Deliverable D4.2

Studies in Parameter Setting

Lorenza Saitta	Università del Piemonte Orientale
Marco Botta	Università del Piemonte Orientale
Giuseppe Beccari	TILab
Ralf Klinkenberg	University of Dortmund

Abstract

Deliverable D4.2 is an integral part of Deliverable D4.3, which contains the running software for automated parameter setting. For some algorithms the types of meaningful constraints the user is allowed to specify, the method to encode these constraints, and, finally, the function to be optimized are reported. The details of the overall process must be hidden from the user, who is only required, through a graphical interface, to specify the constraints.

Studies in Parameter Setting

As described in Deliverable D4.1, when setting an algorithm/system to work, several parameters are usually to be defined. Often, a preliminary series of test runs is performed in order to find some sub-optimal setting. This procedure may be time consuming. On the other hand, in most algorithms suitable to complex real-world applications, the link between a parameter's value and the output cannot be easily a-priori specified. On the other hand, the user may be able to specify a set of constraints on the desired output. If the satisfaction of each constraint can be codified into a function, then this function can be (semi-)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.

Deliverable D4.1 described the task- and algorithm-independent part of the loop into which the user and the algorithm are inserted, with the specification of the types of information they should exchange. The whole process is described in Figure 1.

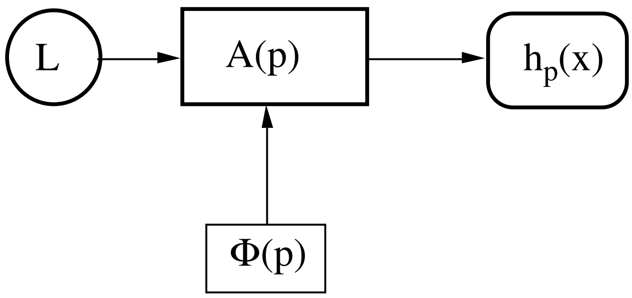

Figure 1 – Task- and algorithm-independent part of the parameter setting process. Algorithm A depends on a vector **p** of parameters. The function $\Phi(\mathbf{p})$ contains the evaluation of the constraints, which contributes to the selection of a particular hypothesis $h_p(x)$ depending upon **p**.

The algorithm A has usually an internal function Ψ to optimize (for example, classification algorithms try to minimize prediction error, clustering algorithms may maximize inter-cluster distance and/or minimize intra-cluster distance). It is important to notice that functions Φ and Ψ cannot be identical, otherwise overfitting occurs. Nevertheless, function Ψ implicitly depends upon the parameter vector **p**, so that optimizing Φ does have effects also on Ψ.

1. Algorithm SEGMENT

SEGMENT is a density-based, non-hierarchical clustering algorithm, which generates flat partitions of an example set, and is currently available to TILab and DISTA in the Mining Mart project. It is used to segment a set of examples, described with <Attribute – Value > pairs. More precisely, let $A = \{A_1, ..., A_H\}$ be a set of attributes and X the set of examples to partition. Let v_h be a generic value of attribute A_h. The notation $v_h^{(i)}$ indicates the value of attribute A_h in example $\mathbf{X}_i$. For each attribute A_h a *distance* measure $d_h(v_h^{(1)}, v_h^{(2)})$ is defined, depending on the attribute nature. The global distance between two examples $\mathbf{X}_1$ and $\mathbf{X}_2$ is given by the average distance over the attributes:

$$D(\mathbf{X}_1, \mathbf{X}_2) = \frac{1}{H}\sum_{h=1}^{H} d_h(v_h^{(1)}, v_h^{(2)})$$

Let now $w_1, ..., w_H$ be the weights assigned to the attributes. Then:

$$D(\mathbf{X}_1, \mathbf{X}_2) = \frac{1}{W}\sum_{h=1}^{H} w_h\, d_h(v_h^{(1)}, v_h^{(2)})$$

$$W = \sum_{h=1}^{H} w_h$$

The basic component of the SEGMENT system is the procedure *Cluster* (X, W, K_M), which has as input the set of examples to be partitioned, the set of weights, and a maximum value for the number of clusters in any given partition. The value K_M is usually twice the sought number of clusters. An abstract description of the procedure is reported in Figure 2.

<u>Cluster</u>(X, K_M)
Select $\{\mathbf{Z}_1, ..., \mathbf{Z}_{K_M}\}$ from X; $X' = X - \{\mathbf{Z}_1, ..., \mathbf{Z}_{K_M}\}$; $N' = N - K_M$
CLUST(1) = X
for i = 1, K_M **do** $C_i = \{\mathbf{Z}_i\}$ **end**
for each $\mathbf{X} \in X'$ **do**
    if $D(\mathbf{X}, \mathbf{Z}_1) \le D(\mathbf{X}, \mathbf{Z}_2)$ **then** $C_1 = C_1 \cup \{\mathbf{X}\}$
        else $C_2 = C_2 \cup \{\mathbf{X}\}$
    endif
end
Memorize CLUST(2) = $\{ C_1, C_2 \}$

```
for  j = 3, K_M  do
     for  i = 1, j-1  do
     for each X ∈ C_i do
               if D(X, Z_j) ≤ D(X, Z_i)  then  C_i = C_i - {X}; C_j = C_j ∪ {X}
          end
end
Memorize CLUST(j) = { C_1, ..., C_j }
end
```

Figura 2 – Algorithm *Cluster* (X, W, K_M), used inside SEGMENT to generate partitions with number of clusters ranging from 2 to K_M.

When *Cluster* terminates, a series of different clusterings, CLUST(j) ($1 \leq j \leq K_M$), is generated: each clustering contains j clusters. Let $M_i^{(j)} = |C_i^{(j)}|$ ($1 \leq j \leq K_M$, $1 \leq i \leq j$) be the cardinality of the i-th cluster in the j-th grouping. Being the clusters disjoint, we have:

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

The *Cluster* algorithm is inserted into a loop which generates K_M clustering with a number of clusters raging from 1 to K_M, as described in Figure 3.

```
SEGMENT (X, K_M, K*)

if  K_M = K*
     then Cluster (X, K*)
     else    Cluster(X, K_M)
Choose K* ≤ K_M
endif
```

Figure 3 – Global algorithm SEGMENT. The value K* is the optimal number of clusters.

When the set of clustering with different numbers of clusters has been generated, the optimal one must be chosen. Let K* be this optimal value. The K* value is chosen using two evaluation function, the *cohesion* and the *separation* functions. The first one measures how similar are the examples inside each cluster, whereas the second one measures how different are examples in separate clusters. Let us consider first the cohesion function.

Cohesion function

Let us introduce the following notations:

$\gamma_C(A)$ = Cohesion function of a cluster C w.r.t. the values of attribute A in its elements.
$\Gamma(C)$ = Cohesion of a cluster C
$\Gamma_{tot}(K)$ = Cohesion of a group of K clusters C_j ($1 \le j \le K$)

Moreover, let $D_{max}(C,A)$ be the maximum distance, w.r.t. A, of element pairs occurring in C. As the distance function is defined in the interval [0, 1], then $0 \le D_{max}(C,A) \le 1$.

Let us define the *Cohesion function* $\gamma_C(A)$ of C w.r.t. A as follows:

+ *Numerical attributes :* $\gamma_C(A) = 1 - D_{max}(C,A)$

+ *Categorical attributes:*

$$\gamma_C(A) = \begin{cases} 1 - \dfrac{l-1}{L} & \text{se } l < L \\ 1 - \dfrac{\|\vec{v}(X,A) - \vec{v}(C,A)\|}{\sqrt{2}} & \text{se } l = L \end{cases}$$

Let $\vec{v}(X,A)$ and $\vec{v}(C,A)$ be the L-dimensional vectors containing the distributions of the A's values in C and in X.

+ *Hierarchical attribute:*

$$\gamma_C(A) = 1 - D_{max}(C,A)$$

According the above definitions, the cohesion w.r.t. a single attribute of a clustering is 0, when each example constitutes a cluster, to 1, when there is a unique single cluster.

The *Cohesion* $\Gamma(C)$ of a cluster C w.r.t. all attributes is given by:

$$\Gamma(C) = \frac{1}{W} \sum_{h=1}^{H} w_h \; \gamma_C(A_h)$$

The maximum cohesion $\Gamma(C) = 1$ corresponds to the case that $\gamma_C(A_h) = 1$ for each attribute, namely, when the clustering contains a unique cluster C. On the contrary, $\Gamma(C) = 0$ when $C \equiv X$.

If a clustering contains K clusters C_j with cardinality M_j ($1 \le j \le K$), the total cohesion $\Gamma_{tot}(K)$ will be:

$$\Gamma_{tot}(K) = \frac{1}{N}\sum_{j=1}^{K} M_i \;\; \Gamma(C_j)$$

The typical behavior of the cohesion function is reported in Figure 4.

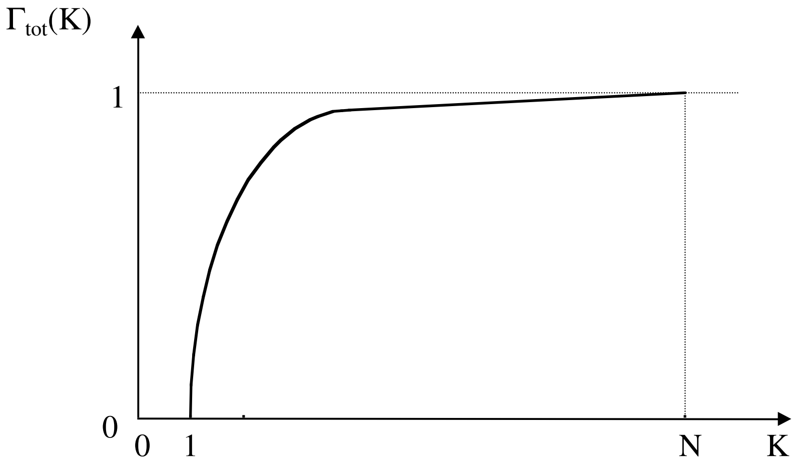

Figure 4 – Typical behaviour of the cohesion function.

Separation Function

Let us introduce the following notations:

$\sigma_{C1,C2}(A)$ = Separation function between clusters C_1 and C_2 w.r.t. the values of attribute A.
$\Sigma(C_1, C_2)$ = Separation between clusters C_1 and C_2 w.r.t. the values of all attributes.
$\Sigma_{tot}(K)$ = Separation among a group of K clusters C_j ($1 \leq j \leq K$)

Moreover, let $D_{min}(C_1, C_2, A)$ and $D_{max}(C_1, C_2, A)$ be the minimum and maximum distance, w.r.t. A, of element pairs occurring one in C_1 and the other in C_2.

Let us define the *Separation function* $\sigma_{C1,C2}(A)$ as follows:
+ *Numerical and Hierarchical attributes :*

$$\sigma_{C_1C_2}(A) = \frac{D_{min}(C_1, C_2, A)}{D_{Max}(C_1, C_2, A)}$$

+ *Categorical attributes:*

$$\sigma_{C_1C_2}(A) = \frac{\| \vec{v}(C_1, A) - \vec{v}(C_2, A) \|}{\sqrt{2}}$$

where $\vec{v}(X,A)$ and $\vec{v}(C,A)$ are the L-dimensional vectors containing the distributions of the A's values in C_1 and C_2.

+ *Hierarchical attribute:*

$$\gamma_C(A) = 1 - D_{max}(C,A)$$

According the above definitions, the separation w.r.t. a single attribute of a clustering is 1, when each example constitutes a cluster, and to 0, when there is a unique single cluster.

The *Separation* $\Sigma(C_1, C_2)$ w.r.t. all attributes is given by:

$$\Sigma(C_1,C_2) = \frac{1}{W} \sum_{h=1}^{H} w_h \ \sigma_{C_1C_2}(A_h)$$

The minimum separation $\Gamma(C) = 1$ corresponds to the case that $\sigma_{C_1C_2}(A) = 1$ for each attribute, namely, when the clustering contains the unique cluster X. On the contrary, $\Sigma(C_1,C_2) = 1$ when $C_1,C_2 \equiv X$.

If a clustering contains K clusters C_j with cardinality M_j $(1 \leq j \leq K)$, the total separation $\Sigma_{tot}(K)$ will be:

$$\Sigma_{tot}(K) = \frac{1}{N\ (K-1)} \sum_{i=1}^{K-1} \sum_{j=i+1}^{K} (M_i + M_j)\ \Sigma(C_i, C_j)$$

Ideally, the optimal number K* of clusters is the K value corresponding to the maximum curvature of the cohesion and separation functions.

Constraints Given by the User

The user may provide constraints of the following types:

TYPE 1: *Instances Association*

The instances $x_1, x_2, \ldots, x_n$ must be in the same cluster.

TYPE 2: *Values Association*

The instances that have the same value v for the attribute A must be in the same cluster.

TYPE 3: *Values Separation*

The values distribution of the attribute A must have separated modality in the clusters.

TYPE 4: *Number of clusters*

The number of clusters must be in the interval $[K_1, K_2]$

TYPE 5: *Cardinality of cluster*

The cardinality of each cluster must be bigger then M.

TYPE 6: *Cardinality ratio*

The ratio of the max cardinality to the min one must not exceede r.

TYPE 7: *Modality conservation*

The values distribution of the attribute A must have modality separated in the clusters and each modality must be in only one cluster.

Constraint of Type 2 require that the instances that have a certain value for an attribute A are concentrated in the same cluster. Nothing is required from the other values that can have any distribution in the cluster.

Constraint of Type 3 require that every cluster has only instances with the same value for a certain attribute. Nothing is required from instances with the same value in the same cluster.

Constraint of Type 7 is the logical AND of the constraints of Type 2 and Type 3.

The constraints of Type 1, 2, 3 and 7 can be satisfied acting on the weights of the attributes. The remaining constraints must be dealt refining the struct of cluster (the number and the centers of the cluster).

These two need of different nature have suggested to introduce two different modules. One, the optimization module, is used to change the weights, the other, the refinement module, is used to change the number and the center of the cluster.

Analytical Expression of the Constraints

The above constraints must be expressed in terms of the parametrers, namely the weights of the attributes, in order to be otpimized. To this aim, let us consider the types of constraints one at a time.

Type 1 Constraint: ***Instance association***

The user can specify that two particular instances, A and B, must belong to the same cluster. This constraint can be expressed as follows:

$$\exists\ \mathbf{Z}_i\ \ [\forall\ j \neq i : D(\mathbf{A},\mathbf{Z}_i) \leq D(\mathbf{A},\mathbf{Z}_j)\ \text{ e }\ D(\mathbf{B},\mathbf{Z}_i) \leq D(\mathbf{B},\mathbf{Z}_j)]$$

If A and B have been assigned to the same cluster, then the constraint is satisfied. Otherwise, let $A \in C_i$ and $B \in C_j$. Then, one of the two points has to move to the other's cluster, or both have to move to a third cluster. If B must move to A's cluster, then, before the move, we have:

$$D(\mathbf{B},\mathbf{Z}_i) > D(\mathbf{B},\mathbf{Z}_j)$$
$$D(\mathbf{A},\mathbf{Z}_i) \leq D(\mathbf{A},\mathbf{Z}_j)$$

After, it must be:

$$D'(\mathbf{B},\mathbf{Z}_i) < D'(\mathbf{B},\mathbf{Z}_j)$$
$$D'(\mathbf{A},\mathbf{Z}_i) < D'(\mathbf{A},\mathbf{Z}_j).$$

By varying the attribute weights, the distance between B and $\mathbf{Z_i}$ shall decrease, whereas the distane between **B** and $\mathbf{Z_j}$ must increase. Then, the following conditions must be verified:

$$D'(\mathbf{B}, \mathbf{Z}_i) = D(\mathbf{B}, \mathbf{Z}_i) + \Delta D(\mathbf{B}, \mathbf{Z}_i) \leq D'(\mathbf{B}, \mathbf{Z}_j) = D(\mathbf{B}, \mathbf{Z}_j) + \Delta D(\mathbf{B}, \mathbf{Z}_j)$$

There are standard algorithms to solve an disequation system of the above kind. The user has only to specify the pairs of instances, as the optimization module translates them automatically into disequations.

Vincoli di Typo 2 and 3 Constraints: ***Value associaton and Modality separation***

The user may specify that instances with the same value for a given attribute A must belong to the same cluster. This constraint can be specified independently of the current cluster centers, by forcing the involved instances to be very "close", as illustrated in Figure 5. Then, most likely, most of the instances will be assigned to the same cluster.

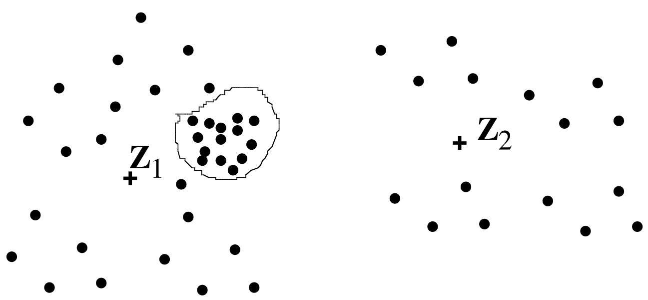

Figura 5 – Instance grouping. The instances tend to behave as a "single example".

The analytical condition that realizes the constraint is the following:

$$D(A,B) = \frac{1}{W} \sum_{\substack{h=1 \\ h \neq r}}^{H} w_h \; d_h(v_h^{(A)}, v_h^{(B)}) \qquad \text{for A and B such that } v_r^{(A)} = v_r^{(B)}$$

$$D(C,D) = \frac{1}{W} \sum_{h=1}^{H} w_h \; d_h(v_h^{(C)}, v_h^{(D)}) \qquad \text{for C and D such that } v_r^{(C)} \neq v_r^{(D)}$$

Clearly, the weight of attribute A_r must be higher than the one of the others, so that:

$$D(A,B) << D(C,D)$$

We have:

$$D(C,D) - D(A,B) \leq w_r \, Max\{d_h\}$$

We impose that:

$$w_r \, Max\{d_r\} \leq Max\{\text{average intra-cluster distance}\}$$

In other words:

$$w_r \leq \frac{Max\{Dist\_Media\_Intra\_Cluster - cluster\}}{Max\{d_r\}}$$

The above condition automatically tries to separate examples that have different values of the attributes. Type 2 and 3 constraints may be conflicting.

The user can utilizes a graphical interface to specify the constraints and the optimization module generates the set of corresponding disequations. If the disequation system is solvable, then the best parameter setting is found. Otherwise the system tries to satisfy the set of constraints as much as possible, starting from the most impostant ones.

Type 4 Constraint: ***Number of cluster***

If the user knows the optimal number K* of clusters, this information can be used directly by SEGMENT. Otherwise, a refinement module shall change K to K*. During the weight optimization phase, the number of clusters does not change. At the end, in order to get close to K*, some clusters can be merged or splitted.

Type 5 and 6 Constraints: ***Minimum cluster cardinality and Maximum cardinality ratio***

If too small clusters have been generated, their elements, during the final refinement, can be reassigned to neibourgh clusters, and the modifications are accepted if the cohesion and separation functions do not change too much. Otherwise, the neibourgh cluster centers are moved closer each other, so that the they will hopefully capture the examples in the small cluster.

The constraint related to the maximum cardinality ratio can be translated into the previous one.

Type 7 Constraint : ***Modality***

This constraint is the logical AND of Type 2 and 3 constraints, then it can be treated as illustrated before.

Evaluation of the Constraint Satisfaction

Let us consider a generic clustering C = {$\mathbf{C}_1$, ..., $\mathbf{C}_K$}, with K cluster $\mathbf{C}_j$, each of cardinality M_j ($1 \leq j \leq K$). Let V = {V_1, ..., V_T} be the set of constraints, and Q = {q_1, ..., q_T} the associated **relevance** set. The relevance values are:

{Fundamental, Very important, Important, Desirable, Fair}

The above linguistic values, given by the user, are translated into numerical values as follows:

Fundamental	$\rightarrow 5$	$\rightarrow$ 5/15 = 0.333
Very important	$\rightarrow 4$	$\rightarrow$ 4/15 = 0.267
Important	$\rightarrow 3$	$\rightarrow$ 3/15 = 0.200
Desirable	$\rightarrow 2$	$\rightarrow$ 2/15 = 0.133
Fair	$\rightarrow 1$	$\rightarrow$ 1/15 = 0.067

Let us define a Satisfaction function *Funzione di Soddisfacimento* $\Theta(C)$ as follows:

$$\Theta(C) = \frac{1}{Q}\sum_{j=1}^{T}\left(q_j \cdot \delta(V_j)\right)$$

where $\delta(V_j)$ is not a binary function, but it is:

$$0 \leq \delta(V_j) \leq 1$$

This approach has the advantage to consider intermediate situations of constraints satisfaction.

The evaluate function $\Theta(C)$ has values in [0,1] and $\Theta(C) = 1$ if all constraints are satisfy and $\Theta(C) = 0$ if no constraint is satisfied. We have now to define the $\delta(V_j)$ function for each type of constraint.

Type 1 Constraint: ***Instance association***

This constraint can only be binary:

$\delta_1(V) =$ **If** (A and B are in the same cluster) **Then** 1, **Else** 0.

If r instances must be in the same cluster, and only s of them actually are, then:

$$\delta_1(V) = \frac{s}{r}$$

Type 2 Constraint: ***Value association***

Let $\tau_{h,j}$ be the number of instances with value v_j for attribute A_h that are in the same cluster, and let $S(A_h, v_j)$ be the total number of instances with value v_j for attribute A_h. Then :

$$\delta_2(V) = \frac{\tau_{h,j}}{S(A_h, v_j)}$$

The function $\delta_2(V)$ is 1 when all examples with value v_j for attribute A_h are in the same cluster. The minimum value of $\delta_2(V)$ is reached when the maximum number of examples in the same cluster is minimum, i.e., when the examples are uniformly subdivided among clusters. Then :

$$\left\lceil \frac{S(A_h, v_j)}{K} \right\rceil \leq \delta_2(V) \leq 1$$

Type 3 Constraint: ***Value separation***

Let C be a clustering, containing K clusters $\mathbf{C}_i$ ($1 \leq i \leq K$) with the hystogram $\vec{\tau}^{(i)}(A_h)$. Let us define:

$$\delta^{(i)} = 1 - 2 \cdot \frac{\sqrt{\mathrm{VAR}\ (\vec{\tau}^{(i)}(A_h))}}{\mathrm{R}\ (\vec{\tau}^{(i)}(A_h))} \quad \text{for continuous-valued attributes}$$

$$\delta^{(i)} = 1 - \frac{m_i}{L_h} \quad \text{for discrete attributes}$$

where m_i is the number of different values of A_h in C_i. Then:

$$\delta_3(V) = \frac{1}{K} \sum_{i=1}^{K} \delta^{(i)}$$

The evaluation function $\delta_3(V)$ is equal to 1 when all $\delta^{(i)}$ are 1, i.e., when all the $\mathrm{VAR}(\vec{\tau}^{(i)}(A_h))$ are equal to zero. The minimum $\delta_3(V)$ is reached when all $\delta^{(i)}$ are zero. Then:

$$0 \leq \delta_3(V) \leq 1$$

Type 7 Constraint: ***Modality***

This type of constraint captures, at the same time, a Type 2 and a Type 3 constraint. Then:

$$\delta^{(i)} = 1 - 2 \cdot \frac{\sqrt{\mathrm{VAR}\ (\vec{\tau}^{(i)}(A_h))}}{\mathrm{R}\ (\vec{\tau}^{(i)}(A_h))} \quad \text{for continuous-valued attributes}$$

$$\delta^{(i)} = 1 - \frac{m_i}{L_h} \quad \text{for discrete attributes}$$

The value m_i is the number of different values of A_h in C_i. Let us consider the average value over all clusters :

$$\frac{1}{K}\sum_{i=1}^{K}\delta^{(i)}$$

If i* is the position of the peak value in the hystograms, the constraint evaluation is good when the i*'s are as much different as possible. Let us define:

$$\delta_6 = \left(\frac{1}{K}\sum_{i=1}^{K}\delta^{(i)}\right)\left(\frac{2}{K\,(K-1)}\sum_{i=1}^{K-1}\sum_{j=i+1}^{K}(1-\delta_{i^*j^*})\right)$$

where:

$$\delta_{i^*j^*} = \begin{cases} 0 & \text{se } i^* \leq j^* \\ 1 & \text{se } i^* = j^* \end{cases} \qquad \text{for discrete attributes}$$

$$\delta_{i^*j^*} = 1 - \frac{E(\vec{\tau}^{(i^*)}(A_h)) - E(\vec{\tau}^{(j^*)}(A_h))}{R(A_h)} \qquad \text{for continuous-valued attributes}$$

Type 4 Constraint: ***Number of clusters***

This constraint specifies that the number of clusters must be in the interval $[K_i, K_s]$. Let us define the function:

$$\sigma(x,a,b,c) = \begin{cases} \dfrac{x-1}{a-1} & \text{se } 1 \leq x \leq a \\ 1 & \text{se } a \leq x \leq b \\ \dfrac{b-x}{c-b} & \text{se } b \leq x \leq c \end{cases}$$

Finally:

$$\delta_4(V) = \sigma(K, K_i, K_s, K_{s+1})$$

Type 5 Constraint: ***Cluster cardinality***

If the minimum cardinality of any cluster is M, then let us define:

$$\delta^{(i)}(V) = \sigma(M, M_i, N, N)$$

Globally:

$$\delta_5(V) = \frac{1}{K}\sum_{i=1}^{K} \sigma(M, M_i, N, N)$$

Type 6 Constraint: ***Cardinality ratio***

If M_{Max} and M_{min} are the maximum and minimum cardinality values, and η is the minimum ratio value, then:

$$M_{min} = \min_{1 \le i \le K} (M^{(i)}) \quad \text{and} \quad M_{Max} = \max_{1 \le i \le K} (M^{(i)})$$

It must be:

$$\frac{M_{min}}{M_{Max}} \ge \eta$$

Using the function σ, we obtain:

$$\delta_7(V) = \sigma\left(1 + \frac{M_{min}}{M_{Max}}, 1 + \eta, 2, \infty\right)$$

Objective Function to Optimize

We have now to define the global function to optimize. Let ξ_h be new variables, normalized to 1:

$$\vec{\xi} = \frac{\vec{w}}{W} \quad \text{and} \quad \xi_h = \frac{w_h}{W} \quad (1 \le h \le H)$$

Moreover:

$$\sum_{h=1}^{H} \xi_h = 1$$

Type 1 Constraint: ***Instance association***

The following term must be small:

$$\varphi_1(\vec{\xi}) = \left[\frac{1}{n-m}\sum_{j=1}^{n-m} D(B, A_j) - \frac{1}{m}\sum_{i=1}^{m} D(B, A_i)\right]^2 + \frac{1}{n}\sum_{i=1}^{n} D(B, A_i)$$

Type 2 Constraint: ***Value association***

The following term must be small:

$$\varphi_2(\vec{\xi}) = \left[\frac{1}{n-m}\sum_{j=1}^{n-m} D(B, A_j) - \frac{1}{m}\sum_{i=1}^{m} D(B, A_i)\right]^2 + \frac{1}{n}\sum_{i=1}^{n} D(B, A_i)$$

Type 3 Constraint: ***Value separtation***

Let A_h be an attribute and let v_i $(1 \le i \le L_h)$ be one of its value. The groups of instances with $A_h = v_i$ must be separated. Let $\mathbf{B_i}$ be the center of gravity of the examples for which $A_h = v_i$ and let n_i be their number. Then:

$$D_i = \frac{1}{n_i}\sum_{j=1}^{n_i} D(B_i, A_j)$$

$$D_{ij} = \frac{D_i + D_j}{2}$$

$$\varphi_{ij}(\vec{\xi}) = 1 - [D(\mathbf{B_i}, \mathbf{B_j}) - D_{ij}]^2$$

Finally:

$$\varphi_3(\vec{\xi}) = \frac{2}{L_h(L_h - 1)}\sum_{i=1}^{L_h - 1}\sum_{j=i+1}^{L_h}\varphi_{ij}(\vec{\xi})$$

Type 7 Constraint: ***Modality***

Using the terminology of the previous case, we can define:

$$\varphi_6(\vec{\xi}) = \frac{1}{L_h}\sum_{i=1}^{L_h} D_i + \frac{2}{L_h(L_h - 1)}\sum_{i=1}^{L_h - 1}\sum_{j=i+1}^{L_h}\varphi_{ij}(\vec{\xi})$$

2. Algorithm Autoclass

AutoClass is an unsupervised Bayesian classification system that seeks a maximum posterior probability classification. Its key features are the following ones:

* it determines the number of classes automatically;
* it can use mixed discrete and real valued data;
* it can handle missing values;
* its processing time is roughly linear in the amount of the data;
* each element has probabilistic class membership;
* it allows correlation between attributes within a class;
* it generates reports describing the classes found; and
* it predicts "test" case class memberships from a "training" classification.

AutoClass uses only vector valued data, in which each instance to be classified is represented by a vector of values, each value characterizing some attribute of the instance. Values can be either real numbers, or they can be discrete. In principle, each attribute represents a measurement of some instance property common to all instances. AutoClass models the data as mixture of conditionally independent classes. Each class is defined in terms of a probability distribution over the meta-space defined by the attributes. AutoClass uses Gaussian distributions over the real valued attributes, and Bernoulli distributions over the discrete attributes. Default class models are also provided.

AutoClass finds the set of classes that is maximally probable with respect to the data and model. In the Bayesian approach to unsupervised classification, the goal is to find the most probable set of class descriptions given the data and prior expectations. The introduction of priors automatically enforces a tradeoff between the fit to the data and the complexity of the class descriptions, giving an automatic form of Occam's razor.

In discussing a probabilistic model, we refer to a probability distribution or density function (pdf) that gives the probability of observing an instance possessing any particular attribute value vector. Ideally, such a modelwould take into account everything known about the processes potentially involved in generating and observing an instance. A Bayes Net relating input and output attributes would be suitable for instances of a well-understood process. For general KDD systems like AutoClass, where little is known about underlying processes, relatively simple statistical models are used.

Probabilistic models invariably contain free parameters, such as Bernoulli probabilities or the Gaussian mean and variance, which must either be fixed or removed (by integration) before instance probabilities can be computed. Thus, it is useful distinguish between the pdf's functional form and its parameter values, and we denote these by T and **p** respectively. S will denote the space of allowed pdf's models and parameters, while I denotes implicit information not specifically represented.

For AutoClass, the fundamental model is the classical finite mixture distribution. This is a two part model. The first gives the interclass mixture probability that an instance X_i is a member of class C_j, independently of anything else we may know of the instance:

$$\Pr\,(X_i \in C_j \mid \mathbf{p}_c, T_c, S, I\,).$$

The interclass pdf T_c is a Bernoulli distribution characterized by the class number J and the probabilities of $\mathbf{p}_{c.}$ Each class C_j is then modeled by a class pdf:

$$\Pr\,(X_i \in C_j \mid \mathbf{p}_j, T_j, S, I\,),$$

giving the probability of observing the instance attribute values X_i conditional on the assumption that instance X_i belongs in class C_j. The class pdf T_j is a product of individual or covariant attribute pdf's T_{jk}, e.g., Bernoulli distributions for nominal attributes, Gaussian densities for real numbers, Poisson distributions for number counts. It is not necessary that the various T_j be identical, only that they all model the same subset of the instance attributes.

AutoClass differs from most other classifiers in that the examples have a degree of membership w.r.t. the classes: they have a weighted assignment, based on the probability of class membership:

$$\Pr\,(\mathbf{X_i}, X_i \in C_j \mid \mathbf{p}_j, T_j, S, I\,).$$

As a practical matter, the weighted assignment approach eliminates the brittle behavior that boundary surface instances can induce in classification systems that decide assignments. More importantly, it allows any user to apply decision rules appropriate to that user's current goals.

Given a set of data X, we seek two things: for any classification pdf T we seek the maximum posterior (map) parameter values **p**, and irrespective ofany **p** we seek the most probable T. Thus there are two levels of search. For any fixed T, specifying the number of classes and their class models, we search the space of allowed parameter values for the maximally probable **p**. This is a real valued space of generally high dimensions, subject to strong constraints between the parameters. There are many local maxima and we have no simple way to determine the global maximum except by generate and test. Thus parameter level search requires an expensive numerical optimization. The model level search involves the number of classes J and alternate class models T_j.

There are several levels of complexity. The basic level involves a single pdf T_j common to all classes, with search over the number of classes. A second level allows the individual T_j to vary from class to class. Model level search is subject to the usual combinatorial explosion of possibilities as attribute number increases, but the Occam factor inherent in the Bayesian approach limits the probability of complex class models for any choice of model and non-delta priors. The model level search retains some of its combinatorial complexity, but with known class memberships we can seek the most probable model for each class individually. The additional information obtained by knowing the class assignments makes it much easier to explore the space of allowed class models, and obtain maximally informative class descriptions.

Autoclass assumes that the data instances X_i are conditionally independent, given the classification pdf **p**, T. Thus any similarity between two instances shiuld be accounted for by their class memberships. Under this assumption the joint data probability is just the product of the individual instance probabilities.

At the classification level, or interclass, model p_c, T_c is the classical Finite Mixture model. This postulates that each instance belongs to one and only one, unknown, member of a set of J classes C_j, with a probability

$$\Pr(X_i \in C_j \mid V_c, T_c, S, I).$$

Note that this probability is independent of the instance attribute vector $\mathbf{X_i}$. In principle, the classes constitute a discrete partitioning of the data, and thus the appropriate pdf is a Bernoulli distribution. Its parameters $p_{\mathbf{c}}$ are a set of probabilities $\{\pi_1, \ldots, \pi_J\}$, constrained to satisfy the conditions $0 \le \pi_j \le 1$ and $\mathrm{Sum}_j\, \pi_j = 1$. Thus we have:

$$\Pr(X_i \in C_j \mid \mathbf{p}_c, T_c, S, I) = \pi_j.$$

Since the Dirichlet (multiple Beta) distribution is conjugate to the Bernoulli, Autoclass uses a uniform minimum information version for the prior probability distribution on the π_j:

$$\Pr(\pi_1, \ldots, \pi_J \mid T_c, S, I) = \Gamma(j+1)/\,\Gamma(1+1/J)^J \,\mathrm{Prod}_j(\pi_j)^{1/J}$$

The MAP parameter estimates for the supervised case, where I_j is the known number of instances assigned to C_j, are then

$$\pi_j^{(est)} = (I_j + 1/J)/(I+1)$$

The instances Xi from each class are assumed to possess attribute vectors $\mathbf{X_i}$ that are independently and identically distributed w.r.t. the class as

$$\Pr(\mathbf{X_i} \mid C_j, \mathbf{p}_j, T_j, S, I).$$

The pdf $\mathbf{p_j}$, T_j thus gives the conditional probability that an instance X_i would have attribute values $\mathbf{X_i}$ if it were known that the instance is a member of class C_j. This class distribution function is a product of distributions modeling conditionally independent attributes k:

$$\Pr(\mathbf{X_i} \mid C_j, p_j, T_j, S, I) = \mathrm{Prod}_k \Pr(X_{ik} \mid X_i \in C_j, \mathbf{p_{jk}}, T_{jk}, S, I)$$

Individual attribute models $\Pr(X_{ik} \mid X_i \in C_j, \mathbf{p_{jk}}, T_{jk}, S, I)$ include the Bernoulli and Poisson distributions, and Gaussian densities.

Combining the interclass and intraclass probabilities, the direct probability that an instance X_i with attribute values $\mathbf{X_i}$ is a member of class C_j is the following:

$$\Pr(\mathbf{X_i}, X_i \in C_j \mid \mathbf{p_j}, T_j, \mathbf{p_c}, T_c, S, I) = \pi_j \,\mathrm{Prod}_k \Pr(X_{ik} \mid X_i \in C_j, \mathbf{p_{jk}}, T_{jk}, S, I)$$

The normalized class membership probability is obtained from this by normalizing over the set of classes. The probability of observing an instance X_i with attribute values $\mathbf{X_i}$, regardless of its class is then:

$$\Pr(\mathbf{X_i} \mid \mathbf{p}, T, S, I) = \mathrm{Sum}_j(\pi_j \,\mathrm{Prod}_k \Pr(X_{ik} \mid X_i \in C_j, \mathbf{p_{jk}}, T_{jk}, S, I))$$

Thus the probability of observing the set X is:

$$\Pr(X \mid \mathbf{p}, T, S, I) = \mathrm{Prod}_i[\mathrm{Sum}_j(\pi_j \,\mathrm{Prod}_k \Pr(X_{ik} \mid X_i \in C_j, \mathbf{p_{jk}}, T_{jk}, S, I))]$$

So far we have only described a classical finite mixture model, which can be converted to a
Bayesian model by introducing priors, at this point only on the parameters, obtaining the joint probability of the data and the parameter values.

AutoClass Attribute Models

Each class model is a product of conditionally independent probability distributions over singleton and/or covariant subsets of the attributes. The only hard constraint is that all class models, used in any classifications that are to be compared, must model the same attribute set. Attributes deemed irrelevant to a particular classification cannot simply be ignored, since this would affect the marginal probabilities. AutoClass provides basic models for simple discrete (nominal) and several types of numerical data. In each case it adopts a minimum or near minimum information prior.

+ Discrete valued attributes : Bernoulli distributions with uniform Dirichlet conjugate prior.
+ Real valued location attributes (spatial locations) : Gaussian densities with either a uniform or Gaussian prior on the means. We use a Jeffreys prior on a singleton attribute's standard deviation, and the inverse Wishart distribution as the variance prior of covariant attribute subsets.
+ Real valued scalar attributes (age, weight) : Log-Gaussian density model.
+ Bounded real valued attributes (probabilities): Gaussian Log-Odds.
+ Integer count valued attributes :Poisson distribution with uniform prior per Loredo.
+ Hierarchical models: this case represents a reorganization of the standard mixture model, where each class is fully independent, to a tree structure, where multiple classes can share one or more model terms.

The Occam Factor

We have several times mentioned an Occam Factor, implying that Bayesian parameter priors can somehow prevent the overfitting that is a problem with maximum likelihood optimization of any kind of probabilistic model. Consider that every single parameter introduced into a Bayesian model brings its own multiplicative prior to the joint probability, which always lowers the marginal. If a parameter fails to raise the marginal by increasing the direct probability by a greater factor than the prior lowers the marginal, we reject the model incorporating that parameter. In the mixture models used by AutoClass, each class requires a full set of attribute model parameters, each with its own prior.

Similar effects limit model complexity within the classes. Simple independent attribute models are usually favored simply because they require fewer parameters than the corresponding covariant models. Both of the foregoing effects are confirmed throughout our experience with AutoClass. For data sets of a few hundred to a few thousand instances, class models with large order covariant terms are generally rated far lower than those combining independent and/or small order covariant terms. We have yet to find a case where the most probable number of classes was not a small fraction of the number of instances classified. Nor have we found a case where the most probable number of model parameters was more than a small fraction of the total number of attribute values. Over fitting simply does not occur when Bayesian mixture models are correctly applied.

Constraints Given by the User

The user may provide constraints similar to those introduced for SEGMENT. In particular:

TYPE 1: *Instances Association*

The instances $x_1, x_2, \ldots, x_n$ must be in the same cluster.

TYPE 2: *Values Association*

The instances that have the same value v for the attribute A must be in the same cluster.

TYPE 3: *Values Separation*

The values distribution of the attribute A must have separated modality in the clusters.

TYPE 4: *Number of clusters*

The number of clusters must be in the interval $[J_1, J_2]$

Constraints of Type 5, 6 and 7 do not have meaning for Autoclass, because each example belongs to all clusters, with different probabilities.

The parameters with respect to which optimization may occur are the a-priori class probability in the Bernoulli inter-class distribution $\Pr(X_i \in C_j \mid \mathbf{p}_c, T_c, S, I)$. Let us denote them by $\pi = \{\pi_1, \ldots, \pi_J\}$. All the other parameters are estimated by AutoClass itself.

In order to apply the criteria of optimization, we have to re-interpret "belonging" to a cluster as "having the maximum probability" w.r.t. that cluster.

TYPE 1: *Instances Association*

Let X_1 and X_2 be two instances that have to belong to the same cluster. Let

$$\Pr(X_1 \in C_j \mid \mathbf{p}_j, T_j, S, I) \quad \text{and} \quad \Pr(X_2 \in C_j \mid \mathbf{p}_j, T_j, S, I)$$

be the corresponding probability sets ($1 \le j \le J$).

Let $j_1^* = \text{Arg Max}_j \Pr(X_1 \in C_j \mid \mathbf{p}_j, T_j, S, I)$ and $j_2^* = \text{Arg Max}_j \Pr(X_2 \in C_j \mid \mathbf{p}_j, T_j, S, I)$. It must be: $j_1^* = j_2^*$. The evaluation function will be:

$$\delta_1(V) = \textbf{If } (j_1^* = j_2^*) \textbf{ Then } 1, \textbf{ Else } 0.$$

If r instances must be in the same cluster, and only s of them actually are, then:

$$\delta_1(V) = \frac{s}{r}$$

Membership is evaluated according to the probabilities.

Type 2 Constraint: ***Value association***

Let $\tau_{h,j}$ be the number of instances with value v_j for attribute A_h that are in the same cluster (in this case, for which the probability of belonging to the cluster is maximum for the class), and let $S(A_h, v_j)$ be the total number of instances with value v_j for attribute A_h. Then :

$$\delta_2(V) = \frac{\tau_{h,j}}{S(A_h, v_j)}$$

The function $\delta_2(V)$ is 1 when all examples with value v_j for attribute A_h are in the same cluster. The minimum value of $\delta_2(V)$ is reached when the maximum number of examples in the same cluster is minimum, i.e., when the examples are uniformly subdivided among clusters. Then :

$$\left\lceil \frac{S(A_h, v_j)}{K} \right\rceil \le \delta_2(V) \le 1$$

In order to evaluate $\tau_{h,j}$, we must decompose $\Pr(X_1 \in C_j \mid \mathbf{p}_j, T_j, S, I)$ as follows:

$$\Pr(X \in C_j \mid A_1 = v_1, \ldots, A_h = v_j, \ldots, A_H = v_H\} = \prod_h [\Pr\{A_h = v_j \mid X \in C_j\}]\, \pi_j / \prod_h [\Pr\{A_h = v_j\}]$$

The constraint will be satisfied when the above probability reaches the maximum for the same value whenever $A_h = v_j$.

TYPE 3: *Values Separation*

This constraint is the complementary of the previous one. The formalization is the same, but the constraint is satisfied whenever the membership reaches its maximum for different classes when the values of attribute A_h are different.

Let C be a clustering, containing K clusters $\mathbf{C}_i$ ($1 \le i \le K$) with the hystogram $\vec{\tau}^{(i)}(A_h)$. Let us define:

$$\delta^{(i)} = 1 - 2 \cdot \frac{\sqrt{\mathrm{VAR}\ (\vec{\tau}^{(i)}(A_h))}}{\mathrm{R}\ (\vec{\tau}^{(i)}(A_h))} \quad \text{for continuous-valued attributes}$$

$$\delta^{(i)} = 1 - \frac{m_i}{L_h} \quad \text{for discrete attributes}$$

where m_i is the number of different values of A_h in C_i. Then:

$$\delta_3(V) = \frac{1}{K} \sum_{i=1}^{K} \delta^{(i)}$$

The evaluation function $\delta_3(V)$ is equal to 1 when all $\delta^{(i)}$ are 1, i.e., when all the $\mathrm{VAR}(\vec{\tau}^{(i)}(A_h))$ are equal to zero. The minimum $\delta_3(V)$ is reached when all $\delta^{(i)}$ are zero.

Type 4 Constraint: ***Number of clusters***

This constraint specifies that the number of clusters must be in the interval $[J_1, J_2]$. As for SEGMENT, let us define the function:

$$\sigma(x, a, b) = (x - a)(b - x)$$

Then:

$$\delta_4(V) = \sigma(J, J_1, J_2)$$

Objective Function to Optimize

For AutoClass, the variables ξ_h directly correspond to the π_j, because they are already normalized to 1. For the rest, the functions to optimize are the same as for SEGMENT, provided that the above defined δ functions are substituted.

3. CART

In order to see whether the proposed methodology is also applicable to classification algorithm, we have considered Cart. Cart is an algorithm for building decision trees. In the learning sample Lfor the J-class problem, let N_j be the number of cases in class j. Often the prior probabilities $\{\pi(j)\}$ are taken to be the proportions $\{N_j/N\}$. If they are unknown, the set of priors $\{\pi(j)\}$ are either estimated from the data as $\{N_j/N\}$ or supplied by the analyst.

In a node t, let N(t) be the total number of cases in L with $x_n \in t$, and $N_j(t)$ the number of class j cases in t. The proportion of the class j cases in L falling into t is $N_j(t)/N_j$. For a given set of priors, $\pi(j)$ is interpreted as the probability that a class j case will be presented to the tree. Therefore, we take

$$p(j, t) = \pi(j)\, N_j(t)/N_j$$

as the resubstitution estimate for the probability that a case will both be in class j and fall into node t. The resubstitution estimate of the probability that a case is in class j, given that it falls into node t, is given by

$$p(j|t) = p(j, t)/p(t)$$

and satisfies:

$$\mathrm{Sum}_j\{p(j|t)\} = 1$$

When $\{\pi(j)\} = \{N_j/N\}$ then $p(j|t) = N_j(t)/N(t)$, so the $p(j|t)\}$are the relative proportions of class j cases in node t.

The four elements needed for the tree growing procedure are:

1. A set Q of binary questions of the form (Is $x \in A$?).
2. A goodness of split criterion $\Phi(s, t)$ that can be evaluated for any split s of any node t

3. A stop-splitting rule
4. 4. A rule for assigning every terminal node to a class.

The set Q of binary questions generates a set S of splits s of every node t. Those cases in t answering "yes" go to left descendant node t_L and those answering "no" to the right descendant t_R. At each intermediate node t, the split selected is that split s* that maximizes $\Phi(s, t)$.

If the data have standard structure, the class Q of questions can be standardized. Assume that the measurement vectors have the form:

$$\mathbf{x} = (x_1, \ldots, x_H)$$

where H is the fixed dimensionality and the variables $x_1, \ldots, x_H$ can be a mixture of ordered and categorical types.The standardized set of questions Q is defined as follows:

1. Each split depends on the value of only a single variable

2. For each ordered variable x_m, Q includes all questions of the form:

$$\{\text{Is } x_m \leq c?\} \qquad \text{for all c.}$$

3. If x_m is categorical, taking values in $\{b_1, \ldots, b_V\}$, then Q includes all questions of the form:

$$(\text{Is } x_m \in S?),$$

   for all S subset of $\{b_1, \ldots, b_V\}$.

The splits in items 2 and 3 for all H variables constitute the standardized set.

At each node the tree algorithm searches through the variables one by one, beginning with x_l and continuing up to x_H. For each variable it finds the best split. Then it compares the H best single variable splits and selects the best of the best. The computer program CART incorporates this standardized set of splits. Since most of the problems ordinarily encountered have a standard data structure, it has become a flexible and widely used tool. When fixed-dimensional data have only ordered variables, another way of looking at the tree structured procedure is as a recursive partitioning of the data space into rectangles.

The goodness of split criterion was originally derived from an *impurity* function. An impurity function is a function Φ defined on the set of all J-tuples of numbers $(p_1, \ldots, p_J)$ satisfying $p_1 \geq 0$ and normalized to 1, satisfying the properties:

(i) Φ is a maximum only at the point $\{1/J, 1/J, \ldots, 1/J\}$
(ii) Φ achieves its minimum only at the points (1, 0, ..., 0), (0,1,0,…0), … (0, 0, …, 1)
(iii) Φ is a symmetric function of $p_1, \ldots, p_J$

Given an impurity function $p_1, \ldots, p_J$, define the impurity measure i(t) of any node t as

$$i(t) = \Phi(p(1|t), p(2|t), \ldots, p(J|t))$$

When a node is split and its elements go to the left and right son nodes, t_L and t_R, we define a decrease in impurity as:

$$\Delta i(s,t) = i(t) - p_R\, i(t_R) - p_L\, i(t_L),$$

being p_R and p_L the proportions of examples that went to t_R and t_L, respectively.
Then take the goodness of split $\Phi(a, t)$ to be $\Delta i(s,t)$.

Parameter Optimization

The problem with classification algorithms is that they already optimize a function Ψ, which is the prediction error. Then, in order to avoid overfitting, the function Φ must not be related to the classification error. Then, the user can only specify constraints related to other aspects of the learned decision tree. One possibility is:

1. To set limits on the tree's depth
2. To set limits on the total number of nodes

Obviously, it may not be possible to satisfy the user's constraints without increase the classification error. In this case, the procedure may serve as a test for the user's intuition.

Another way of using the optimization module is to identify some examples that cannot be misclassified. In other words, examples may be weighted and the weight can be optimized in such a way that the overall error of classification does not increase, yet satisfying the user's constraints. This can be done by extending the idea of cost-sensitive classification, in which each example is weighted singularly, instead of considering only a cost fore the false positives and a cost for the false negative examples.

As a conclusion, we can say that the parameter optimization technique is best suited to clustering algorithms and that, for these, it is mainly independent from the specific algorithm used. For classification algorithms, the best use is for weighting examples, allowing to specify the severity of misclassifying any single example.