Unsupervised Data Mining: From Batch to Stream Mining Algorithms

Prof. Dr. Stefan Kramer

Johannes Gutenberg-Universität Mainz

Outline

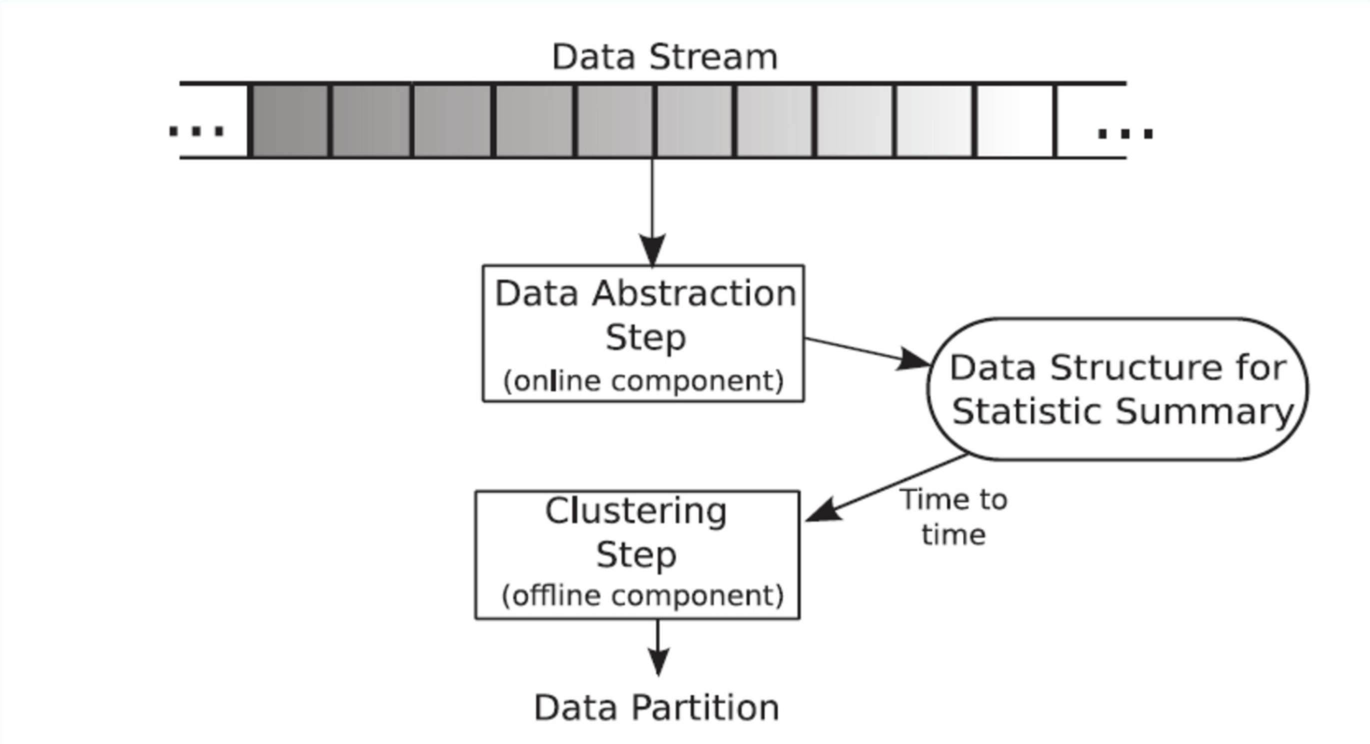
- Stream mining: clustering

Acknowledgements

- Albert Bifet
- Bernhard Pfahringer
- Thomas Seidl
- Rhada Chitta
- Christian Sohler

Stream Mining: Clustering

Stream Clustering

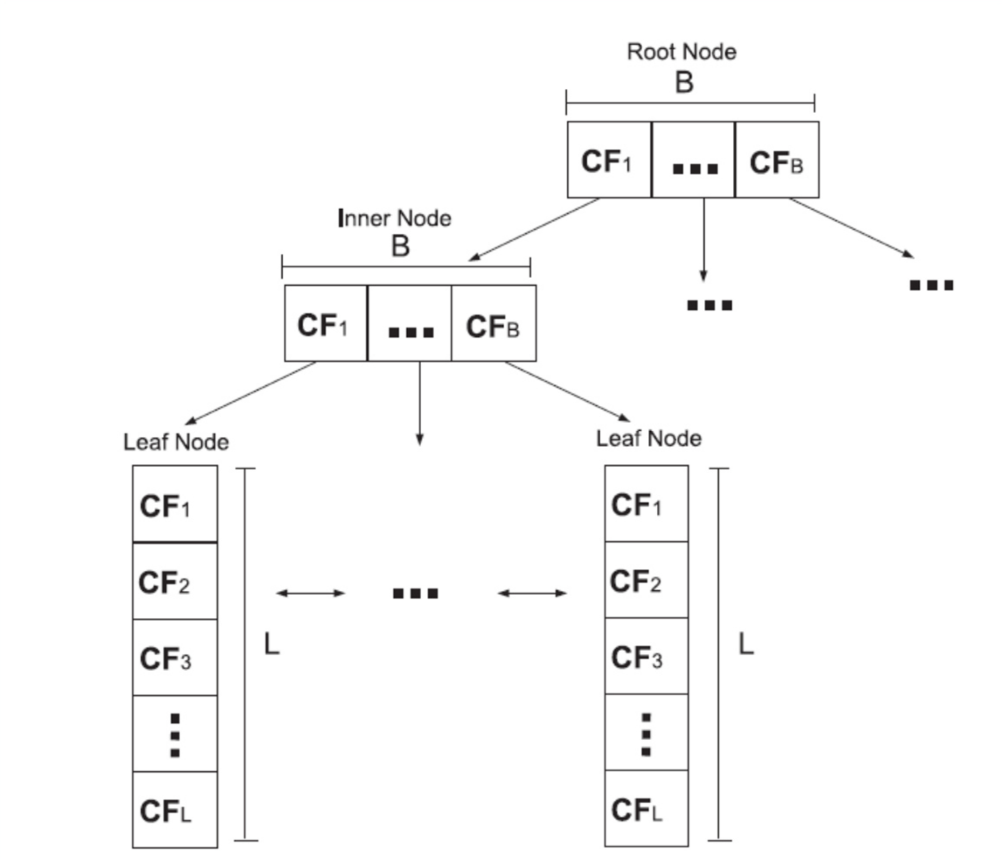


- Online Phase: summarize the data into memory-efficient data structures
- Offline Phase: use a clustering algorithm to find the data partition

Elements of Stream Clustering Algorithms

- Fast data structures to avoid linear or worse performance/look-up/insertion: trees
- *Summarizations* of data:
  - **CF-Trees**: scalable k-Means, single pass k-Means, BIRCH
  - **Microcluster Trees:** ClusTree, DenStream
  - **Coreset Trees:** StreamKM++
  - **Prototypes and point attractors:** CIPA
- Online and offline phase
- Anytime or not anytime
- Sliding window or damped window model: $f(t) = 2^{-\lambda \cdot t}$, where $\lambda > 0$ sum of weights is constant, t_c or t is current time (see DenStream)

CF-Trees



Summarize the data in each CF-vector:

- Linear sum of data points
- Squared sum of data points
- Number of points

Balanced Iterative Reducing and Clustering Using Hierarchies (BIRCH)

- Phase 1: scan all data and build an initial *in-memory* CF-tree (also based on threshold, i.e. diameter parameter; assign to closest cluster)
- Phase 2: condense into desirable range by building a smaller CF tree (optional)
- Phase 3: global clustering of CFs (leaf CFs clustered by hierarchical agglomerative clustering)
- Phase 4: cluster refinement (optional and offline, requires additional passes)

$\Rightarrow$ *Original data only scanned once!*

Definition of Clustering Feature (CF)

CF Definition : *Given N d-dimensional data points in a cluster: $\{\vec{X_i}\}$ where $i = 1, 2, ..., N$, the* **Clustering Feature (CF)** *entry of the cluster is defined as a triple:* **CF** $= (N, \vec{LS}, SS)$, *where N is the number of data points in the cluster, $\vec{LS}$ is the linear sum of the N data points, i.e., $\sum_{i=1}^{N} \vec{X_i}$, and SS is the square sum of the N data points, i.e., $\sum_{i=1}^{N} \vec{X_i}^2$.*

Quite practical! Can (1.) be used to define cluster properties, like centroids (X0), the radius (R) or the diameter:

$$\vec{X0} = \frac{\sum_{i=1}^{N} \vec{X_i}}{N}$$

$$R = \left(\frac{\sum_{i=1}^{N} (\vec{X_i} - \vec{X0})^2}{N}\right)^{\frac{1}{2}}$$

$$D = \left(\frac{\sum_{i=1}^{N} \sum_{j=1}^{N} (\vec{X_i} - \vec{X_j})^2}{N(N-1)}\right)^{\frac{1}{2}}$$

Cluster Distances Based On Clustering Features

Can (2.) used to define cluster distances, like the average inter-cluster distance (D2) and the average intra-cluster distance (D3)

$$D2 = \left(\frac{\sum_{i=1}^{N_1}\sum_{j=N_1+1}^{N_1+N_2}(\vec{X}_i - \vec{X}_j)^2}{N_1 N_2}\right)^{\frac{1}{2}}$$

$$D3 = \left(\frac{\sum_{i=1}^{N_1+N_2}\sum_{j=1}^{N_1+N_2}(\vec{X}_i - \vec{X}_j)^2}{(N_1+N_2)(N_1+N_2-1)}\right)^{\frac{1}{2}}$$

Addition and Subtraction

CF Additivity Theorem : *Assume that* $\mathbf{CF_1} = (N_1, \vec{LS_1}, SS_1)$, *and* $\mathbf{CF_2} = (N_2, \vec{LS_2}, SS_2)$ *are the CF entries of two disjoint subclusters. Then the CF entry of the subcluster that is formed by merging the two disjoint subclusters is:*

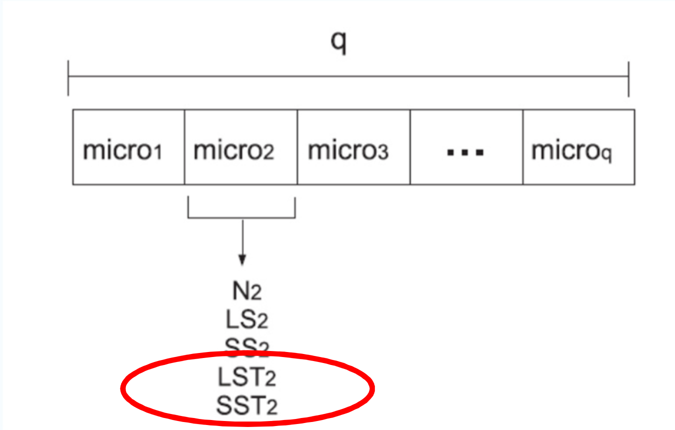
$$\mathbf{CF_1} + \mathbf{CF_2} = (N_1 + N_2, \vec{LS_1} + \vec{LS_2}, SS_1 + SS_2) \tag{11}$$

Thus, merge and split operations can be efficiently supported based on addition and subtraction of clustering features.

Microclusters

- CluStream
  - Linear sum and square sum of timestamps
  - Delete merging microclusters if timestamps are close

CF-Trees with time element



- DenStream
  - Microclusters are associated with weights based on recency
  - Outliers detected by creating separate microcluster

CluStream in More Detail

Online phase:

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

Offline phase

- Apply k-means using microclusters as points

DenStream

- ε-neighborhood(p): set of points that are at a distance of p less or equal to ε
- Core object: object whose ε-neighborhood has an overall weight at least μ
- Density area: union of the ε-neighborhood of core objects
- β: threshold of outliers relative to microclusters

DenStream in More Detail

For a group of points $p_{i_1}, p_{i_2}, \ldots, p_{i_n}$ with time stamps $T_{i_1}, T_{i_2}, \ldots, T_{i_n}$, we distinguish:

Core-micro-clusters:

$$w = \sum_{j=1}^{n} f(t - T_{i_j}) \text{ where } f(t) = 2^{-\lambda t} \text{ and } w \geq \mu$$
$$c = \sum_{j=1}^{n} f(t - T_{i_j}) p_{i_j} / w$$
$$r = \sum_{j=1}^{n} f(t - T_{i_j}) dist(p_{i_j}, c) / w \text{ where } r \leq \epsilon$$

Potential core-micro-clusters:

$$w = \sum_{j=1}^{n} f(t - T_{i_j}) \text{ where } f(t) = 2^{-\lambda t} \text{ and } w \geq \beta\mu$$
$$\overline{CF^1} = \sum_{j=1}^{n} f(t - T_{i_j}) p_{i_j}$$
$$\overline{CF^2} = \sum_{j=1}^{n} f(t - T_{i_j}) p_{i_j}^2 \text{ where } r \leq \epsilon$$

Outlier micro-clusters: $w < \beta\mu$

DenStream: Online Phase

For each new point that arrives

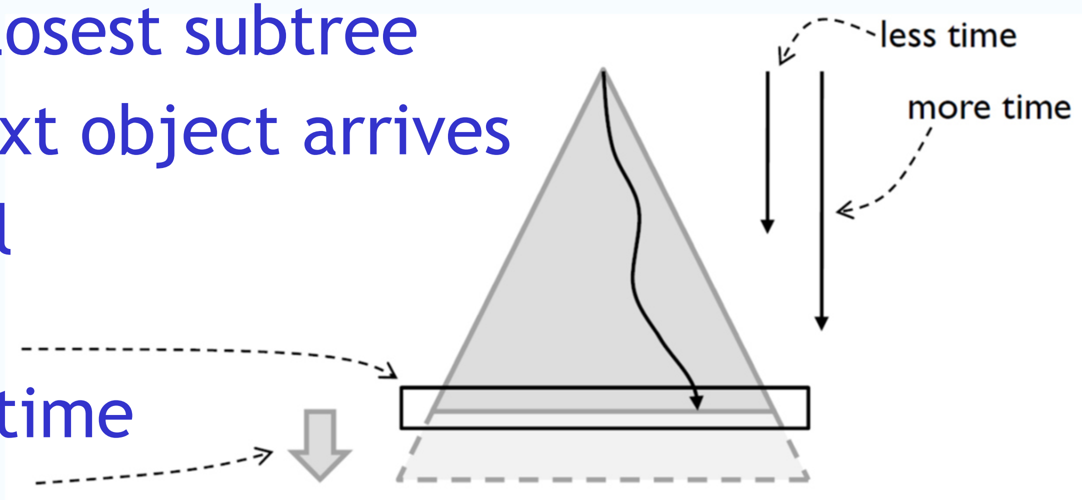
- try to merge to a p-micro-cluster
- else, try to merge to nearest o-micro-cluster
  - if $w > \beta\mu$
    then convert the o-micro-cluster to p-micro-cluster
- otherwise create a new o-microcluster

DenStream: Offline Phase

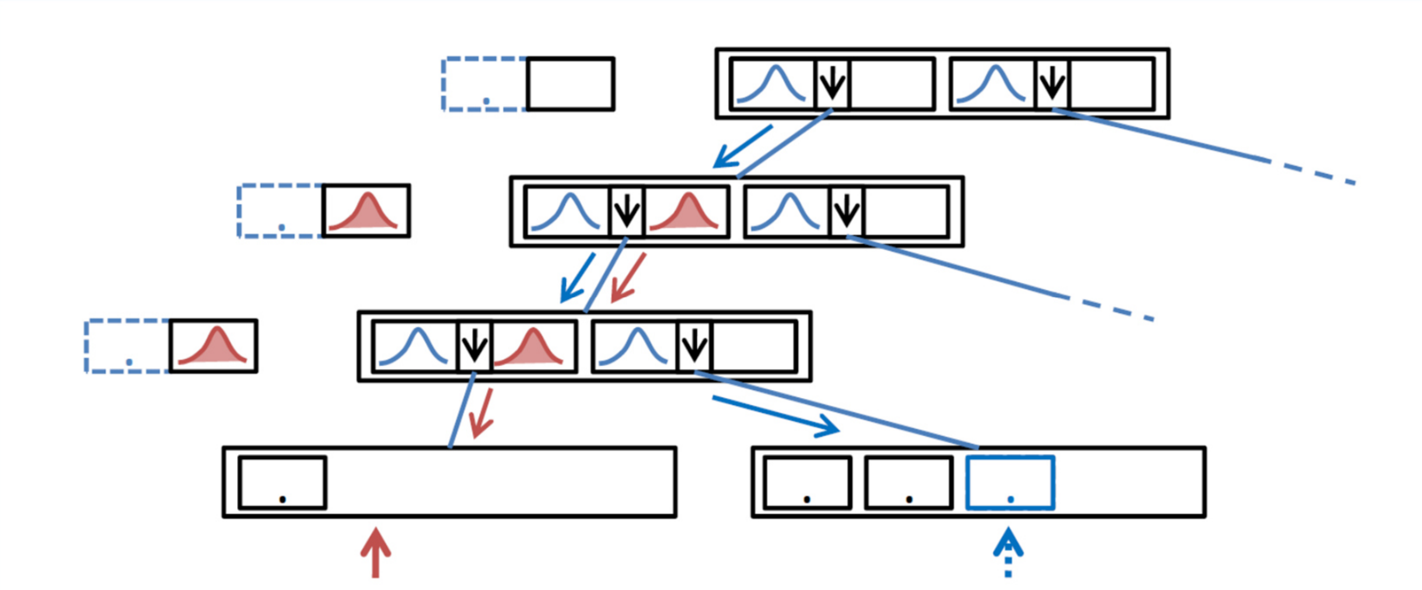
- for each p-micro-cluster c_p
  - if $w < \beta\mu$
    then remove c_p
- for each o-micro-cluster c_o
  - if $w < (2^{-\lambda(t-t_o+T_p)} - 1)/(2^{-\lambda T_p} - 1)$
    then remove c_o
- Apply DBScan using microclusters as points
- t_O creation time of micro cluster, t is current time; o-micro clusters checked every T_p time periods

ClusTree: *Anytime* Approach

- Cluster features CF = (N, LS, SS) represent micro-clusters
- Maintain a balanced tree
- Insert new object closest subtree
- Insertion stops if next object arrives
- Most detailed model stored at leaf level
- Tree grows if more time available
- For very fast streams



ClusTree



- **Buffer:** store objects on interrupt, if new object arrives at the root
- **Hitchhiker:** resume insertion, take buffer along, if they take the same way (max. two objects descend together)
- Tree grows by splitting nodes starting from the leaf

Problem 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:

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

- 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++

- k-Means with a different initialization
- Basis for the stream mining variant of k-Means
- Proof that k-Means++ approximates the best possible partitioning clustering by some factor

- Choose an initial center c_1
- for k = 2, …, K
  - select $c_k = p \in I$ with probability $d^2(p, C)/\text{cost}(C, I)$

StreamKM++: Def. Coreset for k-Means Clustering Problem

A weighted set S is a (k, ε) coreset for a data set D if for each $C \subset \mathbf{R}^d$, $|C| = k$, S approximates the partitioning of D using C with an error margin of ε:

$$(1-\varepsilon)\mathrm{cost}(D,C) \leq \mathrm{cost}_w(S,C) \leq (1+\varepsilon)\mathrm{cost}(D,C)$$

cost_w calculates the cost taking into account the weights.

StreamKM++

Coreset Tree Construction

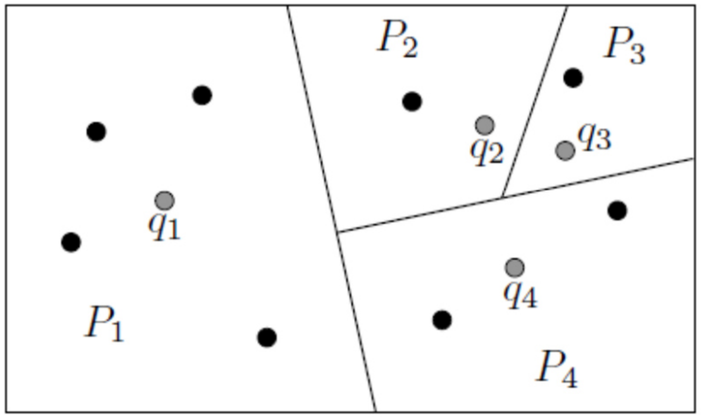
- Choose a leaf l node at random
- Choose a new sample point denoted by q_{t+1} from P_l according to d^2
- Based on q_l and q_{t+1}, split P_l into two subclusters and create two child nodes

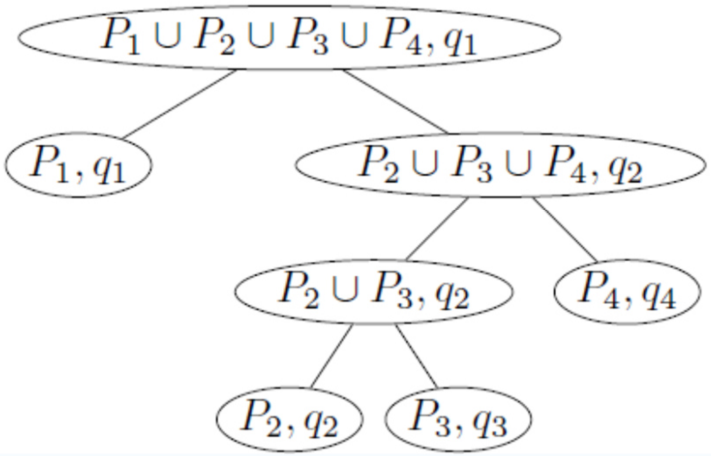
StreamKM++

- Maintain $L = \lceil \log_2(n/m) + 2 \rceil$ buckets B_1, B_2, …, B_L

StreamKM++ (Coresets)

- Maintain data in buckets $B_1, B_2 \dots B_L$. Buckets B_2 to B_L contain either exactly contains 0 or m points. B_1 can have any number of points between 0 to m points.
- Merge data in buckets using coreset tree.

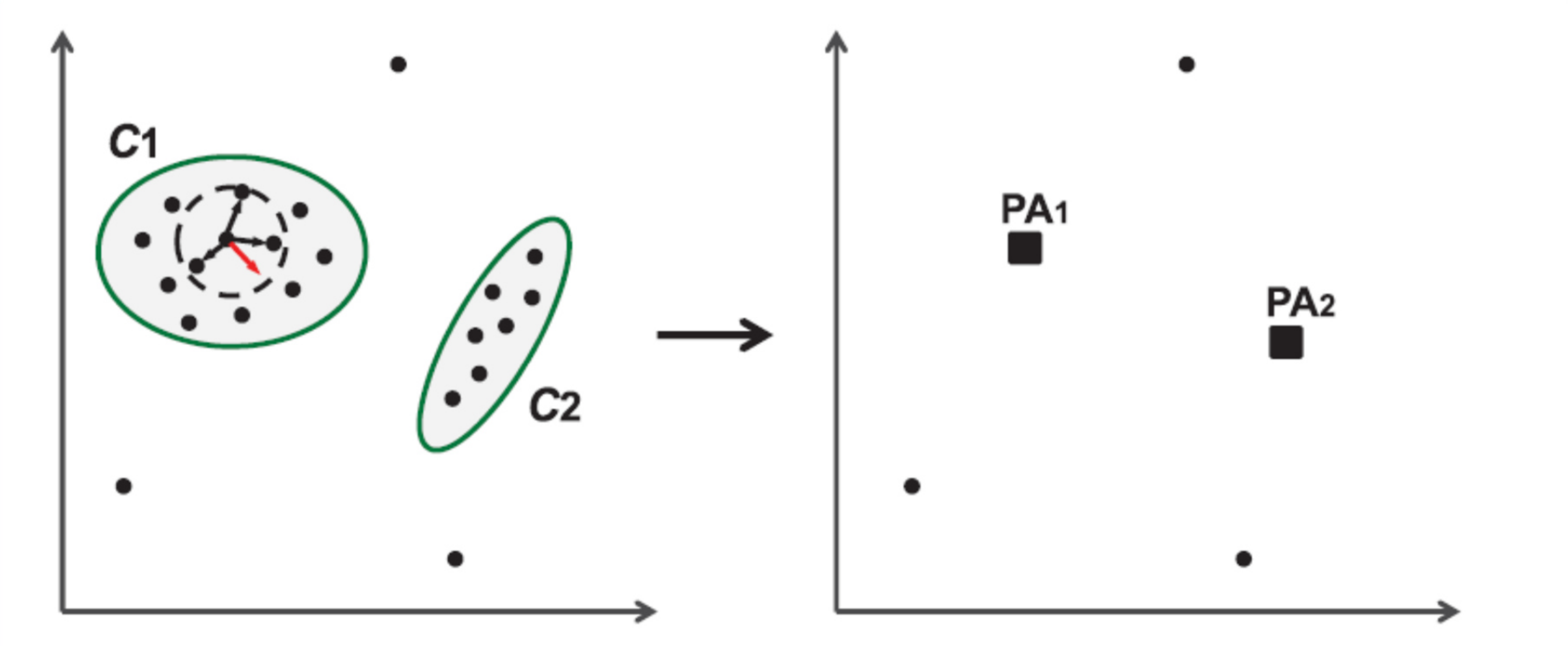




StreamKM++: A Clustering Algorithm for Data Streams, Ackermann, *Journal of Experimental Algorithmics* 2012

CIPA: Synchronization-Based Clustering

- Summarizing data by prototypes / point attractor representations



(a) Synchronization-based Clustering (b) Point Attractor Representation

CIPA: Hierarchical / Iterative Synchronization Based Clustering

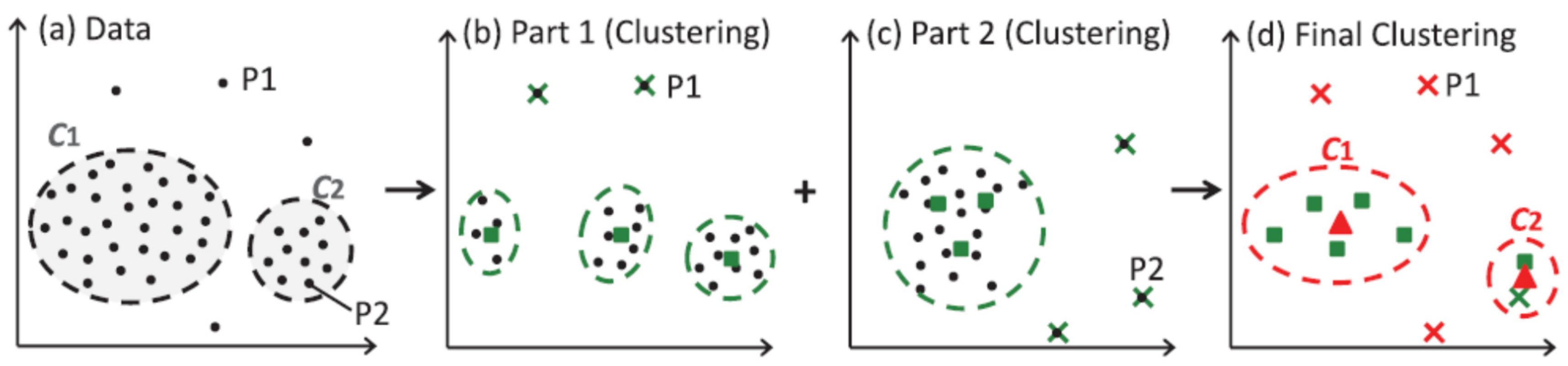


Fig. 3. Data partitioning for synchronization-based clustering.