Generation 5 Hybrid Clustering System and its Applications

Xianping Liu

Generation 5

$M\alpha\tau h\epsilon m\alpha\tau\iota c\alpha\xi\ T\epsilon ch\pi o\xi og\iota\epsilon s$

Toronto, On M2J 1M5

xianping@generation5.net

October 28, 2004

Outline

1. Motivation for G5 MWM Suite

2. Objective of G5Clustering

3. Design

   - Core Algorithms

   - Automation

   - Validity Indexes

   - Data Structures

4. Experiments and Applications

5. Conclusions

Motivation for G5 MWM Suite

With so many data mining systems already developed, the question arises: **Why Develop Yet Another System?**

- **User accessibility:** Most systems are inaccessible to the users without mathematical/statistical backgrounds.

- **Business requirements:** Gen5 itself is the first user. We know everything required by business.

- G5 MWM Suite: fully automated analytics for **business-oriented users**.
  - VIVa (feature selection)
  - G5MWM (prediction/classification)

– Clustering

– ......

– http://www.generation5.net/mwm/

M – Dr. Milorad Krneta
W – Dr. Wenxue Huang
M – Dr. Michael Vainder

Objective of G5Clustering

Business Objective: clustering large database with millions of records and thousands of variables in reasonable processing time without user's interaction. *For example*,

Postal code level customer segmentation:

- Canada: 700K six-digit postal codes (10–15 hhlds)
- USA: 35M zip+4 (3–5 hhlds)
- Generation 5 variables: approximately 10000 for every PC & zip+4

Challenges:

- handle large dataset with any types of data
- determine the optimal number of clusters automatically
- useful results for marketing applications

Design: Core Algorithms

<u>**Which core algorithms ?**</u>

- **Center-based partitional: k-means and its extensions**
  - *time:* O(n)
  - *space:* O(n)
  - *centroids:*
    - means (L_2 distance)
    - modes (Hamming distance)
    - prototypes: means+modes ((L_2 distance) + weight * (Hamming distance))
  - *advantage:*

* most efficient algorithms
  $\Rightarrow$ affordable to run multiple times
  $\Rightarrow$ reasonable approximation to the global minimum

* may use cheap validity indexes: center-based validity indexes

- *comments:*

  * converge to a local minimum

  * prefer sphere–like equal size clusters

  * prefer non-overlapping clusters

  * results depend on the initial guess

- **Hierarchical**
  - *time:* $O(n^2)$
  - *space:* $O(n^2)$
  - *advantage:*
    * based on proximity matrix <u>only</u>
      $\Rightarrow$ can use any similarity measures
      $\Rightarrow$ deal with any types of data directly
    * algorithm itself is independent of similarity measures
  - *comments:*
    * no global objective function is being optimized
    * merging decisions are final
    * good local merging decisions may <u>not</u> result in good global results.

* sensitive to outliers
* tendency to break large clusters.

- **Center-based partitional: k-medoid**
  - *comments:*
    * no ways to find representative points for categorical and mixed-type cases.
    * expensive

- More ...

G5Clustering choices:

Algorithm	Type of data	Core Reference
k-means	numerical	J. Hartigan AS136
k-modes	categorical	Z. Huang
k-prototypes	mixed-type	Z. Huang

k-prototypes algorithm

The **k-prototypes** algorithm is a algorithm which integrates **k-means** (working on numeric domains) and **k-modes** (working on categorical domains) algorithms by defining a combined dissimilarity measure.

k-prototypes objective function:

$$\mathbf{J} = \sum_{i=1}^{n} \sum_{l=1}^{k} u_{il}\, d(x_i, v_l),$$

where v_l are the **cluster centers** defined by prototypes and **hard membership coefficients** $u_{il} \in \{0, 1\}$ are defined as:

$$u_{il} = \begin{cases} 1, & \text{if} \quad l = \arg\min_{l} d(x_i, v_l) \\ 0, & \text{otherwise} \end{cases}$$

k-prototypes dissimilarity measure:

$$d(x_i, x_h) = d_N(x_i, x_h) + \gamma\, d_C(x_i, x_h),$$

where $d_N(x_i, x_h)$ is the dissimilar measure on the numeric attributes and $d_C(x_i, x_h)$ is the dissimilar measure on the categorical attributes. γ is a weight used to avoid favoring either type of attribute.

Theorem. (k-prototypes) *We can divide J into two parts $J = J_N + J_C$. Minimizing J is equivalent to minimizing both J_N and J_C because J_N and J_C are nonnegative and we are working on the local minima.* $\mathbf{J_N}$ *is minimized iff the cluster centers are represented by means:*

$$v_{lj}^N = \frac{1}{|C_l|} \sum_{x_i^N \in C_l} x_{ij}^N.$$

$\mathbf{J_C}$ *is minimized iff the cluster centers are represented by modes:*

$$v_{lj}^C = a_j^{(r)} \in DOM(A_j).$$

On-line k-prototypes algorithm

Off-line k-prototypes algorithm

Gen5 hybrid k-prototypes algorithm

Require: Data set and number of clusters:

- Data set X,
- Number of records N,
- Dimensions MN and MC,
- Number of clusters k.

1. Initialization Phase:

$\{C_l$ is the lth cluster$\}$

1. Select k initial prototypes
2. Allocate all objects to the nearest prototype
3. Initial partition of $X = (C_1, C_2, ..., C_k)$
4. Compute the prototypes of the clusters

2. Iteration Phase:

repeat

Reallocate the object to the nearest prototype

Update prototypes of both clusters immediately

until no object has changed after a full cycle test of the whole data set

3. Final Phase:

1. Allocate all objects to the nearest prototype
2. Final partition of $X = (C_1, C_2, ..., C_k)$
3. Compute the prototypes of the clusters

Design: Automation

<u>**Natural Clustering Algorithm**</u>

Require: Data set:

- Data set X,
- Number of records N,
- Dimensions M.

Require: Options (using defaults if you don't know how to choose):

- Number of samples: nSample,
- Number of tries: nTry,
- Sample size: SampleSize
- Range of number of clusters: MinK, MaxK,

1. Sampling Phase:

{Find a good initial cluster centers from samples}

repeat
    Produce a sample
    repeat
        Compute MaxK initial cluster centers
        repeat
            Apply core algorithm
            Compute validity index and store better cluster centers.
        until k from MinK to MaxK
    until nTry times
until nSample times

2. Final Phase:

{Whole dataset}

- Use above "best" cluster centers as initial center
- Apply core algorithm
- Compute validity index

Special Clustering Algorithm for Marketing

Require: Data set:

- Data set X,
- Number of records N,
- Dimensions M.

Require: Options (using defaults if you don't know how to choose):

- Number of samples: nSample,
- Number of tries: nTry,
- Sample size: SampleSize
- Range of number of clusters: MinK, MaxK,

1. Sampling Phase:

{Find a good initial cluster centers from samples}

repeat

Produce a sample

- Compute $f(MaxK) >> MaxK$ initial cluster centers
- Apply natural clustering algorithm
- Sort clusters according to the cluster size from the largest to the smallest
- Pick the 1st MaxK cluster centers
- Apply natural clustering algorithm from MinK to MaxK
- Compute validity index and store the best cluster centers.

until nSample times

2. Final Phase:

{Whole dataset}

- Use above "best" cluster centers as initial center
- Apply core algorithm
- Compute validity index

Design: Validity Indexes

- Modified Davies-Bouldin (DB) index:

$$V_{DB}(k) = \frac{1}{k} \sum_{l=1}^{k} R_l,$$

where

$$R_l = \max_{h(\neq l)} \frac{Comp_l + Comp_h}{Sep_{lh}}.$$

The number of clusters k^* that **minimizes** $V_{DB}(k)$ is taken as the optimal value of k.

- Ratio (Modified Bezdek Pal) indexes:

$$V_{bezdek}(k) = \frac{\text{Measure of Compactness}}{\text{Measure of Separation}},$$

The number of clusters k^* that **minimizes** $V_{bezdek}(k)$ is taken as the optimal value of k. V_{bezdek} is not defined for $k = 1$ and $k = n$.

Measures of Compactness: $comp(C_l)$

- **Variance:**

- **Radius:**

- **Diameter:**

- **Pairwise average:**

There are two schemes to use the measure of compactness: *maximum* and *average*.

- **maximum compactness**: maximum compactness of clusters.

$$Comp_1 = \max_l comp(C_l).$$

- **average compactness**: average compactness of clusters.

$$Comp_2 = \frac{1}{k}\sum_{l=1}^{k} comp(C_l).$$

- **average compactness**: total variation divided by the number of records.

$$Comp_3 = \frac{1}{n}\sum_{l=1}^{k}\sum_{x\in C_l} dist^2(x, v_{C_l}).$$

Measures of Separation: $Sep(C_l, C_h)$

- **Centroid:**

- **Average-centroid:**

- **Average linkage:**

- **Single linkage:**

- **Complete linkage:**

- **Hausdorff metric:**

There are two schemes to use the measure of separation: *minimum* and *average*.

- **Minimum separation**:

$$Sep_1 = \min_{l \neq h} D(C_l, C_h).$$

- **Average separation**:

$$\begin{aligned} Sep_2 &= \frac{1}{k(k-1)} \sum_{l=1}^{k} \sum_{h \neq i} D(C_l, C_h) \\ &= \frac{2}{k(k-1)} \sum_{l=1}^{k-1} \sum_{h > i}^{k} D(C_l, C_h). \end{aligned}$$

- **Average minimum separation**:

$$Sep_3 = \frac{1}{k} \sum_{l=1}^{k} \min_{h \neq l} D(C_l, C_h).$$

Data Structures

- STL (Standard Template Library) $vector < vector >$.

- VLA (Very Large Array) Arrays on virtual memory.

Data capacity: 2 TB limited by hardware.

Experiments and Applications

Artificial Datasets with 8 clusters:

- A – well-separated
- B – Separated
- C – 5% overlapped

Two-dimensional datasets:

Data	Rec#	Var#	CPU	C#	VI
Data2A	1171	2	00:00:04	8	0.23
Data2B	1171	2	00:00:08	8	0.27
Data2C	1171	2	00:00:09	8	0.30

Twenty-dimensional datasets:

Data	Rec#	Var#	CPU	C#	VI
Data20A	11071	20	00:00:07	8	0.14
Data20B	12671	20	00:00:18	8	0.34
Data20C	12671	20	00:00:09	8	0.41

Real world datasets:

Data	Rec#	Var#	CPU	C#	VI
X30	30	2	00:00:01	3	0.01
IRIS	150	4	00:00:01	2	0.24
Soybean	47	21	00:00:03	4	0.62
tr23372	23372	122	00:17:54	3	0.98
Proj1	9727	26	00:00:39	5	0.96
Proj2	14731	255	00:06:18	2	0.85
Proj3	99761	17	00:11:34	11	1.05
Proj4	30000	12	00:08:34	10	0.28
Proj5	502136	803	46:42:06	4	0.34

Conclusions

An fully automatic clustering system was developed for business-oriented users:

- Handle any types of data.
- Size of dataset is limited by hardware.
- Support multiple database formats.
- Produce the results without user's interaction.

Final remarks:

- Missing values: handled by G5 MWM: Fill missing values.

- Dimension reduction: handled by G5 MWM: Redundancy.