The Integration of user knowledge to learn a specialized decision tree from a real-life data: an empirical and computational study

Semghouni Redouane[1], Rahal Sid Ahmed[1] and Benyoucef Othmane[1]

[1] Département d'informatique, Université des Sciences et de la Technologie USTO, ORAN, 31000, Algeria.

Abstract

Decision trees are the most applicable technique of data mining, because of its power and its simplicity of interpretation. However, learning decision trees from medium to large dataset are different from learning from small dataset, especially when data contain instance that are semantically independent, this lead to lose in accuracy. In our approach, we build patterns according to some criteria with the help of the users' knowledge. We use knowledge to filter and reduce the database and remove the data; this is considered as a noise. Learning from the filtered data can generate more accurate and small decision tree. In our experimentation, we show the difference in accuracy between learning over the entire data and filtered data.

Keywords: *Classification, Domain Knowledge, Data reduction, Decision tree.*

1. Introduction

Variable selection and data reduction have become the focus of actual research in many areas of application, especially when datasets are large "large number of rows or columns", feature selection solve many problems in data mining, it can reduce computational costs "time/memory", and solve the problem of over-fitting, so the reduction can enhance the system interpretability, many research prove that these reductions can be benefit in term of accuracy of learned pattern.

In general, feature selection has a relation with data dimensionality reduction, many techniques has been developed like: the Information Bottleneck [11], Locally Linear Embedding [14], Sufficient Dimensionality Reduction [2], Margin Based Feature Selection [13] and the new filter method for categorical variables selection [5]. All these approaches, can improve the classification accuracy.

Unlike feature selection, there are other kinds of reduction, this reduction operate in the row of data "number of examples".

The most researches demonstrates that in model construction algorithms, more data is not always better for learning, in [17] they show that increasing the amount of data used to build patterns often results in a linear increase in pattern size, probably with no significant increase in accuracy. In the same philosophy, in [10,17], the study shows a randomization testing approach and a progressive sampling of data until we find the best accuracy. In other approaches we cite: the permutation based in p-values technique, this can improve the accuracy of a classifier, and this technique was studied in [16,4], and other technique based in random selection was studied in [12, 3].

2. The Framework

Splitting data randomly or selecting a random instance for learning can be fatal, especially when there is a relation between instances of data and its attribute, this can appear when we learn from real-life data, when each instance contains a significant value, and ignoring it can be critical, including it in the process of learning may change radically the generated pattern and its accuracy.

In our research, and to avoid a bad split, we try to split the data according to some criteria defined by users or experts "not randomly like other approach", and then we learn from different fragments of data. Splitting data in several parts is considered as a kind of a reduction technique.

In our approach, we split data that are judged by users or experts that are independent semantically, so we apply our proposed approach in data that are from real-life, and not from data that are simulated or generated randomly.

When we split data into two or many sub data that are semantically independent, we consider that the first is a noise for the other parts, and vice versa. All the parts will be learned separately, and we generate many decision trees "the number of decision trees is the number of the sub parts" rather than one decision tree.

In our experience, we will show that this simple method can rivals with boosting [19] and bagging [9] techniques, because in boosting and bagging, not all data are used, and instance are chosen randomly, so in our approach, we study also the impact of splitting data, so we study the change in some measure like Gain-ratio [7], and RELIEF [8].

The rest of this paper is structured as follow: section 3 describes the steps and the components that needs to accomplish the task, section 4 describes how to modulate user knowledge, in section 5, we detail how to create the reduced database using the user knowledge, section 6 is reserved for the experimentation and some empirical comparison.

3. Method

Unlike the simulated data, a real data contain a significant value, so expert can make rule that operate in the value of attribute, for example: when we study a data about marketing, we can split data by subject: the first part considering developed country and the second for underdeveloped country; because we know that there is a difference between the comportment and the purchasing power of the two worlds, so the first data is considered as noise for the second.

To demonstrate our proposition we need to modulate user knowledge and using it to split and reduce data, we use ontology [15] for the formalization of user knowledge. In figure 1, we show the global architecture of our proposed approach.

For each part of data, we generate a decision tree that corresponding to a given rule or concept, so a single tree cannot predict all types of new instances. For each new instance we need to find a corresponding tree to predict it. Otherwise, for a new instance, we will need to find its corresponding concept, and then use its tree to predict this new instance. (figure.2)

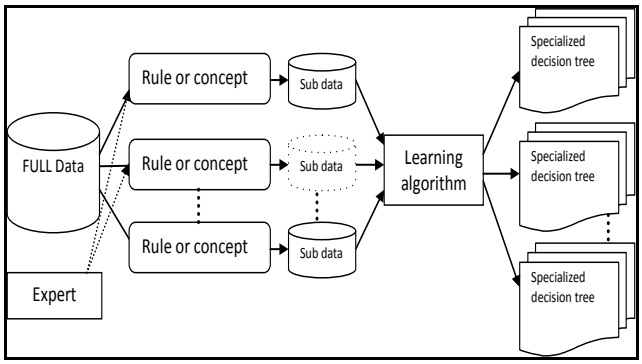

Figure 1: the integration of user knowledge in the process of learning.

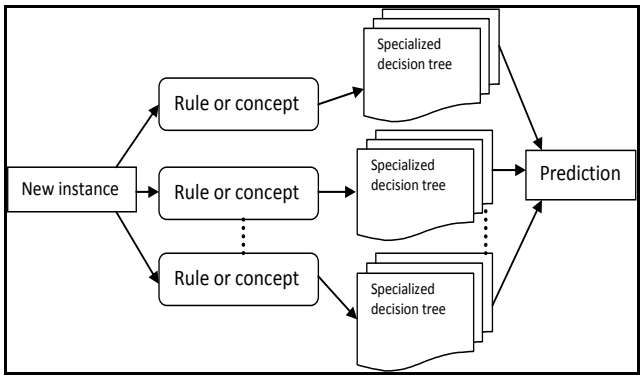

Figure 2: the process of prediction.

4. User knowledge

The data base is constituted of a set of N tuple describes through P attributes.

Let $T = \{t_1, t_2, ..., t_n\}$ be the set of tuples, and $A = \{a_1, a_2, ... a_p\}$ be the set of P attributes.

User knowledge is modulate or defined using an ontology. This ontology is defined by a set of concepts $C = \{C_1, C_2, ..., C_i\}$. The concepts are represented by hierarchies, so each concept can include other concepts. To be able to apply the ontology on the database is sufficient to associate a concept directly to an attribute of the database "of course each concept can be applied to a specific attribute", and then construct a constraint for each concept. In other way, we construct a constraint on the values of attributes.

The rules schemas permit to express some knowledge to get specific models for specific data. We can also combine the rules schemas to generate more complex rule and to build more specific decision trees.

The concepts are represented as: (X_1) logic operator (X_2) logic operator (X_3)…, where each Xi is a constraint on the

values of attributes (A), so we write rule and concept like: $C_i=(X_{i1})$ logic operator $(X_{i2}…X_{in})$.; and $R=(C_1)$ logic operator $(C_2)…C_j$. So by decomposing each concept, we get $R=(X_{11}..X_{1n})$ logic operator $(X_{21}…X_{2n})…(X_{i1}….X_{ij})$, where the logic operators are {and, or, not}. Finally, with the concepts and the constraints applied, we can select the tuples concerned for the construction of the patterns.
For example the rule R1 = ($C_1$and C_2 and C_3) means that if each attributes of each tuples of the database "which are concerned only by the concept C_i" verify C_1 and C_2 and C_3 then it will be retained for the construction of the tree, else the tuples will be deleted or simply ignored.

5. The Reduction and selection algorithm

We propose this simple algorithm that split and reduce data in order to obtain sub data that contain less noise. For example, to obtain the first part of data we apply the reduction algorithm with the rule **R**, and for the second part we apply it with the rule $\mathbf{NR_1}$=NOT **R.**

Input:

1. Database **D**.
2. A collection of concept and rule **R**= C1 , C2 … C_i.

Output:

1. database **B** « initialized to Ø »

For each tuple $T=\{T_1,T_2…T_n\}$ from database D
For each attribute $A=\{a_1,a_2…,a_p\}$ of tuple T_i

- Verify if **ai** satisfy the constraint of concerned concepts of the rule **R**.
- If all verification is positive write $\mathbf{T_i}$ in the output database **B**.

End.

Figure 3: the reduction and selection algorithm.

At the end of process, we obtained a reduced database according to the rule R, so we use this database to derive a model using a simple classification algorithm like C4.5.

6. Experimentation and Empirical study

To demonstrate our approach, we will use two datasets from UCI [1], these datasets are not generated randomly, but it is collect from real world, so we split these data in several parts according to some concept written by the users or expert using the syntax of the tool protégé (http://protege.stanford.edu). For each generated part, we experiment a classification and empirical studies and some statistical measure.

The database chosen for the first tests is "Statlog German credit", this one is downloadable on the site of data base repository UCI, the data base contains 1000 tuples and 20 attributes (7 numeric and 13 categorical), we need to understand the data to builds some concept.

We know that in real life, the comportment and characteristic of customers that have high credit are different than costumers have a low credit, and a costumer with a good job or a bad job, so experts can define a rule to separate these kinds of customer. In the database we have the attribute 'Purpose' that has eleven different kind of credit: new car, old car, furniture/equipment, radio/television, domestic appliances, repairs, education, vacation, retraining, business and others, its value in data base is respectively: A40…A410. So we can decompose the problem in two subs problems "HighCustomer and LowCustomer", in the data we separate customer with high/low credit domain. Also we can decompose our problem in 3 subs problem "Low Skilled Customer, Skilled Customer and High Skilled Customer", this three concepts concerning the job of customer.

In figure 4 we show the ontological tree for this data.

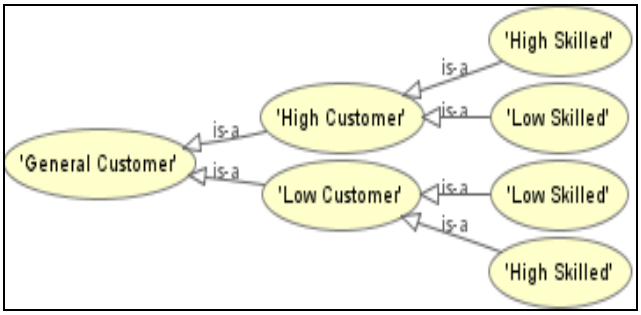

Figure 4: small ontology for the "Statlog German credit" dataset.

We define the first concept using the syntax used by the tool protégé as the following:

- HighCustomer =(hasPurposeValue some string [=”A40”] or string [=”A41”] or string [>= ”A49”]).
- LowCustomer= Not HighCutomer.
- LowSkilledCustomer=(hasJobValue some string [=”A171”] or string[=”A172”]).
- SkilledCustomer=(hasJobValue some string [=” A173 ”]).
- HighSkilledCustomer=(hasJobValue some string [=” A174 ”]).

The entire data contain 1000 instance. In our experience we use 666 for learning and the rest “334” for validation. In our experience we compare our approach using simple C4.5 [16], and boosting over all data. In the boosting experience we use ten iteration, we use two approaches with the boosting “with and without Resampling [18]”. All measure and algorithm are performed using the tool WEKA [6].

We split learning data and testing data according to the concepts. In the following table, we show the number of instance for each part, and the accuracy of each part.

Table 1: The accuracy of each concept.

Concept/learning algorithm	***# Learning***	**# Testing**	**Accuracy**
1	666	334	71.2575%
2	666	334	70.6587%
3	666	334	69.1617%
4	276	158	71.519 %
5	390	176	73.8636%
6	102	46	76.087%
7	421	209	76.0766%
8	143	79	73.4177%

In the above table each number is related to a specific learning to a specific concept as fellow:

1. Learning over the full data using the C4.5 algorithm.
2. Learning over the full data using the boosting algorithm.
3. Learning over the full data using the boosting algorithm with the R-Sampling method.
4. Learning over the data from the ***HighCustomer*** concept using the C4.5 algorithm.
5. Learning over the data from the ***LowCustomer*** concept using the C4.5 algorithm.
6. Learning over the data from the **LowSkilledCustomer** concept using the C4.5 algorithm.
7. Learning over the data from the **SkilledCustomer** concept using the C4.5 algorithm.
8. Learning over the data from the **HighSkilledCustomer** using the C4.5 algorithm.

All result in table 1, indicate that: by splitting data semantically using user knowledge is better than learning over all data, and algorithm based in a random selection can give a less accuracy” like the boosting algorithm”, the average of accuracy using the concepts “High Customer and Low Customer” is 72.69% that is more than 71.25%, and 75.1937% using the concepts “Low Skilled Customer, Skilled Customer, High Skilled Customer”.

Reduce data may cause difference in some measure like the entropy, or Gain-ratio, so to demonstrate that our technique of reduction can improve these measures, we apply some measure to all attribute in all data and compared them in each part of data that is generated using the concepts listed above.

In table 2 “see **Appendix**” we show the difference “subtraction” between the measure of all attribute in each part of data and the entire data. The measures that are used: entropy, gain-ratio and RELIEF.

We indicate that the measure of Gain-ratio in the full data is less than the measure in the data generated with the concepts “HighCustomer and LowCustomer“, the total average is 0.00141 and 0.01614 in the order. For the measure RELIEF, there is a very little change with an average of 0.00259 and -0.0009.

We apply the same measure for the concepts “High Skilled Customer (1), Skilled Customer (2) and Low Skilled Customer (3) ” . Table 3 shows the different results:

Table 3: The measures relative to each concept.

Concept	**Entropy**	**Gain-ratio**	**RELIEF**
1	0.0404085	0.02676504	-0.0135826
2	-0.0138225	0.00317728	0.0105214
3	-0.0214804	0.02608142	0.0045734

The average of deference between measure in all data and the concepts, so we see that in most time the measure Gain-ratio and RELIEF are improved, this is because the data are more pruned. We consider all instance concerned by the concept High Skilled Customer as a noise for the data that have an instance concerned by the concept Low Skilled Customer and Skilled Customer and vice versa.

In other experimentation we use two data from UCI, the first is Thyroid Disease "the name exactly is thyroid0387" it contains 9172 instance "60% for learning and the rest for testing". In the second test, we use a large data set called Adult with 48842 instances "32561 for learning and 16281 for testing".

Concerning the data set Thyroid, it treats the problem of the infection, and in real life we are known that an infection in adults is different in children, so simply we can decompose our problem in an interval of age.

We construct a rule operating in the attribute age and then learn from each data generated from each rule or concept. Figure 5 show the ontology for this data set.

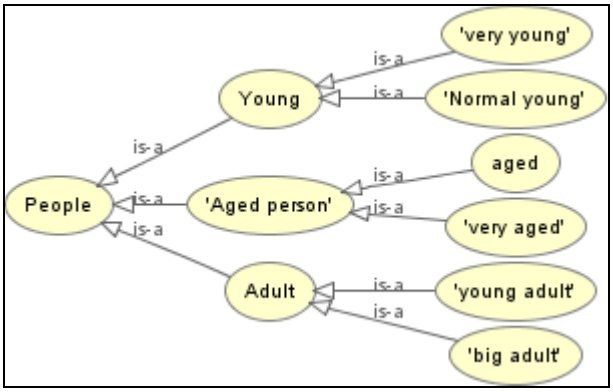

Figure 5: ontology tree for the Thyroid data set.

We define the six concepts that cover all data as follow:

- VeryYoung =(hasAgeValue some integer [<= "16"]).
- NormalYoung= not VeryYoung and ((hasAgeValue some integer [<="30"]).
- YoungAdult= (hasAgeValue some integer [>"30"] and integer [<="40"]).
- BigAdult =(hasAgeValue some integer [>"40"] and integer [<="55"]).
- Aged=(hasAgeValue some integer [>"55"] and integer [<="75"]).
- VeryAged=hasAgeValue some integer [>"75"] .

We built in each part of data, which are generated from this six concept, a decision tree. The result of accuracy is in table 4:

Table 4: The accuracy of each concept.

concepts	learning	testing	Accuracy
1	6416	2752	66.2064 %
2	6416	2752	64.4622%
3	6416	2752	64.0625%
4	146	61	72.1311%
5	958	366	72.9508 %
6	908	359	72.1448 %
7	1356	564	69.3262 %
8	2435	1073	61.137%
9	613	329	61.0942 %

In the precedent table each number is related to a specific learning to a specific concept as fellow:

1. Learning over the full data using the C4.5 algorithm.
2. Learning over the full data using the boosting algorithm.
3. Learning over the full data using the boosting algorithm with the R-Sampling method.
4. Learning over the data from the **VeryYoung** concept using the C4.5 algorithm.
5. Learning over the data from the **NormalYoung** concept using the C4.5 algorithm.
6. Learning over the data from the **YoungAdult** concept using the C4.5 algorithm.
7. Learning over the data from the **BigAdult** concept using the C4.5 algorithm.
8. Learning over the data from the **Aged** concept using the C4.5 algorithm.
9. Learning over the data from the **VeryAged** concept using the C4.5 algorithm.

The average of accuracy of all concepts is 68.1306% which is better than 66.2064 % over the full data, and we show that the accuracy of boosting is less than the simple C4.5.

For this last test, we perform a scalability test. We use the data set "Adult". The problem is to determine wither an individual income is great or less then 50K $/year. We define two categories of people: the first is about a person that has a family and the second without a family, because in real life when we have a family there is a big chance that its income exceeds 50K $/year. We define the two concepts as follow:

- AlonePersson=(hasRelationshipValue some string [="Not-in-family"] or string[="Other-relative"] or string[="Unmarried"]).
- PerssoneWithFamily= not AlonePersson.

We built in each part of data, which are generated from this 2 concept, a decision tree. The result of accuracy is given in table 5:

Table 5: The accuracy of each concept.

Concepts	# Learning	# Testing	Accuracy
1	32561	16281	85.8485 %
2	32561	16281	83.5391 %
3	32561	16281	83.6988 %
4	12732	6482	94.0759 %
5	19829	9799	80.9062%

From table 5 we see that the average of accuracy is 87.491% which is better than 85.84% (through all data), and the boosting algorithm give a less accuracy because of its randomization in selecting data.

7. Conclusion and perspective:

This paper introduces the integration of user knowledge in the process of learning. The data is considered as not cleaned semantically, so it be split in several part that each part will be learned separately, this technique is more efficient than using all data, so we motivate the integration of user knowledge that can be helpful for learning.

from all our experience, we can conclude that reducing data semantically can be benefic and generate more accurate model than the entire data, and more efficient than the approach that operate by selecting a randomly data for learning, and this randomization can give a less accuracy like the boosting algorithm.

In the future work we try to perform more tests with other data from real-life and perform a deep study of the impact of reducing data, especially when data become very small where it is generated from a very complex concept.

References

[1] A. Asuncion, and D. J. Newman, UCI Machine Learning Repository, Irvine, CA: University of California, School of Information and Computer Science http://www.ics.uci.edu/~mlearn/MLRepository.html 2011.

[2] A. Globerson, N. Tishby, I. Guyon, and A. Elisseeff, Sufficient Dimensionality Reduction, Journal of Machine Learning Research, 2003.

[3] A. M. Molinaro, R. Simon, and R. M. Pfeiffer, Prediction error estimation: a comparison of resampling methods. Bioinformatics, 2005, pp. 3301–3307.

[4] D. Jensen, Induction with Randomization Testing: Decision-Oriented Analysis of Large Data Sets. Department of Engineering and Policy, Washington University, St.Louis Missouri, Doctoral Dissertation, 1992.

[5] H. Bouhamed,T. Lecroq, and A. Rebai, New Filter method for categorical variables selection, ijcsi Vol 9, May 2012: pp. 10–19.

[6] H. Mark, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and H. Ian. Witten The WEKA Data Mining Software: An Update, SIGKDD Explorations, Vol 11, 2009.

[7] J. R. Quinlan, C4.5 Programs for Machine Learning. Morgan Kaufmann Publishers,1993.

[8] K. Kira, and L. Rendell, The Feature Selection Problem: Traditional Methods and a New Algorithm, AAAI-92 proceedings, 10th International Conference on Artificial Intelligence, 1992.

[9] L. Breiman, Bagging Predictors Machine Learning, 1996, 123-140.

[10] M. Ojala and G. C. Garriga, Permutation Tests for Studying Classifier Performance 11 (Jun):1833− 1863, 2010.

[11] N. Tishby, F.C. Pereira and W. Bialek, The Information Bottleneck method. The 37th annual Allerton Conference on Communication, Control, and Computing, Sep 1999, pp. 368–377.

[12] P. Golland, F. Liang, S. Mukherjee, and D. Panchenko, Permutation tests for classification. In Annual Conference on Learning Theory, 2005, pp. 501-515.

[13] R. Gilad-Bachrach, A. Navot, and N. Tishby, Margin based feature selection: theory and algorithms, Proceedings of the twenty-first international conference on Machine learning, ACM, 2004, pp. 43-50.

[14] S. T. Roweis and L. K. Saul, Nonlinear Dimensionality Reduction by Locally Linear Embedding, Science Vol 290, 22 December 2000, pp. 2323–2326.

[15] T. Gruber, "Toward Principles for the Design of Ontologies Used for Knowledge Sharing". International Journal of Human-Computer Studies Vol 43, 1995, pp. 907-928.

[16] T. Hsing, S. Attoor, and D. Edward, Relation between permutation-test p-values and classifier error estimates. Mach. Learn., Vol 12, 2003, pp. 11–30.

[17] T. Oates, and D. Jensen, Large data sets lead to overly complex models: an explanation and a solution. In Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD-99).

[18] Y. Chong Ho, Resampling methods: concepts, applications, and justification. Practical Assessment, Research & Evaluation Vol 19, September 2003, 1-23.

[19] Y. Freund, and R. E. Schapire, Experiments with a New Boosting Algorithm, 1996 .

Appendix

Table 2 difference between measures over all data, and data generated with the concept **HighCustomer** and **LowCustomer**.

	Low Customer			**High Customer**		
Attribute	Entropy	Gain-ratio	RELIEF	Entropy	Gain-ratio	RELIEF
1	-0.03127	0.0151734	0.0253317	0.030937	-0.012195	-0.032896
2	-0.012811	0.0011656	0.0025493	0.012666	0.003617	-0.001212
3	-0.006429	-0.00314	0.0078571	0.006865	0.004704	0.0004929
4	-0.003705	-0.002422	-0.017811	0.005235	0.017741	0.0523371
5	-0.012017	0.0043134	-0.001314	0.007792	0.126392	0.0094429
6	-0.018213	0.0061115	-0.005845	0.015073	-0.000348	-0.011669
7	-0.017772	0.0041049	0.0271669	0.010821	0.002805	-0.027513
8	-0.011468	-0.000637	-0.006446	0.013845	0.002235	-0.006711
9	-0.009570	-0.002298	-0.006910	0.008154	0.007187	0.0151946
11	-0.018745	0.0139564	0.0101727	0.018661	-0.008343	-0.002775
12	-0.012503	0.0009004	0.0051686	0.014808	0.000662	-0.008634
13	-0.016024	0.0032032	-0.007764	0.017486	-0.001706	0.0004635
14	-0.018489	0.0101335	0.0056826	0.018282	-0.003629	-0.008621
15	-0.008429	-0.005418	-0.011192	0.009667	0.00786	0.0266908
16	-0.015779	0.0058941	0.0063753	0.015729	-0.001016	-0.008278
17	-0.015173	-0.007435	-0.001840	0.011677	0.156489	0.0082832
18	-0.015780	0.004775	0.0200493	0.0179	-0.002764	-0.020721
19	-0.012004	0.0002696	-0.005446	0.015281	-0.000058	-0.003535
20	-0.011862	-0.000068	0.011289	0.014939	0.000463	-0.006891
21	-0.007673	-0.020238	-0.005144	0.007343	0.022867	0.0083529
22	-0.013786	0.001417	0.0025964	0.013658	0.016148	-0.00091