the tree until reaching a leave and choose the nearest prototype
2. On this leave we could introduce the instance into an existing group or create a new prototype depending on if the distance is larger than parameter T
3. If the current leave has no space for the new prototype, then create a new terminal node, and distribute the prototypes among the current node and the new node

4. The distribution is performed choosing the two most different prototypes and dividing the rest using their proximity to these two prototypes
5. This creates a new node in the ascendant node, if the new node exceeds the capacity of the father then it is split, and the process is continued until the root of the tree is reached if necessary
6. Additionally, we could perform merge operations to compact the tree and reduce space

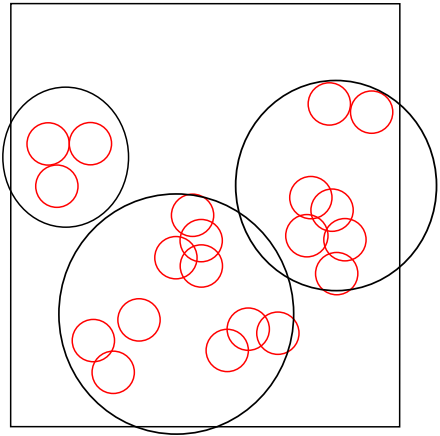

Insertion + division

1. **Phase 1:** Construction of the CF-tree, we obtain a hierarchy that summarizes the database as a set of groups which granularity is defined by T
2. **Phase 2:** Optionally we modify the CF-tree in order to reduce its size by merging near groups and deleting outliers
3. **Phase 3:** We use the prototypes inside the leaves of the trees as new instances, and we run a clustering algorithm with them (for instance K-means)
4. **Phase 4:** We refine the groups assigning the instances from the original database to the prototypes obtained in the previous phase

1st Phase - CFTree



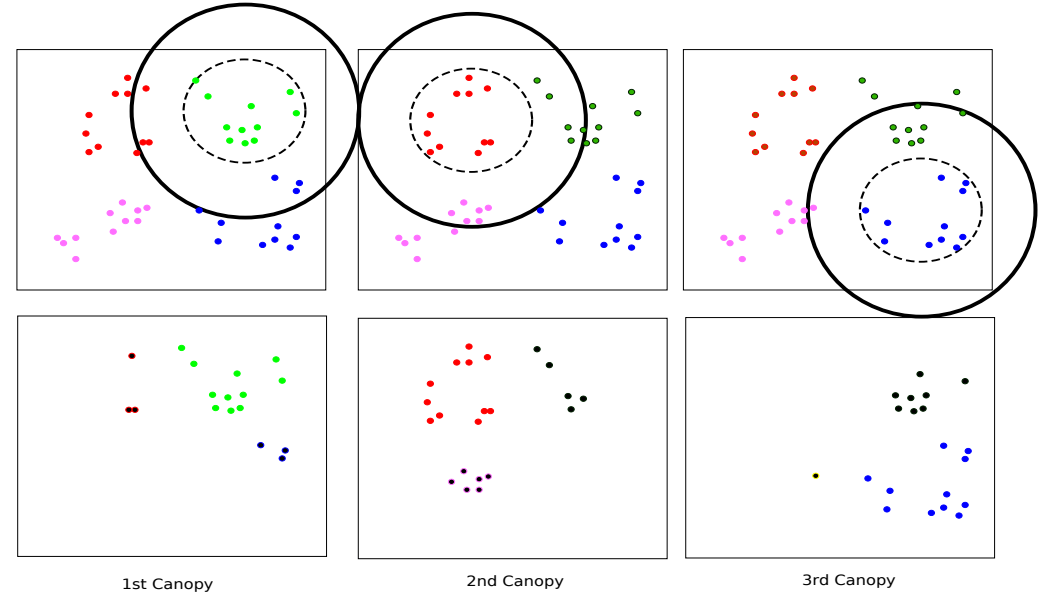
2nd Phase - Kmeans



McCallum, Nigam, Ungar **Efficient clustering of high-dimensional data sets with application to reference matching** (2002)

- Strategy: Divide & Conquer + Approximation
- The approach is based on a two stages clustering
- The first stage can be seen as a pre-process to determine the neighbourhood of the densities and aimed to reduce the number of distances to compute on the second stage
- This first stage is the called *canopy clustering*, relies on a cheap distance and two parameters $T_1 > T_2$
- These parameters are used as two centered spheres that determine how to classify the examples.

- Algorithm:
  1. One example is picked at random, and the cheap distance from this example to the rest is computed
  2. All the examples that are at less than T2 are deleted and included in the canopy
  3. The points at less than T1 are added to the canopy of these examples without deleting them
  4. The procedure is repeated until the example list is empty
  5. Canopies can share examples
- After that the data can be clustered with different algorithms
- For agglomerative clustering only the distances among the examples in the canopies have to be computed


1st Canopy
2nd Canopy
3rd Canopy

Sculley **Web-scale k-means clustering** Proceedings of the 19th international conference on World wide web, 2010, 1177-1178

- Strategy: Sampling
- Apply K-means to a sequence of bootstrapped samples of the data
- Each iteration the samples are assigned to prototypes, and the prototypes are updated with the new sample
- Each iteration the weight of the samples is reduced (learning rate)
- The quality of the results depends on the size of the batches
- Convergence is detected when prototypes are stable

Given: k, mini-batch size b, iterations t, data set X

Initialize each c $\in$ C with an x picked randomly from X

v $\leftarrow$ 0

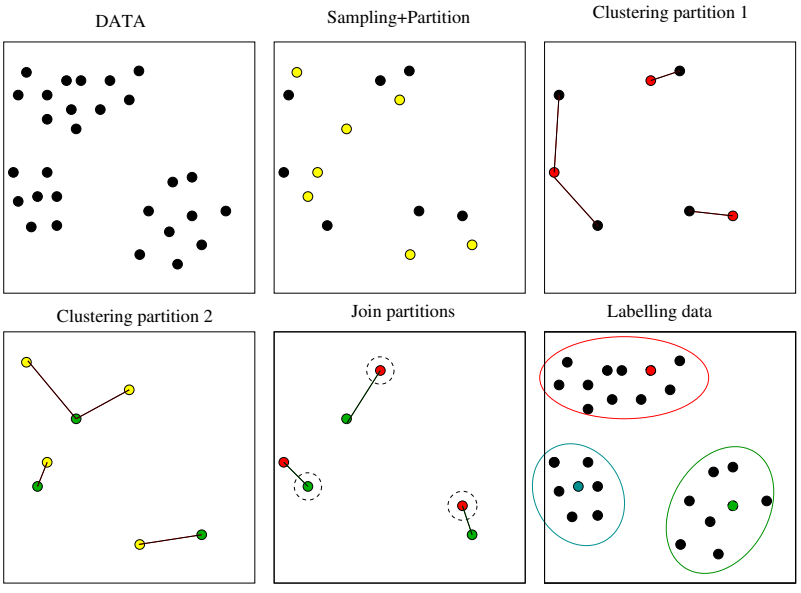
for $i \leftarrow 1$ **to** t **do**

- M $\leftarrow$ b examples picked randomly from X
- **for** $x \in M$ **do**
  - d[x] $\leftarrow$ f(C,x)
- **for** $x \in M$ **do**
  - c $\leftarrow$ d[x]
  - v[c] $\leftarrow$ v[c] + 1
  - $\eta \leftarrow \frac{1}{v[c]}$
  - c $\leftarrow$ (1-η)c+ηx

Guha, Rastogi, Shim **CURE: An efficient clustering algorithm for large databases** (1998)

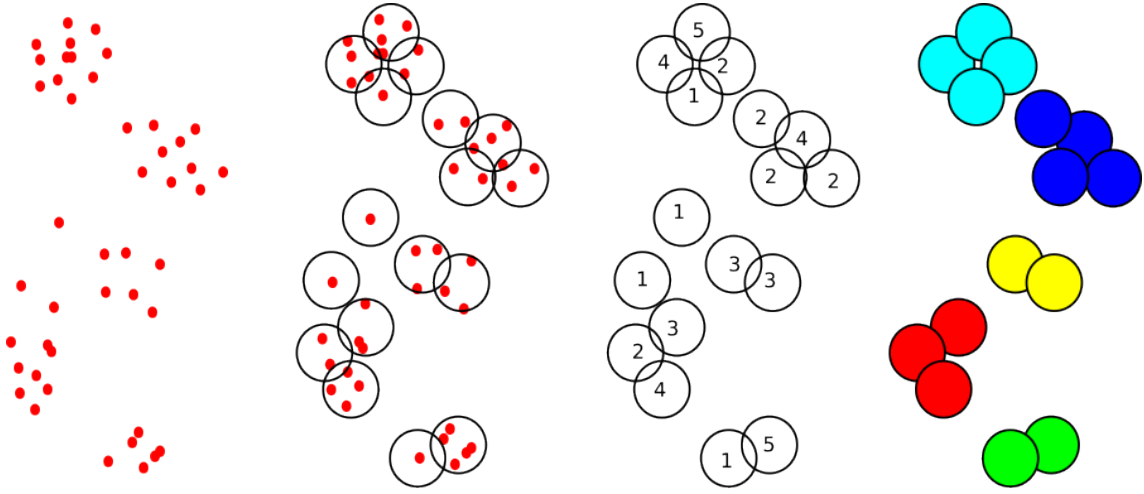
- Strategy: Sampling + Divide & Conquer
- Hierarchical agglomerative clustering
- Scalability is obtained by using sampling techniques and partitioning the dataset
- Uses a set of representatives (c) for a cluster instead of centroids (non-spherical groups)
- Distance is computed as the nearest representative among groups
- The clustering algorithm is agglomerative and merges pairs of groups until k groups are obtained

1. Draws a random sample from the dataset
2. Partitions the sample in p groups
3. Executes the clustering algorithm on each partition
4. Deletes outliers
5. Runs the clustering algorithm on the union of all groups until it obtains k groups
6. Label the data accordingly to the similarity to the k groups


DATA
Sampling+Partition
Clustering partition 1
Clustering partition 2
Join partitions
Labelling data

Viswanath, Babu **Rough-DBSCAN: A fast hybrid density based clustering method for large data sets** Pattern Recognition Letters, 2009, 30, 1477 - 1488

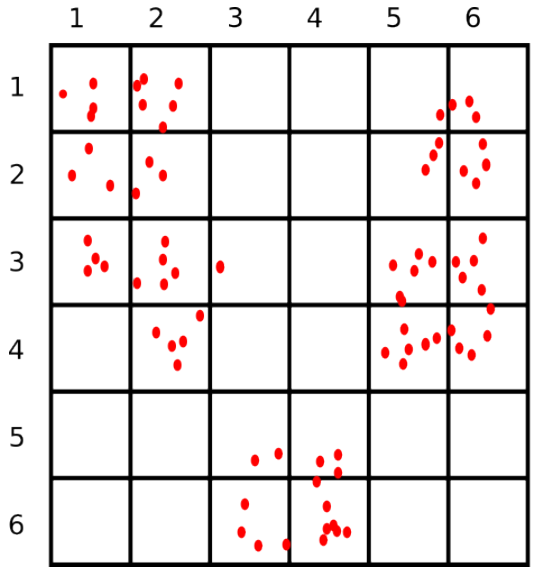
- Strategy: One-pass + Summarization
- Two stages algorithm:
  1. Preprocess using the leader algorithm
     - Determine the examples that belong to the higher densities and their number of neighbours
  2. Apply DBSCAN algorithm
     - Determine the densities for the selected instances
     - Approximate the values of the densities from their distances, and the sizes of the neighbourhood
     - Assign the neighbours accordingly to the found densities



1 2 3

Esfandiari, Mirrokni, Zhong **Almost linear time density level set estimation via dbscan** Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, no. 8, pp. 7349-7357. 2021

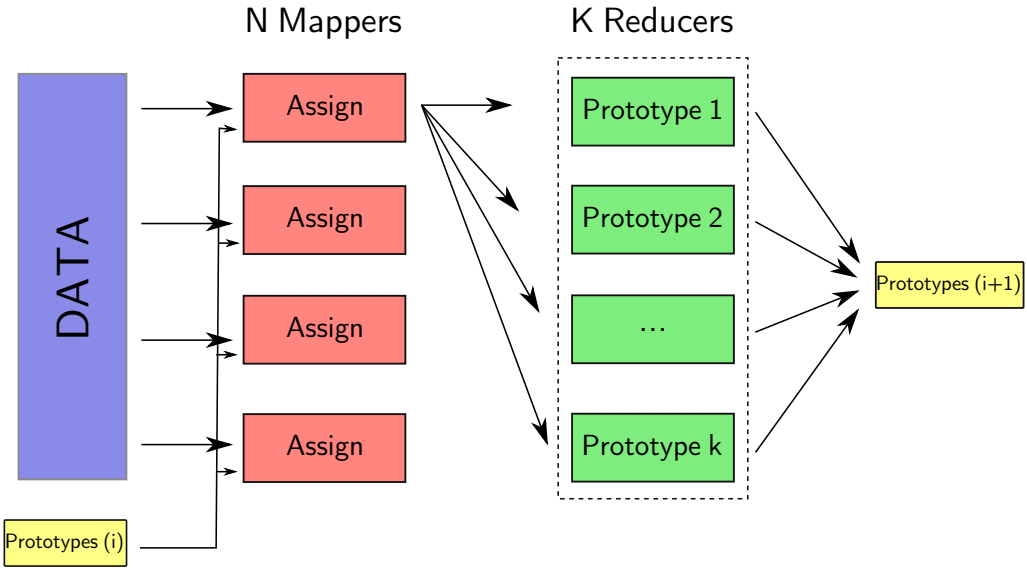
- Strategy: Approximation
- Uses discretization to approximate the densities and computing the core points (discretization to cubes of side 2ϵ) avoiding computing K-nn
- Matches the core points using Locality Sentitive Hashing connecting them in a graph
- Connects non core points connecting to core points that share at least one hashing
- Obtains the cluster by computing the connected components
- The algorithm is log linear


1
2
3
4
5
6
1
2
3
4
5
6
1,1=4
1,2=6
1,5=1
1,3=3
.
.
.
.
.
.
.
6,3=4
6,4=7

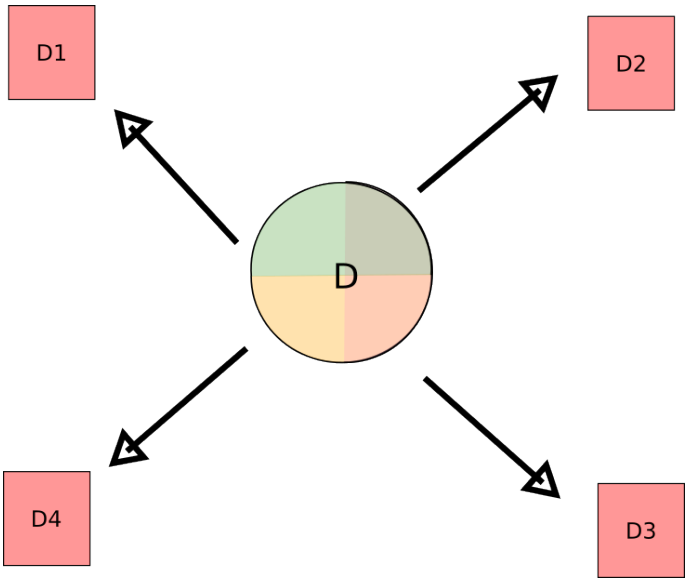
- Obtain t hashing functions obtaining values uniformly from $[0, 2\epsilon]$ where the hash is the discretization of $x + \epsilon$
- For all core points, compute the sets corresponding to the hash buckets and add edges from a random core point in the set to the rest (core points are connected t times)
- For all non core points, if there is any core point that shares a bucket with them (any hashing), add an edge to the core point (unconnected points are noise)
- Compute the connected components of the graph

Zhao, W., Ma, H., He, Q. **Parallel K-Means Clustering Based on MapReduce** Cloud Computing, LNCS 5931, 674-679, Springer 2009

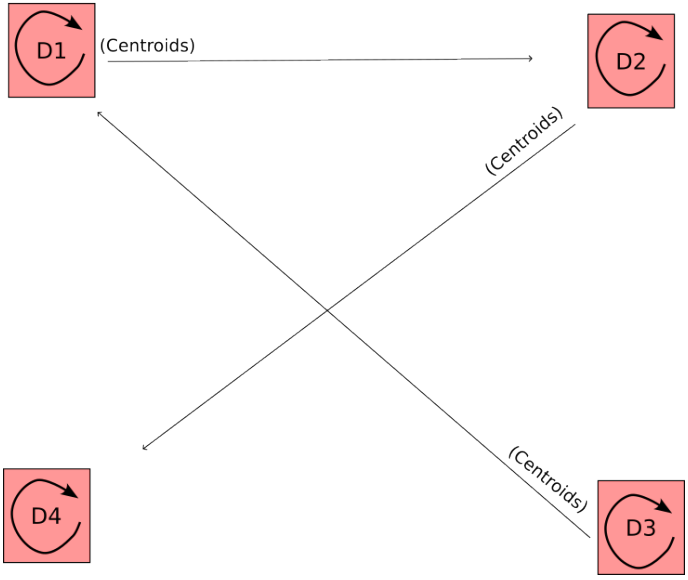
- Strategy: Distribution/Divide & Conquer
- Applied to K-means and GMM
- Mappers have a copy of the current centroids and assigns the closest one to examples
- Reducers compute the new centroids according to the assignments


N Mappers
K Reducers
DATA
Assign
Assign
Assign
Assign
Prototype 1
Prototype 2
...
Prototype k
Prototypes (i+1)
Prototypes (i)

- Use Peer2Peer networks as a divide and conquer strategy
- Each member of the network processes a chunk of the data
- Peers interchange messages with data or intermediate results
- Strategies:
  - **Synchronous**: All peers interchange information at timed intervals
  - **Asynchronous**: All peers work independently and interchange information randomly
- Messages:
  - Messages consist of prototypes that are integrated (possibly with a weighting strategy)
  - Messages consist of examples or summarized examples


D1
D2
D
D4
D3

Iterate K-means - comunicate centroids - integrate centroids



Clustering of Data Streams

Data streams: Modelling an on-line continuous series of data

- Each item of the series is an example (one value, a vector of values, structured data)
  - For instance, sensory data (one or multiple synchronized data), stream of documents (twitter/news)
- Data are generated from a set of clusters (stable or changing over time)
  - For instance, states from a process or semantic topics

- Data are processed incrementally (model changes with time)
  - Only the current model
  - Periodic snapshots
- Different goals:
  - Model the domain
  - Detect anomalies/novelty/bursts
  - Detect change (Concept drift)

- Clustering has an on-line and an off-line phase
- Elements:
  - The data structure used to summarize the data
  - The window model used to decide the influence of the current and past data
  - The mechanism for identifying outliers
  - The clustering algorithm used to obtain the partition of the data

- Involved in the on-line phase
- Data are summarized using sufficient statistics (num of examples, sum of values, sum of squared products of values...)
- Usually a hierarchical data-structure (different levels of granularity)
- Indexing structure that can be updated incrementally
- Stores raw data or prototypes depending on space constraints

- Sliding window model
  - Fixed time window
  - Only data inside the window updates the structure
- Damped window model
  - A weight is associated to examples and clusters
  - Influence of data depends on time, old data fade away or are discarded
- Landmark window model
  - Defines points of interest in time or amount of data
  - Data before the landmark are discarded

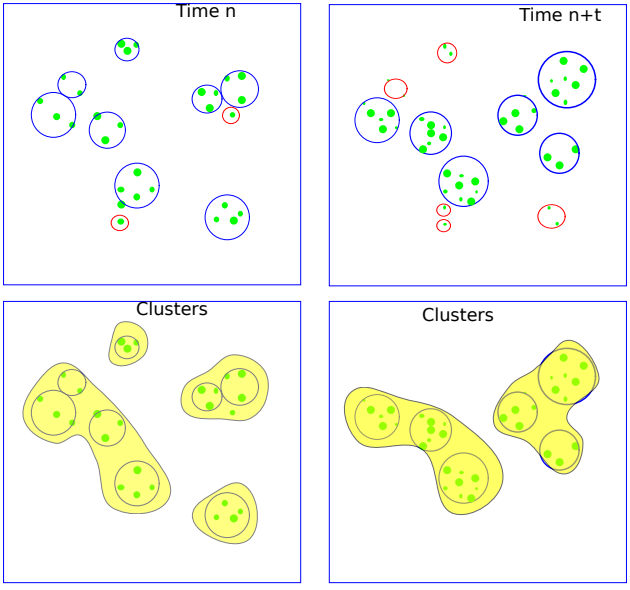
- Difficult task because data evolve with time
- Most methods work around the idea of microclusters
- A micro-cluster represents a dense area in the space of examples
- The indexing structure tracks the evolution of the micro-clusters
- Different thresholds determine if a micro-cluster is kept or discarded

Aggarwal et al. **On Clustering Massive Data Streams: A Summarization Paradigm** Data Streams-Models and Algorithms, Springer, 2007, 31, 9-38

- On-line phase:
  - Maintains micro-clusters (more than final number of clusters)
  - New data is incorporated to a micro-cluster or generates new micro-clusters
  - The number of micro-clusters is fixed, they are merged to maintain the number
  - Periodically the micro-clusters are stored
- Off-line phase:
  - Given a time window the stored micro-clusters are used to compute the micro-clusters inside the time frame
  - K-means used to compute the clusters for the time window

Cao, Ester, Qian, Zhou **Density-Based Clustering over an Evolving Data Stream with Noise** Proceedings of the Sixth SIAM International Conference on Data Mining, 2006

- On-line phase:
  - Core-micro-clusters (a weighted sum of close points)
  - The weight of a point fades exponentially with time (damping window model)
  - New examples are merged and mc are classified as:
    - core-mc, sets of points with weight over a threshold
    - potential-mc
    - outlier-mc, sets of points with weight below a threshold
  - outlier-mc dissapear with time
- Off-line phase: Modified version of DBSCAN


Time n
Time n+t
Clusters
Clusters

This Python Notebook has examples comparing K-means algorithm with two scalable algorithms Mini Batch K-means and BIRCH

- Clustering DM Notebook (click here to open the notebook in colab)

If you download the notebook you will be able to use it locally (run jupyter notebook to open the notebooks)