Enabling End-User Datawarehouse Mining
Contract No. IST-1999-11993
Deliverable No. D4.1

# Informed Parameter Setting

Lorenza Saitta[1], Giuseppe Beccari[2], and Alessandro Serra[1]

[1] University of Piemonte Orientale
Dipartimento di Scienze e Tecnologie Avanzate
C.so Borsalino 54, 15100 Alessandria, Italy
{saitta,serra}@mfn.unipmn.it
http://www.mfn.unipmn.it

[2] Cselt - Telecom Italia Group
Via Reiss Romoli 274, 10143 Torino, Italy
Giuseppe.Beccari@cselt.it
http://www.cselt.it

December 22, 2000

**Abstract**

When setting an algorithm/system to work, several parameters, both categorical and numerical, are usually to be defined. Often, a preliminary series of test runs is performed in order to find some sub-optimal parameter setting. This procedure may be time consuming. On the other hand, in most algorithms suitable to complex real-world applications, the relation between a parameter's value and the output cannot be easily specified, not even qualitatively, because the interrelation between different parameters may mask the effect of each one. On the other hand, the user, even though not knowing in advance the results of the mining, may nevertheless be able to explicitly specify some set of constraints on the desired output.

Deliverable D4.1 furnishes the abstract description of the methodology trying to capture those aspects that do not depend upon a specific data mining task. An example of this task-independent part is the description of the loop into which the user and the algorithm are inserted, with the specification of the types of information that they should exchange.

# Chapter 1

# Introduction

When setting an algorithm/system to work, several categorical and/or numerical parameters are usually to be defined. Often, a preliminary series of test runs is performed in order to find some sub-optimal parameter setting. This procedure may be boring and very time consuming, especially when the faced task requires the use of algorithms with a high computational complexity. On the other hand, in most algorithms suitable to complex real-world applications, the relation between a parameter's value and the output cannot be easily specified, not even qualitatively, because the interrelation between different parameters may mask each other's effects. On the other hand, the user, even though not knowing in advance the results of mining, may nevertheless be able to explicitly specify some set of constraints on the desired output. For instance, in a segmentation task, he/she may want that customers with given characteristics mostly belong to the same group, and so on. The satisfaction of the constraints defined by the user can be codified into a function, which reflects the degree to which the constraints have been satisfied by the algorithm. This function can be automatically optimized with respect to the algorithm's parameters. In order to introduce this step, constraint definition and algorithm's run must be inserted into a closed loop, which the user is an integral part of. Two fundamental aspects differentiate this approach from the trial-and-error one: on the one hand, the loop is executed only one time, and, on the other, the parameter setting is determined automatically.

In general, the satisfaction of the user's constraints are codified into a function that reflects the degree to which the constraints are satisfied by the algorithm. This function can be automatically optimized with respect to the algorithm's parameters. In order to introduce this step, constraint definition and algorithm's run must be inserted into a closed loop, which the user is an integral part of. Two fundamental aspects differentiate this approach from the trial-and-error one: on the one hand, the loop is executed only one time, and, on the other, the parameter setting is determined automatically.

Previous experience in using this approach in market segmentation tasks gave good results in terms of both speed up and user agreement on the quality of the results.

The problem of parameter setting in Machine Learning is a very important one, because different parameter values may drastically change the output of a learning algorithm. Some efforts have been done in the past (see, for example, [Kohavi and John, 1995][1]) and also commercial products, such as, for instance, the MLC++, contains methods for automatic setting of C4.5s parameters. However, the problem has mostly been left, up to now, to the intuition of the machine learning developer or to a time consuming trial-and-error procedure.

The idea underlying our approach is to allow the user to express his/her preference for solutions and to codify these preferences so as to link them experim entally to the algorithms parameter values through an optimization process.

[1] Ron Kohavi and George John. "Automatic Parameter Selection by Minimizing Estimated Error". In *Proceeding of the Twelfth International Conference on Machine Learning* (Lake Tahoe,CA), pp.304-312.

# Chapter 2

# Methodology

Given an algorithm **A**, designed to solve a specific data mining task, let $X$ be its input (typically a dataset), $Y$ be its output (for instance, a set of classification rules, a partition of the data, a set of association rules, and so on), and $P = \{p_1, \ldots, p_s\}$ be a set of $s$ numerical parameters:

$$Y = A(X, P) \tag{2.1}$$

Each parameter $p_j$ in $P$ has usually a default value:

$$P(0) = \{p_1(0), \ldots, p_s(0)\} \tag{2.2}$$

The default values are employed when the user does not have a better suggestion. Even though the user does not know the results $Y$, he/she may nevertheless know that the solution $Y$ should (preferably) satisfy some constraints. These constraints, possibly expressed in an unstructured language, are collected in a set $Q = \{q_1, \ldots, q_T\}$.

The first step consists in codifying each constraint into a numerical function, which expresses, in an interval [0, 1] the degree to which a given output satisfies the constraint. The functions associated to every constraints are summed up together, to build up a global objective function to be maximized. This soft approach is necessary because it may often happen that the constraints expressed by the user contradict each other, so that a compromise has to be sought.

A way of satisfying the constraints is to modify the parameters present in the algorithm. If the analytical link between parameters and output is known, the problem is simpler, because it is sufficient to determine the region of the parameter space where the output is the desired one. But this situation seldom occurs. In general, the effects of the parameters is much more far-reaching. However, the output, considered as an implicit function of the parameters, as in (2.1), may be experimentally modified by letting the learning algorithm run. The difference between this approach and the trial-and-error one consists in the fact that if the maximum (or a setting

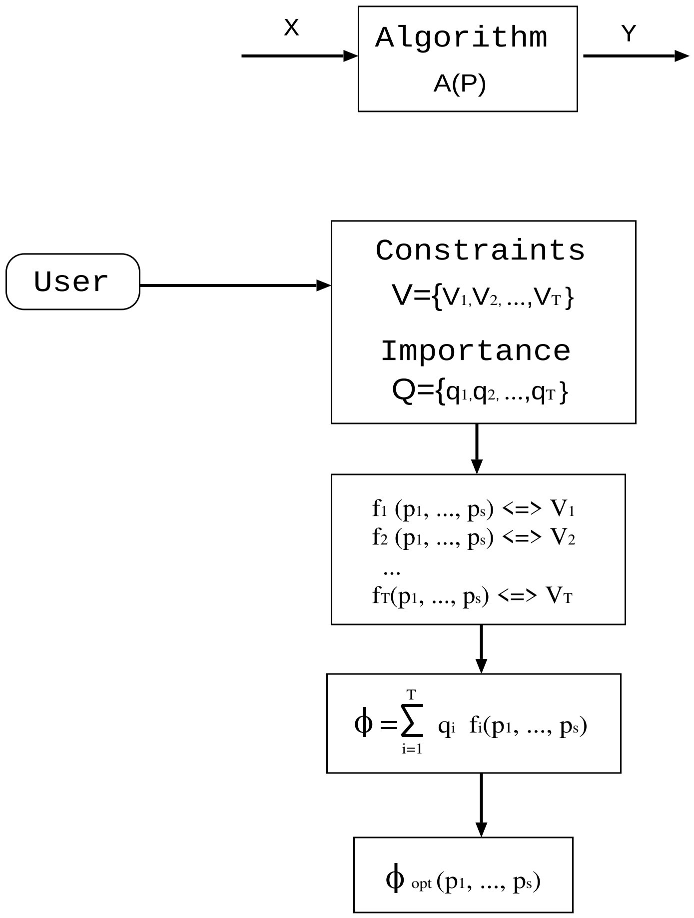


Figure 2.1:

close to the maximum) of the objective function is found, then is no need to re-try, because we cannot expect to do better.

In Figure 2.1, a graphical representation of the steps to perform is reported. In the next section, an exemplification of the method shall help to clarify the method.

# Chapter 3

# Parameter Tuning in a Segmentation Algorithm

In this Section we describe, for the sake of exemplification, how the methodology outlined in Section 3 has been applied to a segmentation algorithm, SEGMENT, already available at CSELT.

SEGMENT is an not hierarchical clustering algorithm based on the concept of density, that produces separated groups and operates over data described by couple $[Attribute, Value]$. This approach requires introducing the *notion of distance* between points and eventually between clusters. In general, this *distance* is a function that both increases with the inter-cluster distance and diminishes with the intra-cluster distance. An optimal segmentation will maximize an objective function that is related to this distance notion. The attributes are usually weighed and the weights of the attributes are the main algorithm parameters of whom values have to be determined.

The attributes can be categorical, numerical or structured. Let $X$ the set of N examples to be partitioned, and $\mathcal{A} = \{A_1, \ldots, A_H\}$ the attributes set, where each value of the attribute $A_h$ belongs to $\Lambda_h$, with

$$\forall A_h \in \mathcal{A} : A_h = v_j \in \Lambda_h = \{v_1, \ldots, v_{L_h}\};\ |\Lambda_h| = L_h \tag{3.1}$$

Let $v_j^{(i)}$ the value of the attribute $A_h$ in the example $x_i$. For each attribute we introduce a distance function, that depends on the attribute type, between the attribute values. The distance for the attribute $A_h$ between two example $x_1$ and $x_2$ is denoted by $d_h(v_j^{(1)}, v_j^{(2)})$, while the global distance between two example is the average of the distances of the single attributes, namely

$$D(x_1, x_2) = \frac{1}{H} \sum_{h=1}^{H} d_h(v_j^{(1)}, v_j^{(2)}) \tag{3.2}$$

Let $w_1, \ldots, w_H$ the attributes weights, equation 3.2 is modified as in the following

$$D(x_1, x_2) = \frac{1}{W} \sum_{h=1}^{H} w_h \, d_h(v_j^{(1)}, v_j^{(2)}) \tag{3.3}$$
$$W = \sum_{h=1}^{H} w_h.$$

Equation 3.3 is the *distance* function we use in SEGMENT.

## 3.1 SEGMENT: the algorithm

In this section we briefly furnish a description of SEGMENT. The algorithm receives as an input a set $X$ of examples to be partitioned off into $K$ clusters with $K \leq K_M$, where $K_M$ roughly is twice as much as the final number of clusters we would obtain. SEGMENT partitions the set $X$ in groups throughout a straightforward dinstance criteria. An abstract description of the algorithm is shown in Figure 3.1, and its step are summarized in the following:

1. Require to the user
   - the X examples to be partitioned;
   - the couples attribute-weight to be taken into account;
   - the couples constraints-value to be taken into account;
   - the parameter $K_M$.
2. Partition $X$ and produce a collection of different clustering [1].
3. Select one clustering.
4. Analyze the selected clustering: If the clustering is adequate then *goto 6* .
5. Otherwise ask to the user to:
   - Change/Add constraints;
   - Require the execution of the Optimization module (weights' tuning);

[1] The algorithm produces a collection of different clustering, $CLUST(j)$ $(1 \leq j \leq K_M)$, containing 1 upto $K_M$ clusters. Let $M_i^{(j)} = |C_i^{(j)}|$ $(1 \leq j \leq K_M,\ 1 \leq i \leq j)$ the cardinality of the $i_{th}$ cluster into the $j_{th}$ group, as the clusters are separated, we have:

$$N = \sum_{i=1}^{j} M_i^{(j)}. \tag{3.4}$$

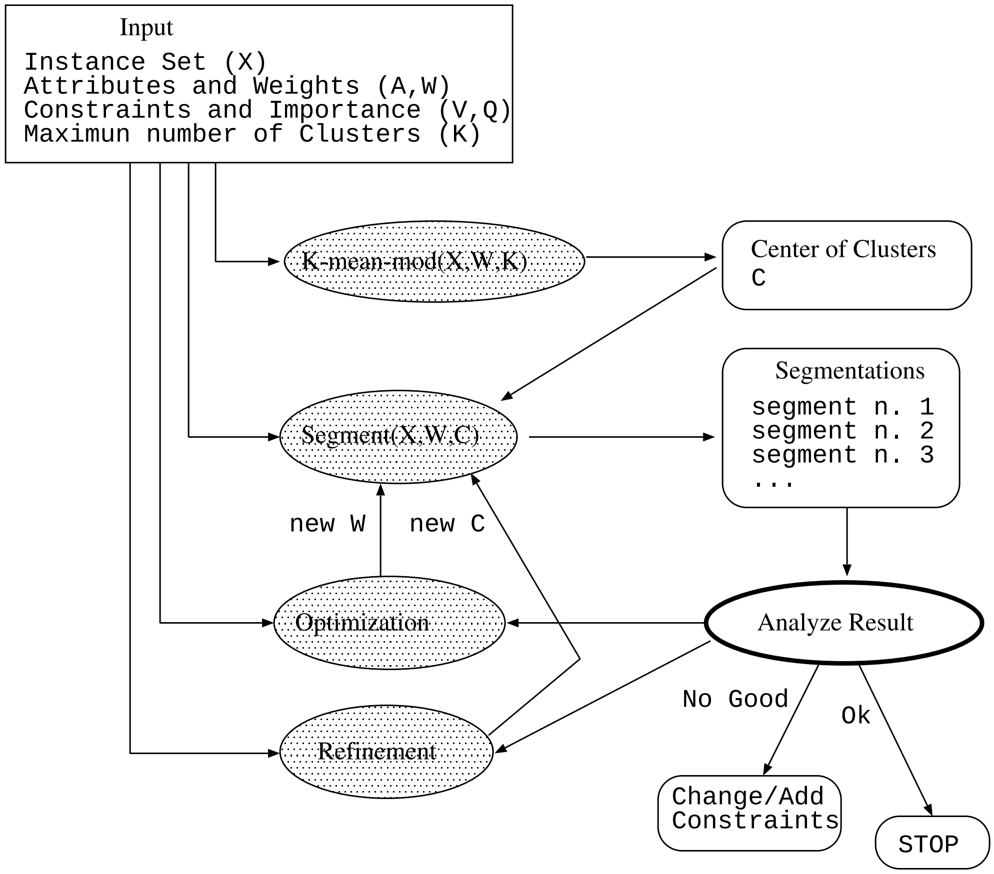


Figure 3.1: A abstract description of the SEGMENT$(X, K_M)$ algorithm, that partitions X examples off in a certain number of separated clusters.

- Require the execution of the Refinement module (clusters' centers tuning);

*goto 3.*

6. Iteration ends.

In many applications constraints are usually imprecise at the beginning of the iteration, and they are often assigned by default especially when the user cannot furnish them. For this reason, the initial loops might fail, and a modulation of instances' distances, obtained by a weight tuning (Optimization module), and/or an adjustment of the number of clusters (Refinement module) might produce the desired results.

## 3.2 Clusters' centers selection

The selection of the initial clusters' centers is critical even for algorithms that are able to change the centers' positions (e.g. K-Means [Fayyad and Bradley 1998][2]), hence this is particularly difficult in SEGMENT where you cannot modify the initial centers' postures. As they are not provided by the user, the clusters' centers are selected by an *Stochastic Windowing* algorithm (see Figure 3.1). The method has been inspired by the solution proposed by Fayyad and Bradley, that has been demonstrated to be adequately accurate and efficient also with big dataset. The main idea of the Fayyad and Bradley's algorithm can be summarize in the following steps:

1. Given a set $X$ of examples we generate $J$ subset $x_i$ $(1 \leq i \leq J)$ by a random sampling algorithm.
2. Partition any $x_i$, e.g. by the K-Means algorithm, and produce $Z_i$, namely the set of clusters' centers obtained by the partion ofter the partition.
3. Let $\mathcal{Z} = \bigcup_{i=1}^{J} Z_i$, partition $\mathcal{Z}$, e.g. by the K-Means algorithm, $J$ time and obtain $Y_i$ solutions, with $i \in [1 \ldots J]$ (where each $Y_i$ is obtained exploiting $Z_i$ as initial centers' set to be furnished to K-Means).
4. Select the $Y_i$ minimizing the distortion over $\mathcal{Z}$.

## 3.3 User's Constraints

The constraints the user furnishes, express his/her knowledge about the problem domain. They are described by syntactical expressions, as closed as possible to the natural language, that have to be translate into adequate analytical functions or automatic procedures. This section briefly describes the constraints the user exploit into the informed segmentation task.

**Type 1: Instance Association.** Instances $x_i \in X' \subseteq X$, have mostly to belong to the same cluster.

**Type 2: Value Association.** Instances with a given value, $v_j \in \Lambda_h$, for the categorical or numerical attribute $A_h$, have mostly to belong to the same cluster; more specifically, the constraint requires that instances with a given value for a certain attribute are grouped into one cluster, while other attributes could be also remarkable different.

**Type 3: Value Separation.** The value distribution of a given attribute $A_h$ mostly has to be partitioned over clusters; more specifically, the

[2]U. Fayyad and P. Bradley, *Refining Initial Points for K-Means Clustering*, Interantional Conference on Machine Learning, 1998

constraint requires that any cluster has to exhibit instances with one value for the attribute $A_h$, while many clusters might exhibit instances with that $A_h = v_j \in \Lambda_h$.

**Type 4: Number of Clusters.** Let K the number $K$ of clusters, it has to be $K_{min} \leq K \leq K_{max}$.

**Type 5: Cluster Cardinality.** The cardinality of each cluster has to be bigger then a given value $M$.

**Type 6: Cardinality Ratio.** The Max/Min cardinality ratio has to be less or equal to a given value $R$.

**Type 7: Cluster Adherence.** The values' distribution of the attribute $A_h$ mostly has to be partitioned over clusters, and instances with a given value $v_j \in \Lambda_h$ have mostly to be grouped into one cluster; actually, this constraint is the logical AND between constraints 2 and 3.

## 3.4 Optimization module

The Optimization module adjusts the instances' distances (the attributes' weights) in order to increase the number of satisfied user constraints, thus eventually producing the desired result. Actually, an attribute weight modulation allows modifying the obtained clustering in the way that constraints 1, 2, 3 and 7 might be guaranteed. This section describes how an attribute weights perturbation affect the instances' distances and how this correlation can be exploited to modify the solution SEGMENT proposes. The first paragraph describes the case of single weight tuning, while the second paragraph presents the case of multiple weight tuning.

### Single weight tuning

Let $x_1, x_2$ two instances in $X$, the distance $D(x_1, x_2)$ is defined by the following equation:

$$D = D(x_1, x_2) = \frac{1}{W} \sum_{h=1}^{H} w_h \; d_h(v_j^{(1)}, v_j^{(2)}) = \frac{1}{W} \sum_{h=1}^{H} w_h \; d_h \tag{3.5}$$

The distance $D$ can be also written as a function of $w_r$:

$$D(w_r) = D(x_1, x_2) = \frac{1}{W} \sum_{h=1}^{H} w_h \; d_h \tag{3.6}$$

Let perturbate the $w_r$ weight by a little value$\Delta w_r$, we have (using a error calculation)

$$\Delta D = \frac{\partial D(w_r)}{\partial w_r} \Delta w_r \tag{3.7}$$

with

$$\frac{\partial D(w_r)}{\partial w_r} = \frac{d_r - D(w_r)}{W} \tag{3.8}$$

Equation 3.8 shows that:

- $\frac{\partial D}{\partial w_r} = 0$ *iff* $d_r = D$, $\Delta w_r$ does not affect $\Delta D$, thus $d_r = D$ returns a stable state;
- $\frac{\partial D}{\partial w_r} > 0$ *iff* $d_r < D$, $\Delta w_r$ is in direct proportion with $\Delta D$, thus a positive $\Delta w_r$ results into a positive $\Delta D$;
- $\frac{\partial D}{\partial w_r} < 0$ *iff* $d_r > D$, $\Delta w_r$ is in inverse proportion with $\Delta D$, thus a positive $\Delta w_r$ results into a negative $\Delta D$.

Hence, a single weight perturbation affects the distance $D(x_1, x_2)$ so that:

$$D^*(x_1, x_2) = D(x_1, x_2) + \frac{d_r - D(w_r)}{W} \Delta w_r \tag{3.9}$$

Let $z_s$ the center of the cluster $C_s$. Let $\Delta w_r$ a perturbation of the attribute $w_r$. Let $B_s$ the ball defined by the points $x_j$ so that $d_h(x_j, z_s) = D(x_j, z_s)$. Points over $B_s$ are not affected by any $\Delta w_r$. Points inside the ball $B_s$ has $d_r > D$, and $\frac{\partial D}{\partial w_r} < 0$, thus any positive $\Delta w_r$ returns a negative $\Delta D$, and points are *attracted* into the cluster $C_s$. Points outside the ball $B_s$ has $d_r < D$, and $\frac{\partial D}{\partial w_r} > 0$, thus any positive $\Delta w_r$ gives returns a positive $\Delta D$, and points are *moved away* to the cluster $C_s$.

### Instances' displacement by a single weight tuning

Given an instance $x_j$ and a ball $B_s$, related to the center $z_s$ of cluster $C_s$, one of the following conditions holds:

- $x_j$ does not belong to the ball $B_s$;
- $x_j$ is onto the ball $B_s$;
- $x_j$ belongs to the ball $B_s$.

Let the third condition holds for a specific $x_j^*$, and assume that $x_j^*$ belongs to $C_s$ while it is also close to the cluster $C_{new} \neq C_s$. If we want displace $x_j^*$ from $C_s$ to $C_{new}$, we should apply a positive $\Delta w_r$ so that, after the weight tuning we will obtain:

$$D'(x_j^*, z_{new}) \leq D'(x_j^*, z_s) \tag{3.10}$$

with

$$D(x_j^*, z_{new}) \geq D(x_j^*, z_s) \tag{3.11}$$

before the displacement, and

$$\begin{aligned} D'(x_j^*, z_s) &= D(x_j^*, z_s) + \frac{d_r - D(w_r)}{W} \Delta w_r \\ D'(x_j^*, z_{new}) &= D(x_j^*, z_{new}) + \frac{d_r - D(w_r)}{W} \Delta w_r \end{aligned}$$

**Multiple weight tuning**

Let $x_1, x_2$ two instances in $X$, the distance $D(x_1, x_2)$ is defined by the following equation:

$$D = D(x_1, x_2) = \frac{1}{W} \sum_{h=1}^{H} w_h d_h(v_j^{(1)}, v_j^{(2)}) = \frac{1}{W} \vec{d} \bullet \vec{w} \tag{3.12}$$

Let perturbate $\vec{w}$ by $\vec{\Delta w}$, we have

$$\Delta D = \Delta D(x_1, x_2) = \vec{\nabla}_w D \bullet \vec{\Delta w} \tag{3.13}$$

where $\nabla_w$ is the the gradient operator;

$$\vec{\nabla}_w D = \vec{\nabla}_w \left( \frac{\vec{d} \bullet \vec{w}}{W} \right) = \frac{\vec{d} - D\vec{\xi}}{W} \quad with \quad \vec{\xi} = \frac{\vec{w}}{W} \tag{3.14}$$

where $\vec{\xi}$ is the unit vector of $\vec{w}$, then

$$\Delta D = \Delta D(x_1, x_2) = \frac{\vec{d} - D\vec{\xi}}{W} \bullet \vec{\Delta w} \tag{3.15}$$

Equation 3.15 shows that $\Delta D$ is not affected by $\vec{\Delta w}$ if $\vec{d} = D\vec{\xi}$ or when $\vec{d} \neq D\vec{\xi}$ but $\vec{u} = \frac{\vec{d} - D\vec{\xi}}{W}$ is orthogonal to $\vec{\Delta w}$. In general, a multiple weight tuning affect the distance $D$ so that:

$$D^*(x_1, x_2) = D(x_1, x_2) + \frac{\vec{d} - D\vec{\xi}}{W} \bullet \vec{\Delta w} \tag{3.16}$$

Explicitly, we have

$$D^*(x_1, x_2) = D(x_1, x_2) + \frac{1}{W} \sum_{h=1}^{H} \left[ d_h(v_j^{(1)}, v_j^{(2)}) - D(x_1, x_2) \right] \Delta w_h \tag{3.17}$$

## 3.5 Refinement module

The Refinement module modifies the clusters' centers in order to increase the number of satisfied user constraints, thus eventually producing the desired result. This module exploits both a merging and a splitting techniques that affect the clusters' shapes. Actually, this approach allows tuning the obtained clustering in the way that constraints 4, 5 and 6 might be guaranteed. The merging and splitting techniques are performed by a search into the cluster space with the goal of finding the most relevant solutions that allow satisfying the user constraints. User is asked to choose among these solutions.

## 3.6 Constraint analytical expression

Constraints 1, 2, 3 and 7 can be translated into a set of differential disequations of whom variables are the differentials of the attribute weights $\vec{\Delta w}$. When a solution of these disequation exist, the attributes' weights are properly modified and the Optimization module succeedes. On the other hand a perturbation $\vec{\Delta w}$ both that minimizes a given objective function or that satisfies a subset of differential disequations obtained by an selection that involves the user, is often accepted. The latter strategy invokes an iterative algorithm that increasly reduces the number of differential disequations by a selection discarding the most relevant ones. Disequations importance is mainly measured in term of constraint values but a constraint evaluation criteria is also furnished to prioritize them (see paragraph 3.7). Given the constraint priorities obtained by the evaluation criteria, the user is allowed modifying the disequations' subset that at each step is taken into account throughout a tuning of the constraints values.

Constraints 4, 5 and 6 can be handled by merging and splitting operations that have been implemented by automatic procedures.

**Type 1: Instance Association** requires that instances $x_i \in X' \subseteq X$, have to belong to the same cluster $C_s$, thus we have to satisfy the following disequations' sets (with $K-1$ disequations' per set):

$$\forall\, x_i \in X'\ D(x_i, z_s) \leq D(x_i, z_j) \quad j = 1..K \wedge j \neq s \tag{3.18}$$

where $z_s$ is center of cluster $C_s$ and $z_j$ is the center of cluster $C_j$.

If for some $x_p \in X'' \subseteq X'$ disequations $D(x_p, z_s) \leq D(x_p, z_j)$ do not hold, a $\vec{\Delta w}$ is applied.

Let

$$
\begin{aligned}
D'(x_p, z_s) &= D(x_p, z_s) + \Delta D(x_p, z_s) \\
D'(x_p, z_j) &= D(x_p, z_j) + \Delta D(x_p, z_j)
\end{aligned}
\tag{3.19}
$$

we have the disequations

$$
D'(x_p, z_s) \leq D'(x_p, z_j) \tag{3.20}
$$

while equation 3.18 has still to hold for any $x_i \in X' - X''$.

**Type 2 and 3: Value Association and Value Separation** requires that instances $x_i \in X' \subseteq X$ with the a given value $v_j$ of the attribute $A_h$ have mostly to belong to the same cluster (in the former case only one, while in the latter the attribute distribution has to be partitioned). The analytical expression for these constraints can be indirectly obtained by a disequation that forces these instances to be close each other. Let $X'' = X - X'$, we have:

$$
\forall\, x_a, x_b \in X' \quad \forall\, x_c, x_d \in X'' \quad D(x_a, x_b) \ll D(x_c, x_d) \tag{3.21}
$$

Let

$$
\begin{aligned}
D(x_a, x_b) &= \frac{1}{W} \sum_{h=1}^{H} w_h\, d_h(v_j^{(a)}, v_j^{(b)}) \quad (d_\alpha = 0) \\
D(x_c, x_d) &= \frac{1}{W} \sum_{h=1}^{H} w_h\, d_h(v_j^{(c)}, v_j^{(d)})
\end{aligned}
\tag{3.22}
$$

as $D(x_a, x_b)$ is not affected by $w_\alpha$ when this is particularly high, we have:

$$
D(x_c, x_d) - D(x_a, x_b) \leq w_\alpha * d_{h,max} \tag{3.23}
$$

where $d_{h,max} = Max\{d_h(x_c, x_d)\}$.

In general,

$$
w_\alpha * d_{h,max} \leq max\{average(IntraCluster\ Distances)\} \tag{3.24}
$$

holds, and

$$
w_\alpha \leq \frac{max\{average(IntraCluster\ Distances)\}}{d_{h,max}} \tag{3.25}
$$

If for some $\{x_a, x_b\} \wedge \{x_c, x_d\}$ disequation 3.23 does not hold, a $\vec{\Delta} w$ is applied.

Let

$$D'(x_a, x_b) = D(x_a, x_b) + \Delta D(x_a, x_b) \quad (3.26)$$
$$D'(x_c, x_d) = D(x_c, x_d) + \Delta D(x_c, x_d)$$

we have the disequations:

$$D'(x_c, x_d) - D'(x_a, x_b) \leq w_\alpha * d_{h,max} \quad (3.27)$$

**Type 4: Number of Clusters** - the constraint can be handle by a proper selection of the $K_{max}$ parameter injected into the segmentation algorithm.

**Type 5: Cluster Cardinality** - the constraint can be handled by a re-assignment of subsets of instances to clusters. When an instance $x_i$ is assigned to a cluster $C_s$ of whom cardinality is smaller then $K_s$ then cluster $C_s$ is destroyed, and $x_i$ is re-assigned to its closer cluster $C_\alpha$ but $C_s$. In general, instances' displacements might requires an adjustment of the center $z_\alpha$ of cluster $C_\alpha$.

**Type 6: Cardinality Ratio** - the constraint can be handled with an algorithm that works as described above.

**Type 7: Cluster Adherence** requires the logical AND of constraints 2 and 3, thus it can be according handled.

## 3.7 Constraint evaluation criteria

In practical applications, constraints' values are expressed as a collection of adjectives. Actually, users often classify a constraint to be

$$\{\text{Fundamental}, \text{Very important}, \text{Important}, \text{Desirable}, \text{Secondary}\}$$

thus the need of transforming them into proper numerical values $v_i$. The straightforward relation below can be usefully exploited for this purpose:

$$\begin{array}{lclclcl} \text{Fundamental} & \rightarrow & 5 & \rightarrow & \frac{5}{15} & = & 0.333 \\ \text{Very important} & \rightarrow & 4 & \rightarrow & \frac{4}{15} & = & 0.267 \\ \text{Important} & \rightarrow & 3 & \rightarrow & \frac{3}{15} & = & 0.200 \\ \text{Desirable} & \rightarrow & 2 & \rightarrow & \frac{2}{15} & = & 0.133 \\ \text{Secondary} & \rightarrow & 1 & \rightarrow & \frac{1}{15} & = & 0.067 \end{array}$$

The global constraints satisfaction function is defined by the following equation:

$$\Theta(C) = \frac{1}{Q}\sum_{i=1}^{T} q_i * \theta_i(C) \quad (3.28)$$
$$Q = \sum_{i=1}^{T} q_i$$

where $q_i$ are the constraints' values, while $\theta_i(C)$ in [0..1] are single constraints satisfaction function. $\theta_i(C)$ might also be less then one, while different to zero. This choice is mainly motivated by the need of evaluating solutions that while non satisfying all constraints, can be accepted by the user. As a matter of the fact, given the complexity of differential disequations introduced in Section 3.6, it is often hard to find a complete solution while partial results can still be really precious for the user. For this reason, $\Theta(C)$ ranges in [0..1], if all constraints are satisfied $\Theta(C) = 1$, while $\Theta(C) = 0$ when all constraints are not satisfied.

In order to introduce the function $\theta_i(C)$ for any constraint, let define the following objects:

**Definition 1** - Let $A_h$ a categorical attribute, $\Lambda'_h \subseteq \Lambda_h$ a subset of its possible value, and $X(A_h, \Lambda'_h)$ the subset of examples for which the attribute $A_h$ exhibit a value in $\Lambda'_h$. Let $N(A_h, \Lambda'_h)$ the number of these examples.

**Definition 2** - Let $A_h$ a categorical attribute and $\Lambda_h$ the set of its values. Let $\vec{\tau}(A_h)$ the vector, with $L_h$ componets, that contains the histogram of the values of $A_h$ over $X$. In particular, $\tau_j(A_h)$ is the number of examples in $X$ so that $A_h = v_j$.

When $A_h$ is numerical we will take into account intervals instead of values, while if $A_h$ is structured we will evaluate leaves (or interval nodes when leaves share a father) of the taxonomy instead of single values.

Let introduce the following equations:

$$\begin{aligned} N(A_h, \Lambda'_h) &= \sum_{v_j \in \Lambda'_h} \tau_j(A_h) \qquad (3.29) \\ N &= \sum_{j=1}^{L_h} \tau_j(A_h) \end{aligned}$$

Let also introduce the range, the average and the standard deviation of $\vec{\tau}(A_h)$ with $A_h$ numerical:

$$R(\vec{\tau}(A_h)) = Max_j\{v_j|\tau_j(A_h) \leq 0\}_{j\in[1..L_h]} - Min_j\{v_j|\tau_j(A_h) \leq 0\}_{j\in[1..L_h]} \qquad (3.30)$$

$$\begin{aligned} E(\vec{\tau}(A_h)) &= \frac{1}{\sum_{j=1}^{L_h} \tau_j(A_h)} \sum_{j=1}^{L_h} v_j \tau_j(A_h) \\ VAR(\vec{\tau}(A_h)) &= \frac{1}{\sum_{j=1}^{L_h} \tau_j(A_h)} \sqrt{\sum_{j=1}^{L_h} [\tau_j(A_h)(v_j - E(\vec{\tau}(A_h)))^2]} \end{aligned}$$

Let now taking into account a single cluster $C_i \in C$ and extend definitions 1 and 2 to $C_i$, so that:

**Definition 3** - Let $C_i \in C$ a single cluster, $A_h$ a categorical attribute, $\Lambda'_h \subseteq \Lambda_h$ a subset of its possible value, and $X^{(i)}(A_h, \Lambda'_h)$ the subset of examples in $C_i$ for which the attribute $A_h$ exhibit a value in $\Lambda'_h$. Let $N^{(i)}(A_h, \Lambda'_h)$ the number of these examples.

**Definition 4** - Let $C_i \in C$ a single cluster, $A_h$ a categorical attribute and $\Lambda_h$ the set of its values. Let $\vec{\tau}^{(i)}(A_h)$ the vector, with $L_h$ componets, that contains the histogram of the values of $A_h$ over $C_i$. In particular, $\tau_j^{(i)} A_h)$ is the number of examples in $C_i$ so that $A_h = v_j$.

Let introduce the following equation:

$$N^{(i)}(A_h, \Lambda'_h) = \sum_{v_j \in \Lambda'_h} \tau_j^{(i)}(A_h) \tag{3.31}$$

and the range, the average and the standard deviation analogously.

**Type 1: Instance Association** requires that instances $x_i \in X' \subseteq X$, have mostly to belong to the same cluster. Let A and B the instances that have to belong to the same cluster $C_s$, the constraint satisfaction function is a binary function, namely:

$$\theta_1(C) = \begin{cases} 1 & \text{if } A \in C_s \text{ and } B \in C_s \\ 0 & \text{otherwise} \end{cases} \tag{3.32}$$

In general, let $N'$ the number of instances $x_i \in X'$ that have to belong to the same cluster $C_s$, let $M_s$ the instances in $X_s$ that already belong to $C_s$, the constraint evaluation function is:

$$\theta_1(C) = \frac{M_s - 1}{N'} \tag{3.33}$$

(We assume that at least one instance already belong to $C_s$).

**Type 2: Values Association** requires that instances with a given value, $v_j \in \Lambda_h$, for the categorical or numerical attribute $A_h$, have mostly to belong to the same cluster. The constraint satisfaction function for this constraint can be obtained by the $\tau_j^{(i)}(A_h)$ (with $1 \leq i \leq K$) distribution over clusters, more specifically we have:

$$\theta_2(C) = \frac{Max_i(\tau_j^{(i)}(A_h))}{N(A_h, \Lambda'_h)} \quad 1 \leq i \leq K \tag{3.34}$$

$\theta_2(C)$ ranges in $[\frac{1}{K}..1]$, if all instances with the same value $v_j$ of the attribute $A_h$ are grouped into the same cluster $\theta_2(C) = 1$, on the other hand, if they are uniformly distributed over clusters $\theta_2(C) = \frac{1}{K}$.

**Type 3: Value Separation** requires that the value distribution of a given attribute $A_h$ mostly has to be partitioned over clusters, namely instances that exhibit different values for $A_h$ have to belong to different clusters. The constraint satisfaction function for this constraint can be obtained by the $\vec{\tau}^{(i)}(A_h)$ vector distribution over clusters (with $1 \leq i \leq K$) , more specifically we have:

$$\begin{aligned}\delta^{(i)} &= 1 - 2\frac{VAR(\vec{\tau}^{(i)}(A_h))}{R(\vec{\tau}^{(i)}(A_h))} \quad \textit{if } A_h \textit{ is numerical} \\ \delta^{(i)} &= 1 - \frac{m_i}{L_h} \quad \textit{if } A_h \textit{ is categorical}\end{aligned} \tag{3.35}$$

where $m_i$ is the number of different values of $A_h$ in cluster $C_i$, the constraint function is:

$$\theta_3(C) = \frac{1}{K}\sum_{i=1}^{K}\delta^{(i)}. \tag{3.36}$$

$\theta_3(C)$ ranges in [0..1]. $\theta_3(C)$ is one if all $\delta^{(i)}$ are equal to one, i.e. when all $VAR(\vec{\tau}^{(i)}(A_h))$ are zero. This condition occurs when the vector $\vec{\tau}^{(i)}(A_h)$ (with $1 \leq i \leq K$) exhibits only one not null component (namely, instances with $A_h = v_j$ mostly belong to a cluster). $\theta_3(C)$ is zero when all $\delta^{(i)}$ are zero, i.e. when the the standard deviation of each distribution is uniform over the range.

**Type 7: Cluster Adherence** requires that values' distribution of the attribute $A_h$ mostly has to be partitioned over clusters, and instances with a given value $v_j \in \Lambda_h$ have mostly to be grouped into one cluster, namely the logical AND between constraints 2 and 3. The condition requires that $\vec{\tau}^{(i)}(A_h)$ exhibits only one not null component for each $i \in [1..K]$, and that this histogram component differs from cluster to cluster (off course, a complete solution can be found *iff* $K = L_h$). The constraint satisfaction function for this constraint is:

$$\begin{aligned}\delta^{(i)} &= 1 - 2\frac{VAR(\vec{\tau}^{(i)}(A_h))}{R(\vec{\tau}^{(i)}(A_h))} \quad \textit{if } A_h \textit{ is numerical} \\ \delta^{(i)} &= 1 - \frac{m_i}{L_h} \quad \textit{if } A_h \textit{ is categorical} \\ \delta_{i^*,j^*} &= 1 - \frac{E(\vec{\tau}^{(i^*)}(A_h)) - E(\vec{\tau}^{(j^*)}(A_h))}{R(\vec{\tau}(A_h))}\end{aligned}$$

the constraint function is:

$$\theta_7(C) = \left[\frac{1}{K}\sum_{i=1}^{K}\delta^{(i)}\right]\left[\frac{2}{K(K-1)}\sum_{i=1}^{K-1}\sum_{j=1}^{K}(1-\delta_{i,j})\right] \tag{3.37}$$

Let now introduce the following function:

$$\sigma(x,a,b,c) = \begin{cases} \frac{x-1}{a-1} & 1 \leq x \leq a \\ 1 & a \leq x \leq b \\ \frac{b-x}{c-b} & b \leq x \leq c \end{cases} \tag{3.38}$$

**Type 4: Number of Clusters**. The number of clusters has to belong to the range $[K_{min}..K_{max}]$, the constraint evaluation function is:

$$\theta_4(c) = \sigma(K,\ K_{min},\ K_{max},\ K_{max}+1) \tag{3.39}$$

where $K$ is the number of clusters obtained by the segmentation algorithm.

**Type 5: Cluster Cardinality**. The cardinality of each cluster has to be bigger then a given value $M_{min}$, the constraint evaluation function is:

$$\theta_5(C) = \frac{1}{K}\sum_{i=1}^{K} \sigma(M_i, M_{min}, N, N) \tag{3.40}$$

where $M_i$ is the cardinality of cluster $C_i$, while $N$ is the number of instances.

**Type 6: Cardinality Ratio**. The Max/Min cardinality ratio has to be less or equal to a given value $R$. Let

$$\begin{aligned} M_{min} &= Min_i\{M_i\} \quad i = 1..K \\ M_{max} &= Max_i\{M_i\} \quad i = 1..K \end{aligned} \tag{3.41}$$

the constraint evaluation function is:

$$\theta_6(C) = \sigma(1 + \frac{M_{min}}{M_{max}}, 1+R, 2, \inf) \tag{3.42}$$

## 3.8 Objective function

This section presents the objective function we exploit into the Optimization module of SEGMENT. This function is obtained as the average of single optimization function ( $\varphi_i$ ) of constraints 1, 2, 3 and 7, namely

$$\varphi(\vec{\xi}) = \sum_{i \in V} q_i\ \varphi_i(C, \vec{\xi}). \tag{3.43}$$

Let $\vec{w}$ be the weight vector, and $\vec{\xi}$ its unit vector, for each constraint, we have the following functions.

**Type 1: Instance Association** requires that instances $x_i \in X' \subseteq X$, have mostly to belong to the same cluster. Let $N'$ the number of instances $x_i \in X'$ that have to belong to the same cluster $C_s$, let $M_s$ the instances in $X_s$ that already belong to $C_s$, we should move $N' - M_s$ instances into $C_s$. The objective function is:

$$\varphi_1(C, \vec{\xi}) = \left[ \frac{1}{N' - M_s} \sum_{i=1}^{N'-M_s} D(x_j, z_s) - \frac{1}{M_s} \sum_{i=1}^{M_s} D(x_i, z_s) \right]^2 + \frac{1}{N'} \sum_{i=1}^{N'} D(x_j, z_s) \tag{3.44}$$

where $z_s$ is the center of cluster $C_s$.

**Type 2: Value Association** requires that instances with a given value, $v_j \in \Lambda_h$, for the categorical or numerical attribute $A_h$, have mostly to belong to the same cluster. Let $N'$ the number of instances with the a given value, $v_j$, for the attribute $A_h$, let $M_s$ the number of instances that already belong to a cluster $C_s$; we have to be push $N' - M_s$ instances into $C_s$. The objective function is:

$$\varphi_2(C, \vec{\xi}) = \left[ \frac{1}{N' - M_s} \sum_{i=1}^{N'-M_s} D(x_j, z_s) - \frac{1}{M_s} \sum_{i=1}^{M_s} D(x_i, z_s) \right]^2 + \frac{1}{N'} \sum_{i=1}^{N'} D(x_j, z_s) \tag{3.45}$$

where $z_s$ is the center of cluster $C_s$.

**Type 3: Value Separation** requires that the value distribution of a given attribute $A_h$ mostly has to be partitioned over clusters, namely instances that exhibit different values for $A_h$ have to belong to different clusters. Let $z_i$ the center of instances for whom $A_h = v_j$, let $N_i$ the number of these instances ($1 \leq i \leq L_h$), let

$$\begin{aligned} \varphi_i &= \frac{1}{N_i} \sum_{j=1}^{N_i} D(x_j, z_i) \qquad (3.46) \\ \varphi_{p,q} &= \frac{\varphi_p + \varphi_q}{2} \end{aligned}$$

we have

$$\varphi_3(C, \vec{\xi}) = \frac{2}{L_h(L_h - 1)} \sum_{i=1}^{L_h - 1} \sum_{j=i+1}^{L_h} \left(1 - [D(z_i, z_j) - \varphi_{i,j}]^2\right) \tag{3.47}$$

**Type 7: Cluster Adherence** requires that values' distribution of the attribute $A_h$ mostly has to be partitioned over clusters, and instances with a given value $v_j \in \Lambda_h$ have mostly to be grouped into one cluster, namely the logical AND between constraints 2 and 3. Let $z_i$ the center of instances for whom $A_h = v_j$, let $N_i$ the number of these instances $(1 \leq i \leq L_h)$, with the position in equations 3.46, we have

$$\varphi_4(C, \vec{\xi}) = \frac{1}{L_h} \sum_{i=1}^{L_h} \varphi_i + \frac{2}{L_h(L_h - 1)} \sum_{i=1}^{L_h - 1} \sum_{j=i+1}^{L_h} \left(1 - [D(z_i, z_j) - D_{i,j}]^2\right) \tag{3.48}$$

# Appendix A

# Symbols

- $X$ - dataset
- $x_1, ..., x_N$ - instances or examples
- $N$ - number of instances (cardinality of $X$)
- $K$ - number of clusters
- $K_M$ - max number of clusters
- $A_1, ..., A_H$ - attributes
- $H$ - number of attributes
- $w_1, ..., w_H$ - attributes' weights
- $W = \sum_{i=0}^{H} w_i$
- $\Lambda_h = \{v_1, ..., v_{Lh}\}$ - values of attribute $A_h$
- $D$ - distance between two instances
- $C_i$ - cluster
- $M_i$ - cardinality of cluster $C_i$
- $C = \bigcup_{i=1}^{K} C_i$ clustering
- $T$ number of constraints
- $V_1, \ldots, V_T$ constraints
- $q_1, \ldots, q_T$ constraints values
- $Q = \sum_{i=0}^{T} q_i$
- $z_i$ center of the cluster $C_i$
- $Z_j$ set of centers of a clustering