A Distributed Graph Based Approach for Rough Classifications Considering Dominance Relations Between Overlapping Classes
Khalil Laghmari, Mohammed Ramdani, Christophe Marsala

To cite this version:

HAL Id: hal-01303006
https://hal.sorbonne-universite.fr/hal-01303006
Submitted on 15 Apr 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Abstract—Several data from real world applications involves overlapping classes. Data is allowed to belong to multiple classes with different membership degrees. In this paper, we explore a different concept characterizing social networks, documents, and most of biological and chemical datasets: data could have multiple classes, but dominant classes are better noticed than dominated classes. For example, a document could discuss economy and politics, but it would be more focused on politics. A molecule could have multiple odors, but experts could notice some odors better than others. We are interested in this type of data, where a dominance relation exists between classes. Experts could easily make mistakes because dominated classes are hardly noticed. Data incoherence is a serious problem but not the only one. There is too much irrelevant and redundant attributes. Unfortunately this increases the computational time of generating classifiers. Our first challenge is to find an adapted model to overlapping classes considering dominance relations. The second challenge is to find the most relevant attributes. Finally the third challenge is to ensure that the approach gives results in an acceptable time. We address those challenges by taking advantage of the rough set theory, which is suited for incoherent data and allows multiple classes and attributes selection. The proposed approach works in a parallel and decentralized way to reduce the computational time. We tested it on real chemical data and the collected results are very promising.

I. INTRODUCTION

Classification is a supervised machine learning method where data labels are already known. The process uses a training dataset as input, generates a classifier and then predicts a class label for each unlabeled data. A test dataset is used to compute the classifier accuracy and to validate the predictive model. Sometimes data could have more than one label, this problem is known as overlapping classification. In this paper we first discuss the state of the art for overlapping classification in Section II, then we present our proposed approach in Section III, and finally we discuss our experiment results in Section IV.

II. STATE OF ART

In the following, we first discuss mathematical tools applied to overlapping classifications, then we discuss some algorithms designed for dimensionality reduction. Finally we present some tools designed for distributed data classification as a solution to reduce the computational time.

A. The overlapping problem

Building a good classifier is the objective of the classification process. Traditional classifiers work well with simple data with only one possible class, but they produce bad results for complex data with multiple possible classes. The fuzzy sets theory developed by [1] is a popular and a powerful tool for solving the classes overlapping problem. By means of a membership function, data is allowed to be assigned to all classes with different membership degrees that range between 0 and 1. In recent work [2], a new neuro fuzzy-classification technique is proposed. Experiments proved that this approach produces better results than Radial Basis Function Neural Network [3] or Adaptive Neuro-fuzzy Inference System [4] algorithms. However, even with medium data sizes $n$, if the size $k$ of possible classes is large, then setting and transposing a $n \times k$ matrix would be very time consuming. The rough sets theory (RST) developed by [5] is a good choice when the objective is reducing the computational time. The RST-based approaches are usually easy to distribute. Each class is defined by means of upper and lower approximations. Depending on the classification certainty, a data is allowed to belong either to exactly the lower approximation of a single class, or to the upper approximation of multiple classes. The RST covers many application domains, especially biology and medicine: In [6] a RST-method is proposed to generate classification rules for Breast cancer data, and more recently in [7] a RST-method has been applied to generate classification rules for Hepatitis C virus dataset. Mixture model approaches are an interesting alternative to fuzzy set and rough set theories. Each class is represented by a distribution. Each data is assumed to be generated by one or more distributions. The objective is to find the relevant parameter values to suit distributions as much as possible. The Expectation-Minimization (EM) [8] algorithm is often used to determine those values.
Gaussian mixture models is proposed to detect overlapped speeches, and the proposed approach was successfully tested on artificial and real datasets. A less flexible alternative could be hierarchical algorithms such as the pyramidal classification [10], the week and the k-week hierarchies [11]. The pyramidal classification allows overlaps only between adjacent classes. The week hierarchies assumes that the intersection of 3 classes is equal to the intersection of just 2 of them. The k-week hierarchies are a generalization of the week hierarchies. They assume that the intersection of k+1 classes is equal to the intersection of just k of them. Those hierarchical classification algorithms reduce clearly the solutions search space. Unfortunately, without any preliminary knowledge about data we could not decide if k-week hierarchies could be adapted to our problem or not. We decided to use rough sets because no preliminary knowledge about data is required, and because the classification could easily be distributed to reduce the computational time.

B. The attributes selection problem

Rough set approaches are based on a discernibility relation. Each minimal attributes subset discerning data is called a reduct. Reducts computation could be done by generating a discernibility matrix and simplifying the extracted boolean function, or by using the concept of positive regions to add discernibility. A less flexible alternative could be hierarchical algorithms such as the pyramidal classification on artificial and real datasets. A less flexible alternative could be the right choice for our problem. It’s a graph database server designed for big data. A graph database is easier to distribute than a relational database because there is no relational constraints to care about. A graph database has also the advantage of an easy generation of decision trees, and a simple modeling of dominance relations by means of edges. This motivates us to build a distributed graph based approach using the RST.

III. THE PROPOSED ROUGH CLASSIFICATION APPROACH

In this section, we introduce a mathematical formulation of our problem and we review the necessary mathematical foundations of the RST, then we present our proposed approach.

A. Problem formulation and preliminary knowledge

Let \( M = \{m_1, \ldots, m_x\} \) be the set of available machines, \( A = \{a_1, \ldots, a_p\} \) be the set of attributes, \( (g_j)_{1 \leq j \leq p} \) be the number of possible values for each attribute \( a_j \), \( (W_j = \{w_{j1}, w_{j2}\})_{1 \leq j \leq p} \) be the set of possible values for each attribute \( a_j \), \( C = \{c_1, \ldots, c_l\} \) be the set of classes, \( X \) be the set of data, \( X_t = \{x_1, \ldots, x_n\} \subset X \) be the training set, and \( X_v = X - X_t \) be the validation set. Each data \( x_i \) is described by values \( (v_{i1}, \ldots, v_{ip}, c_{i1}, \ldots, c_{im}) \), \( v_{ij} \in W_j \forall j \in [1, p] \), \( c_{ih} \in C \forall h \in [1, s] \). A data \( x_i \) could belong to one or more classes from \( C \) according to different ranks from \( R = \{r_1, \ldots, r_s\} \). A class at a rank \( r_h \) dominates its successor at the rank \( r_{h+1} \). The dominance relation gives us information about classes which never coexist, classes always together, strong (dominant) and weak (dominated) classes. Our objective is to find out simple classification rules, taking into consideration the dominance relation, and using the minimal subset of needed attributes.

A rough class is defined by means of upper and lower approximations. The set of all data which can be classified with certainty as belonging to a class \( c_h \) is called the lower approximation of \( c_h \), denoted by \( c_{h*} \). The set of all data which can be possibility classified as belonging to \( c_h \) is called the upper approximation of \( c_h \) denoted by \( c_{h}^* \). The set \( c_{h}^* = c_{h}^* - c_{h*} \) containing elements which cannot be classified neither as members of \( c_h \) nor as members of \( r_h \) is called the boundary region of \( c_h \). The class \( c_h \) is crisp if \( c_{h} = \emptyset \), otherwise \( c_h \) is rough. A rough class can be described by means of some coefficients:

- Quality coefficient of the lower approximation: \( \alpha(c_h) = |c_{h*}|/|c_h| \) where \(|.|\) denotes cardinality.
- Quality coefficient of the upper approximation: \( \beta(c_h) = |c_h^*|/|c_{h*}| \).
- Imprecision coefficient: \( \gamma(c_h) = |c_{h*}|/|c_h^*| = \alpha(c_h) \times \beta(c_h) \).

Let \( X_h = \{x_i \in X_t, c_{ih} = c_l\} \) be the set of data having the class label \( c_l \) at the rank \( r_h \). In order to discern for a rank \( r_h \), a class \( c_l \) from all other classes using a RST-approach, we could compute a symmetric \(|X_t - X_h| \times |X_h|\) matrix \( D_h \), called the discernibility matrix. Where \( D_{ij} = \{a_k \in A, v_{ih} \neq v_{jk}\} \) contains the set of all attributes that could be used to discern data \( x_i \in X_t - X_h \) from data \( x_j \in X_h \). Once the discernibility matrix is constructed, we could extract the called discernibility function \( f = \land (\lor (a_k \in m_{ij} (a_k))) \), where \( \land \) and \( \lor \) denotes the conjunction and the disjunction operators respectively. Each possible simplification of \( f \) is called a reduct. The intersection of all reducts is called the core. A reduct represent a minimal attributes set to discern two datasets. It is used to generate rules based on attributes values. For a specific reduct, a rule for a rank \( r_h \) is in the form: if \( (\land (a_j = v_j)) \) then \( c_l \).
The set of all rules defines the classifier we are looking for. In our proposed approach we generate a decision tree, and since generating all reducts is a NP Hard problem, we generate a single subset of attributes, with good chances that it could be a reduct. We give the needed details in III-B.

B. The proposed approach

Our proposed approach includes 10 steps that are summarized in (Fig. 1). First, the Main Server (M.S) read the initial data (Fig. 1 step 1) and generates the training \((X_t)\) and validation \((X_v)\) sets (Fig. 1 step 2). Next, each DataBase Server (DB.S) receives a subset of the training set (Fig. 1 step 3) and do some preprocessing tasks, like generating new attributes and discretizing values (Fig. 1 step 4). Each DB.S generates a graph from the prepared data (Fig. 1 step 5). The graph database includes data and classes nodes, and two types of edges: Edges to link data with noticed classes, and edges to link dominant classes with dominated classes. Each DB.S generates discernibility edges between specific data pairs (Fig. 1 step 6). Classes discernibility edges (Fig. 1 step 7) and the decision tree (Fig. 1 step 8) are then generated for each rank \(r_h \in R\). Finally The M.S generates the global decision tree (Fig. 1 step 9) and validates the model using the validation set (Fig. 1 step 10). Details are given for main steps below.

1) Training set and validation set selection (step 2): In order to build a good classifier without need to a cross validation step, the proposed approach try to generate a training set \(X_t\) containing all classes arrangements \((c_{i1},...,c_{ik}), i \in [1,N]\) existing in the initial data set \(X\). The main server count how many times each arrangement appears in the initial data. The parameter \(\lambda = |X_v|/|X|\) defines how many data with a specific classes arrangement should go to the validation set \(X_v\). We start the selection process from the most frequent arrangement, and we stop when \(X_v\) reaches the wanted size.

2) Generating Data discernibility edges (step 6): In order to quickly count how many times each attribute was used to discern data, we generate for each data pair \((x_i,x_i')\) a vector \(b_{i'} = (b_{i'1},...,b_{i'p})\) of booleans instead of a set of selected attributes. The vector \(b_{i'}\) (Fig. 2 step 6) is defined as follows: \(\forall j \in [1,p] b_{ij'} = 1\) if \(v_{ij} \neq v_{ij'}\), and \(b_{ij'} = 0\) if not.

3) Generating class discernibility edges (step 7): Some classes could never be noticed at a rank \(r_h\). Also some classes could be noticed only if another class is noticed before at rank \(r_{h-1}\). Considering this, we generate classes discernibility edges (Fig. 2 step 7) between classes for a given \(r_h\) and a given class \(c_f\) at \(r_{h-1}\). \(c_f^h = X^{h,l} \cap X^{h,l'}\) denotes the set of data having the class \(c_l\) at the rank \(r_h\) and the class \(c_f\) at the rank \(r_{h-1}\). For each pair \((c_l^h,c_f^h)\), we count how many times an attribute \(a_j\) was used to discern data belonging to \(c_f^h\) from data belonging to \(c_l^h\). The following vectors are then generated for each DB.S:

- \((\omega_{c_l^h, c_f^h})_{1\leq j \leq p} = (\sum_{i=1}^{n-1} \sum_{v' = i+1}^{n} b_{i'v})_{1\leq j \leq p}\) \(z_i \in c_l^h, x'_{v} \in c_f^h\)
- \((\omega_{c_f^h})_{1\leq j \leq p} = (\sum_{i=1}^{n} b_{i'v})_{1\leq j \leq p}\)

4) Generating local decision tree (step 8): The M.S collects \(\omega^h\) vectors from all \(z\) DB.S (Fig. 2 step 9). \(\omega^h(d)\) denotes the vector \(\omega^h\) generated by the DB.S \(d\). The M.S generates \((\Omega^h)_{1\leq h \leq s}\) vectors as follows: \((\Omega^h)_{1\leq h \leq s} = (\sum_{i=1}^{d} \omega^h(d))_{1\leq j \leq p}\). The most used attribute \(\hat{a}_j^h\) to discern data at the rank \(r_h\) corresponds to the attribute with the highest value in the \(\Omega^h\) vector: \(\hat{a}_j^h = \arg\max \{\max_{1\leq j \leq p} (\Omega^h)_{1\leq j \leq p}\}\). Usually we need to select more than one attribute to discern data and classes. Let \(A^h\) be the set of selected attributes at \(r_h\), and \(f_h = \{ j \in [1,p], a_j \in A^h \}\) be the set of corresponding indexes.

We then create a node \(F^{h,f} (\text{Fig. 2 step 8})\) with parameters:

- \(r_h\): current rank.
- \(c_f\): noticed class at the rank \(r_{h-1}\).
- \(\hat{a}_j^h\): the set of data to discern.
- \(L_{h,f} = \{l \in [1,k], c_l^h \neq \emptyset\}\): Indexes corresponding to possible classes at the rank \(r_h\) considering the class \(c_f\) at the rank \(r_{h-1}\).

\(F\) nodes help us at the aggregation step and they allow us to evaluate some coefficients at the validation step. Referenced classes in \(L_{h,f}^j\) should be discerned using the selected attribute \(\hat{a}_j^h\). For each value \(v_{je}1 \leq e \leq q_v^h\) we create a node \(F^{h,f,w_{je}}\) with the parameters described below:

- \(c_l^{h,w_{je}} = \{ x_i \in c_l^h, v_{ij} = w_{je}\}\): The set of data to discern at the node \(F^{h,f,w_{je}}\).
- \(L_{h,f,w_{je}} = \{l \in [1,k], c_l^{h,w_{je}} \neq \emptyset\}\): Indexes corresponding to the set of classes to discern at \(F^{h,f,w_{je}}\).

Next, for each node \(F^{h,f,w_{je}}\), we repeat our method from step 7 (Fig. 2 step 7) using \(F^{h,f,w_{je}}\) parameters instead of \(F^{h,f}\) parameters. Each time we select a new attribute \(\hat{a}_j^h\) to discern data and classes, we add \(\hat{a}_j^h\) to \(A^h\). We stop when classes referenced by \(F^{h,f,w_{je}}\) cannot be discerned using the set of available attributes \((\max (\Omega^h) = 0)\).

Usually in a decision tree, classes are discerned by a selected attribute values at each step, until only isolated classes are kept at the bottom of the decision tree. The proposed approach is different because if only one decision tree couldn’t discern a class \((c_l)_{1\leq l \leq k}\) using the current subset of selected attributes, then all decision trees should be extended by selecting an additional attribute, even if some of decision trees could discern \(c_l\) without the additional attribute. This redundancy is important to facilitate the global decision tree generation.

A node \(F^{h,f,w_{je}}\) on which we stop denoted by \(F^{h+1,f,w_{je}}\) represent a rule node. We create edges to all classes referenced by each rule node \((F^{h+1,f,w_{je}})_{1 \leq l \leq q_v^h}\). If a rule node references exactly one class, then it represents a crisp rule. Otherwise it represents a rough rule. Finally we link each class \(c_f\) where \(f' \in E^{h,f,w_{je}}\) with a new node \(F^{h+1,f'}\), and we repeat our
method from step7 starting at $F^{h+1}\cdot f'$ instead of $F^{h-f}$. We stop if one of those conditions holds:

- $h = s$: We reached the last rank.
- $f' = 0$: $c_f'$ is the empty class, and then $c_0$ is the corresponding class for all next ranks ($r_{hi}\leq h'\leq s$).

5) Generating global decision trees (step 9): After generating all local decision trees using the same attributes for each rank, we generate the global decision tree as follows:

- We generate global $F$ nodes: $F^u_Gh,f,w_{je}$ taking into consideration all data to discern from all $z$ DB.S: $c^u_G^{h,f,w_{je}} = \bigcup_{1\leq d\leq z} c^{h,f,w_{je}}(d)$, and all classes to discern from all $z$ DB.S: $L^u_G^{h,f,w_{je}} = \bigcup_{1\leq d\leq z} L^{h,f,w_{je}}(d)$.

- We copy all edges linking $F$ nodes from the $z$ local decision trees (without duplicates) to the global generated decision tree.

A partial view of 2 local decision trees is shown in (Fig. 3). The resulting global decision tree is shown in (Fig. 4).

6) Model validation (step10): Some classes could overlap with too much other classes at different ranks. This make the decision tree larger and less efficient. A solution could be to build another classifier for those classes with higher overlap degrees. The overlap parameter $\theta(c_l)$ for a class $c_l$ represents the proportion of classes that could overlap with $c_l$. We introduce the parameters defined below:
Finally, the rough coefficients described in III-A are computed.

Next, we let the global decision tree decide classes for data.

The parameter $X$ is computed on each $DBS$.

The $M.S$ aggregate the previous parameters as follows:

$$\hat{L}_{cl} = \bigcup_{1 \leq l \leq s, 1 \leq h \leq s} \hat{L}_{cl}(d).$$

The overlap parameter $\theta(c_l)$ is then defined as: $\theta(c_l) = |\hat{L}_{cl}|/|C|$. The lower and the upper approximations of each class $c_l$ are computed on each $DBS$ as follows:

$$c_{l}^{\prime}(d) = \bigcup_{1 \leq h \leq s, 1 \leq j \leq h} \bigcup_{1 \leq e \leq q_j} \hat{L}_{h,f,w_{ej}} \cap c_{l}^{f}, 2 \leq |\hat{L}_{h,f,w_{ej}}|.$$ $c_{l}^{\prime}(d) = X^{h,l} - c_{l}^{\prime}(d).$

$c_{l}^{\ast}(d) = c_{l}^{\ast}(d) \cup c_{l}^{\prime}(d).$

The $M.S$ aggregate the previous parameters as follows:

$c_{l}^{\ast} = \bigcup_{1 \leq h \leq s} c_{l}^{\ast}(d), c_{l}^{\ast} = \bigcap_{1 \leq h \leq s} c_{l}(d), c_{l}^{\ast} = c_{l}^{\ast} \cup c_{l}^{\prime}.$

Finally, the rough coefficients described in III-A are computed.

Next, we let the global decision tree decide classes for data.

IV. EXPERIMENTS

The proposed approach has been experimented on molecules data from [20] (TABLE 1). 2000 molecules were used as a learning dataset and 400 molecules were used to validate the model. Each molecule could have one or more odors. The maximum number of odors is 6 per molecule.

### TABLE I: Molecules dataset: A view of initial data

<table>
<thead>
<tr>
<th>data</th>
<th>$r_1$</th>
<th>$r_2$</th>
<th>$r_3$</th>
<th>$r_4$</th>
<th>$r_5$</th>
<th>$r_6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$x_1$</td>
<td>HERBAC</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_2$</td>
<td>FLORAL</td>
<td>FRUITY</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_3$</td>
<td>FRUITY</td>
<td>TEA</td>
<td>HERBAC</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$x_4$</td>
<td>BALSAM</td>
<td>FRUITY</td>
<td>FLORAL</td>
<td>ORANG</td>
<td>JASMIN</td>
<td></td>
</tr>
<tr>
<td>$x_5$</td>
<td>HERBAL</td>
<td>FRUITY</td>
<td>FLORAL</td>
<td>ORANG</td>
<td>JASMIN</td>
<td></td>
</tr>
<tr>
<td>$x_6$</td>
<td>FRUITY</td>
<td>ROSE</td>
<td>BALSAM</td>
<td>WINEY</td>
<td>LEAFY</td>
<td>FLORAL</td>
</tr>
</tbody>
</table>

The set of classes which overlap at rules nodes with a class $c_l$ considering only the $DBS$ $d$:

$$\hat{L}_{cl}(d) = \bigcup_{1 \leq h \leq s, 1 \leq f \leq k} \bigcup_{1 \leq j \leq h} \bigcup_{1 \leq e \leq q_j} \{\hat{L}_{h,f,w_{ej}} \cap \{l\}, l \in h,f,w, 2 \leq |\hat{L}_{h,f,w_{ej}}|\}.$$ The parameter $\hat{L}_{cl}$ considering all the $z$ DBS:

$$\hat{L}_{cl} = \bigcup_{1 \leq d \leq z} \hat{L}_{cl}(d).$$

$X^{h,l} = \bigcup_{1 \leq h \leq s} X^{h,l}$; The set of data having the class $c_l$ at any rank $(r_h)_{1 \leq h \leq s}$.

The overlap parameter $\theta(c_l)$ is then defined as: $\theta(c_l) = |\hat{L}_{cl}|/|C|$. The lower and the upper approximations of each class $c_l$ are computed on each $DBS$ as follows:

$$c_{l}^{\prime}(d) = \bigcup_{1 \leq h \leq s, 1 \leq j \leq h} \bigcup_{1 \leq e \leq q_j} \{\hat{L}_{h,f,w_{ej}} \cap c_{l}^{f}, 2 \leq |\hat{L}_{h,f,w_{ej}}|\}.$$ $c_{l}^{\ast}(d) = X^{h,l} - c_{l}^{\prime}(d).$

$c_{l}^{\ast}(d) = c_{l}^{\ast}(d) \cup c_{l}^{\prime}(d).$

The $M.S$ aggregate the previous parameters as follows:

$c_{l}^{\ast} = \bigcup_{1 \leq d \leq z} c_{l}^{\ast}(d), c_{l}^{\ast} = \bigcap_{1 \leq d \leq z} c_{l}(d), c_{l}^{\ast} = c_{l}^{\ast} \cup c_{l}^{\prime}.$

Finally, the rough coefficients described in III-A are computed.

### TABLE II: Selected attributes

<table>
<thead>
<tr>
<th>rank</th>
<th>rules</th>
<th>crisp rules</th>
<th>rough rules</th>
<th>avg rough classes per rule</th>
<th>validation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1627</td>
<td>412</td>
<td>1215</td>
<td>4.15</td>
<td>100%</td>
</tr>
<tr>
<td>2</td>
<td>1913</td>
<td>408</td>
<td>1505</td>
<td>5.07</td>
<td>32%</td>
</tr>
<tr>
<td>3</td>
<td>1609</td>
<td>337</td>
<td>1332</td>
<td>4.83</td>
<td>23%</td>
</tr>
<tr>
<td>4</td>
<td>1017</td>
<td>240</td>
<td>777</td>
<td>4.18</td>
<td>56%</td>
</tr>
<tr>
<td>5</td>
<td>413</td>
<td>128</td>
<td>288</td>
<td>3.32</td>
<td>87%</td>
</tr>
<tr>
<td>6</td>
<td>93</td>
<td>53</td>
<td>42</td>
<td>2.62</td>
<td>98%</td>
</tr>
</tbody>
</table>

### TABLE III: Rules validation

Using a local network with 1 server and 10 similar computers, we could sample data to 4 samples of 500 elements. 4 computers were used to store and compute discernibility inside each sample. The left 6 computers were used to compute discernibility between elements from each 2 different samples. Unfortunately initial data don’t contain descriptive attributes, but we could generate 15 attributes from only the name and formula of each molecule. Among them there is a boolean attribute determining if a molecule is cyclic or not, frequencies of some atoms, and the molecular weight. Numerical attributes were discretized to have less than 10 equal range values. Since we didn’t address the discretization problem, this is the simplest discretization strategy for us even if it’s not the best one.

At the attributes selection step, only 2 attributes were discarded, and a considerable number of data pairs could not

<table>
<thead>
<tr>
<th>class</th>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\gamma$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>GREEN</td>
<td>0.21</td>
<td>0.21</td>
<td>0.04</td>
<td>0.85</td>
</tr>
<tr>
<td>FRUITY</td>
<td>0.26</td>
<td>0.23</td>
<td>0.06</td>
<td>0.78</td>
</tr>
<tr>
<td>HERBAC</td>
<td>0.24</td>
<td>0.22</td>
<td>0.05</td>
<td>0.78</td>
</tr>
<tr>
<td>MOSSY</td>
<td>0.5</td>
<td>0.36</td>
<td>0.18</td>
<td>0.1</td>
</tr>
<tr>
<td>SULFUR</td>
<td>0.9</td>
<td>0.84</td>
<td>0.76</td>
<td>0.06</td>
</tr>
</tbody>
</table>

### TABLE IV: Classes evaluation

<table>
<thead>
<tr>
<th>rank</th>
<th>classes</th>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\gamma$</th>
<th>$\theta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$r_1$</td>
<td>54</td>
<td>0.24</td>
<td>0.23</td>
<td>0.08</td>
<td>0.45</td>
</tr>
<tr>
<td>$r_2$</td>
<td>70</td>
<td>0.75</td>
<td>0.59</td>
<td>0.48</td>
<td>0.30</td>
</tr>
<tr>
<td>$r_4$</td>
<td>72</td>
<td>0.71</td>
<td>0.72</td>
<td>0.54</td>
<td>0.23</td>
</tr>
<tr>
<td>$r_4$</td>
<td>57</td>
<td>0.79</td>
<td>0.83</td>
<td>0.69</td>
<td>0.12</td>
</tr>
<tr>
<td>$r_5$</td>
<td>25</td>
<td>0.92</td>
<td>0.94</td>
<td>0.88</td>
<td>0.04</td>
</tr>
<tr>
<td>$r_6$</td>
<td>5</td>
<td>0.97</td>
<td>0.97</td>
<td>0.94</td>
<td>0.16</td>
</tr>
</tbody>
</table>

### TABLE V: Classes evaluation by rank
be discerned using the selected attributes (TABLE II). The attributes we could generate were not relevant. We obtained too much rough rules (Table III). Some classes overlap with almost all other classes ($\theta \uparrow$) by means of many data ($\beta \downarrow$) (TABLE IV). The results is that classes could not be well discerned and the average of classes imprecision coefficient $\gamma$ was very low (TABLE V). By using both the selected attributes and the information about the class in the previous rank, we could reduce the average of rough classes per rule(TABLE III).

The classifier could always find the correct class for the first rank, because even if classes at $r_1$ are rough in a high degree, experts don’t make mistakes while noticing them. Noticed classes at $r_1$ are correct. The difficult task for experts is to determine classes at next ranks. The giving classes could be wrong. That explain why the accuracy of ranks $r_2$ and $r_3$ is low. The accuracy of ranks $(r_h)_{4 \leq h \leq 6}$ is higher because there is not too much data having more than 4 classes.

In order to evaluate the computational time gain of the proposed approach, we run it on a single machine in a sequential and centralized way, and we turned it on similar 11 machines (10 Database servers + 1 main server). The generated global decision tree by aggregation of local decision trees and the generated decision tree by the centralized method were similar. TABLE VI shows that the distributed approach could save 95, 6% of the centralized approach computational time. This is because except some few simple operations, all operations in the proposed approach could be executed in a parallel and decentralized way.

V. CONCLUSION AND FUTURE WORKS

In this paper, an adapted framework to rough classifications with dominance relations is proposed. The proposed approach work on a distributed and parallel way, and as shown in experiments, the computational time is clearly reduced. Using Neo4j server, we could build a special decision tree adapted to our problem definition. Experiment on a real chemical dataset shows that thanks to dominance relations, our model can find the correct classification among some few other possible classifications. In our future works, we will try to reduce the number of needed attributes, by integrating the positive region concept to our model. We also plan to find a good discretization strategy adapted to our problem. Finally, more accurate results would be obtained by generating more descriptive attributes, and by aggregating decision trees corresponding to each rank. The aggregation should be weighted because decision trees at lower ranks are more accurate.

<table>
<thead>
<tr>
<th>Time (min)</th>
<th>Sequential and centralized method</th>
<th>Distributed and parallel method</th>
<th>Saved time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data discernibility</td>
<td>1844</td>
<td>188</td>
<td>97%</td>
</tr>
<tr>
<td>Class discernibility</td>
<td>1815</td>
<td>53</td>
<td>94%</td>
</tr>
<tr>
<td>Total</td>
<td>3659</td>
<td>161</td>
<td>95.6%</td>
</tr>
</tbody>
</table>

TABLE VI: Computational time reduction

REFERENCES