



# Clustering

Albert Bifet



THE UNIVERSITY OF  
**WAIKATO**  
*Te Whare Wānanga o Waikato*

May 2012

# COMP423A/COMP523A Data Stream Mining

## Outline

1. Introduction
2. Stream Algorithmics
3. Concept drift
4. Evaluation
5. Classification
6. Ensemble Methods
7. Regression
8. **Clustering**
9. Frequent Pattern Mining
10. Distributed Streaming



## Big Data & Real Time

# Clustering

## Definition

Clustering is the distribution of a set of instances of examples into non-known groups according to some common relations or affinities.

## Example

Market segmentation of customers

## Example

Social network communities

# Clustering

## Definition

Given

- ▶ a set of instances  $I$
- ▶ a number of clusters  $K$
- ▶ an objective function  $cost(I)$

a clustering algorithm computes an assignment of a cluster for each instance

$$f : I \rightarrow \{1, \dots, K\}$$

that minimizes the objective function  $cost(I)$

# Clustering

## Definition

Given

- ▶ a set of instances  $I$
- ▶ a number of clusters  $K$
- ▶ an objective function  $cost(C, I)$

a clustering algorithm computes a set  $C$  of instances with  $|C| = K$  that minimizes the objective function

$$cost(C, I) = \sum_{x \in I} d^2(x, C)$$

where

- ▶  $d(x, c)$ : distance function between  $x$  and  $c$
- ▶  $d^2(x, C) = \min_{c \in C} d^2(x, c)$ : distance from  $x$  to the nearest point in  $C$

# k-means

- ▶ 1. Choose  $k$  initial centers  $C = \{c_1, \dots, c_k\}$
- ▶ 2. while stopping criterion has not been met
  - ▶ For  $i = 1, \dots, N$ 
    - ▶ find closest center  $c_k \in C$  to each instance  $p_i$
    - ▶ assign instance  $p_i$  to cluster  $C_k$
  - ▶ For  $k = 1, \dots, K$ 
    - ▶ set  $c_k$  to be the center of mass of all points in  $C_i$

# k-means++

- ▶ 1. Choose a initial center  $c_1$
- ▶ For  $k = 2, \dots, K$ 
  - ▶ select  $c_k = p \in I$  with probability  $d^2(p, C)/cost(C, I)$
- ▶ 2. while stopping criterion has not been met
  - ▶ For  $i = 1, \dots, N$ 
    - ▶ find closest center  $c_k \in C$  to each instance  $p_i$
    - ▶ assign instance  $p_i$  to cluster  $C_k$
  - ▶ For  $k = 1, \dots, K$ 
    - ▶ set  $c_k$  to be the center of mass of all points in  $C_i$



# Performance Measures

## Internal Measures

- ▶ Sum square distance
- ▶ Dunn index  $D = \frac{d_{min}}{d_{max}}$
- ▶ C-Index  $C = \frac{S - S_{min}}{S_{max} - S_{min}}$

## External Measures

- ▶ Rand Measure
- ▶ F Measure
- ▶ Jaccard
- ▶ Purity

# BIRCH

## BALANCED ITERATIVE REDUCING AND CLUSTERING USING HIERARCHIES

- ▶ Clustering Features  $CF = (N, LS, SS)$ 
  - ▶ N: number of data points
  - ▶ LS: linear sum of the N data points
  - ▶ SS: square sum of the N data points
  - ▶ Properties:
    - ▶ Additivity:  $CF_1 + CF_2 = (N_1 + N_2, LS_1 + LS_2, SS_1 + SS_2)$
    - ▶ Easy to compute: average inter-cluster distance and average intra-cluster distance
- ▶ Uses CF tree
  - ▶ Height-balanced tree with two parameters
    - ▶ B: branching factor
    - ▶ T: radius leaf threshold

# BIRCH

## BALANCED ITERATIVE REDUCING AND CLUSTERING USING HIERARCHIES

- Phase 1: Scan all data and build an initial in-memory CF tree
- Phase 2: Condense into desirable range by building a smaller CF tree (optional)
- Phase 3: Global clustering
- Phase 4: Cluster refining (optional and off line, as requires more passes)

# Clu-Stream

## Clu-Stream

- ▶ Uses micro-clusters to store statistics on-line
  - ▶ Clustering Features  $CF = (N, LS, SS, LT, ST)$ 
    - ▶ N: numer of data points
    - ▶ LS: linear sum of the N data points
    - ▶ SS: square sum of the N data points
    - ▶ LT: linear sum of the time stamps
    - ▶ ST: square sum of the time stamps
- ▶ Uses pyramidal time frame

# Clu-Stream

## On-line Phase

- ▶ For each new point that arrives
  - ▶ the point is absorbed by a micro-cluster
  - ▶ the point starts a new micro-cluster of its own
    - ▶ delete oldest micro-cluster
    - ▶ merge two of the oldest micro-cluster

## Off-line Phase

- ▶ Apply k-means using microclusters as points

# Density based methods

## DBSCAN

- ▶  $\epsilon$ -neighborhood( $p$ ): set of points that are at a distance of  $p$  less or equal to  $\epsilon$
- ▶ Core object: object whose  $\epsilon$ -neighborhood has an overall weight at least  $\mu$
- ▶ A point  $p$  is *directly density-reachable* from  $q$  if
  - ▶  $p$  is in  $\epsilon$ -neighborhood( $q$ )
  - ▶  $q$  is a core object
- ▶ A point  $p$  is *density-reachable* from  $q$  if
  - ▶ there is a chain of points  $p_1, \dots, p_n$  such that  $p_{i+1}$  is directly density-reachable from  $p_i$
- ▶ A point  $p$  is *density-connected* from  $q$  if
  - ▶ there is point  $o$  such that  $p$  and  $q$  are density-reachable from  $o$

# Density based methods

## DBSCAN

- ▶ A *cluster*  $C$  of points satisfies
  - ▶ if  $p \in C$  and  $q$  is density-reachable from  $p$ , then  $q \in C$
  - ▶ all points  $p, q \in C$  are density-connected
- ▶ A *cluster* is uniquely determined by any of its core points
- ▶ A *cluster* can be obtained
  - ▶ choosing an arbitrary core point as a seed
  - ▶ retrieve all points that are density-reachable from the seed

# Density based methods

## DBSCAN

- ▶ select an arbitrary point  $p$
- ▶ retrieve all points density-reachable from  $p$
- ▶ if  $p$  is a core point, a cluster is formed
- ▶ If  $p$  is a border point
  - ▶ no points are density-reachable from  $p$
  - ▶ DBSCAN visits the next point of the database
- ▶ Continue the process until all of the points have been processed



# Density based methods

## DenStream

- ▶  $\epsilon$ -neighborhood( $p$ ): set of points that are at a distance of  $p$  less or equal to  $\epsilon$
- ▶ Core object: object whose  $\epsilon$ -neighborhood has an overall weight at least  $\mu$
- ▶ Density area: union of the  $\epsilon$ -neighborhood of core objects

# Density based methods

## DenStream

For a group of points  $p_{i_1}, p_{i_2}, \dots, p_{i_n}$ ,  
with time stamps  $T_{i_1}, T_{i_2}, \dots, T_{i_n}$

- ▶ core-micro-cluster
  - ▶  $w = \sum_{j=1}^n f(t - T_{i_j})$  where  $f(t) = 2^{-\lambda t}$  and  $w \geq \mu$
  - ▶  $c = \sum_{j=1}^n f(t - T_{i_j}) p_{i_j} / w$
  - ▶  $r = \sum_{j=1}^n f(t - T_{i_j}) \text{dist}(p_{i_j}, c) / w$  where  $r \leq \epsilon$
- ▶ potential core-micro-cluster
  - ▶  $w = \sum_{j=1}^n f(t - T_{i_j})$  where  $f(t) = 2^{-\lambda t}$  and  $w \geq \beta \mu$
  - ▶  $\overline{CF^1} = \sum_{j=1}^n f(t - T_{i_j}) p_{i_j}$
  - ▶  $\overline{CF^2} = \sum_{j=1}^n f(t - T_{i_j}) p_{i_j}^2$  where  $r \leq \epsilon$
- ▶ outlier micro-cluster:  $w < \beta \mu$

# DenStream

## On-line Phase

- ▶ For each new point that arrives
  - ▶ try to merge to a p-micro-cluster
  - ▶ else, try to merge to nearest o-micro-cluster
    - ▶ if  $w > \beta\mu$  then
      - ▶ convert the o-micro-cluster to p-micro-cluster
  - ▶ otherwise create a new o-microcluster

## Off-line Phase

- ▶ for each p-micro-cluster  $c_p$ 
  - ▶ if  $w < \beta\mu$  then remove  $c_p$
- ▶ for each o-micro-cluster  $c_o$ 
  - ▶ if  $w < (2^{-\lambda(t-t_o+T_p)} - 1)/(2^{-\lambda T_p} - 1)$  then remove  $c_o$
- ▶ Apply DBSCAN using microclusters as points

# ClusTree

## ClusTree: anytime clustering

- ▶ Hierarchical data structure: logarithmic insertion complexity
- ▶ Buffer and hitchhiker concept: enable anytime clustering
- ▶ Exponential decay
- ▶ Aggregation: for very fast streams

# StreamKM++: Coresets

Coreset of a set  $P$  with respect to some problem

Small subset that approximates the original set  $P$ .

- ▶ Solving the problem for the coreset provides an approximate solution for the problem on  $P$ .

$(k, \epsilon)$ -coreset

A  $(k, \epsilon)$ -coreset  $S$  of  $P$  is a subset of  $P$  that for each  $C$  of size  $k$

$$(1 - \epsilon)cost(P, C) \leq cost_w(S, C) \leq (1 + \epsilon)cost(P, C)$$

# StreamKM++: Coresets

## Coreset Tree

- ▶ Choose a leaf  $l$  node at random
- ▶ Choose a new sample point denoted by  $q_{t+1}$  from  $P_l$  according to  $d^2$
- ▶ Based on  $q_l$  and  $q_{t+1}$ , split  $P_l$  into two subclusters and create two child nodes

## StreamKM++

- ▶ Maintain  $L = \lceil \log_2(\frac{n}{m}) + 2 \rceil$  buckets  $B_0, B_1, \dots, B_{L-1}$