Discovering decision rules from numerical data streams

Conference Paper · January 2004

DOI: 10.1145/967900.968036 · Source: DBLP

3 authors:

Francisco Javier Ferrer-Troyano
Universidad de Sevilla
28 PUBLICATIONS **145** CITATIONS
SEE PROFILE

Jesús S. Aguilar-Ruiz
Universidad Pablo de Olavide
227 PUBLICATIONS **2,478** CITATIONS
SEE PROFILE

José C Riquelme
Universidad de Sevilla
250 PUBLICATIONS **2,393** CITATIONS
SEE PROFILE

Discovering Decision Rules from Numerical Data Streams *

Francisco Ferrer–Troyano
Dept. of Computer Science
University of Seville, Spain
ferrer@lsi.us.es

Jesús S. Aguilar–Ruiz
Dept. of Computer Science
University of Seville, Spain
aguilar@lsi.us.es

José C. Riquelme
Dept. of Computer Science
University of Seville, Spain
riquelme@lsi.us.es

ABSTRACT

This paper presents a scalable learning algorithm to classify numerical, low dimensionality, high–cardinality, time–changing data streams. Our approach, named SCALLOP, provides a set of decision rules on demand which improves its simplicity and helpfulness for the user. SCALLOP updates the knowledge model every time a new example is read, adding interesting rules and removing out–of–date rules. As the model is dynamic, it maintains the tendency of data. Experimental results with synthetic data streams show a good performance with respect to running time, accuracy and simplicity of the model.

Keywords

Decision rules, scalable algorithms, data streams

1. INTRODUCTION

Medicine, meteorology, ATM transactions, retail chains, or scientific projects are some examples of different fields where gigabytes of numerical data streams are daily generated and mined under the assumption that they hide valuable information. Progress in hardware storage and data–warehouse technologies allow modern organizations to collect vast amounts of data from proprietary and client case histories. The inherent nonstop data traffic among heterogeneous sources gives rise to noise, missing, and inconsistency on attribute values. In addition, when data distribution is not stationary (examples are collected over months), algorithms based on data partitioning techniques (instance/feature sampling) are oversensitive to both underfitting and overfitting. Furthermore, memory and time limitations compel such systems to give an approximate answer from few scans (ideally only one) assuring that both result and performance are not adversely affected by the order of the examples. Mining potentially infinite data sequences implies high computational cost and usually results in large, complex and incomprehensible knowledge models, so interactive and user–controlled systems are becoming increasingly developed moving on the user's priorities to less accurate but more comprehensible answers. For all these reasons, designing new scaling–up and scalable learning algorithms has consolidated as an important challenge in recent years [11].

This paper introduces a scalable classification algorithm named SCALLOP (Scalable Classification ALgorithm by Learning decisiOn Patterns) that provides a model on demand according to several user–defined parameters. In the next sections we describe the motivation and the basis of our approach, discussing its major drawbacks. Next we present experimental results on numerical data sets that show its performance mining numerical, low–dimensionality, high–speed data steams.

2. MOTIVATION

Many scalable learning algorithms are based on decision trees, modelling the whole search space hierarchically as disjointed hypercubes. The highly complex trees given by these systems cast doubts on its capabilities as suitable knowledge representation due to the user need to explore paths of several dozen of levels to know interesting patterns. In addition, mining time–changing data streams may involve to rebuild an *out–of–date* sub–tree, increasing the computational cost to a greater extent. Within incremental learning, a common approach to extract the concepts to be learned consists in repeatedly applying the learner to a sliding window of w examples. An important issue of these approaches is to find the best value for w that optimizes the performance as a function of the input data [13].

Our proposal obtains a reduced set of *updated* decision rules sorted in a relevance order according to the user's demand. From several user–defined parameters, SCALLOP only models the regions whose characteristics interest the user, showing visually the obtained rules (see Figure 1). Contrary to decision–tree–based approaches, the whole search space is not modelled. Using a window of size 1, those examples located inside the most influential regions turn into hypercubes and extend its limits to the nearest different label regions. This approach makes the model to be initially unstable since some rules could be wrongly expanded, intersecting different labelled regions whose examples have not been read yet. To attain the stabilization of the model, SCALLOP associates growth limits with each rule preventing them to be extended. Such growth limits give an excellent way to classify by voting with a reduced set of rules, differently to decision lists.

3. THE SCALLOP ALGORITHM

Classification is generally defined as follows. An input finite data set of training examples is given. Every training example is a pair $e = (x, y)$ where x is a vector of m attribute values (each of which may be numeric or symbolic) and y is a class discrete value

*The research has been supported by the Spanish Research Agency CICYT under grant TIC2001-1143-C03-02.

SAC'04, March 14–17, 2004, Nicosia, Cyprus.

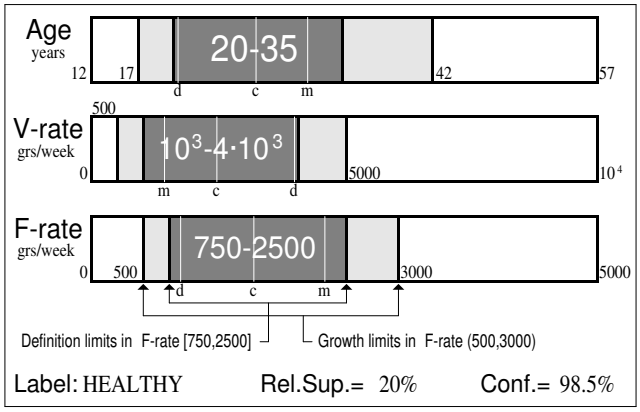

Figure 1: Example of a rule in SCALLOP: A young person will not be healthy if he/she eats less than 500 grs. of vegetables per week.

named label. The goal is to obtain a model $y = f(x)$ to classify or decide the label for new non–labelled test examples named queries. SCALLOP builds a model formed by several sets of decision rules, one set per label. Henceforth, the next notation is used to describe our approach. Let m be the number of continuous attributes. Let $Y = \{y_1, \ldots, y_z\}$ be the set of class labels. Let $e_i = (x_i, y_i)$ be the i^{th} training example to be read, where x_i is a normalized vector in $\mathscr{R}^m$ and y_i is a discrete value in Y.

Every decision rule in SCALLOP is a set of m closed intervals $[I_{jl}, I_{ju}]$ (one per dimension) named definition limits which define an hypercube inside the search space. l denotes lower bound and u upper bound.

DEFINITION 1 (EXAMPLE COVERING). *An example is covered by a rule when it belongs to the space given by its definition limits.*

DEFINITION 2 (POSITIVE SUPPORT OF A RULE). *The number of examples with label y covered by a rule R with label y is said positive support of R.*

DEFINITION 3 (NEGATIVE SUPPORT OF A RULE). *The number of examples with label y covered by a rule R with label y' is the negative support of R.*

DEFINITION 4 (CONFIDENCE OF A RULE). *Let ps and ns the positive support and the negative support of a rule R, respectively. The confidence of R is defined as: $C(R) = \frac{ps}{ps+ns}$.*

In order to achieve a balanced performance between the running time and the classification accuracy, each rule R has associated four relevant elements:

- **Centroid *C***: vector in $\mathscr{R}^m$ generated as the weighted mean of the vectors belonging to all the examples covered by R.
- **Delimiters *D***: set of the β examples covered by R that are farthest to each other. β is an user–defined parameter.
- **Markers *M***: set of β representative examples covered by R. M, D, and C are used to modify an invalid rule.
- **Growth limits *B***: set of m open intervals (B_{jl}, B_{ju}), so that:

$$B_{jl} \leq I_{jl} \leq I_{ju} \leq B_{ju};\ \forall j \in \{1, \ldots, m\}$$

The growth limits play an important role since if we know where every rule can be expanded to, the algorithm makes stable the model faster. Furthermore, this information is very helpful to classify new queries because there is no need for them to be covered.

In addition, every rule keeps its positive support and its negative support, the index of the last covered example, a boolean value that indicates if the rule was formed from another rule instead of an example, and a set of links to the rules with which overlap. Only rules associated with the same label may overlap.

Moreover, the rules are kept or removed according to several user–defined parameters: the maximum number of rules per label (α), the update rate, the minimum positive support, and the minimum confidence.

The algorithm starts with α rules per label generated from the first α read examples of each label. These rules are not hypercubes but points. When there have been read more than α examples for a certain label y_i, three different situations are differentiated with every new example $e_i = (x_i, y_i)$:

- ***Positive covering***: x_i is covered by one or several rules associated with the same label y_i.
- ***Possible expansion***: x_i is not covered by any rule in the model but there is at least one rule associate with the same label that can be extended to cover it without overlapping with a different labelled rule.
- ***Negative covering***: x_i is covered by one or several rules associated with a different label $y' \neq y_i$.

Cases 1 and 3 take turns to be firstly checked. If none of them come true, then the actions associated with the case 2 are run. After each pruning (every γ read examples), SCALLOP counts how many times the cases 1 and 3 come true. For the next γ examples SCALLOP will check firstly that case with the highest count according to the preceding γ examples.

In the two first cases SCALLOP updates the delimiters and the markers of the involved rules when they have covered β new examples (Figure 2). Let E be the set of the β latest examples covered by a rule so that $X = D \cup M \cup E$. The first delimiter selected $d1$ is the most distant point $d1 \in X$ to the centroid (C) of the rule. The first marker selected $m1$ is the nearest point to the middle point of $d1$ and C. The remaining delimiters are selected from X according to the greatest Euclidean distance from the centroid of new delimiters: the second delimiter $d2$ is the most distant point to $d1$ (c'), the third delimiter $d3$ is the most distant point to the centroid of $d1$ and $d2$ (c''), the fourth delimiter $d4$ is the most distant point to the centroid of $d1$, $d2$ and $d3$, and so on. The markers are updated with the same criterion of $m1$. The second marker $m2$ is the nearest point to the middle point of $d2$ and C. The third marker $m3$ is the nearest point to the middle point of $d3$ and C, and so on.

Positive covering: every rule that covers x_i increases its positive support by one unit, updates the index of the last covered example and moves its centroid. When the first rule R_p that covers the new example is found, the other rules that may also cover it are found thanks to the links of R_p to them.

Possible expansion: a rule R_g can be expanded to seize the point x_i if it fulfills two conditions:

- x_i is not beyond the growth–bounds of R_g, that is: $\forall j \in \{1, \ldots, m\} \cdot\ x_{ij} \in (B_{jl}, B_{ju})$
- The resulting extended rule does not intersect with any other rule associated with a label different from that of R_g.

Only one rule out of all the candidate ones is expanded: the one whose *growth* is the smallest and now covers x_i.

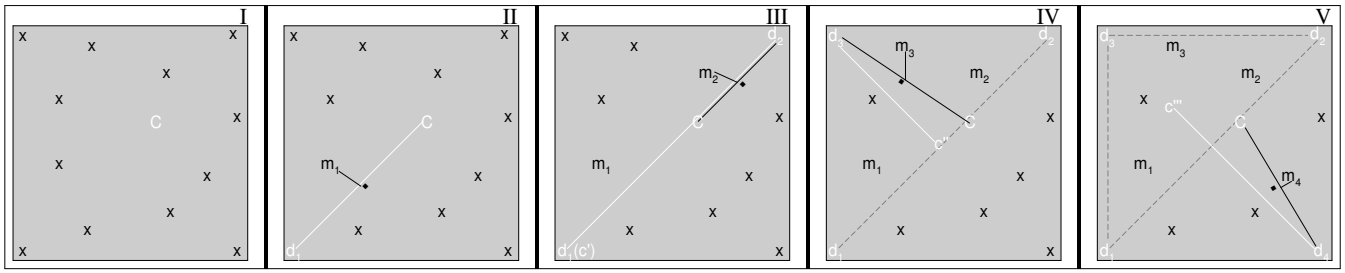

Figure 2: Updating the delimiters (d_i) and the markers (m_i) of a rule with $\beta = 4$.

DEFINITION 5 (GROWTH OF A RULE). *Let R be a rule in $\mathscr{R}^m$. Let x be a point in $\mathscr{R}^m$. The growth G of the rule R to cover the point x_i is defined as:*

$$G(R,x_i) = \prod_{j=1,\ g_j>0}^{m} 10^{\phi} g_j \ - \prod_{j=1,\ r_j>0}^{m} 10^{\phi} r_j;$$

$$g_j = u_j - l_j;\ u_j = \max(x_{ij}, R.I_{ju});$$
$$l_j = \min(x_{ij}, R.I_{jl});\ r_j = R.I_{ju} - R.I_{jl};\ \phi \in \mathscr{N}$$

In order to measure only the new region that is taken when a rule R extends, the *growth* takes into account only those dimensions for which there is expansion. Since SCALLOP normalizes each attribute value in [0,1] before processing an example, the term 10^{ϕ} is used to avoid that a rule without expansion in a certain dimension k ($I_{kl} = I_{ku}$) has greater *growth* than a rule with expansion in such a dimension and the same intervals in the rest of dimensions. For example, consider two rules in $\mathscr{R}^2$, R_a and R_b, which can be extended to cover a new point $x = \{0.5, 0.5\}$, so that: $R_a.I = \{[0.1, 0.4]; [0.5, 0.5]\}$ (a segment) and $R_b.I = \{[0.1, 0.4]; [0.6, 0.7]\}$ (a rectangle). Without the term 10^{ϕ}, it results in: $G(R_a, x) = 0.4 - 0.3$; $G(R_b, x) = 0.4 \cdot 0.2 - 0.3 \cdot 0.1$. That is, contrary to expected, *R_a grows more* than R_b. We have used $\phi = 3$ in our experiments.

When a rule R_g extends, it may overlap with other rules associated with the same label y_i so that SCALLOP updates the set of links for each rule in both directions.

Negative covering: when x_i is covered by a rule associated with a different label y', the covering rule R_n with the nearest centroid to x_i is founded as in the first case. If the new confidence of R_n is still greater than or equal to the minimum given by the user, then the negative support is increased by one unit. If the new confidence is smaller than the minimum given by the user, then a new rule R_x for x_i is added to the model. In addition, R_n is split into two new rules R'_n that do not cover x_i. Each new rule may be partially or totally covered by a previous rule (which must be linked through R_n). If R'_n is totally covered by other rule R_c, then it is not included in the model. If R'_n overlaps with R_c, then both rules update to each other the set of links. Before adding them to the model, the growth limits of each new rule are updated with the growth limits of R_n.

Although the new example x_i may be covered by several rules associated with a different label, only the rule whose centroid is the nearest to x_i is split. We have decided on this criterion under the assumption that if the example x_i belongs to a pattern, then near examples associated with the same label y_i must be read shortly after, and wrong rules will be corrected. If x_i is a noisy example or belongs to a minority pattern, then R_x will be removed in the next pruning. So every time a noisy example x is read, dividing only one rule instead of all the rules that cover x avoids an unnecessary computational cost.

3.1 Refining the model

The set of rules is refined every γ new examples. γ is an user–defined parameter. First, an iterative procedure is run to join rules associated with the same label. When no union is possible the procedure ends. In every iteration, the two nearest rules to each other whose union is possible are analyzed. The two nearest rules are those whose resulting volume is the smallest in relation to the volume of the rest of the possible unions. The union R_u from two rules R_a and R_b of the same label is done if two conditions are fulfilled: 1) R_u does not intersect with any rule associated with a different label; 2) the resulting hypercube is located inside the hypercube obtained from the growth bounds of R_a and R_b.

In a second step every rule has to satisfy two conditions to stay in the model: 1) must cover at least one of the last δ read examples; 2) the positive support must be greater than or equal to the minimum given by the user (as percentage of the total number of examples read at that time). δ is another user parameter. If noise is present in data, those *wrong* rules that stem from noise are likely to have a low support and a variable update rate. If after this prune the number n of rules is still greater than α, then they are sorted by both the positive support and the index of the last covered example, in a decreasing order, so that the last $n - \alpha$ rules are directly removed. The rules that stay in the model reset the negative support.

Before removing a rule R_r, some rules associated with a different label may update their growth bounds. If R_r was extended with one of the last δ read examples and was not split ever, then SCALLOP takes it as a valid minority rule (not noise). Therefore, the rules of different label that will remain in the model should not extend across the region given by the definition limits of R_r (Figure 4). To avoid wrong expansions that may involve a splitting shortly after, SCALLOP updates the growth bounds of every different labelled rule R_s that overlaps with R_r in all dimensions except one j, so that:

if $R_r.I_{jl} > R_s.I_{ju}$ **then** $R_s.B_{ju} \leftarrow \min(R_s.B_{ju}, R_r.I_{jl})$
if $R_r.I_{ju} < R_s.I_{jl}$ **then** $R_s.B_{jl} \leftarrow \max(R_s.B_{jl}, R_r.I_{ju})$

3.2 Classifying new queries by voting

If a new query Q is covered by a rule R_q, then Q is directly classified as the label associated with R_q. If there is no rule that covers the new query, SCALLOP tries to infer which labels are not possible for Q and it is classified by voting. Figure 3 shows this procedure. If the query is beyond the growth bounds of all the rules associated with a certain label l, then l is rejected to classify Q. If Q is beyond the growth bounds of a rule R_y with label y, then the votes against y are increased by one unit. If a rule R_t of label t can be extended to cover Q (it is inside the growth bounds of R_t and the resulting expansion does not intersect with any rule associated with a label different to y), then the votes for t are increased by one unit. Thus, the label assigned is that with the highest number of

Algorithm 1 classifyTest

Input: Q: Vector in $\mathscr{R}^m$;
Output: label: Discrete;
begin
  if there is a rule R_q that covers Q **then**
    label ← the label associated with R_q
  else
    for all label $y_k \in Y$ **do** $\{Y = \{y_1, \ldots, y_z\}\}$
      validLabel[k] ← **false**
      for all rule R associated with the label y_k **do**
        if Q is beyond the growth limits of R **then**
          votesAgainst[k] ← votesAgainst[k] + 1
        else
          if R can extend to cover Q **then**
            validLabel[k] ← **true**
            votesFor[k] ← votesFor[k] + 1
          end if
        end if
      end for
    end for
    label ← **decide**(validLabel,votesFor,votesAgainst,received)
  end if
end

Figure 3: Algorithm to classify new test examples.

votes. When two labels have the same number of votes, the label distribution (*received*) decides which is the class value for the new query.

4. EMPIRICAL EVALUATION

We have run all our experiments on an AMD x86/1.4Ghz and 256Mb DDR RAM PC running Windows XP. SCALLOP have been tested for 15 continuous attributes using a method similar to [4]. The concepts to be learned are created by randomly generation of decision trees with 8 levels (128 concepts to be learned). Each leaf is randomly assigned a class label between only two possible values, 0/1. The tree was grown in each internal node with a random pair (attribute,value) that is consistent with the path from the root to such a node. For every example of the training stream, the 15 attribute values are generated with a simple uniform number generator as a stream of pseudo–random numbers in the real interval [0,1]. The class label associated with each training example is then assigned according to the target tree. As in [4], we carried out all tests without ever writing the training examples to disk (i.e., generating them on the fly and passing them directly to the algorithm).

We have evaluated three aspects of the performance given by SCALLOP: the prediction accuracy, the stabilization speed, and the running time. They have been measured for different sizes of the training stream. We have added a class label noise level of 1% so that every 100 examples, one of them was passed with a random label. For each training set, 10% of examples were used for testing. We have carried out ten evaluations for each training and ten training for each size. The values used for the parameters of the algorithm were: $\alpha = 100$, $\beta = 3$, $\gamma = 10^4$, $\delta = 2 \cdot 10^4$, minimum confidence 90% and minimum positive support 0.01%.

Table 1 shows the results obtained with respect to the accuracy and the stability given by SCALLOP. The last row shows the results for a changing–tree, so that every hundred thousand examples the target tree was replaced making SCALLOP to reject the generated rules. From 1 million examples, the tree was stationary. The accuracy becomes stable from one million examples like the number of covered examples. It is important the high percentage of test examples correctly classified without direct cover for $5 \cdot 10^4$ tests (more than 37% of correctly classified) and for 10^5 (about 30%). The number of split rules does not increase under a linear trend but from one million examples tends to be asymptotic bounded. From five million examples, the number of final rules is very near to the number of concepts to be learned.

Table 1: Performance given by SCALLOP learning 128 concepts of 15 continuous dimensions. NE is the number of examples; CA is the classification accuracy; TC is the percentage of test examples covered by the ruleset; AC is the accuracy obtained by direct covering; NR is the final number of rules; NS is the total number of rules split during the process; and RP is the number of new rules generated before α examples of each label are read.

NE	%CA	%TC	%AC	NR	NS	RP
$5 \cdot 10^4$	66.0 ± 0.70	29.3	28.8	190	780	480
$1 \cdot 10^5$	74.0 ± 0.40	45.5	45.4	185	1400	1000
$2 \cdot 10^5$	84.0 ± 0.17	64.5	64.4	185	2150	2050
$3 \cdot 10^5$	91.5 ± 0.15	73.6	73.5	185	2550	3080
$4 \cdot 10^5$	93.0 ± 0.11	81.8	81.7	185	3150	4180
$5 \cdot 10^5$	94.0 ± 0.11	84.9	84.9	185	3225	5380
$1 \cdot 10^6$	95.3 ± 0.02	90.4	90.4	170	3330	12370
$5 \cdot 10^6$	95.8 ± 0.02	90.3	90.3	137	4000	72200

Table 2: Running time to build the model and classify 10% test examples. NE is the number of examples; TL is the time needed to built the model (in seconds); TC is the time to classify the test examples (in seconds); and %UE is the percentage of examples that are not used to update the model.

NE	TL	TC	%UE
$5 \cdot 10^4$	65	0.4	42
$1 \cdot 10^5$	115	0.6	33
$2 \cdot 10^5$	165	1.0	24
$3 \cdot 10^5$	210	1.3	18
$4 \cdot 10^5$	250	1.6	16
$5 \cdot 10^5$	290	1.6	15
$1 \cdot 10^6$	340	2.2	6
$5 \cdot 10^6$	925	10.8	1

Table 2 shows the results obtained in running time (seconds). Column TL shows that the running time to update the model is proportionally decreasing as the number of examples increases, so that the system's stability increases as the number of examples. Column UE shows that the quality of the rules is increasingly nearer to the real concepts to be extracted. These results lead us to think that SCALLOP is a good choice to mine major patterns from continuous data streams.

When the number of examples is over a million, SCALLOP is able to process about 5000 examples per second, what gives an idea of the good performance of our approach.

5. RELATED WORK

There is a huge literature on incremental learning and rule learning [3, 13, 5]. Decision tree based classifiers for mining very large databases are Gehrke et al.'s BOAT [6] and Agrawal et al.'s SPRINT [12]. BOAT obtains an approximate tree through a sample of fixed size. Previous approaches based on subsampling methods are also proposed by Catlett [2]. In contrast, SPRINT is a disk–based learner

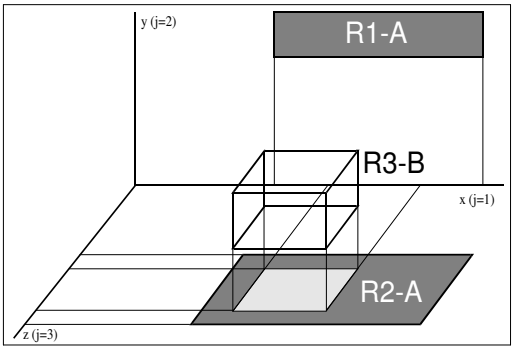

Figure 4: Updating the growth bounds of a rule in $\mathscr{R}^3$. The rules $R2$ and $R3$ overlap in two dimensions (x, z), whereas $R1$ and $R3$ overlap only in one dimension. $R2.B_{3u}$ is updated with $R3.I_{3l}$ and $R3.B_{3l}$ is updated with $R2.I_{3u}$ ($= R2.I_{3l} = 0$), respectively. $R1.B$ is not changed.

that use all the examples and focus on optimizing sequential access to disk.

Recent works on mining data streams has been introduced by Domingos et al. in [4] (VFDT) and [9] (CVFDT), building a decision tree using constant time and memory per example. Their approach is based on Hoeffding bounds [8], which guarantee that the output is asymptotically nearly identical to that given by a batch conventional learner from enough examples. They also apply Hoeffding's inequalities to build a scaling–up method that is applicable to any induction algorithm based on discrete search [10]. There is also large literature on scaling–up algorithms [7, 1].

6. CONCLUSIONS

A scalable classification learning algorithm based on decision rules and prototypes has been introduced in this paper. Providing a model on demand, which improves its simplicity and helpfulness for the user, we have developed a system for mining numerical, low–dimensionality, high–speed, time–changing data streams that updates the model with each new example. With a refining method as part of the algorithm, SCALLOP is able to remove *out–of–date* rules that have become uninteresting for the user and wrong rules caused by noise. This periodical pruning does not adversely affect the computational cost but rather speeds up its subsequent updating by helping to make the model more stable.

The strong point of our algorithm is that the generated rules *know* where can extend to, what provides few rules to classify new queries without decreasing the accuracy. This approach is different to decision tree based algorithms in that the whole search space is not modelled and the new queries are classified by voting. The performance of SCALLOP is excellent, as for prediction accuracy as running time.

7. FUTURE WORK

Our future research directions are oriented to drop irrelevant dimensions, and recover dropped attributes turned relevant later (there is no much literature on feature selection from data streams). We are also studying to deal with nominal attributes in order to be able to compare SCALLOP with another classification algorithms, as CVFDT [9] and SPRINT [12].

8. REFERENCES

[1] P.S. Bradley, U.M. Fayyad, and C. Reina. Scaling clustering algorithms to large database. *Knowledge Discovery and Data Mining*, pages 9–15, 1998.

[2] J. Cattlet. *Megainduction: machine learning on very large databases*. PhD thesis, Basser Department of Computer Science, University of Sydney, Australia, 1991.

[3] W.W. Cohen. Fast effective rule induction. In Armand Prieditis and Stuart Russell, editors, *Proc. of the 12th International Conference on Machine Learning*, pages 115–123, Tahoe City, CA, July 9–12, 1995. Morgan Kaufmann.

[4] P. Domingos and G. Hulten. Mining high-speed data streams. In *Proc.* 6^{th} *ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining*, pages 71–80, Boston, MA, 2000.

[5] V. Ganti, J. Gehrke, and R. Ramakrishnan. DEMON: Mining and monitoring evolving data. *Knowledge and Data Engineering*, 13(1):50–63, 2001.

[6] J. Gehrke, V. Ganti, R. Ramakrishnan, and W.Y. Loh. BOAT – optimistic decision tree construction. In *ACM SIGMOD Conference*, pages 169–180, Philadelphia, Pennsylvania, 1999.

[7] J. Gehrke, R. Ramakrishnan, and V. Ganti. Rainforest – a framework for fast decision tree construction of large datasets. In *Proc.* 24^{th} *Int. Conf. Very Large Data Bases, VLDB*, pages 416–427 , 1998.

[8] W. Hoeffding. Probabilities inequalities for sums of bounded random variables. *Journal of American Statistical Association*, 58:13–30, 1963.

[9] G. Hulten, L. Spencer, and P. Domingos. Mining time-changing data streams. In *Proc.* 7^{th} *ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining*, pages 97–106, San Francisco, CA, 2001. ACM Press.

[10] G. Hulten, L. Spencer, and P. Domingos. Mining complex models from arbitrarily large databases in constant time. In *Proc.* 8^{th} *ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining*, Edmonton, Alberta, Canada, 2002. ACM Press.

[11] F. Provost and V. Kolluri. A survey of methods for scaling up inductive algorithms. *Data Mining and Knowledge Discovery*, 3(2):131–169, 1999.

[12] J.C. Shafer, R. Agrawal, and M. Mehta. SPRINT: A scalable parallel classifier for data mining. In *Proc.* 22^{th} *International Conf. Very Large Databases, VLDB*, pages 544–555, 1996.

[13] G. Widmer and M. Kubat. Learning in the presence of concept drift and hidden contexts. *Machine Learning*, 23(1):69–101, 1996.