The algorithm GENREG for generating rules from symbolic or numerical data

Henri Ralambondrainy

INRIA

UNITE DE RECHERCHE
INRIA-ROCQUENCOURT

Institut National
de Recherche
en Informatique
et en Automatique

Domaine de Voluceau
Rocquencourt
B.P 105
78153 Le Chesnay Cedex
France
Tél. (1) 39 63 55 11

Rapports de Recherche

N° 910

THE ALGORITHM GENREG FOR GENERATING RULES FROM SYMBOLIC OR NUMERICAL DATA

Programme 5

Henri RALAMBONDRAINY

Octobre 1988

THE ALGORITHM GENREG FOR GENERATING RULES FROM SYMBOLIC OR NUMERICAL DATA

L'ALGORITHME GENREG POUR LA GENERATION DE REGLES A PARTIR DE DONNEES SYMBOLIQUES OU NUMERIQUES

Henri RALAMBONDRAINY

INRIA : Domaine de Voluceau Rocquencourt B.P.105, 78153 LE CHESNAY Cedex,FRANCE.

RESUME :

Une des difficultés essentielles lors du développement de systèmes experts est l'acquisition de la connaissance nécessaire pour la construction d'une base de connaissances. Cet article présente un algorithme d'apprentissage, qui à partir de tableaux de données rectangulaires, classiquement traités par les techniques d'Analyse de Données, ou de données symboliques, se présentant sous forme d'assertions et munies de connaissances supplémentaires, de générer des règles de production directement exploitables par un système expert.

ABSTRACT :

One of the most important problem which occurs when developing expert systems is to find the required knowledge for building the knowledge base. We present here a learning algorithm, which generates from data arrays or symbolic data with background knowledge, a set of production rules which can be inserted in an expert system knowledge base.

KEY WORDS : Data analysis , learning, expert system,.

1. INTRODUCTION

At present, there is a growing interest in learning techniques. One main reason is that learning techniques offer a solution to the knowledge acquisition bottleneck for a wide development of Expert Systems. Different approaches have been proposed (Michalski 1983)(Kodratoff 1986) : learning by analogy, learning by discovery,etc.We are concerned here with techniques of learning from examples. The goal of these methods is to find rules to recognize a given concept described by a set of examples and counter-examples. For instance, the series of programs AQ11, INDUCE, proposed by Michalski, described data in an extended predicate calculus notation. They find discriminating rules, which recognize the examples, using knowledge about the domain such as structures on descriptor, theorems,etc,(the background knowledge). On the other hand, the ID3 program (Quinlan 1982) uses vectors to represent data. It builds classification rules, in the form of a decision tree, using information theory criteria to measure the discriminating power of a descriptor. This last method may be related to the decision tree methods proposed in Data Analysis (D.A.) (Breiman & al 1982). They both use a low level representation of data and numerical or statistical measures for selecting discriminating descriptors, in the generating rules process.

In the system SICLA, powerful decision tree methods of D.A. are proposed for dealing with numerical data (Celeux 1982) without background knowledge. The GENREG program is an example of a series of programs which are going to deal with symbolic data (assertions, rules, etc.) defined in Diday (1988).The algorithms proposed have the following characteristics:

-the input data are represented by couples (attribute, value); missing, multivalued values are allowed,

-production rules are provided having the format : IF... THEN...

-if knowledge domain is provided, it will be used in the generating rules step, but the program can process rectangular data without background knowledge,

-the method is suited to noisy data and the rules provided are probalistic.

Section 2 presents the different types of descriptors and the generalization rules used. Section 3 describes the algorithms and section 4 deals with the case of nominal data with an application example.

2. THE DESCRIPTION LANGUAGE

2.1. The different types of descriptor :

We note Ω a set of n objects given on which p descriptors are measured . To each descriptor is associated :

1) A text which is used to denote the descriptor,

2) A specification of the domain D or the set of values that is possible for the descriptor. The domain determines the type of the descriptor (described below),

3) A function or variable denoted X defined on the objects Ω having values in D.

Depending on the structure of the domain, we have the following types of descriptors :

Nominal descriptors :

The value set D is finite and unstructured. For instance the descriptor "sex" is nominal and its domain is : sex ={man,woman}.

Hierarchical descriptors :

The value set D is finite and has a hierarchical structure. Let D^0 denote a set of elementary values, D has a hierarchical structure on D^0 if :

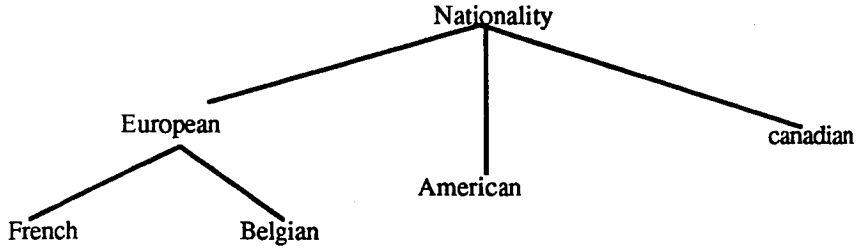
1) D is a set of parties of D^0 : $D \in \mathbf{P}(D^0)$,
2) $D \supset D^0$, D^0 is the set of the leaves,
3) if d_j and d_k are two elements of D, we have : $d_j \cap d_k \in \{\varnothing, d_j, d_k\}$.

We shall say that the hierarchy is total if the root $\{D^0\} \in D$ and partial otherwise. We shall consider only total hierarchy because it is easy to make a total hierarchy from a partial one by introducing the set $\{D^0\}$ in D. For example the descriptor "nationality" is a partial hierarchy:

"nationality" = {French, Belgian, European={French, Belgian}, American, Canadian}

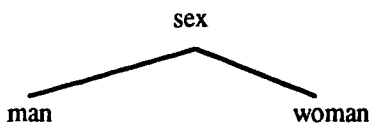
we make it total , by considering :

"nationality" = {French, Belgian,European={French, Belgian}, American, Canadian, Nationality = {French, Belgian, American, Canadian} }



If ω is an object of Ω and X the function associated to the descriptor, the value of the function X(ω)=European is interpreted as : X(ω)=French or Belgian.

A nominal descriptor will be seen as a hierarchical descriptor by considering the trivial hierarchy obtained by introducing the set of the values in the domain. For example, the nominal descriptor sex has the following hierarchy structure : sex={man, woman, sex={man,woman}}



Graph descriptors :

A hierarchy is a particular graph. If D^0 is still a set of basic and elementary values, a graph descriptor, associated to D^0, has for domain D which is a set of parties of D^0. If d_j and d_k are two elements of D, there will be a vertain : $d_j \rightarrow d_k$ if $d_k \supset d_j$. As previously, we shall consider that $\{D^0\} \in D$.

Numerical descriptors :

The domain is the set of numbers or intervals. For example, weather, temperature or the weight of a person are numerical descriptors.In this paper, we only consider the case of descriptors whose domains are finite and discrete. If the descriptors are numerical, they will be coded into nominal descriptors. Different D.A. techniques (histograms for instance) may be used to make such a coding. For instance, the numerical variable age may be coded as follow:

	age <20	: young
20<=	age <55	: middle - age
55<=	age	: old

2.2. Conceptual structure on the descriptors :

To each value d of D, we denote $\omega_{X=d}$ the set of observations of Ω which takes the value d for the variable X : $\omega_{X=d} = \{\omega \mid \omega \in \Omega \text{ and } X(\omega)=d\}$ and A=(X=d) the characteristic function of $\omega_{X=d}$. The assertion A=(X=d) is a logical function defined on Ω such that : $A(\omega)$='true', if $X(\omega)$=d and 'false' otherwise.

Each value d of D is interpreted as a *concept* which is characterized by the assertion (X=d). The values of D are partially ordered by the operator inclusion. The structure of D is not simply a hierarchy on a set of parties but it is a conceptual hierarchy which reflects the *generalization* relation between the concepts. We will say that the concept d_2 is *"more general "* than the concept d_1 : $d_1 < d_2$, if the following equivalent conditions are verified :

1) $d_2 \supset d_1$
2) $(X=d_1) \Rightarrow (X=d_2)$
3) $\omega_{X=d2} \supset \omega_{X=d1}$

2.3.The generalization rules:

Let d be a concept of D, the *climbing hill generalization* rule allows us to define the set of concepts of D more general than d : $gen(d) = \{d_m \mid d_m \in D \text{ and } d < d_m\}$. When X is a hierarchical descriptor, the set gen(d) contains d and its parents. If X has a graph structure, the set gen(d) contains d and its successors.The set gen(d) contains at least d and D^0.

We extend this definition to a set of concepts $(d_1, d_2,...,d_q)$ by defining :

$$gen(d_1, d_2,...,d_q) = \{ d_m \mid d_m \in D \text{ and } d_j < d_m \text{ for } j=1,...,q\}$$

We have p descriptors, we denote Xj the function, D_j the domain constructed on the basic values D^0_j , related to the j-th descriptor. We denote by j the current index of the set $J=\{1,...,p\}$.

We will consider only assertions having the following form : $A=\wedge\{ (Xj=aj) \mid j\in J'\}$ where $J \supset J'$ assertions which are conjunctions of elementary assertions $(Xj=aj)$. The assertion A may be seen as an assertion with p terms $A=\wedge\{ (Xj=aj) \mid j\in J\}$ by adding to the assertion A, for $j\in J - J'$, the terms $(Xj=aj)$ where $a_j = D^0_j$. The assertions $(Xj=D^0_j)$ are logical functions which are always true. Each observation ω of Ω will be represented by an assertion $A_\omega=\wedge\{(Xj=aj) \mid j\in J\}$.

The assertion $B =\wedge\{(Xj=b_j \mid j\in J\}$ is more general than the assertion $A=\wedge\{ (Xj=aj) \mid j\in J\}$ if each b_j is a concept of D which is more general than a_j : $b_j \in$ gen (a_j) for $j\in J$.

The *dropping generalization* rule comes down to taking the root D^0_j as the generalization of a_j.The length of an assertion is defined as the number of its terms $(Xj=b_j)$ where b_j is different from D^0_j.

If we have two assertions $A=\wedge\{ (Xj=aj) \mid j\in J\}$ and $B=\wedge\{ (Xj=b_j \mid j\in J\}$, we generalize them with an assertion $C=\wedge\{ (Xj=cj) \mid j\in J\}$ where c_j is a concept which is more general than a_j and b_j : $c_j \in$ gen(a_j,b_j).

If the variable Xj is hierarchical, we choose for c_j the minorant of the set gen(a_j,b_j). When a descriptor has a graph structure the set gen(a_j,b_j) may have many minorants. In this paper, we do not yet consider this case.When we have interval variables Xj=[a,b] and Xj=[c,d] we generalize these assertions, using the *extension domain generalization* rule : Xj=[min(a,c),max(b,d)]

2.4. Extension to multivalued data :

In the previous section, we always suppose that the variables are monovalued. But data in input may be multivalued as for instance : $A=(X^1=a_1\vee a_2)\wedge(X^2=b_1)\wedge(X^3=c_1\vee c_2\vee c_3)$.The extension of the generalization rules to multivalued data is trivial because the assertions $(Xj=a_1\vee a_2...\vee a_m)$ and $(Xj=b_1\vee b_2...\vee b_p)$ are generalized by the assertion Xj=c where c is a concept of D_j which is more general than the set $(a_1,...,a_m,b_1,...,b_p)$.

3. THE ALGORITHM :

The observations are partitioned into two classes, the example set E and the counter-example CE of a concept to be discriminated. The problem is to find a set of *good* assertions which recognizes the examples and not the counter-examples. We will first give the properties required by the assertions solution and the describe two algorithms to obtain them.

Properties of the assertions :

The systematic research of all the assertions which recognize the examples and not the counter-examples is a combinational problem. To reduce the complexity, we look for a set of assertions which verifies discriminating and covering criteria (conditions 2 and 3). More precisely the assertions must satisfy the three following conditions:

1) An assertion A has to be composed of l_{min} terms at least and l_{max} terms at most :

$$l_{min} < \text{length}(A) < l_{max}$$

Numbers l_{min} and l_{max} are integers between 2 and p. This condition is essentially used to reduce the research space. Otherwise, it is admitted that assertions with more than 7 terms are difficult to be apprehended by a human then l_{max} will be in general inferior to 7 .

2)An assertion A has to be α-discriminating. This means that the number of counter-examples recognized by A must be inferior to α :

$$\text{card}(\omega_A \cap CE) <= \alpha$$

If α=0, the assertion A does not recognize any counter-examples. Most of the time, learning algorithms assume that examples and counter-examples describe perfectly the concept and research assertions which recognize no counter-examples. In practice, for statistic data it is rarely true, which is why the threshold α may be superior to 0.

3) An assertion A must recognize β examples at least :

$$\beta < \text{card}(\omega_A \cap E)$$

This condition allows the elimination of assertions which reflect too particular observations and provides security for a certain robustness of results. The study of the number of examples and counter-examples recognized by the most general assertions ($X_j=d_j$) for $j \in J$, where $d_j \neq D^0_j$ is the most general concept of D_j ,will give an idea for the choice of α and β.

The generalization algorithm:

The first algorithm is a *data driven* algorithm. This means that starting from the observations, the algorithm generalizes each of them to obtain assertions which verify the previous conditions. More precisely, we have two steps :

The generation step :

The assertions which generalize each couple of observations is computed using climbing hill and dropping generalization rules. Assertions, which are too small (condition 1) and not

sufficiently discriminating (condition 2), are eliminated. The whole process is repeated on these first generated assertions, and so on, until stabilization. The process converges because we have a finite number of examples and the generated assertions' lengths become smaller and smaller. Each assertion of this last set are still generalized using the climbing generalization rule and only assertions which satisfy the discriminating criterion are kept. We then take out of the rule base those particular assertions which does not recover enough examples (condition 3).

The simplification step :

Let S be the resulted assertions set from the previous step. The simplification of S is made using the following rules :

-If A and A$\wedge$B are assertions of S, the assertion A$\wedge$B will be suppressed because it is more particular than A.

-If A, B,C, ...,Q are assertions of the set S and such that A $\Rightarrow$ B$\vee$ C $\vee$ ...$\vee$ Q, A is suppressed because it is less general than B$\vee$ C $\vee$ ...$\vee$ Q. To make such a simplification, we order the assertions of S according to a criterion (the number of examples recognized or the probability associated to an assertion). We start the list of the assertions solution with the best assertion of S according to the previous criterion. A new assertion candidate will be added to the solution list if it is *not* more particular than the union of the assertions selected in the list.

If we denote the final assertions set $\{S_m \mid m \in M\}$, the recognizing function of the examples will be the assertion : $S=\vee\{S_m \mid m \in M\}$.

The specialization algorithm:

The second algorithm is a *model driven* algorithm (Quinqueton 1983), (Gascuel 1986). Starting from the most generalized assertions, it specializes them in order to find the good assertions which verify the previous conditions. It differs from the previous algorithm only at the generation step.

The generation step :

Starting from the most general assertions $(X_j=d_j)$, where $d_j \neq D^0_j$ is the most general concept of D_j, these are specialized by adding new terms such as $(X^k=d_k)$ with $j \neq k$ until the resulted assertions do not recognize more than β examples and less than α counter examples :

$$A=(X_j=d_j)\wedge(X^k=d_k)\wedge...$$

where $d_j \in D_j$ and $d_k \in D_k$. Thus we build a tree of assertions candidates whose lengths are inferior to l_{max}. These assertions will be the input of the simplification step.

The generalization algorithm is more efficient than the specialization one when there are many descriptors, important background knowledge and few observations because the generalization process starts from the observations. The specialization algorithm is better when there are many observations and not great knowledge about descriptors because the generalization process generates the candidate assertions by combining the different concepts.

Quality criteria for an assertion :

Let $E_S=\omega_S \cap E$ be the examples set recognized by an assertion (or a set of assertions) S. We denote ρ_{wc} the percentage of well classified observations that is to say the percentage of examples recognized by S. It is a criterion for measuring the quality of the recognition potential of an assertion :

$$\rho_{wc}= \text{card}(E_S) \times 100 / \text{card}(E)$$

In the same way, let $CE_S=\omega_S \cap CE$ be the counter-example set recognized by S . The discrimination quality of the system S is measured by the percentage of misclassified objects ρ_{bc}:

$$\rho_{bc}= \text{card}(CE_S) \times 100 / \text{card}(E)$$

If we choose for α the value 0, the assertions of S do not recognize any counter-example and $\rho_{bc}= 0$.

The recognizing rule of the examples E has the following form : if S then E with the probability P_S :

$$P_S = P(E \mid S) = \frac{P(E \wedge S)}{P(S)} = \frac{\text{card}(\omega_S \cap E)}{\text{card}(\omega_S)} = \frac{\text{number of examples recognized}}{\text{number of observations recognized}}$$

The probability P_S is maximal, equal to 1 when S does not recognize counter-examples. P_S is the probability of good classification as an example for an observation which verifies S and 1-P_S the probability of misclassification.

Algorithm optimality :

We can prove that the proposed algorithms are optimal from the following point of view : it finds the most covering assertion(s) at the imposed conditions. The specialization algorithm starts from the most general assertions and specializes them. It is clear that it finds the most general assertion(s) at the imposed conditions and then the assertion(s) which recognizes the most numerous of examples.

4. THE STUDY OF NOMINAL DATA :

4.1 Generalization and similarity measure :

We consider in this section that the observations are totally described with nominal descriptors i.e. there is no background knowledge. Such data (surveys for instance) are currently studied with Data Analysis techniques such as decision tree method. D.A. techniques deal with data matrices where observations are represented by vectors with p components. In our case the observation $\omega \in \Omega$ may be represented either by the vector $\omega=(a_1,a_2,...,a_p)$ or by the assertion $A=\wedge\{ (X_j=a_j) \mid j \in J\}$.

When the descriptor Xj is nominal, the generalization of the assertions $(X_j=a_j)$ and $(X_j=b_j)$ is the assertion $(X_j=c_j)$ where $c_j = gen(a_j,b_j) = a_j$ if $a_j = b_j$ and D^0 if not.

The assertion which generalizes : $A=\wedge\{ (X_j=a_j) \mid j\in J\}$ and $B=\wedge\{ (X_j=b_j \mid j\in J\}$ is the assertion : $C=\wedge\{ (X_j=c_j) \mid j\in J\}$ where $c_j = gen(a_j,b_j)$ for $j\in J$. The generalization rule used, in this case, is the dropping rule because we do not have background knowledge.

Let M be the indexes set related to the common terms of A and B: $M=\{m \mid m\in J$ and $a_m=b_m\}$. If M is not empty then the *generalization* of A and B is the longest assertion which is more general than A and B which is defined as follows:

$$gen(A,B) =\wedge\{ (X^m=a_m) \mid m\in M\}$$

Let s(A,B) denote the number of the common terms of A and B.

s(A,B) = 0 if A and B have no common term
s(A,B)=m if they have m terms in common, m is the length of gen(A,B)

If we consider the vectorial representation of the observations, the number s of the common terms or components of two vectors defines a similarity measure on the observations :

$$s : \Omega \times \Omega \rightarrow \{0,1,...,p\}$$
$$(\omega,\omega') \rightarrow s(\omega,\omega')=m$$ where m is the number of common components of ω and ω'

The calculation of the length of the assertion which generalizes the assertions related to two observations ω,ω' is equivalent to the calculation of the similarity measure between the related vectors.

At the first iteration,if n_E denotes the number of examples, one must calculate $n_E(n_E-1)/2$ similarity measures to obtain the generalized assertions. Algorithm complexity is at least n_E^2. Nevertheless, it is possible by appropriate methods to reduce it. If it is possible to find a partition $P=(P_1,...,P_K)$ of the set of examples E, such as : $\forall\omega_i\in P_i$ and $\forall\omega_j\in P_j$ with $i \neq j$ and $s(\omega_i,\omega_j) > l_{min}$ then it is not necessary to calculate similarity measures for objects which belong to distinct classes of the partition. If n_k denotes the cardinality of the class k, the algorithm complexity becomes : $n^2_1 + n^2_2 + ...+ n^2_K << n_E^2$ Such a partition can be realized by a descending connexity algorithm related to the single link method (Diday 1972).

4.2 An application example :

The data concern 169 desk computers which cost more than 20000 francs (Ordinateur Individuel March 1985). Each computer is described with the following descriptors :

1. color monitor : (no, optional, yes)
2. CP/M system : (no, optional, yes)
3. MS/DOS system : (no, optional, yes)
4. other operating system : (no, optional, yes)
5. micro-processor : (8bits, 16bits, 32 bits)
6. parallel interface : (no, optional, yes)
7. serie interface : (no, optional, yes)
8. IEE488 interface : (no, optional, yes)
9. hard disk: (no, yes)

We first process a clustering D.A. method on the data and we obtain a partition having 5 clusters. We are only interested in the cluster 1 (30%) and the variables characteristic of the class, according to the chi-square measure, are listed in the figure 1. This class is related to 16 bits computers running under MS/DOS system and with optional color monitor and parallel or series interfaces.

Class : 1 **number** : 50, 30%

variable	chi-square	number	perc./class	perc.
Color monitor : optional	106	47	94	36
Micro-processor : 16 bits	51	49	98	56
MS/DOS operating system : yes	44	45	90	50
Other operating system : no	9	44	88	71
Parallel interface : yes	8	47	94	80
Series interface : yes	6	48	96	86

Figure 1

Then, we consider this class as a set of examples and apply the generalization algorithm GENREG to find recognizing rules. The resulted rules are listed below :

Rule 1: well classified 44 (88%), misclassified 0

If color monitor = optional and microprocessor = 16 bits and parallel interface = yes
then class 1

Rule 2: well classified 41(82%), misclassified 0

If color monitor = optional and MS/DOS system = yes and microprocessor = 16 bits
then class 1

Rule 3 : well classified 41(82%), misclassified 0

If color monitor = optional and MS/DOS system = yes and other system = no
then class 1

Rule 4 : well classified 41 (82%), misclassified 0

If color monitor = optional and MS/DOS system = yes and parallel interface = yes
then class 1

This set of rules recognizes 92% of the class 1.

5. Concluding remarks

The algorithms GENEREG is actually applied to various problems :

-to build a knowledge base to control a flow shop, this application is described in (Pierreval 1988).

-to build a rule-based system which can be applied in the decision process of a mail-order firm concerning its consumers.

The major interest of our approach is that the algorithms proposed can process both important data described by low level features and symbolic data with background knowledge. We expect that the methods presented here will be easily extended to process more complex objects described in Diday (1988).

6. References

Brieman, L, and al, Classification and regression by trees .
Addison-Wesley (1982).

Celeux, G, Lechevallier, Y, Non Parametric Decision Trees by Baesian Approach.
Compstat 82, 5th Symposium, Physica Verlag Wien (1982).

Diday, E, Optimisation en Classification Automatique et en Reconnaissance des Formes.
Note scientifique numéro 6, bulletin Inria numéro 12 (1972).

Diday, E, The symbolic approach in Clustering and related Methods of data Analysis.
Classification and Related methods of Data Analysis. Proceedings of the first Conference of the Federation of the classification societies. North Holland (1988).

Gascuel , O, PLAGE : A way to Give and Use Knowledge in learning.
First European Working Session on Learning,LRI, Orsay (1986).

Kodratoff, Y, Leçons d'Apprentissage Symbolique Automatique.
Collection Techniques Avancées de l'Informatique Cepadues-Editions (1986).

Michalski, R, A theory and method of inductive learning.
Machnie Learning : an artificial intelligence approach. Chapitre 4.
Michalski R, Carbonell J, Mitchell T, Tiago Press, Palo Alto(1983).

Pierreval ,H, Ralambondrainy, H, Generation of knowledge about the control of a flow shop using Data-Analysis oriented learning techniques and simulation.
Rapport Inria numéro 897, Septembre (1988).

Quinlan,J, Semi-autonomous Acquisition of Pattern-based knowledge.
Introductory Readings in Experts Systems, editor Donald Michie, Gordon & Breach (1982).

Quinqueton, J, Sallantin, J, Algorithms for learning logical formulas.
William Kaufmann,CA, (1983).

Ralambondrainy, H, A clustering method for nominal data and mixture of numerical and nominal data.
First conference of the international federation of classification societies, Aachen (1987).