Inductive Inference of First-Order Models from Numeric-Symbolic Data

Floriana Esposito, Sergio Caggese, Donato Malerba, and Giovanni Semeraro

Dipartimento di Informatica, Università degli Studi
via Orabona 4, 70126 Bari - Italy
{esposito, caggese, malerba, semeraro@lacam.uniba.it}

Abstract

A factor common to statistical techniques of data analysis is the adopted representation formalism: A tabular (zeroth-order) model with almost exclusively numerical features. On the contrary, several studies on machine learning concern the induction of first-order models from symbolic data, but are inadequate for continuous data. In the paper, we face the problem of handling both numerical and symbolic data in first-order models, distinguishing the moment of model generation from examples (induction) from the moment of model recognition by means of a flexible, probabilistic subsumption test. We demonstrate the proposed solutions on a problem in document understanding, where the objective is to induce the models of the logical structure of some real business letters.

1. Introduction

For decades, data analysis has been considered to be of exclusive interest to applied statisticians, who have developed several techniques for multi-dimensional classification and clustering, discriminant analysis, regression and factorial analysis. A factor common to all these techniques is the adopted representation formalism: A tabular model with almost exclusively numerical features. Studies on categorical data analysis [1] and symbolic data analysis [7] are an exception. Statistical methodology for categorical data analysis is mainly concerned with the analysis of categorical (that is, non-numerical) dependent/independent variables, but it still pays little attention to the problem of how to search for good models. The main goal is that of testing whether a model adequately fits the sample data, while strategies for model generation and selection are only superficially examined. Actually, the problem of model selection has been faced since early studies on information measures for classification [21], but no attention has been paid to the process of model generation. As a matter of fact, the problem of defining a generality order of models that organizes the space of hypotheses has been completely ignored in statistics, even though the comprehension of the algebraic structure of the model space is the first step toward the construction of efficient search strategies [14]. The first attempt to fill this gap can be found in symbolic data analysis with the idea of partially ordered assertion objects [5].

The steady growth of the machine learning branch within the area of artificial intelligence has introduced alternative representations and methods to analyze the data. One major dimension along which to differentiate machine learning techniques is the complexity of the representation language they adopt. Most of the machine learning studies involve propositional attribute-value languages, which are comparable to tabular representations adopted in statistics, but differ in the possibility of mixing both numeric and symbolic data.[1]

A typical example is represented by decision tree induction methods that work on continuous, nominal, ordinal and tree-structured attributes [2]. The main weakness of such methods derives from a lack of expressive power when relations among sub-parts have to be described. Indeed, even in the case of a fixed number of sub-parts and binary relations, the idea of defining a distinct attribute for each possible relation between two sub-parts would be realizable only if sub-parts could be totally ordered according to some criterion. Otherwise, the association of an attribute to pairs of sub-parts would not be deterministic, hence not expressible in tabular form.

To overcome these limitations of propositional attribute-value representations, it is necessary to move to first-order

1. Henceforth, the term numeric will denote integer and ratio-level measurements, while symbolic data will correspond to nominal and ordinal level measurements.

logic (FOL). However, the change of language is not painless, since the intractability of predicate calculus and the NP-completeness of several decision problems raise new problems that can only be solved by imposing appropriate restrictions. In other words, a trade-off between expressiveness and computational tractability is necessary. These aspects have been widely investigated in machine learning, particularly by researchers working in the field of inductive logic programming (ILP) [15]. A great deal of research in the field of ILP concerns generalization models and the corresponding generalization/specialization operators. However, very few studies have been performed on the problem of inductive inference from both numerical and symbolic data, and no studies have faced the problem of dealing with rules that near-miss the correct entailment of positive examples. As a consequence, the applicability of several learning systems to a range of problems is jeopardized.

A first attempt to deal with continuous-valued attributes in FOL was made by Bergadano and Bisio [3], who proposed a method to automatically set some parameters of predicate "schemes" with a parametric semantics. Some years later, the idea was resumed by Botta and Giordana [4], who implemented a two-step approach in the system ML-SMART: First a tentative numerical parameter is learned and then a standard genetic algorithm is applied to refine the numerical knowledge. Esposito *et al.* [10] proposed a different approach to the problem of handling both numerical and symbolic data. It was based on the integration of statistical data analysis with symbolic concept learning methods. More precisely, the authors combined a discriminant analysis technique for linear classification with a first-order learning method, so that the numerical information was handled by linear classifiers, while the symbolic attributes and relations were used by the first-order learning system.

One limit of all these approaches is that they have been defined for clauses with nullary predicates in the head, that is, with predicates corresponding to propositional classes. This means that such rules can be used to predict the membership class of an observation *as a whole*, but they cannot entail properties of sub-parts of an observation. For instance, the following clause:

House ← *red(x), ontop(x,y), triangle(x), block(y).*

can be used to classify houses with a red triangular sub-part on top of a square, but cannot be applied to the problem of understanding which part of the house is the roof. In the latter case, non-nullary predicates are necessary in the head of a clause, as in the following definite clause:

roof(x) ← *red(x), ontop(x,y), triangle(x), square(y).*

Recently, Dzeroski *et al.* (1995) have proposed the transformation of first-order representations into propositional form, in order to handle real numbers by means of techniques already tested in decision tree induction systems. However, the transformation into propositional form is possible only when all variables appearing in the body of a clause also appear in the head [12]. This restriction on the type of hypotheses sets a limit on the applicability of the approach. On the contrary, FOIL 6.0 [19] does not suffer from such a limit. The system automatically produces comparative literals of type $V_i > k$, $V_i \geq k$, $V_i > V_j$, $V_i \leq V_j$, where V_j are numerical variables already present in other non-comparative literals and k is a numerical threshold. One problem with FOIL is that it neglects the issue of rules that near-miss the correct entailment of observations. The problem becomes particularly relevant when numerical descriptions are involved. Indeed, a comparative literal of the type $V_i > k$ is falsified for any value just smaller than or equal to k. Therefore, a classical subsumption test should be replaced by an approximate subsumption test.

Esposito *et al.* [10] proposed a solution to the problems raised by noise during the classification phase: It was based on a probabilistic interpretation of the matching predicate. The result of the matching process is no longer a true/false answer, but a probability that two well-formed formulae of a variable-valued logic matched because of a change in one of them. Once again, one limitation of the proposed flexible matching process is that it has been defined for rules with nullary predicates in the head.

In this paper the authors extend their previous work along two dimensions: First, the way to handle both numerical and symbolic information in the learning process and second, the definition of a flexible matching process for definite clauses in FOL. As to the inductive inference from numeric and symbolic data, we have defined an information-theoretic specialization operator that can specialize a clause by adding literals of the type

$f(x_1, \ldots, x_n) \in [a..b]$

where $f(x_1, \ldots, x_n)$ is a function taking values in a numeric domain, while $[a..b]$ denotes a closed interval. It is worthwhile to observe that the arity of the function may be greater than one, that is the operator is applicable to numerical relations like *distance(x,y)*. This operator has been embedded into a first-order learning system called INDUBI/CSL [13]. Rules generated by INDUBI/CSL are input to REFLEX (REcognition by FLEXible subsumption), another system that implements the approximate subsumption algorithm. Both systems have been implemented in C language and tested

on the problem of understanding images of single-page documents [11]. Preliminary results have demonstrated the importance of handling both numeric and symbolic descriptions, since the former increase the sensitivity while the latter increase the stability of the internal representation of visual objects [6].

2. Induction of first-order models

Models considered in this paper are defined by means of a first-order logic language, whose basic component is the *literal* or *selector*, which has two distinct forms:

$$f(t_1, \ldots, t_n)=Value \quad (simple\ literal) \qquad \text{and} \qquad f(t_1, \ldots, t_n)\in Range \quad (set\ literal)$$

where f is an n-ary function symbol, called *descriptor*, t_i's can be either variable or constant terms, and *Range* is a set of possible values taken by f. Some examples of literals are the following: $color(x_1)=red$, $height(x_2)\in[1.1 \ldots 2.1]$, and $ontop(x_1, x_2)=true$. Literals can be combined to form *definite clauses*, which can be written as:

$$L_0 \leftarrow L_1, L_2, \ldots, L_m$$

where the simple literal L_0 is called *head* of the clause, while the conjunction of simple or set literals $L_1, L_2, \ldots, L_m$ is named *body*. A clause with literal $f(t_1, \ldots, t_n)=Value$ in the head defines conditions that should be satisfied by the arguments $t_1, \ldots, t_n$ so that the function f can take the value *Value* when applied to $t_1, \ldots, t_n$. For instance, the clause

$$identifier(x)=roof \leftarrow color(x)=red, ontop(x,y)=true, shape(x)=triangle, shape(y)=square$$

defines the conditions that should be satisfied by x so that *identifier* can take the value *roof* when applied to x. A *function definition* is a set of clauses that define f for all possible values in its domain. A *value definition* is a set of clauses that define the conditions that should hold for the arguments of f, so that f can take a *certain* value of its domain. Models considered in this paper are *value definitions*. Henceforth, we will concentrate our attention on value definitions of functions taking values in finite unordered domains.

Some particular definite clauses are obtained by imposing different constraints:

- *linkedness*: conditions in the body should be directly or indirectly related to the arguments of the function f defined by the clause;
- *range-restrictedness*: all arguments of the function f defined by the clause must be constrained by at least a condition in the body of the clause.

The clause reported above is both linked and range-restricted, while the following clause

$$f(x,y)=a \leftarrow g(y)=b, h(z)=c$$

is neither linked (condition $h(z)=c$ is not related to either x or y) nor range-restricted (we have no condition for x).

First-order models generated by INDUBI/CSL are expressed as sets of linked, range-restricted definite clauses. These models are generated from a set of training (positive and negative) examples for a given function value, each of which is described as a single ground, linked and range-restricted definite clause. The generation proceeds according to a *separate-and-conquer* search strategy. The *separate* stage is a loop that checks for the completeness of the current model, that is checks that all examples are explained (or *covered*) by the generated model. If this check fails, another clause to be added to the partial model is searched for. The new clause has to cover some of those examples still unexplained by the partial model. For instance, if we have the following:

positive examples for the value a for the function f and first-order model[2]

e_1: $f(x,y)=a \leftarrow g(x)=b, h(y)=c, r(x,z)=d, g(z)=b$

e_2: $f(u,v)=a \leftarrow g(u)=b, g(v)=b$

H: $f(X,Y)=a \leftarrow g(X)=b, h(Y)=c$

then it is easy to see that H explains e_1, not e_2, thus we have to complete it by adding a further clause that covers at least e_2. In the separate stage the learning system searches in the space of all value definitions for a complete model.

The separate-and-conquer search strategy is adopted in other well-known learning systems, such as FOIL [18]. The main difference with INDUBI/CSL is the use of a *seed* example, whose function is that of guiding the learning process. Since each positive example should be covered by at least one clause of the model returned by the procedure separate-and-conquer, each example becomes a valuable source of information on the structure of the covering clause. Indeed, if e^+ is a positive example to be explained by an induced model H, then H should contain at least one clause C that covers e^+. Therefore, INDUBI/CSL starts with a seed example e^+ and generates a set of at least M distinct range-restricted clauses which are consistent and cover e^+. Then, the best generalization is selected from such a set according to a

2. Henceforth, we will follow the usual Prolog convention of starting variables with a capital letter, all other atoms being constants.

preference criterion. Finally, positive examples covered by the best generalization are removed from the set of positive examples and a new clause is generated, if the set of remaining positive examples is not empty.

In the *conquer* stage, the system performs a beam-search in the space of definite clauses, looking for a linked, range-restricted definite clause that explains some positive examples but no negative example (*consistency* property). The search starts with the most general clause:

$$f(t_1, \ldots, t_n)=Value \leftarrow$$

and proceeds from general-to-specific, or top-down, by adding literals to the body until the obtained clause becomes consistent. In order to specialize a clause G, INDUBI/CSL has to choose some literals to be added. Both numeric and symbolic data are handled in exactly the same way, that is, they have to comply with the property of linkedness and should be sorted according to the very same preference criterion. The only difference is that numeric literals already present in the generalization can be reconsidered, in order to specialize the interval. It is worthwhile to observe that all selected literals are generalizations of literals in the seed example e^+ obtained by applying a simple inverse substitution [20] that replaces all occurrences of a term t_i by the same variable X_i. Thus, the seed example provides the learner with useful information on the structure of the generalizations.

The computation of the interval for numerical set literals is described in Figure 1. A table associated to the function $f(X_1, X_2, \ldots, X_n)$ is built by matching the specialized clause

$$G, f(X_1, X_2, \ldots, X_n) \in [-\infty \ldots +\infty]$$

against positive and negative examples. Then an information-theoretic heuristic is used to select the best interval of values. The table, initially empty, contains pairs $\langle Value, Class \rangle$, where *Class* can be either + or - according to the sign of the example e from which *Value* is taken. The *Value* is determined by considering the literal of the example e that matched against $f(X_1, X_2, \ldots, X_n) \in [-\infty \ldots +\infty]$.

Now the problem is that of finding the interval that best discriminates positive from negative examples. Any threshold value α, lying between two consecutive distinct values, will have the effect of producing two disjoint intervals: the left interval $[l_1, l_2]$ and the right interval $[r_1, r_2]$. The lower bound of the left interval is the smallest value in the table with sign +, while the upper bound in the same interval is the largest value in the table that does not exceed the

```
procedure determine_range([f(X1, X2, ..., Xn)=Seed_value],e+,G,E+,E-)
  initialize table T[f(X1, X2, ..., Xn)]
  foreach example e in E+ do
    foreach substitution θ such that G,f(X1, X2, ..., Xn)∈ [-∞ .. +∞] covers e do
      select the literal [f(X1, X2, ..., Xn)=Value]θ of e
      add the tuple ⟨Value ,+⟩ to the table T[f(X1, X2, ..., Xn)]
    endforeach
  endforeach
  foreach example e in E- do
    foreach substitution θ such that G,f(X1, X2, ..., Xn)∈ [-∞ .. +∞] covers e do
      select the literal [f(X1, X2, ..., Xn)=Value]θ of e
      add the tuple ⟨Value ,-⟩ to the table T[f(X1, X2, ..., Xn)]
    endforeach
  endforeach
  sort table T[f(X1, X2, ..., Xn)] on the Value field
  Cut := determine_all_cut-points(T[f(X1, X2, ..., Xn)])
  MinWE := +∞
  foreach cut-point α in Cut do
    determine the left and right intervals [l1,l2], [r1,r2] with l2<α<r1
    if Seed_value∈ [l1 .. l2] then Admissible_interval := [l1, l2] else Admissible_interval := [r1, r2] endif
    WE := weigthed_entropy(T[f(X1, X2, ..., Xn)], Admissible_interval)
    if WE < MinWE then
      Best_interval:=Admissible_interval
      MinWE := WE
    endif
  endforeach
  if MinWE ≠ +∞ then return [f(X1, X2, ..., Xn)∈ Best_interval] else return nil endif
```

Figure 1. Choice of the best range for numerical descriptors.

threshold α. On the contrary, the lower bound of the right interval is the smallest value in the table that exceeds α, while the upper bound is the largest value with sign +. When one of the two intervals contains no positive value, then it is set to *undefined*. However, at least one of the two intervals must be defined, since the table contains at least one value + corresponding to the *Seed_value*, that is the value taken by $f(X_1, X_2, \ldots, X_n)$ in the seed example.

Not all definite intervals have to be considered, since the specialized clause $G, f(X_1, X_2, \ldots, X_n) \in Range$ for a given *Range* might no longer cover the seed example e^+. Those definite intervals that include the *Seed_value* are said to be *admissible*, because they guarantee that the corresponding specializations still cover the seed example e^+. When the condition of admissibility holds for an interval, the weighted entropy is computed and compared to the minimum weighted entropy found till that moment.

By looking at the table as a source of messages labelled + and -, then the expected information on the class membership conveyed from a randomly selected message is:

$$inf\, o(n^+, n^-) = -\frac{n^+}{n^+ + n^-} log_2 \frac{n^+}{n^+ + n^-} - \frac{n^-}{n^+ + n^-} log_2 \frac{n^-}{n^+ + n^-}$$

where n^+ and n^- are the number of values in the table with positive and negative sign, respectively.

Now consider a similar measurement after $T[f(X_1, X_2, \ldots, X_n)]$ has been partitioned into two subsets, S_1 and S_2, the former containing $n_I^+ + n_I^-$ values falling within an admissible interval, the latter containing the remaining values. The information provided by S_1 will be close to zero when almost all cases have the same sign, + or -. However, we are interested in intervals with a low entropy but a high number of positive examples. The following weighted entropy:

$$E(n_I^+, n_I^-) = \frac{n_I^-}{n_I^+} inf\, o(n_I^+, n_I^-)$$

does actually penalize those admissible intervals with a low percentage of positive examples. A good rule of thumb would be to choose the admissible interval that minimizes $E(n_I^+, n_I^-)$.

As a concrete illustration, let us reconsider the table reported below.

Value	0.5	0.7	0.9	1.0	1.5	1.5	1.5	1.7	2.5	2.5
Sign	+	-	-	-	-	-	+	+	-	+

There are four possible cut points that generate the following intervals:

α	*0.60*	*1.25*	*1.60*	*2.10*
[l1, l2]	[0.50 .. 0.50]	[0.50 .. 1.00]	[0.50 .. 1.50]	[0.50 .. 1.70]
[r1, r2]	[0.70 .. 2.50]	[1.50 .. 2.50]	[1.70 .. 2.50]	[2.50 .. 2.50]

Let us suppose that *Seed_value* equals 1.50. Then only those intervals including 1.50 are admissible since they allow the specialized clause to cover the seed example. The weighted entropy for each of them is

Adm. interval	[0.70 .. 2.50]	[1.50 .. 2.50]	[0.50 .. 1.50]	[0.50 .. 1.70]
E	1.836592	1.000000	2.157801	1.590723

Thus, the best interval is the second one, with a weighted entropy equal to 1.0.

When the table is huge, there could be numerous cut-points. Only some of them will actually be considered, namely those between two consecutive distinct values with different sign (*boundary points*). This explains why the cut points 0.80 and 0.95 have not been considered. On the contrary the cut point 1.25 has been considered because the value 1.0 has a negative sign, while there is a value 1.5 with a positive sign. The reason for this choice is due to the following:

Theorem 1 (Best cut-point) If a cut-point α minimizes the measure $E(n_I^+, n_I^-)$, then α is a boundary point.

The proof can be obtained electronically from http://lacam.uniba.it:8000/pagine/proofs.html.

This result helps to discard several computations of the weighted entropy by considering only boundary points, so improving the efficiency of the procedure *determine_range*.

3. Probabilistic subsumption for first-order models

Models induced by INDUBI/CSL can be used to predict function values for new observations. More precisely, each clause

$$C: \quad f(X_1, \ldots, X_n) = Value \leftarrow L_1, L_2, \ldots, L_m$$

of an induced model can be used forward as an inference rule: If conditions in the body are satisfied by the new observation according to a grounding substitution θ, we can conclude that the function f takes the value *Value* when applied to $X_1\theta$, ..., $X_n\theta$. It is worthwhile to observe that $X_1\theta$, ... ,$X_n\theta$ are ground terms, since the property of range-restrictedness guarantees that when conditions in the body of a clause are satisfied, then the arguments of f are completely determined. For instance, the body of the clause $f(X,Y)=a \leftarrow g(X)=b, h(Y)=c$ is satisfied by the observed facts $g(x)=b, h(y)=c, r(x,z)=d, g(z)=b$, according to the substitution $\theta = \{X \leftarrow x, Y \leftarrow y\}$ that grounds all literals in the clause. Thus, we can conclude that f takes the value a when applied to x and y.

The matching procedure adopted in this deductive step requires that all conditions in the body of a clause are satisfied by the observed facts, or, more technically, that the body of the clause θ-*subsumes*[3] the set of observed facts [17].

Definition 1 (θ-subsumption). Let C and D be two clauses.[4] Then C θ-*subsumes* D, denoted as $C \leq_\theta D$, if and only if there exists a substitution θ such that $C\theta \subseteq D$.

The result of a θ-subsumption test is either *true* or *false*. If $\mathbb{C}$ denotes the space of clauses, then

$$\theta\text{-}subsumption: \mathbb{C} \times \mathbb{C} \rightarrow \{false, true\}$$

However, this requirement might be too strict for real-world problems, because of their inherent vagueness. The presence of noise or the imprecision of the measuring instruments or the variability of the phenomenon described by the induced model often cause the subsumption test to fail. For this reason it becomes necessary to rely on a more flexible definition of subsumption that aims at comparing two descriptions in order to identify their similarities rather than their equality. The result of a flexible subsumption should produce a number in the unit interval [0,1] that indicates a degree of similarity between two clauses:

$$flexible\text{-}subsumption: \mathbb{C} \times \mathbb{C} \rightarrow [0,1]$$

such that, for any two clauses C and D,

i) $flexible\text{-}subsumption(C,D)=1$ if C θ-subsumes D,

ii) $flexible\text{-}subsumption(C,D) \in [0,1)$ otherwise.

The definition of such a similarity measure should be based on a theory which is able to reason about chance and uncertainty, such as the probability theory. In this case we can interpret the result of the flexible-subsumption function as the probability of C θ-subsuming D provided that a change is made in D. More precisely, let D' be a ground clause obtained from D by means of some syntactic changes, such that C θ-subsumes D'. We can associate a probability to D', $P(D \mid D')$, representing the likelihood of observing D given that the original observation was D'. Then, we can set

$$flexible\text{-}subsumption(C,D) \underset{def}{=} \max_{D' \ \theta\text{-}subsumed\ by\ C} P(D \mid D')$$

that is *flexible-subsumption(C,D)* equals the maximum value of the likelihood computed over the space of clauses D' θ-subsumed by C.

Therefore, one of the main problems is that of estimating $P(D \mid D')$ for all clauses D' θ-subsumed by C. For instance, let us consider the following definite clause:

$f(X,Y)=a \leftarrow g(X)=b, h(Y)=c$

and the following set of observed facts:

D: $g(x)=b, h(y)=d.$

Let C denote the body of the definite clause. It can immediately be seen that C does not θ-subsume D. On the contrary,

D': $g(x)=b, h(y)=c$

is θ-subsumed by C, so that we can draw the conclusion $f(x,y)=a$ with probability equal to $P(D \mid D')$. This probability is the likelihood that the original observation was D', but we measured D because of noise. Before explaining how to

---

3. The canonical definition of θ-subsumption given by Plotkin [17] is appropriate for simple literals. However, it can be straightforwardly extended to set literals by requiring that the range of values of literals in C is a subset of the range of values of the corresponding literals in D.

4. A clause $C=L_1 \vee L_2 \vee \ldots \vee L_m$ is also considered as the set of its literals, that is, $C = \{L_1, L_2, \ldots, L_m\}$.

compute $P(D \mid D')$ let us observe that there are many other clauses D' θ-subsumed by C, therefore we have to search in the (possibly infinite) space of clauses θ-subsumed by C for that clause D' that maximizes $P(D \mid D')$. REFLEX, the system that implements a flexible-subsumption test, performs a branch-and-bound search that expands the least-cost partial path. The cost function is given by $1\text{-}P(D \mid D')$.

Let D be the set of literals $\{L_1, L_2, \ldots, L_m\}$. Under the assumption that all facts $L_1, L_2, \ldots, L_m$ are conditionally independent, given D', the probability $P(D \mid D')$ can be defined as follows:

$$P(D/D') = \prod_{i=1}^{m} P(L_i/D')$$

where $P(L_i \mid D')$ denotes the probability of observing the ground fact L_i given D'. Suppose that L_i is $f(t_1, \ldots, t_n)=Value$, then, if D' contains the literal $f(t_1, \ldots, t_n)=Value'$, $P(L_i \mid D')$ is the probability that the real value was $Value'$, but we observed $Value$. We relate this probability to the type of domain of the function f (unordered, partially ordered, or totally ordered) as well as to the probability distribution of values in the function domain. When no information is available on the probability distribution of values, we make an assumption of uniform distribution. Formulae for the computation of $P(L_i \mid D')$ under this assumption are reported in [9, Section V] for several types of domains.

When D' does not contain a literal $f(t_1, \ldots, t_n)=Value'$ we can say that the information on $f(t_1, \ldots, t_n)$ is missing or unknown. In this case $P(L_i \mid D')$ is computed as the expected value of $P(L_i \mid D' \cup \{f(t_1, \ldots, t_n)=Value'\})$, where $Value'$ is the generic domain value, that is

$$P(L_i/D') = \sum_{Value'} P(f(t_1,\ldots,t_n) = Value') \cdot P(L_i/D' \cup \{f(t_1,\ldots,t_n) = Value'\})$$

Once again, formulae for the computation of the expected value, under the assumption of uniform distribution are reported in [9, Section VII].

We conclude this section by observing that under the assumption of uniform distribution, we have $P(D \mid D') > 0$ for any D'. This means that any ground instance of the head of a definite clause can be probabilistically entailed, although some instances are more likely than others. If we are interested into logical entailment with probability at least p, then the definition of *p-subsumption* can be considered.

Definition 2 (p-subsumption). Let C and D be two clauses. Then C *p-subsumes* D, denoted as $C \leq_p D$, if and only if *flexible-subsumption(C,D)*$\geq p$.

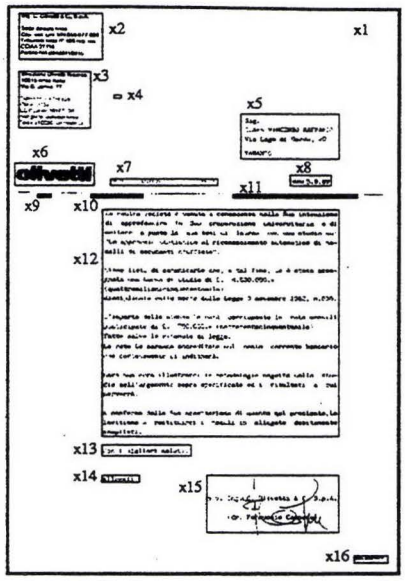
In the application presented in the next chapter, we use a p-subsumption test to identify the logical components of some real business letters.

4. Application to document understanding

In this section the application of INDUBI/CSL and REFLEX to the field of document understanding is presented. The term *document understanding* denotes the process of identification of logical components of a document and the subsequent extraction of relationships between them, such as the reading order [11]. Often a document can be understood by means of its layout structure. This happens when the document has a standard format, as, for instance, business letters sent by a certain company. Indeed, in this case, the date, the logo, the receiver of the letter and other semantically relevant parts can be easily located. Nevertheless, writing down the models useful to understand a particular type of document can be a demanding task, thus we adopted a different approach: We learn the models from a set of training documents. In previous experiments we used only symbolic descriptors by discretizing numeric attributes, such as height, width and position of a block. Preliminary results were encouraging, but not totally satisfactory. One of the main issues seemed to be the prior discretization of numeric attributes. Since the current release of INDUBI/CSL is able to handle numerical descriptors as well, we decided to organize an experiment to test the improvement of the generated rules in terms of accuracy, learning time and simplicity.

We considered a set of 30 single page documents, namely copies of business letters sent by a company. The page layout of each document was described with only symbolic descriptors or mixed numeric/symbolic descriptors (see Figure 2). The logical components we are interested in recognizing are logo, sender, receiver, date, reference number, body and (possibly) signature of the sender. Experimental results for a 10-fold cross-validation are summarized below:

Average Number of Errors		p-value Wilcoxon signed ranks test	Number of Clauses		Average Learning Time	
symbolic	mixed		symbolic	mixed	symbolic	mixed
3.6	2.9	0.3114	28.0	11.5	20:24	19:17



part_of(x1,x2)=true, part_of(x1,x3)=true, ..., part_of(x1,x16)=true,
width(x2)= medium, width(x3)= medium, ..., width(x16)= medium_small,
height(x2)= medium_small, height(x3)= medium_small, ..., height(x16)= smallest,
type(x2)= text, type(x3)= text, ..., type(x16)= text, position(x2)= top_left,
position(x3)= top_left, ..., position(x8)= top_right, ..., position(x16)= bottom_right,
ontop(x2,x3)=true, ontop(x5,x8)=true, ..., ontop(x13,x14)=true,
toright(x6,x7)=true, toright(x3,x4)=true, ..., toright(x9,x10)=true,
aligned(x2,x3)=only_left_col,aligned(x6,x7)=only_lower_row, ..., aligned(x13,x14)=only_left_col

part_of(x1,x2)=true, ..., part_of(x1,x16)=true,
width(x2)= 120.0, ..., width(x8)= 57.0, ..., width(x16)= 44.0,
height(x2)= 63.0, ..., height(x16)= 5.0,
type(x2)= text, ..., type(x16)= text,
x_pos_centre(x2)= 89.0, ..., x_pos_centre(x8)=478.0, ..., x_pos_centre(x16)=568.0,
y_pos_centre(x2)= 40.0, ..., y_pos_centre(x8)= 263.0, ..., y_pos_centre(x16)= 836.0,
ontop(x2,x3)=true, ..., ontop(x13,x14)=true,
toright(x6,x7)=true, ..., toright(x9,x10)=true,
aligned(x2,x3)=only_left_col, ..., aligned(x13,x14)=only_left_col

Figure 2. A business letter and its layout descriptions, symbolic (up) and mixed (down).

The significance test used is a non-parametric test, namely the Wilcoxon signed-ranks test [16], pairing across the folds for the cross-validations. The table shows that the average number of errors decreases, although not significantly, when numerical attributes are discretized on-line. By decomposing the average number of errors into omission and commission errors[5] we can conclude that rules generated from numeric/symbolic descriptions made a significantly lower number of commission errors (0.3 vs. 1.5, p-value=0.0277), and slightly increased the number of omission errors (2.6 vs. 2.1, p-value=0.7353). Since in our application, commission errors are considered more serious than omission errors, we can conclude that the handling of numerical attributes was actually beneficial. As to the other parameters, we observe that the introduction of numerical descriptors simplified the models (see the average number of clauses) and reduced the learning time (expressed in minutes).

The increase in omission errors is due to the presence of literals of the type $f(X_1, \ldots, X_n) \in [a..b]$, that often miss the match against an instance, since the value taken by f is either a little higher than b or a little lower than a. For instance, in one of the ten trials of previous experiment, INDUBI/CSL produced the following definition of *logic_type* for the value *date*:

$logic\_type(X_1)=date \leftarrow width(X_1) \in [42.0 .. 97.0], x\text{-}pos\text{-}centre(X_1) \in [480.0 .. 525.0]$
$logic\_type(X_1)=date \leftarrow y\text{-}pos\text{-}centre(X_1) \in [262.0 .. 279.0], aligned(X_3,X_1)=both\_rows,$
$\quad aligned(X_2,X_3)=both\_rows$
$logic\_type(X_1)=date \leftarrow aligned(X_3,X_1)=only\_lower\_row, aligned(X_2,X_1)=both\_rows$
$logic\_type(X_1)=date \leftarrow x\text{-}pos\text{-}centre(X_1) \in [453.0 .. 525.0], y\text{-}pos\text{-}centre(X_1) \in [266.0 .. 276.0]$

However, the body of none of the four clauses above θ-subsumes the numeric/symbolic description of the document in Figure 2. In fact, the first clause misses the test because the function *x-pos-centre* takes the value 478.0 for the layout component *x8* corresponding to the logical component *date*, while the range of possible values is [480.0 .. 525.0]. On the contrary, the fourth clause misses the test because the function *y-pos-centre* takes the value 263.0 for the argument *x8*, while the range of possible values is [266.0 .. 273.0].

For this reason we decided to match the induced models against test cases using a p-subsumption test instead of the traditional θ-subsumption. The results obtained with a 0.99-subsumption test are the following: 0.4% of commission errors and 2.2% of omission errors. Thus, REFLEX has been able to reduce the rate of omission errors to that obtained with symbolic descriptions, with a small increase of commission errors. Such an increase is mainly due to the proximity of layout components labelled as reference number (see block *x7* in Figure 2) to the other layout components without any logical meaning (see block *x10* in Figure 2). We can conclude that the threshold of a probabilistic subsumption test depends of value of the function to be predicted. In this experiment we have defined a unique threshold for all p-subsumption tests, but for the future we plan to learn automatically the best p from the same training set used to build the models.

5. Omission errors are made when some logical components in the test document are not identified, while commission errors are made when some layout components are given a wrong logical meaning.

Acknowledgments

Thanks to Marzio Cristiano Rotondo for his help in conducting the experiments. Many thanks to Lynn Rudd for her help in rereading the paper.

References

[1] A. Agresti. *Categorical Data Analysis*. New York, NY: Wiley, 1990.

[2] H. Almuallin, Y. Akiba, & S. Kaneda. On handling tree-structured attributes in decision tree learning. *Proceedings of the 12th International Conference on Machine Learning*. San Francisco, CA: Morgan Kaufmann, pp. 12-20, 1995.

[3] F. Bergadano, & R. Bisio. Constructive learning with continuous-valued attributes. In B. Bouchon, L. Saitta, & R.R. Yager (Eds.), *Uncertainty and Intelligent Systems*. Lecture Notes in Computer Science, 313, Berlin: Springer-Verlag, pp. 154-162, 1988.

[4] M. Botta, & A. Giordana. Learning quantitative features in a symbolic environment. In Z.W. Ras, & M. Zemankova (Eds.), *Methodologies for Intelligent Systems*. Lecture Notes in Artificial Intelligence, 542, Berlin: Springer-Verlag, pp. 296-305, 1991.

[5] P. Brito. Order structure of symbolic assertion objects. *IEEE Trans. on Knowledge and Data Engineering*, vol. 6, no. 5, pp. 830-835, 1994.

[6] J.H. Connell, & M. Brady. Generating and generalizing models of visual objects. *Artificial Intelligence*, vol. 31, no. 2, pp. 159-183, 1987.

[7] E. Diday. Knowledge representation and symbolic data analysis. In M. Schader, & W. Gaul (Eds.), *Knowledge, Data and Computer-Assisted Decisions*. Berlin: Springer-Verlag, pp. 17-34, 1990.

[8] S. Dzeroski, L. Todorovski, & T. Urbancic. Handling real numbers in ILP: A step towards better behavioural clones (extended abstract). In N. Lavrac & S. Wrobel (Eds.), *Machine Learning: ECML95*, Lecture Notes in Artificial Intelligence, 912, Berlin: Springer, pp. 283-286, 1995.

[9] F. Esposito, D. Malerba, & G. Semeraro. Classification in noisy environments using a distance measure between structural symbolic descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-14, no. 3, pp. 390-402, 1992.

[10] F. Esposito, D. Malerba, & G. Semeraro. Incorporating statistical techniques into empirical symbolic learning systems. In D.J. Hand (Ed.), *Artificial Intelligence Frontiers in Statistics*, London: Chapman & Hall, pp. 168-181, 1993.

[11] F. Esposito, D. Malerba, & G. Semeraro. Multistrategy learning for document recognition. *Applied Artificial Intelligence*, vol. 8, no. 1, pp. 33-84, 1994.

[12] N. Lavrac, S. Dzeroski, & M. Grobelnik. Nonrecursive definitions of relations with LINUS. In Y. Kodratoff (Ed.), *Machine Learning: EWSL-91*, Lecture Notes in Artificial Intelligence, 482, Berlin: Springer-Verlag, pp. 265-281, 1991.

[13] D. Malerba, G. Semeraro, & F. Esposito. A multistrategy approach to learning multiple dependent concepts. In C. Taylor, R. Nakhaeizadeh (Eds.), *Statistics and Machine Learning: The Interface*, London: Wiley, pp. 87-106, 1996.

[14] T.M. Mitchell. Generalization as search. *Artificial Intelligence*, vol. 18, no. 2, pp. 203-226, 1982.

[15] S. Muggleton (Ed.), *Inductive Logic Programming*. London: Academic Press, 1992.

[16] M. Orkin, R. Drogin. *Vital Statistics*, New York, NY: McGraw Hill, 1990.

[17] G.D. Plotkin. Automatic methods of inductive inference. PhD thesis, Edinburgh University, August 1971.

[18] R. Quinlan. Learning logical definitions from relations. *Machine Learning*, vol. 5, pp. 239-266, 1990.

[19] J.R. Quinlan, & R.M. Cameron-Jones. FOIL: A midterm report. In P.B. Brazdil (Ed.), *Machine Learning: ECML-93*, Lecture Notes in Artificial Intelligence, 667, Berlin: Springer-Verlag, pp. 3-20, 1993.

[20] C. Rouveirol. Flattening and saturation: Two representation changes for generalization. *Machine Learning*, vol. 14, pp. 219-232, 1994.

[21] C.S. Wallace, & D.M. Boulton. An information measure for classification. *Computer Journal*, vol.11, no.2, pp. 185-194, 1968.