A batch self-organizing maps algorithm for interval-valued data

Francisco de A. T. de Carvalho[1]

Centro de Informatica-CIn/UFPE
Av. Prof. Luiz Freire, s/n -Cidade Universitaria, CEP 50740-540, Recife-PE, Brasil,
fatc@cin.ufpe.br

Table of Contents

Kohonen Self-Organising Maps (SOM)

- SOM is an unsupervised neural network method which has both clustering and visualization properties
- It maps a high dimensional data space to a lower dimension (generally 2) which is called a map
- the input data is partitioned into “similar” clusters while preserving their topology
- k-means and related algorithm operates as a SOM without topology preservation and without easy visualization
- Our general aim: to have SOM algorithms able to manage (interval-valued, histogram-valued, etc) symbolic data

Related works

- Bock (2003): stochastic version of SOM for interval-valued data
- Badran et al (2005): batch version of SOM for real-valued data
- This presentation: batch version of SOM for interval-valued data with automatic weighting of the variables
- J. A. Kangas et al (1990), N. Grozavu et al (2009): stochastic versions of SOM for real-valued data with automatic weighting of the variables
- L. D. S. Pacifico and F. A. T. de Carvalho (2011): batch version of SOM for real-valued data with automatic weighting of the variables
- Adaptive distances: Diday and Govaert (1977):

The data

- Let $E = \{e_1, \ldots, e_n\}$ the set of individuals
- Each individual is described by a vector of intervals: $\mathbf{x}_i = (x_{i1}, \ldots, x_{ip}); x_{ij} = [a_{ij}, b_{ij}] \in \Im = \{[a, b] : a, b \in \Re \text{ and } a \leq b\}$ $(i = 1, \ldots, n; j = 1, \ldots, p)$
- Each neuron (cluster) is represented by a prototype described by a vector of intervals: $\mathbf{w}_r = (w_{r1}, \ldots, w_{rp}); w_{rj} = [\alpha_{rj}, \beta_{rj}] \in \Im$ $(r = 1, \ldots, m; j = 1, \ldots, p))$

Adequacy criterion - I

- All the individuals belonging to E are simultaneously presented to the self-organizing map
- The algorithm alternates (iteratively) three steps: representation, weighting and affectation
- Adequacy criterion:

$$J = \sum_{i=1}^{n} \sum_{r=1}^{m} K^T \left(\delta(f(\mathbf{x}_i), r)\right) d^2_{\boldsymbol{\lambda}_r}(\mathbf{x}_i, \mathbf{w}_r)$$

Adequacy criterion - II

- $d^2_{\boldsymbol{\lambda}_r}(\mathbf{x}_i, \mathbf{w}_r) = \sum_{j=1}^{p} \lambda_{rj}[(a_{ij} - \alpha_{rj})^2 + (b_{ij} - \beta_{rj})^2]$ is the square of an adaptive Euclidean distance between vectors of intervals parameterized by a vector of weights $\boldsymbol{\lambda}_r = (\lambda_{r1}, \ldots, \lambda_{rp})$ on the variables;
- the vectors of weights $\boldsymbol{\lambda}_r$ $(r = 1, \ldots, m)$ change at each iteration and are different from one neuron to another;
- the matriz of weights is composed by m vectors of weights $\boldsymbol{\Lambda} = (\boldsymbol{\lambda}_1, \ldots, \boldsymbol{\lambda}_m)$

Adequacy criterion - III

- f is the identification function defined from E on $\{1, , \ldots, m\}$. This function gives the affectation of an individual to a cluster.
- $\delta(k, l)$ is the topological proximity between the clusters C_k and C_l on the grid
- $T = T_{max}(\frac{T_{min}}{T_{max}})^{\frac{t}{N_{iter}}}$ is a (decreasing) function of the number of iterations t already realised
- K^T is the neighborhood function of the self-organizing map; K^T is a function of the topological proximity δ as well as a function of the number T

Adequacy criterion - IV

The function

$$d_{(T,\mathbf{\Lambda})}(\mathbf{x}_i, \mathbf{w}_{f(\mathbf{x}_i)}) = \sum_{r=1}^{m} K^T\left(\delta(f(\mathbf{x}_i), r)\right) d^2_{\boldsymbol{\lambda}_r}(\mathbf{x}_i, \mathbf{w}_r)$$

is a weighting sum of distances between the individual $\mathbf{x}_i$ and the set of prototypes $\mathbf{w}_r$ $(r = 1, \ldots, m)$.

Algorithm - I

- From an initial solution, the algorithm is repetead a fixed number of iterations;
- For each iteration, T being fixed, the adequacy criterion J is minimized in three steps: representation, weighting and affectation

Algorithm - II

1) *Initialization :*

- Fix m (the number of neurons or clusters), the topological distance δ, the neighborhood function K^T with T_{min} and T_{max} and the maximum number of iterations N_{iter};
- Set $t \leftarrow 0$ and compute T;
- Randomly select m distincts prototypes $\mathbf{w}_r^{(0)} \in E \, (r = 1, \ldots, m)$;
- Set the map $L(m, \mathbf{W}^{(0)})$, where $\mathbf{W}^{(0)} = (\mathbf{w}_1^{(0)}, \ldots, \mathbf{w}_m^{(0)})$;
- Set $\boldsymbol{\Lambda}^{(0)} = (\boldsymbol{\lambda}_1{}^{(0)}, \ldots, \boldsymbol{\lambda}_m{}^{(0)})$ with $\boldsymbol{\lambda}_r{}^{(0)} = (1, \ldots, 1) \, (r = 1, \ldots, m)$;
- Affect each individual $\mathbf{x}_i$ to the nearest neuron (cluster) according to

$$r = f^{(0)}(\mathbf{x}_i) = arg \min_{1 \leq h \leq m} d_{(T, \boldsymbol{\Lambda}^{(0)})}(\mathbf{x}_i, \mathbf{w}_h^{(0)})$$

Algorithm - III

2) *Step 1: Representation*

- the function f and the matriz $\mathbf{\Lambda}$ are kept fixed;
- the adequacy criterion J is minimized on the prototypes
- set $t \leftarrow t + 1$ and compute $T = T_{max}(\frac{T_{min}}{T_{max}})^{\frac{t}{N_{iter}-1}}$;
- the components $w_{rj}^{(t)} = [\alpha_{rj}^{(t)}, \beta_{rj}^{(t)}]$ $(j = 1, \ldots, p)$ of the prototype $\mathbf{w}_r^{(t)} = (w_{r1}^{(t)}, \ldots, w_{rp}^{(t)})$ $(r = 1, \ldots, m)$ are computed for each neuron by:

$$\alpha_{rj}^{(t)} = \frac{\sum_{i=1}^{n} K^T\left(\delta(f^{(t-1)}(\mathbf{x}_i), r)\right) a_{ij}}{\sum_{i=1}^{n} K^T[\delta(f^{(t-1)}(\mathbf{x}_i), r)]}$$

$$\beta_{rj}^{(t)} = \frac{\sum_{i=1}^{n} K^T\left(\delta(f^{(t-1)}(\mathbf{x}_i), r\right) b_{ij}}{\sum_{i=1}^{n} K^T[\delta(f^{(t-1)}(\mathbf{x}_i), r)]}$$

Algorithm - IV

3) *Step 2: Weighting*
   - the prototypes and the function f are kept fixed;
   - the adequacy criterion J is minimized on the vectors of weights
   - the components of the vector of weights $\boldsymbol{\lambda}_r^{(t)} = (\lambda_{r1}^{(t)}, \ldots, \lambda_{rp}^{(t)})$, are computed, under the constraints, $\lambda_{rj} > 0$ et $\prod_{j=1}^{p} \lambda_{rj} = 1$, by

$$\lambda_{rj}^{(t)} = \frac{\left\{\prod_{h=1}^{p}\left(\sum_{i=1}^{n} K^T\left(\delta(f^{(t-1)}(\mathbf{x}_i), r)\right)\left[(a_{ih} - \alpha_{rh}^{(t)})^2 + (b_{ih} - \beta_{rh}^{(t)})^2\right]\right)\right\}^{\frac{1}{p}}}{\sum_{i=1}^{n} K^T\left(\delta(f^{(t-1)}(\mathbf{x}_i), r)\right)\left[(a_{ij} - \alpha_{rj}^{(t)})^2 + (b_{ij} - \beta_{rj}^{(t)})^2\right]}$$

Algorithm - V

4) *Step 3: Affectation*
   - the prototypes and the matriz of weights are kept fixed;
   - the adequacy criterion J is minimized on the identification function f;
   - Affect each individual $\mathbf{x}_i$ ($i = 1, \ldots, n$) to the nearest neuron according to

$$r = f^{(t)}(\mathbf{x}_i) = arg \min_{1 \leq h \leq m} d_{(T, \mathbf{\Lambda}^{(t)})}(\mathbf{x}_i, \mathbf{w}_h^{(t)})$$

5) *Stopping criterion*.
   If $t = N_{iter} - 1$ then STOP; else go to 2 (Step 1 : Representation).

Experimental configuration

- Batch SOM with adaptive distances $\times$ Batch SOM without adaptive distances;
- Each algorithm is run 50 times on the data sets. The best result is choosen according to the adequacy criterion J;
- Number of iterations $N_{iter} = 30$;
- δ: Euclidean distance;
- Neighborhood function: $K^T\left(\delta(c,r)\right) = \exp\left\{-\frac{(\delta(c,r))^2}{2T^2}\right\}$;
- Evaluation metrics: overall error rate of classification (OERC), corrected Rand index and F-measure

Fish data set

- 12 species of freshwater fish
- 13 interval-valued variables: Length, Weight, Muscle, Intestine, Stomach, Gills, Liver, Kidneys, Liver/muscle, Kidneys/muscle, Gills/muscle, Intestine/muscle, Stomach/muscle
- Four a priori classes:
  - Class 1 (Carnivorous): 1-Ageneiosusbrevifili/C, 2-Cynodongibbus/C, 3-Hopliasaimara/C, 4-Potamotrygonhystrix/C
  - Class 2 (Detritivorous): 7-Dorasmicropoeus/D, 8-Platydorascostatus/D, 9-Pseudoancistrusbarbatus/D, 10-Semaprochilodusvari/D
  - Class 3 (Omnivorous): 5-Leporinusfasciatus/O 6-Leporinusfrederici/O
  - Class 4 (Herbivorous): 11-Acnodonoligacanthus/H 12-Myleusrubripinis/H

Fish data set: results

Fish data set (grid: 2×3)

T_{min} : T_{max}	OERC		Rand index		F-measure	
	No W.	W.	No W.	W.	No W.	W.
0.3 : 1.0	0.416	0.416	**0.002**	-0.033	**0.503**	0.438
0.3 : 1.5	0.583	**0.083**	-0.140	**0.500**	0.388	**0.747**
0.3 : 2.0	0.416	**0.333**	-0.120	**0.093**	0.438	**0.580**
0.3 : 3.0	0.416	**0.333**	-0.052	**0.043**	0.449	**0.504**
0.3 : 5.0	0.583	**0.333**	-0.104	**0.120**	0.435	**0.644**
0.3 : 7.0	0.333	**0.333**	0.057	**0.120**	0.566	**0.644**

Grid (fish data set)

T: 0.3 - 1.0 (No W.)			T: 0.3 - 1.5 (W.)		
X	1/2	2	4	2	1/2
1	1	3/4	1	1	3

Car data set

- 33 car models
- 8 interval-valued variables: Price, Engine Capacity, Top Speed, Acceleration, Step, Length, Width and Height
- Four a priori classes:
  - Class 1 (Utilitarian): 1-Alfa 145/U, 5-Audi A3/U, 12-Punto/U, 13-Fiesta/U,17-Lancia Y/U, 24-Nissan Micra/U, 25-Corsa/U, 28-Twingo/U, 29-Rover 25/U, 31-Skoda Fabia/U
  - Class 2 (Berlina): 2-Alfa 156/B, 6-Audi A6/B, 8-BMW serie 3/B, 14-Focus/B, 21-Mercedes Classe C/B, 26-Vectra/B, 30-Rover 75/B, 32-Skoda Octavia/B
  - Class 3 (Sporting): 4-Aston Martin/S, 11-Ferrari/S, 15-Honda NSK/S,16-Lamborghini/S, 19-Maserati GT/S, 20-Mercedes SL/S, 27-Porsche/S
  - Class 4 (Luxury): 3-Alfa 166/L, 7-Audi A8/L, 9-BMW serie 5/L, 10-BMW serie 7/L, 18-Lancia K/L, 22-Mercedes Classe E/L,23-Mercedes Classe S/L, 33-Passat/L

Car data set: results

Car data set (grid: 2×5)

T_{min} : T_{max}	OERC		Rand index		F-measure	
	No W.	W.	No W.	W.	No W.	W.
0.3 : 2.0	0.303	**0.181**	0.299	**0.318**	0.515	**0.572**
0.3 : 3.0	0.303	**0.242**	0.310	**0.510**	0.557	**0.760**
0.3 : 3.5	0.303	**0.212**	0.315	**0.583**	0.565	**0.797**
0.3 : 5.0	0.454	**0.333**	0.253	**0.392**	0.585	**0.746**
0.3 : 9.0	0.242	**0.212**	0.333	**0.615**	0.570	**0.852**
0.3 : 13.0	0.393	**0.333**	0.269	**0.392**	0.583	**0.746**

Grid (car data set)

T: 0.3 - 3.5 (No W.)					T: 0.3 - 9.0 (W.)				
4	3	3	1	2	4	X	X	X	3
4	4	X	3/4	4	1	X	X	X	2

Temperature data set - 34 cities

- 34 cities described by 12 interval-valued variables
- this data set gives the minimum and the maximum monthly temperatures of cities in degrees centigrade
- Two a priori classes:
  - Class 1 (cities mainly located between 0^0 and 40^0 latitudes): 3-Bahraim, 4-Bombay, 5-Cairo, 6-Calcutta, 7-Colombo, 9-Dubai, 12-Hong Kong,13-Kula Lampur, 16-Madras, 18-Manila, 20-Mexico, 23-New Delhi, 30-Sydney
  - Class 2 (cities mainly located between 40^0 and 60^0 latitudes): 1-Amsterdam, 2-Athens, 8-Copenhagen,10-Frankfurt ,11-Geneva ,14-Lisbon, 15-London,17-Madrid, 21-Moscow, 22-Munich, 24-New York, 25-Paris, 26-Rome, 27-San Francisco, 28-Seoul, 29-Stockholm, 32-Tokyo, 33-Toronto, 34-Vienna, 35-Zurich

Temperature data set - 34 cities (grid: 2 $\times$ 8)

T_{min} : T_{max}	OERC		Rand index		F-measure	
	No W.	W.	No W.	W.	No W.	W.
0.3 : 3.5	0.000	0.000	0.170	**0.295**	0.362	**0.504**
0.3 : 4.5	0.000	0.000	0.213	**0.558**	0.406	**0.732**
0.3 : 6.0	0.029	**0.000**	0.266	**0.487**	0.483	**0.678**
0.3 : 8.5	0.000	0.000	0.415	**0.686**	0.591	**0.798**
0.3 : 15.5	**0.000**	0.029	0.408	**0.839**	0.587	**0.891**
0.3 : 22.0	0.000	0.000	0.295	**0.825**	0.490	**0.884**

Grid (temperature data set - 34 cities)

T: 0.3 - 8.5 (No W.)								T: 0.3 - 15.5 (W.)							
2	X	1	X	X	2	X	2	X	X	X	X	X	X	X	2
2	X	1	X	X	2	X	1	1	X	X	X	X	X	X	1

Temperature data set - 492 cities

- 492 cities described by 12 interval-valued variables
- this data set gives the average minimum and the maximum monthly temperatures of cities in degrees centigrades
- There is no a priori classification:

Grid (temperature data set - 492 cities)

T: 0.3 - 7.0 (W.)							
113	X	X	X	X	X	X	100
X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X
X	X	X	X	X	X	X	X
106	X	X	X	X	X	X	173

Temperature data set - 492 cities

Grid (prototypes: January temperatures)

T: 0.3 - 7.0 (W.)							
[−11.1, −3.9]	[−8.0, −1.2]	[−8.2, −1.4]	[−5.7, 1.4]	[9.0, 18.7]	[12.4, 22.7]	[12.8, 23.0]	[21.0, 27.0]
[−6.4, 0.7]	[−6.6, 0.4]	[−7.1, −0.2]	[−5.5, 1.6]	[10.5, 20.2]	[14.0, 24.0]	[14.7, 24.6]	[14.9, 24.9]
[−5.1, 2.2]	[−5.1, 2.2]	[−5.0, 2.2]	[−2.8, 4.7]	[14.7, 28.2]	[16.2, 25.9]	16.2, 25.9]	[16.2, 25.9]
[−3.7, 3.8]	[−3.5, 4.0]	[−2.9, 4.8]	[1.4, 9.5]	[16.5, 25.8]	[17.7, 27.2]	[17.3, 26.9]	[17.2, 6.8]
[−4.0, 1.0]	[−1.9, 5.9]	[−1.6, 6.2]	[2.6, 10.9]	[16.6, 25.8]	[18.4, 27.8]	[18.3, 27.7]	[19.0, 29.0]

- Prototypes average minimum and maximum January temperatures increases from left to right and from top to down in the grid

- Cluster 1 (106): barcelona, beijing, belgrade, budapest, dubrovnik, frankfurt, geneva, lisbon, lyon, madrid, marseille, milan, naple, paris, porto, rome, shangai...
- Cluster 8 (173): baghdad, bankok, brazzaville, cairo, calcutta, cayenne, dakar, hanoi, havana, hong kong, islamabad, jakarta, karthoum, kuweit, rio de janeiro, ...
- Cluster 33 (113): berlin, brussels, copenhagen, helsink, kiev, london, moscow, oslo, prague, quebec, reykjavic, seatle, st petersburg, stockholm, totonto, viena, warsaw, ...
- Cluster 40 (100): athens, beirut, buenos aires, canberra, jerusalem, lima, los angeles, mexico city, montevideo, palermo, porto alegre, pretoria, santiago, sydney, ...

Concluding Remarks

Contributions

- extension of the batch SOM algorithm to interval-valued data
- automatic weighting of the interval-valued variables based on adaptive distances

Evaluation and example

- adaptive batch SOM $\times$ non adaptive batch SOM
- four interval-valued data sets: fish, car, city temperatures
- evaluation metrics: overall error rate of classification, corrected Rand index, F-measure
- conclusion: adaptive batch SOM outperforms non adaptive batch SOM in the majority of the cases

Work in progress

- Batch SOM with city-block and Hasudorff distances to manage interval-valued data (modelling already finished; program implementations in progress)
- Batch SOM to manage histogram-valued data (modelling already finished; program implementations in progress)
- Different automatic weightings of the interval-valued variables (sum, product)

References

1. H. H. Bock and E. Diday, Analysis of Symbolic Data, Springer-Verlag, Heidelberg, 2000
2. E. Diday and M. Noirhome, Symbolic Data Analysis and the SODAS Software, Wiley, 2008
3. T.K. Kohonen, Self-Organizing Maps. Berlin, Springer, 2001
4. H.-H. Bock, Clustering algorithms and kohonen maps for symbolic data, Journal of the Japanese Society of Computational Statistics 15, 217–229, 2003
5. F. Badran, M. Yacoub, et S. Thiria, Self-organizing maps and unsupervised classification. In G. Dreyfus (Ed.), Neural Networks. Methodology and Applications, pp. 379–442, Springer, 2005
6. J. A. Kangas, T. K. Kohonen and J. T. Laaksonen, Variants of self organizing maps, IEEE Transactions on Neural Networks 1, 93–99, 1990
7. N. Grozavu, Y. Bennani and M. Lebbah, From Variable Weighting to Cluster Characterization in Topographic Unsupervised Learning, Proceedings of the International Joint Conference on Neural Networks (IJCNN'09), 1005–1010, 2009
8. E. Diday et G. Govaert, Classification automatique avec distances adaptatives, R.A.I.R.O. Informatique Computer Science 11, 329–349, 1977
9. L. D. S. Pacifico and F. A. T. de Carvalho, A batch self-organizing maps algorithm based on adaptive distances, Proceedings of IJCNN 2011.
10. F. A. T. de Carvalho and L. D. S. Pacifico, Une version batch de l'algorithme SOM pour des données de type intervalle, Comptes Rendus des Rencontres de la SFC 2011

Let $P = \{P_1, \ldots, P_i, \ldots, P_m\}$ be the *a priori* partition into m classes and $Q = \{Q_1, \ldots, Q_j, \ldots, Q_K\}$ be the hard partition into K clusters given by a clustering algorithm.

Table: Confusion matrix

<table>
<tr><th rowspan="2">Classes</th><th colspan="6">Clusters</th></tr>
<tr><th>Q_1</th><th>$\ldots$</th><th>Q_j</th><th>$\ldots$</th><th>Q_K</th><th>$\sum$</th></tr>
<tr><td>P_1</td><td>n_{11}</td><td>$\ldots$</td><td>n_{1j}</td><td>$\ldots$</td><td>n_{1K}</td><td>$n_{1\bullet} = \sum_{j=1}^{K} n_{1j}$</td></tr>
<tr><td>$\vdots$</td><td>$\vdots$</td><td>$\ldots$</td><td>$\vdots$</td><td>$\ldots$</td><td>$\vdots$</td><td>$\vdots$</td></tr>
<tr><td>P_i</td><td>n_{i1}</td><td>$\ldots$</td><td>n_{ij}</td><td>$\ldots$</td><td>n_{iK}</td><td>$n_{i\bullet} = \sum_{j=1}^{K} n_{ij}$</td></tr>
<tr><td>$\vdots$</td><td>$\vdots$</td><td>$\ldots$</td><td>$\vdots$</td><td>$\ldots$</td><td>$\vdots$</td><td></td></tr>
<tr><td>P_m</td><td>n_{m1}</td><td>$\ldots$</td><td>n_{mj}</td><td>$\ldots$</td><td>n_{mK}</td><td>$n_{m\bullet} = \sum_{j=1}^{K} n_{mj}$</td></tr>
<tr><td>$\sum$</td><td>$n_{\bullet 1} = \sum_{i=1}^{m} n_{i1}$</td><td>$\ldots$</td><td>$n_{\bullet j} = \sum_{i=1}^{m} n_{ij}$</td><td>$\ldots$</td><td>$n_{\bullet K} = \sum_{i=1}^{m} n_{iK}$</td><td>$n = \sum_{i=1}^{m} \sum_{j=1}^{K} n_{ij}$</td></tr>
</table>

The corrected Rand index is:

$$CR = \frac{\sum_{i=1}^{m}\sum_{j=1}^{K}\binom{n_{ij}}{2} - \binom{n}{2}^{-1}\sum_{i=1}^{m}\binom{n_{i\bullet}}{2}\sum_{j=1}^{K}\binom{n_{\bullet j}}{2}}{\frac{1}{2}[\sum_{i=1}^{m}\binom{n_{i\bullet}}{2} + \sum_{j=1}^{K}\binom{n_{\bullet j}}{2}] - \binom{n}{2}^{-1}\sum_{i=1}^{m}\binom{n_{i\bullet}}{2}\sum_{j=1}^{K}\binom{n_{\bullet j}}{2}} \quad (1)$$

where $\binom{n}{2} = \frac{n(n-1)}{2}$ and n_{ij} represents the number of objects that are in class P_i and cluster Q_j; $n_{i\bullet}$ indicates the number of objects in class P_i; $n_{\bullet j}$ indicates the number of objects in cluster Q_j; and n is the total number of objects in the data set.

The traditional $F - measure$ between class $P_i \, (i = 1, \ldots, m)$ and cluster $Q_j \, (j = 1, \ldots, K)$ is the harmonic mean of precision and recall:

$$F - measure(P_i, Q_j) = 2 \, \frac{Precision(P_i, Q_j) \times Recall(P_i, Q_j)}{Precision(P_i, Q_j) + Recall(P_i, Q_j)} \quad (2)$$

The *Precision* between class $P_i \, (i = 1, \ldots, m)$ and cluster $Q_j \, (j = 1, \ldots, K)$ is defined as the ratio between the number of objects that are in class P_i and cluster Q_j and the number of objects in cluster Q_j:

$$Precision(P_i, Q_j) = \frac{n_{ij}}{n_{\bullet j}} = \frac{n_{ij}}{\sum_{i=1}^{m} n_{ij}} \quad (3)$$

The *Recall* between class $P_i \, (i = 1, \ldots, m)$ and cluster $Q_j \, (j = 1, \ldots, K)$ is defined as the ratio between the number of objects that are in class P_i and cluster Q_j and the number of objects in class P_i:

$$Recall(P_i, Q_j) = \frac{n_{ij}}{n_{i\bullet}} = \frac{n_{ij}}{\sum_{j=1}^{K} n_{ij}} \tag{4}$$

The $F-measure$ between the *a priori* partition $P = \{P_1, \ldots, P_i, \ldots, P_m\}$ and the hard partition $Q = \{Q_1, \ldots, Q_j, \ldots, Q_K\}$ given by a cluster algorithm is defined as:

$$F-measure(P,Q) = \frac{1}{n} \sum_{i=1}^{m} n_{i\bullet} \ \max{}_{1 \le j \le K} \ F-measure(P_i, Q_j) \tag{5}$$