The Use of Galois Lattices to Fine-Tune a Meta-Classifier

Mohamed Aoun-Allah and Guy Mineau

Laval University, Computer Science and Software Engineering Dept.,
Laboratory of Computational Intelligence (LCI/LIC),
Quebec City, Quebec, G1K 7P4, Canada,
{Mohamed.Aoun-Allah, Guy.Mineau}@ift.ulaval.ca,
http://w3.ift.ulaval.ca/~moaoa,
http://www.ift.ulaval.ca/~LIC/guy/guy.htm

Abstract. This paper deals with the problem of mining very large distributed databases. We propose a distributed data mining technique which produces a *meta-classifier* that is both predictive and descriptive. This meta-classifier is in the form of a set of classification rules, which could be refined then validated by fine-tuning its rule set using a concept lattice. A detailed description of this method is presented in the paper, as well as the experimentation proving the viability of our technique and the usefulness of using a concept lattice to validate rules of a meta-classifier.

1 Introduction

We witness nowadays an explosion of electronic data. Indeed, almost everything (grocery, medical file, car repair history, etc.) is recorded on databases for future analysis. For this analysis many centralized data mining tools exist but with the increasingly bigger databases, these tools are very time-consuming.

furthermore, distributed data mining tools are created as an alternative to centralized ones, when data are inherently distributed or when distributing data can speed-up the processing time. This paper deals with a distributed data mining algorithm which produces a meta-classifier in form of a set of classification rules. This set of rules could be refined then validated using a concept lattice. As it will be shown, one of the advantages of these refinement and validation steps is to reduce the meta-classifier size in terms of its number of rules, which is very helpful for a human analyst.

The paper proceeds as follows. In Section 2, we present the proposed technique of distributed data mining where we detail the use of a concept lattice to fine-tune the resulting meta-classifier. In Section 3, we present experimental results which prove the viability of our method. We finally present a conclusion and our future work.

2 The Proposed Meta-Classifier

Before detailing the proposed meta-classifier, we have to note that our meta-classifier is formed by a set of classification rules having a non disjoint coverage. Compared to existing techniques, all algorithms found in the literature produce a meta-classifier in the form of a disjoint cover set of rules [1] [2] [3]. Hence, the key subject of this paper, namely the validation of non disjoint rules using concept lattices is not comparable to any existing work.

The proposed algorithm goes roughly through two tasks: a distributed one achieved by "miner agents" which have to mine distributed databases on remote sites and to extract useful information, and a centralized task achieved by a "collector agent" which is responsible of aggregating information gathered by miner agents in order to produce the meta-classifier. Hereafter, we detail the tasks of these two types of agents.

2.1 The Tasks of a Miner Agent

We have already detailed the task of a miner agent in previous papers [4] [5] [6]. In what follows we just present an overview of these tasks through the algorithm of Fig 1.

Do, by a miner agent Am_i, working on database DB_i on a remote site:

1. Apply on DB_i a classification algorithm producing a set of rules with disjoint cover. The produced set is: $R_i = \{r_{ik} \mid k \in [1..n_i]\}$ where n_i is the number of rules;
2. Compute for each r_{ik} a confidence coefficient $c_{r_{ik}}$;
3. Extract a random sample S_i from DB_i.

Fig. 1. Algorithm showing the tasks of a miner agent.

The algorithm of Fig. 1 shows that a miner agent produces a set of classification rules, called *base classifier*, then computes for each rule a confidence coefficient. This coefficient, as its name suggests, reflects the confidence that we have in each rule based on some statistic means [4]. The sample S_i is used, later by collector agent, in the process of rule validation. The size of this sample should be very small in order to reduce the amount of data travelling from the database site to the collector agent site.

2.2 The Tasks of a Collector Agent

The tasks of a collector agent are detailed in Fig. 2. This algorithm shows that a collector agent starts by aggregating all rules in order to produce our meta-classifier, R. Indeed, the simple aggregation of base classifiers could represent a good predictive meta-classifier [4].

Do, by a collector agent CA, on the central site:

1. Main step: Create the primary meta-classifier R as follows: $R = \bigcup_{i=1...nd} R_i$ where nd is the number of sites
2. Optional refinement step: From R, eliminate rules which have a confidence coefficient lower than a certain threshold t (determined empirically) and produce R_t: $R_t = \{r_{ik} \in R \mid c_{r_{ik}} \geq t\}$;
3. Optional validation step:
   (a) Create S as follows: $S = \bigcup_{i=1...nd} S_i$;
   (b) Create a binary relation $\mathcal{I}$ defined over $R_t \times S$ where, at each intersection (r_i, s_j), we find 0 if r_i does not cover s_j, 1 otherwise ;
   (c) Apply on $\mathcal{I}$ an algorithm in order to produce a Galois lattice $\mathcal{G}$;
   (d) Use $\mathcal{G}$ to validate rules in R_t and produce the meta-classifier $R_t^{\mathcal{G}}$

Fig. 2. Algorithm showing the tasks of a collector agent.

The main problem that turns up from using the meta-classifier R as a descriptive tool could be the number of rules proposed to the user in order to explain the predictive class of a new object. In fact, the aggregation of x rules from n databases produces in the worst case nx rules. The subsequent steps (the refinement and the validation steps) are proposed to fix this disadvantage. In fact, the refinement step, proposes to remove from R rules that according to their confidence coefficient would not have a good prediction capability when used with new objects. The resulting meta-classifier is the set R_t. The validation step, which is the core of this paper, uses some samples to fine-tune rules in R_t by identifying those that actually have poor prediction performance according to samples. The validation process is detailed hereafter.

2.3 Use of Galois lattice to Fine-Tune a Set of Rules

The Galois lattice $\mathcal{G}$ is built over the binary relation $\mathcal{I}$ defined over $R_t \times S$. Consequently, each formal concept $(Rules, Objects)$ of $\mathcal{G}$ contains maximal pairs of objects and rules covering them. Thus, the obtained lattice represents a preset hierarchy of generalization/specialization of maximal pairs of rules and objects that they cover. This hierarchy presents various advantages; we enumerate a few of them hereafter:

– If there exist two rules covering the same object but predicting different classes, called rules in conflict, these rules will necessarily appear in at least one concept of the lattice. In these concepts we find all rules in conflict, and the objects that they cover.

- In order to delete a rule r from the lattice, we do not need to visit each concept. We must just find the first concept that contains the first occurrence of r when we go upwards in the lattice. Then, all the occurrences of this rule will be in concepts that are superconcepts of the latter identified one. This is due to the structure of the lattice, where when we go up from the bottom most to the upper most concept, the extension of a concept (i.e., rules) is enriched while intention (i.e., objects) is impoverished. In other words, when we go up in the lattice, the number of rules increases thus together they will cover fewer objects.
- Rules in a concept are coded by their numbers. Thus their treatment is very fast since the coverage of each rule is already computed and stored in the lattice. The only information that we may need, the conclusion of the rule, could be obtained instantly.

In order to simplify the presentation of the algorithm that uses $\mathcal{G}$ to validate the rules, we present in the following some notation and terminology:

1. Our algorithm manipulates only binary databases denoting by "+" and "−" (*positive* and *negative* class) the two classes of the data set. Nevertheless, it could be extended to handle multiple class systems.
2. We note by *cpt* a concept of the lattice, R_{cpt} its extension and O_{cpt} its intention.
3. We call a *positive rule* (resp. a *negative rule*), a rule having the positive (resp. negative) class as conclusion.
4. We borrow some notation and terminology from [7], as we call the *least concept* the bottom most concept of the lattice. It contains no rules and all the objects. Dually, we call the *largest concept* the upper most concept of the lattice. It contains all the rules and no objects.
5. We note by $NbRules(ARuleSet)$ (resp. $NbObjects(AnObjectSet)$) the function that returns the number of rules in the rule set $ARuleSet$ (resp. objects in the set of objects $AnObjectSet$).
6. We note by $RulesOfTheClass(ARuleSet, clas)$ the function that returns rules of the set $ARuleSet$ belonging to the class $clas$ ($clas \in \{+, -\}$). The result is a set of rules.
7. We note by $ObjectsOfTheClass(AnObjectSet, clas)$ the function that returns objects of the set $AnObjectSet$ belonging to the class $clas$. The result is a set of objects.
8. $NbObjects(ObjectsOfTheClass(O_{cpt}, +))$ (resp. $NbObjects(ObjectsOfTheClass(O_{cpt}, -))$) is abbreviated by $NbObj^{+}_{cpt}$ (resp. $NbObj^{-}_{cpt}$).
9. $NbRules(RulesOfTheClass(R_{cpt}, +))$ (resp. $NbRules(RulesOfTheClass(R_{cpt}, -))$) is abbreviated by $NbRules^{+}_{cpt}$ (resp. $NbRules^{-}_{cpt}$).

How to Use a concept lattice. We bring to mind that we use a concept lattice in order to validate the rules of R_t that statistically (according to the

confidence coefficient) should behave well when faced with new objects. This validation consists of choosing among rules that do not correctly predict the class of objects of S those to keep in the final meta-classifier $R_t^{\mathcal{G}}$. In other words, it consists of identifying rules to delete from R_t so that each conflicting rule is assessed. The successful rules are assigned to $R_t^{\mathcal{G}}$.

To achieve its task, the validation algorithm starts by identifying concepts having conflicting rules. To do this, we compute for each concept the number of positive rules ($NbRules_{cpt}^{+}$), the number of negative rules ($NbRules_{cpt}^{-}$), the number of objects belonging to the positive class ($NbObj_{cpt}^{+}$), and the number of objects belonging to the negative class ($NbObj_{cpt}^{-}$). Then we associate to each concept a label ("+", "−" or "?") according to the majority of rules. The label "?" is associated to a concept if the number of positive rules equals the number of negative rules (See (1))

$$Label(cpt) = \begin{cases} + \text{ If } NbRules_{cpt}^{+} > NbRules_{cpt}^{-} \\ - \text{ If } NbRules_{cpt}^{+} < NbRules_{cpt}^{-} \\ ? \text{ Otherwise} \end{cases} \quad (1)$$

We have to note that the labelling of concepts could be done during the construction of the lattice, at a negligible cost. Once the labelling of concepts is done, we could resume the algorithm as follows:

1. Go through the lattice from the least to the largest concept.
2. At the first concept containing rules in conflict, we identify rules belonging to the minority class that we will call them *problematic rules*. These rules are positive rules if the concept is labelled negative et vice-versa. If the concept is labelled "?", all its rules are considered problematic (See (2)). In other words, problematic rules are rules that should not be in the concept and hence, eventually, should not be in the final meta-classifier.

$$PrbRules(cpt) = \begin{cases} RulesOfTheClass(R_{cpt}, +) & \text{If } Label(cpt) = - \\ RulesOfTheClass(R_{cpt}, -) & \text{If } Label(cpt) = + \\ R_{cpt} & \text{If } Label(cpt) = ? \end{cases} \quad (2)$$

3. In order to assess the impact of suppressing a rule from the lattice, we associate to each concept a cost function. Lets explain this function through an example. Figure 3(b) details class distribution of the concept of Fig 3(a). We can easily notice that rule F is a problematic one since it is in conflict with rules C and E.

C, E, F	1, 11, 15, 18

(a) Example of a concept

C(+),E(+),F(−)	1(+),11(+),15(+),18(−)

(b) Class distribution in the concept

Fig. 3. An example of a concept from the lattice.

It is clear that suppressing the rule F from this concept has a positive effect or at least no effect, since the majority of objects and the majority of rules

belong to the positive class whereas F is a negative rule. And this is the case of all problematic rules where by definition they are rules belonging to the minority class. Consequently, the cost function for suppressing a problematic rule is computed through out all concepts that contain it. Fortunately, the lattice restrains our exploration of concepts where F will appear only in those that are superconcepts of the bottom most concept containing the first occurrence of the rule under consideration (See Fig. 4).

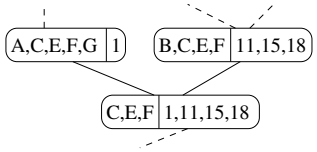

Fig. 4. Example of superconcepts of the one of Fig. 3.

4. The cost function that we proposed represents the gain of objects correctly classified minus the loss of objects incorrectly classified when the label of the concept changes (designated by function MP). In other words, suppose that we suppress from the concept of Fig. 3 the rule C or E. In that case the label of the concept passes from "+" to "?". The cost of this action according to our function is $-3+1=-2$ since objects 1, 11 and 15 are no longer correctly classified and the object 18 gains in classification since we consider that the class "?" is closer to "−" than "+" is close to "−".
5. For presentation purposes, we denote by cpt' the concept cpt abated by problematic rules. Hence, our cost function could be defined as follows:

$$Cost(cpt) = \begin{cases} 0 & \text{If } Label(cpt) = Label(cpt') \\ NbObj^{-}_{cpt} - NbObj^{+}_{cpt} & \text{If } (Label(cpt), Label(cpt')) \in \\ & \{(+,-),(+,?),(?,-)\} \\ NbObj^{+}_{cpt} - NbObj^{-}_{cpt} & \text{If } (Label(cpt), Label(cpt')) \in \\ & \{(-,+),(-,?),(?,+)\} \end{cases}$$

6. This cost is iteratively repeated on the remaining of the lattice by suppressing the same rule already identified as problematic. If the final cost over the lattice is positive, thus it is advantageous to eliminate the rule from the concept and vice versa. If the final cost equals zero it is neither advantageous nor disadvantageous to keep or to ignore the rule. Actually, keeping or ignoring these rules (that we call "*limit rules*") produces two variants of cost function according to the decision taken about these rules.
7. The process of identifying problematic rules and computing the result of the cost function is also done over all possible combinations of problematic rules in one concept. So $2^n - 1$ rule subsets could be assessed if n is the number of problematic rules (obviously, the empty set is ignored). In other words, if in one concept there is more than one problematic rule, we compute the powerset of problematic rules and then for each set from the powerset the process described above is conducted.

Cost Function Variants. The first variant of cost function is deduced from the use of the label "?" where it can be considered as an intermediate class between the "+" and "−". This function is designated by $MPQM$. Thus when the label of a concept goes from "−" to "+" by removing rules identified as problematic, the cost function returns twice the difference between positive and negative objects and vice versa. Whereas, when "?" appears as the label of the concept before or after removing problematic rules, the cost function returns a simple difference.

Another variant of cost function could be proposed by considering only the sign of the difference. Thus this binary function (designated by function BIN) return only $+1$, -1 or 0 according to the sign of the difference and the change of label.

The last variant proposed is deduced from the BIN function when considering "?" as an intermediate class. Thus, when the label changes going from "+" to "−" or inversely, the $BINQM$ function returns ± 2, otherwise it returns ± 1 or 0 if there is no label change.

2.4 The Use of Set R as a Meta-Classifier

The set R represents the aggregation of all base classifiers ($R = \cup_i R_i$). This rule set is used as a predictive model as well as a descriptive one. From a predictive point of view, the predicted class of a new object is the class predicted by a majority vote of all the rules that cover it, where the rules are weighted by their confidence coefficients[1].

It is to be noted that any object can be covered by at most nd rules –knowing that nd is the number of sites. The number of rules is not exactly equal to nd because the confidence coefficient determination process could fail in certain circumstances, due to a lack of cover, and consequently the rule in question would be ignored. Besides, by gathering the sets R_i, a rule can appear in more than one base classifier. In this case, only one occurrence of the rule is kept by assigning it with a confidence coefficient equal to the mean of the confidence coefficients of its various occurrences.

From a descriptive point of view, rules which cover an object explain its class even in the case of a tie of the simple and/or the weighted majority vote. As the whole system is developed as support to decision-making, the rules covering an object maybe proposed to the user who could then judge, from his expertise, of their relevance. Presenting to a decision maker more than one rule in order to explain the class of an object may have its advantages since this provides a larger and more complete view of the "limits" of each class. We bring to mind, that in machine learning, the limit which defines separation between various classes is generally not unique nor clear cut, and consequently, several rules producing the same class can represent the "hyper-planes" separating the various classes, providing various views on these data.

[1] However, in a tie situation, we propose to carry out a simple majority vote. In rare cases, when the simple majority vote leads to a tie, we choose the majority class in the different training sets.

3 Experiments

To assess the performance of our meta-learning method, we conducted a battery of tests in order to assess its prediction (accuracy) rate and its size (i.e., the number of rules it produces). We compared it to a C4.5 algorithm built on the whole data set, i.e., the aggregation of the distributed databases. This C4.5, produces the rule set R', which is used as a reference for its accuracy rate since we assumed in the introduction that it is impossible to gather all these bases onto the same site, and this, either because of downloading time, or because of the difficulty to learn from the aggregated base because of its size.

For this purpose we ran the following 4 experiments:

Exp. 1: Find the best threshold t for producing the meta-classifier R_t.
Exp. 2: Find the best pair (t, $Costfunction$) for producing $R_t^{\mathcal{G}}$.
Exp. 3: Compare R, R_t and $R_t^{\mathcal{G}}$ to R' on the basis of accuracy.
Exp. 4: Compare R, R_t and $R_t^{\mathcal{G}}$ to R' on the basis of the size of their rule set.

All these experiments have been tested on ten data sets: adult, chess endgame (King+Rook versus King+Pawn), Crx, house-votes-84, ionosphere, mushroom, pima-indians-diabetes, tic-tac-toe, Wisconsin Breast Cancer (BCW)[8] and Wisconsin Diagnostic Breast Cancer (WDBC), taken from the UCI repository [9]. The size of these data sets varies from 351 objects to 45222 objects. Please note that we deleted from these data sets objects with missing values, and we have mixed them up in order to avoid "sorted" bases. Furthermore, in order to get more realistic data sets, we introduced noise in the ten aforementioned databases, and this by reversing the class attribute[2] of successively 10%, 20%, 25% and 30% of each data set objects. Hence, since for each data set we have, in addition to the original set, 4 other noisy sets the total number of databases used for our tests is 50.

In order to simulate distributed data sets, we did the following. We divided each database into a test set with proportion of 1/4. This data subset was used as a test set for our meta-classifier and for R', our reference classifier. The remaining data subset (of proportion 3/4), was divided in its turn randomly into 2, 3, 4 or 5 data subsets in order to simulate distributed databases. The size of these bases was chosen to be disparate and in such a way such there was a significant difference between the smallest and the biggest data subset. As an example of such subdivision see Fig. 5.

For the construction of the base classifiers we used C4.5 release 8 [10] which produces a decision tree that is then directly transformed into a set of rules. For the concept lattice construction we used the algorithm proposed in [11].

3.1 Experiment 1: Best Parameter t for R_t

In order to find the best t for R_t, we tried all values ranging from 0.95 to 0.20, with decrements of 0.05 and 0.01. The analysis of results that we got for R_t over

[2] Please note that all data sets have a binary class attribute.

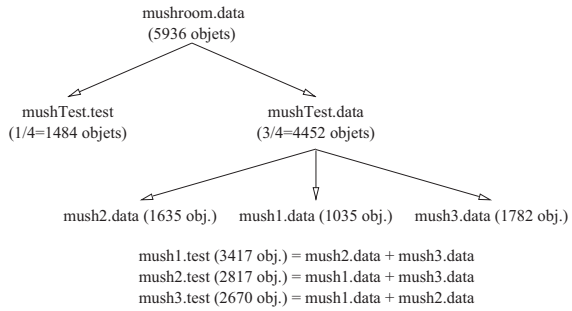

Fig. 5. Example of subdivision for a database from the UCI.

the 50 data sets show that the minimum error rate of R_t is obtained in almost all the cases with the threshold $t = 0.95$ or $t = 0.01$. When this is not the case, the error rate of R_t, with $t = 0.95$ or $t = 0.01$, is worse than the minimum error rate by no more than 0.1% except for the Pima-Indians original database where the difference is 1%.

The choice of a high threshold (such as 0.95) suggests that keeping only rules with a high value of the confidence coefficient produces good results. Moreover, a threshold as low as 0.01 signifies that the aggregation of almost all the rules produces also good results thanks to the weighted majority vote. So weighting the rules by this confidence coefficient seems to be quite sufficient to provide our method with a satisfactory accuracy rate.

In order to choose between these two thresholds, we draw a table of the number of occurrences for which they produce the minimum error rate. In the case where the best accuracy is obtained with t' which is neither 0.01 nor 0.95, we count t' either with 0.01 or 0.95 depending if it is closer to the first or to the second threshold.

From table 1 we can choose the threshold 0.01 as the best one since it produces the minimum error rate of R_t in the majority of the cases.

Table 1. Number of occurrences of the minimum of R_t error rate with 0.01 and 0.95.

Bases	Min with 0.01	Min with 0.95	Error rate steady
Original databases	6	3	2
Bases with 10% noisy	3	3	4
Bases with 20% noisy	4	1	5
Bases with 25% noisy	3	5	3
Bases with 30% noisy	4	3	3
TOTAL	20	15	17

3.2 Experiment 2: Best Pair (t, $CostFunction$) for $R_t^{\mathcal{G}}$

We start by identifying the threshold t that produces the minimum error rate of $R_t^{\mathcal{G}}$ when considering the four cost function (MP, $MPQM$, BIN and $BINQM$)

presented above with for each one a little variant by taking or ignoring limit rules for a total of 8 functions. We did the same analysis that for R_t and we found that $t = 0.01$ produces a score of 124 occurrences of the minimum and 41 occurrences to be closest to the minimum than 0.95. Where the later threshold produces a score of 116 occurrences and 34 occurrences closest to the minimum. Consequently, we can assume that the threshold $t = 0.01$ is the best for $R_t^{\mathcal{G}}$ too. This very low threshold suggests that the proposed technique is robust faced to poor rules. Moreover, it could take into account almost all rules that the validation process will approve or eliminate.

Once threshold t fixed, we have to find the best cost function. To do so, we analyze the Table 2 that presents which of the cost function produces the minimum error rate of $R_t^{\mathcal{G}}$. The winning function is clear that it is the "$MPQM$" or "MP" taking into account limit rules (LR). Functions "$BINQM$" and "BIN" with limit rules produce also good results but they are slightly less efficient from a prediction point of view than their competitors.

Table 2. Number of bases for which cost functions produce $R_t^{\mathcal{G}}$ minimum error rate.

	Original DB	10% noisy DB	20% noisy DB	25% noisy DB	30% noisy DB	Σ
$MPQM$ with LR	4	5	4	3	2	18
$MPQM$ without LR	3	4	1	3	2	13
MP with LR	4	5	4	4	2	19
MP without LR	3	4	1	3	2	13
$BINQM$ with LR	6	2	4	1	3	16
$BINQM$ without LR	3	3	1	4	2	13
BIN with LR	6	2	3	1	4	16
BIN without LR	3	3	1	4	2	13

3.3 Experiment 3: Compare R, R_t and $R_t^{\mathcal{G}}$ to R' on the Basis of Accuracy

Once parameters for R_t and $R_t^{\mathcal{G}}$ has been found, we can compare the prediction performance of our meta-classifiers R, R_t and $R_t^{\mathcal{G}}$ to the classifier R' used as reference since it is the ideal case. To do so, we compare R' with its error rate confidence interval (lower and upper bounds) computed at 95% confidence to our meta-classifiers, over the original databases and over the noisy databases.

As a detailed citation of these results needs more than a few pages, we will restrain our presentation to the most important results. Indeed, we can resume our observations to the following:

1. The prediction performance of R are very comparable to R' since in 34 cases over 50, the error rate of R is statistically comparable (with 95% confidence) to that of R'. It worsens only in 5 cases but it improves in 11 cases.
2. The sets R and R_t have sensibly the same error rate except for 6 cases where in two of them the error rate of R_t is worse than that of R by no more than

0.1% (which is not a significant difference) and in the 4 other cases R_t do better than R with an error rate difference ranging from 1.1% to 3%. Hence, we can conclude that R_t, in general, does predict as well as R or better.
3. Globally, R_t and $R_t^{\mathcal{G}}$ present comparable prediction performance. Indeed, over the 50 data sets, $R_t^{\mathcal{G}}$ presents exactly the same error rate over 30 data sets, better error rate for 11 data sets with a difference of at most 4.7% and a worse error rate for only 9 cases with a difference of at most 2.1%.
4. When databases are very noisy, our meta-classifiers R, R_t and $R_t^{\mathcal{G}}$ produce better error rates (statistically, with confidence of 95%) than R'. We conclude that the error rates of R are comparable to R', but that R outperforms R' as noise increases in the data set.

The reader should notice that even if R_t or $R_t^{\mathcal{G}}$ does not significantly improve R in terms of its error rate, applying the threshold t then validating rules offers some advantages, like decreasing the meta-classifier size.

3.4 Experiment 4: Compare R, R_t and $R_t^{\mathcal{G}}$ to R' on the Basis of the Size of their Rule Set

Table 3 presents the number of rules in the classifiers: R', R, R_t and $R_t^{\mathcal{G}}$. It is clear from this table that R, R_t and $R_t^{\mathcal{G}}$ have a relatively low number of rules which is in certain circumstances inferior that the number of rules in our reference classifier R'. This result is very encouraging since our meta-classifier can be seen as neither more difficult nor easier to interpret than R'.

Table 3. The number of rules in each rule set.

	Adult	BCW	Chess	Crx	Iono.	Mush.	Pima.	Tic.	Vote	Wdbc
R'	523	10	31	25	7	24	21	69	5	11
R	592	50	54	23	11	11	30	77	10	18
R_t	482	33	54	20	9	11	26	64	6	17
$R_t^{\mathcal{G}}$	469	32	46	19	9	11	26	61	6	17

Moreover, we can easily observe that the refinement step as well as the validation steps are very useful since they can reduce the number of rules in R significantly. For instance we have a reduction of $40.0\%, 36.0\%, 20.8\%$ and 20.8% for respectively Vote, BCW, Adult and Tic-Tac-Toe data sets.

4 Conclusion

The objective of this paper is to present an application of concept lattices to a distributed data mining technique (DDM). This technique goes globally through two steps. The first one is to mine in a distributed manner each data set, then in centralized site information gathered from the first process is aggregated.

Concept lattices play a key role in this aggregation process, where they are used to fine-tune rule sets.

We have demonstrated in this paper that concept lattices could be very useful in DDM. Indeed, from a prediction point of view a validated meta-classifier (i.e., rules of the meta-classifier are validated by a concept lattice) performs as well as or even better, than a classifier built on the whole data set, which is used as a reference, depending on the level of noise in it. Moreover, from a description point of view (i.e., number of rules in the classifier), the size of validated meta-classifier is usually comparable to that of the reference centralized classifier. When we compare the simple rule sets aggregation with this set refined then validated by a Galois lattice, a significant decrease of the set of rules (up to 40%) could be observed.

Although the preliminary results as presented in this article are encouraging, real world experiments will soon be the subject of experimentation.

References

1. Williams, G.J.: Inducing and Combining Decision Structures for Expert Systems. PhD thesis, The Australian National University (1990)
2. Hall, O.L., Chawla, N., Bowyer, W.K.: Learning rules from distributed data. In: Workshop on Large-Scale Parallel KDD Systems (KDD99). Also in RPI, CS Dep. Tech. Report 99-8. (1999) 77–83
3. Provost, F.J., Hennessy, D.N.: Scaling up: Distributed machine learning with cooperation. In: Thirteenth National Conference on Artificial Intelligence (AAAI-96). (1996) 74–79
4. Aounallah, M., Mineau, G.: Le forage distribué des données : une méthode simple, rapide et efficace. Revue des Nouvelles Technologies de l'Information, extraction et gestion des connaissances **1**(RNTI-E-6) (2006) 95–106
5. Aounallah, M., Quirion, S., Mineau, G.: Forage distribué des données : une comparaison entre l'agrégation d'échantillons et l'agrégation de règles. Revue des Nouvelles Technologies de l'Information, extraction et gestion des connaissances **1**(RNTI-E-3) (2005) 43–54
6. Aounallah, M., Mineau, G.: Rule confidence produced from disjoint databases: a statistically sound way to regroup rules sets. In: IADIS international conference, Applied Computing 2004, Lisbon, Portugal (2004) II–27 – II31
7. Ganter, B., Wille, R.: Formal Concept Analysis: Mathematical Foundations. Springer (1999)
8. Mangasarian, O.L., Wolberg, W.H.: Cancer diagnosis via linear programming. SIAM News **23**(5) (1990) 1–18
9. Blake, C., Merz, C.: UCI repository of machine learning databases. http://www.ics.uci.edu/∼mlearn/MLRepository.html (1998)
10. Quinlan, J.R.: Improved use of continuous attributes in C4.5. Journal of Artificial Intelligence Research **4** (1996) 77–90
11. Valchev, P., Missaoui, R., Lebrun, P.: A partition-based approach towards building galois (concept) lattices. Technical Report 2000-08, Département d'Informatique, UQAM, Montréal (CA) (2000)