

2^{nd} Order Understandability of Disjunctive Version Spaces

Michèle Sebag

LMS, CNRS URA 317, Ecole Polytechnique, 91128 Palaiseau, France
and LRI, CNRS URA 410, Université Paris Sud, 91400 Orsay, France

Tel : 33 1 69 33 33 42 ; Fax : 33 1 69 33 30 26

e-mail : Michele.Sebag@polytechnique.fr

Abstract

Inductive learning aims at extracting knowledge (usually classification rules), from classified examples. Among the criteria of inductive learning, are the understandability of the extracted knowledge and its accuracy in order to classify further examples.

This paper is concerned with what we called *weakly-adequate learning*, i.e. when the learner abilities and the current description of the learning problem are not well-suited. It is suggested that building redundant knowledge possibly allows to resist weak-adequacy, regarding the predictive accuracy concern. This claim is supported by some first results in a particular case of weakly-adequate learning: we present a learner building (logical) Disjunctive Version Spaces and we apply it on a well-studied numerical problem.

However, redundant learners poorly address the understandability concern, due to the size of the produced knowledge. We then propose to define the *second-order understandability* of a knowledge base, as the ability to provide understandable justifications for any statement inferred from this knowledge base. We show that disjunctive Version Spaces enjoy 2^{nd} order understandability.

Introduction

It is commonly acknowledged in the machine learning field, that providing a learner with a smart description of the problem is necessary to actually tackle complex problems. But providing a "perfect" description of the problem nearly requires it to be solved, which is not the general case. Moreover, a smart description may have unwanted side effects: as emphasized by Kelly and Davis [KD91], the more the learner is provided with description bias, the more this may preclude the discovery of unforeseen relations. The issue of internally refining the initial description of the problem is addressed by constructive learning (see [Wne93, CS93]), and is a hot topic of research in machine learning.

So, this paper focuses on what we called *weakly-adequate learning*, i.e. tackling a problem with a learner that is not well suited (because of its search space and/or heuristics) to the available data. Weak-adequacy is studied with respect to the main concerns of inductive learning, namely : providing the expert with an understandable knowledge, and providing the user with an accurate knowledge, i.e. efficient at classifying further examples.

It is suggested that building a redundant knowledge can address the predictive accuracy concern. This claim is supported by some first experimental results: we introduce a learner building disjunctive Version Spaces [Mit82] in a *logical* language; this learner is experimented on a well-studied *numerical* problem [BFOS84].

However, the understandability concern is poorly addressed by redundant learners, because of the size of the produced knowledge. Therefore we define the property of *second-order* understandability, as the ability of a knowledge base to justify any statement inferred from this knowledge base, through understandable arguments. We show that disjunctive Version Spaces enjoy 2^{nd} -order understandability.

This paper is organized as follows. Section 1 discusses weak-adequacy with respect to some main approaches in inductive learning: decision trees [Qui86], star algorithms [Mic83] and version spaces [Mit82]. Section 2 addresses the learning of redundant knowledge, via embedding the version space frame into a star-like frame. A constraint-based approach polynomially achieves the building of such disjunctive version spaces. In section 3, this learner called DIVS (for *Disjunctive Version Space*) is experimented on a well-studied problem [BFOS84]. Results are discussed and compared to that of some other numerical and logical approaches. Section 4 focuses on the understandability concern; it introduces and discusses the notion of 2^{nd} -order understandability.

1 Goal and State of the Art

This section briefly reviews some well-known inductive learning approaches and the heuristics potentially coping with weak-adequacy. We then discuss the problem of weak-adequacy in the particular frame of learning logical formulas from numerical data and we argue the typicality of this frame.

1.1 Decision trees

The cornerstone of decision trees [Qui86, Qui90, BFOS84] is a global evaluation function (e.g. based on the quantity of information or the Gibbs criterion), that measures the discriminant power of a descriptor with respect to a data set. This criterion governs the choice of the nodes of the tree. However, its basic definition only regards informations defined on finite domains. Its application to continuous informations involves a discretization step, i.e. a change of representation of the initial problem. This change has a crucial impact on the success of learning and many strategies are concerned with judicious discretization (see for instance [FI93,

dM93, BG93]).

Decision trees are reputed for being both understandable and efficient. As a matter of fact, they produce concise knowledge bases and this knowledge is expressed within the language of the examples ; moreover, the optimality criterion allows for noise filtering.

The only heuristics somehow allowing decision trees to overcome weak-adequacy, involves the building of *oblique decision trees* [HKS93, BFOS84]: a predefined subset of functions of the initial attributes (e.g. linear combinations) is systematically explored during the building of the tree.

1.2 Star algorithms

The star algorithm repeatedly generalizes some selected examples termed *seeds* [Mic83, MMHL86]. The set of generalizations of a seed, termed *star*, consists of the best M rules that cover the seed and are optimal in the sense of some quality function ; both the number M of rules to retain and the optimality function, are supplied by the user.

The user thereby takes in charge the control of the produced knowledge: the desired qualities in terms of generality, of redundancy, of discriminant accuracy, are set via the optimality function and parameter M . From this point of view, star algorithms appear much more demanding than decision trees.

In counterpart, star algorithms provide some facilities for constructive induction. For instance, it allows to retain redundant rules ($M > 1$), i.e. rules that generalize same example(s). The comparison of the premises of redundant rules gives hints about defining new descriptors (e.g. XOR or *M-of-N* expressions involving the initial attributes) [WM94]. This ability allows to successfully handle some cases of weak adequacy, such as those arising from learning symmetry- or parity-based concepts with, so to say, bare hands.

1.3 The Version Space

The version space frame [Mit82] determines the upper and lower bounds of the learning search: it builds the set noted S of most specific formulas that cover all positive examples, and the set noted G of most general formulas that do not cover any negative example. In contrast with decision trees and star algorithms, version spaces do not involve any order relationship specific to learning (such as the discriminant power for decision trees and the optimality criterion for star algorithms) but only relies on the generality order.

The price to pay is that this frame fails to handle noisy data and disjunctive concepts. G and S evolve in a monotonous way (the level of generality of S can only increase, while the level of generality of G decreases) ; so there is no way to recover from erroneous choices.

Furthermore, the size of G may be exponential with respect to the number of descriptors [Hau88]. This drawback led Hirsh to propose a computable characterization of

G with a polynomial complexity [Hir92]. This line is continued by another characterization of G based on a constraint-like formalism [Seb94b]. This formalism permits a polynomial characterization of G in propositional logic, and can be extended to first order logic [Seb94a, SR94].

1.4 Logical learners and numerical data

The most common case of what we called *weakly-adequate learning*, consists of using logical learners to process numerical data. Machine discovery emphasizes that sophisticated heuristics are needed in order to inductively explore the search space of numerical functions (see the pioneering paper of Langley et al. [LSB83]). Another trend in induction consists in devolving the part of the search, concerned with finding numerical thresholds [BG90] or functions [DTU95], to an external optimization module (e.g. based on either genetic algorithms [Gol89] or genetic programming [Koz94]).

This particular case of weak-adequacy deserves attention for practical and theoretical reasons:

- It occurs in most real-world problems, whose description involves both numerical and logic informations. This mixed description can be handled by separately considering the numerical and logic parts and then merging the partial results so obtained. But processing the whole information at once would of course, if by any means tractable, be preferred.
- It is a real case of weak-adequacy, to be contrasted with what would be *ill-adequacy*. As a matter of fact, computer science witnesses that any numeric expression can be computed (approximated) by means of logical expressions [WM94]. So, it is not hopeless that logical learners could reach some competence regarding numerical problems (without any pretension to winning over purposely devised algorithms).

All of the reviewed learning approaches basically handle numerical data via selectors [Mic83], i.e. logical functions comparing an initial information (or some predefined function of the initial informations) to a threshold. But the number of selectors in the output knowledge base strongly depends on the chosen approach: it

- tends to be minimal in decision trees, via an embedded optimality criterion,
- is globally controlled via some user-supplied optimality criterion in star algorithms,
- only depends on the data and tends to be huge, in version spaces.

However, in the context of weakly-adequate learning, one may fear the selection of the candidate formulas accordingly to the available heuristics (user-supplied or embedded into the learner), to be potentially harmful.

Assume *a contrario* such selection to be relevant. Then, either the learning is successful and weak-adequacy would not be detected. Or it fails because it explores

an inadequate search space, and this would rather be *ill-adequacy*, than properly speaking weak-adequacy.

According to this remark, some attention should be paid to redundant learning, i.e. learning with as few selection of the candidate formulas, as possible. We describe in section 2 a redundant learner that builds the exhaustive characterization of the candidate solutions in the line of the version space. An experimental study (section 3) considers the artificial waveform problem [BFOS84]. According to the experiments, this problem can be taken as weakly-adequate to many logical and numerical learners, with respect to the accuracy of the produced knowledge. The predictive performances are compared ; this unique problem can only state that, at least once, redundancy turned out to be a good way to reach complexity. Last, section 4 discusses the price to pay for reaching a (possibly) better accuracy by means of redundant learning, in terms of understandability.

2 Disjunctive Version Spaces

This section addresses redundant learning. We first propose an embedding of version spaces into a star-like approach in order to overcome most limitations pertaining to version spaces. Then, in order for this paper to be self contained, we briefly recall the constraint-like formalism devised for a polynomial learning of version spaces. The general scheme of the proposed algorithm, called *DIVS* (for *Disjunctive Version Space*, is finally discussed.

2.1 Coupling version space and star algorithm

Our approach proceeds from the following remark: version spaces do never fail when a single positive example Ex is generalized against several negative examples. In this case, S reduces to Ex and is never generalized; hence it never comes to be inconsistent with G . The erroneous negative examples only result in over-specializing G .

So, it comes to generalize the examples one at a time; the generalization of the current example Ex , denoted $G(Ex)$, is the G set of the most general expressions covering Ex (taken as unique positive example) and discriminating all examples belonging to other classes (taken as negative examples). The version space frame is thus embedded into a star-like approach, the star of seed Ex being set to $G(Ex)$.

The output knowledge noted \mathcal{G} then consists of the disjunction of all G stars built from the various seeds. Note this approach completely avoids any merge between different G sets, in order to preserve a polynomial complexity [Hir92].

2.2 Building a G star

Consider a unique positive example Ex , to generalize against the negative examples $Ce_1, ..Ce_N$. $G(Ex, Ce_1, ..Ce_N)$ denotes the set of the maximally general formulas covering Ex and rejecting any Ce_i . Given the exponential complexity of characterizing G as a disjunction of conjunctions, the constraint formalism rather characterizes

G as a *conjunction of disjunctions*. $G(Ex, Ce_1, ..Ce_N)$ is thus defined as the conjunction of all $G(Ex, Ce_i)$.

$$G(Ex, Ce_1, ..Ce_n) = G(Ex, Ce_1) \wedge .. \wedge G(Ex, Ce_n)$$

Due to space limitations, we restrict ourselves to an attribute-value formalism (the reader interested in an extension of this approach to a subset of first order predicate logic is referred to [Seb94a, SR94]). In this formalism, a maximally general formula is a selector [Mic83]. Let $x_1, .., x_P$ denote the attributes ; a selector is a boolean function denoted $[x_i = V_j]$, that takes value *true* for example E iff value $x_i(E)$ belongs to subset V_j .

Consider the maximally general selector covering Ex and rejecting Ce_i , based on attribute x_j . This selector is defined iff values $x_j(Ex)$ and $x_j(Ce_i)$ are both informed and distinct. When it is defined, this selector is set to $[x_i = V_{i,j}]$ where

- $V_{i,j}$ is the largest interval including $x_j(Ex)$ and excluding $x_j(Ce_i)$ if x_i is an ordered attribute (e.g. taking continuous or integer values).
- $V_{i,j}$ denotes the most general value covering $x_j(Ex)$ and not covering $x_j(Ce_i)$ if x_i is a tree-structured attribute.

Let us take an example and build the maximally discriminant selectors discriminating the Unidentified Flying Object Ex from the well-identified Ce , described in the table below :

	Shape	Height	Weight	UFO
Ex	circular	16	?	yes
Ce	2-wings	160	7	no

Granted that the *Shape* domain is tree-structured, and that *oval* is the most general value covering *circular* and not covering *2-wings*, the maximally discriminant selector based on attribute *Shape* will be¹ $[Shape = oval]$. The selector based on attribute *Height* obviously is $[Height = [0, 160]]$ (or equivalently $[Height < 160]$). No selector based on attribute *Weight* can be discriminant since this attribute is uninformed for Ex . Finally, we have

$$G(Ex, Ce) = \{[Shape = oval], [Height < 160]\}$$

Any formula covering Ex and discriminating Ce must be more specific than at least one above selector ; the disjunction of these selectors is therefore called *constraint* derived from Ce upon the generalization of Ex . Details about the pruning of constraints and attributes will be found in [Seb94b]. The star of seed Ex then consists of the conjunction of all constraints $G(Ex, Ce_i)$, $i = 1..n$.

¹With restriction to operator '='.

Complexity

When no pruning of the constraints is attempted, the (time and space) complexity of building a G -star is $\mathcal{O}(N \times P)$, where N denotes the number of examples and P is the number of attributes.

When no selection of the seeds is attempted, the (time and space) complexity of building \mathcal{G} is in $\mathcal{O}(N^2 \times P)$. When examples belonging to the current star are removed from the data set, as in classical star algorithms [Mic83], the time complexity increases up² to $\mathcal{O}(N^3 \times P)$.

2.3 Classifying from G -Stars

In the presented approach of disjunctive version space, S sets do not play any role: given a unique positive example, S is never generalized. The usual classification function associated to a version space (S, G) must therefore be modified and also extended in order to discriminate among several classes.

The class of a star $G(Ex)$ is naturally defined as the class of its seed Ex . The difficulty arises from the fact that stars belonging to incompatible classes may overlap. Inconsistencies are usually solved through a majority vote process (see [Gam89, Ven93] among others). Each hypothesis (here, G -star) triggered by the current case votes for its class, and the current case is classified in the majority class. In *DIVS*, this majority vote can be thought of as a nearest-neighbor-like process. Let a boolean similarity function be defined as : E is neighbor on Ex iff E belongs to the G -star derived from Ex . Then, *DIVS* classifies E according to the classes of its neighbors³.

Complexity

The classification complexity thus linearly depends on the size of \mathcal{G} , which is in $\mathcal{O}(N^2 \times P)$.

2.4 Heuristics

Two heuristics parameterized by the expert tune the use of the G -stars during the classification phase.

The first heuristic intends to fight against noise in the dataset, in a very classical way [Mic83, Gam89] : example E belongs to star $G(Ex, Ce_1, ..Ce_N)$ iff it satisfies at least a given percentage $(100 - \epsilon)$ of the star constraints $G(Ex, Ce_i)$. This heuristic allows to stand erroneous examples among the $Ce_1, ..Ce_N$; parameter ϵ is supposed to approximate the initial error rate.

²We used another selection of seeds, with same additional complexity: the candidate seed is retained iff at the moment it is considered, it is not yet classified in the right class given the previously built stars.

³The main difference with a traditional k-NN process is that our approach relies on a similarity function which is induced from examples, instead of being given (the usual case) or deduced (as in [Bis92]).

The second heuristic is original, as far as we know, since it is tightly connected to the constraint formalism. Constraint $G(Ex, Ce)$ is a disjunction of maximally discriminant selectors :

$$G(Ex, Ce) = \{Selector_1, Selector_2, ..Selector_K\}$$

But most examples happen to satisfy one selector at least in most constraints (this is particularly true when the problem domain involves many continuous attributes). This entails that most examples belong to most stars, hence are classified in the most frequent class !... To prevent this, an example is now required to satisfy at least M selectors in a constraint, to be reputed to satisfy this constraint. Parameter M is supplied by the expert : it controls the degree of specialization of a constraint.

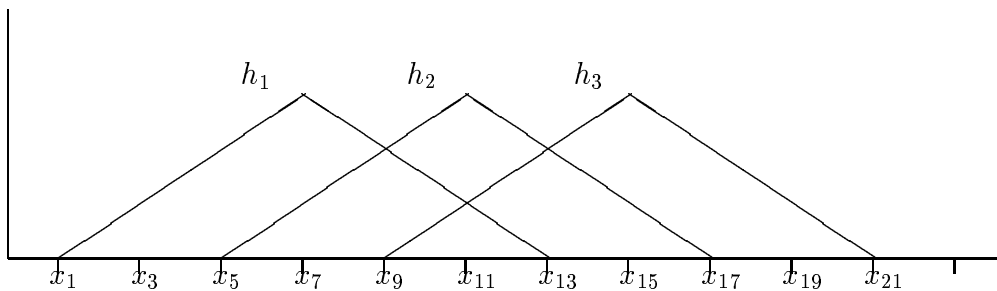
Note this heuristic significantly modifies the semantics of constraints : $G(Ex, Ce)$ now stands for an "at least M -of- K " concept ; the constraint formalism thus permits to explore a much wider search space at no additional computation cost.

3 Experimentation

As argued in 1.4, we focus on the particular case of weak-adequacy that consists of applying a logical learner on a numerical problem; experiments are done on the well-studied artificial waveform problem [BFOS84]. Besides *DIVS*, several numerical and logical learners are experimented on this problem; the diversity of the results gives some hints as to overcoming weak-adequacy.

3.1 Problem and Reference Results

The waveform problem is a numerical artificial problem designed by Breiman et al. [BFOS84] : three classes are built from the waves $h_1(t)$, $h_2(t)$ and $h_3(t)$:



Each class is obtained by randomly combining two waves among the above three waves. Each example is a vector of 21 components, with :

$$Ex \in C_1 : x_j(Ex) = uh_2(j) + (1 - u)h_3(j) + \epsilon_j, \quad j = 1, \dots, 21, \quad u \in [0, 1]$$

$$Ex \in C_2 : x_j(Ex) = uh_1(j) + (1 - u)h_3(j) + \epsilon_j, \quad j = 1, \dots, 21, \quad u \in [0, 1]$$

$$Ex \in C_3 : x_j(Ex) = uh_1(j) + (1 - u)h_2(j) + \epsilon_j, \quad j = 1, \dots, 21, \quad u \in [0, 1]$$

where u is uniformly selected in $[0,1]$ and ϵ_j denote independent random variables, following a Gaussian distribution with mean 0 and variance 1. Note classes C_1, C_2 and C_3 overlap ; when they are equi-represented, the minimal misclassification rate is 14 % [BFOS84].

The training set includes 300 examples randomly generated according the above model. Results are averaged on 10 independent training sets. Validation considers a single test set including 5 000 independently generated examples.

The methods experimented on these continuous data include feed-forward neural nets (NN) [RM86], linear discriminants (LD) and k -nearest-neighbors⁴ [DH73], fuzzy decision trees (DT) [Ram94], and a star algorithm optimized by genetic algorithm (SIA) [Ven93]. These reference results are detailed in [SYM95]. The best results obtained are summarized in Table 1 : each row gives the averaged error rate observed on the training set and on the test set, together with its variance.

The last column gives the dimension of the learning output. It stands respectively for the number of weights, in the case of neural nets ; for the number of parameters characterizing the discriminant hyper-planes, in linear discriminants ; for the number of nodes in decision trees ; for the size of the training set, in k -NN ; and for the number of selectors involved in the stars, in SIA.

<i>Method</i>	<i>Training error</i>		<i>Test error</i>		<i>Result Dimension</i>
NN	7.8	(± 1.3)	17.1	(± 1)	128
LD	12.8	(± 1.3)	20.4	(± 1)	66
DT	5.8	(± 4.4)	28	(± 1.8)	24.3
k-NN	0		27	(± 1.7)	6 300
SIA	4	(± 0.7)	24.3	(± 0.7)	320

Table 1 : Reference results on the waveform problem

As could be expected, logical learners are not good as this problem; furthermore, the best performance in terms of predictive accuracy on the test set (reached by a neural net) is still far from the theoretical optimum (17% against 14%).

3.2 Experimenting redundancy

DIVS is controlled through two parameters : ϵ allows to tune the degree of accuracy prescribed, thus coping with noise ; M rules out the specialization of the constraints, and the complexity of the underlying formalism (cf sec. 2.4). In the experiments presented here, ϵ varies from 0 to 20% and M from 8 to 10 (Table 2).

Parameter ϵ influences the results in a classical way : increasing ϵ decreases the performances on the training set and improves the accuracy on the test set. In fact, the only crisp difference occurs between $\epsilon = 0\%$ and $\epsilon = 5\%$; quite similar performances are obtained for ϵ ranking from 5% to 25%. For a given ϵ , there exists an optimal M (i.e. leading to a minimal error rate on the test set). However, only slight variations are observed near to this optimal value.

⁴Several values of k have been tried. The best one, with respect to the predictive accuracy on the test set, turned out to be 1.

ϵ	M	<i>Training error</i>		<i>Test error</i>		<i>Nb Stars</i>	<i>Nb Sel.</i>
0	8	0	(± 0)	22.8	(± 0.4)	35	20 903
0	9	0	(± 0)	22.3	(± 0.2)	51	18 652
0	10	0	(± 0)	21.6	(± 0.2)	74	17 602
10	8	18.4	(± 0.6)	17.9	(± 0.3)	76	102 938
10	9	17.7	(± 0.7)	18.7	(± 0.3)	72	60 264
10	10	16.7	(± 0.6)	19.3	(± 0.1)	78	37 701
20	8	19.8	(± 0.8)	19.1	(± 0.2)	82	156 007
20	9	19.3	(± 0.6)	17.8	(± 0.3)	78	97 753
20	10	19	(± 0.7)	18	(± 0.2)	77	57 068

Table 2 : DIVS results on the waveform problem⁵

A learning run takes between 1 and 3 minutes on a HP-700 workstation. *DIVS* is written in C++. The dimensions of the learning result are the number of stars, and the total number of selectors involved in the stars. The error rate stands for both misclassified and unclassified examples.

3.3 Discussion

Of course, definite conclusions cannot be drawn from experiments on a single problem. However, this problem is typical of weak-adequacy in the following sense: according to experiments, its solution cannot be easily approximated by simple functions, be they logical (as rules in decision trees or *SIA*), or based on simple numerical functions (such as mean square distance in *k*-NN, or linear combinations in linear discriminants).

Furthermore, the comparison between the different experiments gives some hints about the difficulty of this problem. Let us first compare *SIA* and *DIVS*, for these learners are fairly similar and yet obtain significantly different results. Like *DIVS*, *SIA* follows a star-like approach; the star associated to a seed is optimized by means of a genetic algorithm [Ven93] ; optimization regards both the attributes and the numerical thresholds involved in the star. Several optimization criteria (e.g. significance, predictive accuracy, generality,..) were tried. So the only fixed differences between *SIA* and *DIVS* are

- The way seeds are selected (which is the usual star selection in [Ven93] and described in 2.2 for *DIVS*). However, the final number of seeds is about the same (80 stars in *SIA* against 76-78 in *DIVS*).

⁵The number of selectors is not correlated to the star number, for the following reason. A counter-example gives rise to a constraint iff, at the time it is considered, it belongs to the current star (i.e. it satisfies the constraints previously built). This heuristic is not sound in general (a particular case being when *M* equals the number of selectors in a constraint ; its soundness then derives from [Seb94b]). However, it experimentally speeds up the learning without decreasing the performances.

- The redundancy of the star: a star contains a conjunction of selectors in *SIA* optimal in the sense of whatever criterion, while it contains all conjunctions of selectors that are maximally discriminant in *DIVS*.

This comparison gives evidence that the better performance of *DIVS* can only be blamed on the redundancy of the produced knowledge. The advantage of redundancy can be argued as follows. Assume the classification depends on a numerical function of the attributes ; then, a huge number of thresholds is required to mimic numerical skills : think of an oblique line approximated by a stepwise function. The bad performances of decision trees could also be blamed on the insufficient number of thresholds retained.

Surprisingly, its huge number of selectors even allows *DIVS* to outperform the numerical skills of linear discriminants on this problem [DH73]. Quadratic discriminants turned out to be less good than linear discriminants at this problem, in spite of their higher complexity, or rather, because this higher complexity results in overfitting the data.

The classifiers built by *DIVS* and linear discriminants mainly disagree in the regions where classes (and hyper-planes) overlap. The better performance of *DIVS* in these regions can be attributed to its nearest-neighbor-like approach, because it captures something of the local density of a class. But the notion of neighborhood in *DIVS* is very rough (boolean); so, *DIVS* is less sensitive to noise than a classical *k*-NN (note the variance of the error rate is very low).

More experiments are ongoing; at present, we can only state that redundancy has been found at least once a good strategy to resist weak-adequacy.

4 Understandability

When simple functions fail to provide good approximations of the solution, more complex functions must be considered. This section examines to what extent, the increase of complexity needed to resist weak-adequacy, implies a less understandability of the learning output.

4.1 First-order understandability

In the experiments reported in previous section, the best solutions so far were encoded respectively via a neural net and a disjunctive version space. These approaches share at least one characteristic: a poor understandability of the output knowledge.

The understandability of a knowledge base is of course difficult to define for it much refers to the expectations of the user. So, some intuitive requirements for understandability are : the knowledge must both use intelligible "words" of the domain, and be of readable size, according to the want of the user. This acceptation of understandability is hereafter referred to as *first order understandability*.

In this acceptation, decision trees are understandable (but oblique decision trees are not necessarily so) ; stars may be understandable (depending on the bounds of a

star) ; neural nets are primarily not understandable. Neither are disjunctive version spaces, because of their size.

The poor understandability of neural nets has been long considered a fact. But recently, much attention has been paid to the extraction of meaningful (understandable) knowledge from trained neural nets: Towell's and Shavlik's pioneering work describes how to map a knowledge base into a neural net, and how to use the numerical optimization of this neural net, to improve the original knowledge base [Tow92]. This approach could be viewed in the scheme of [dMD94]: understandability and predictive accuracy are reached by different means, and more precisely, understandability is met by extracting some approximation of the prediction function, that fits the understandability requirements set by the user (here, the use of logical instead of numerical functions).

We similarly plan to extract an approximate and readable knowledge from disjunctive version spaces.

4.2 Second-order understandability

But besides the understandability of knowledge *per se*, we are also interested in some 'operational understandability' of knowledge, i.e. the ability to provide justifications for the results it produces. This acceptance of understandability is referred to as *second-order understandability*.

In the framework of induction, 2^{nd} order understandability is concerned with providing understandable justifications for the classification of an example. Understandable justification remains to be defined. The definition we adopt here is: an understandable justification for a classification is a subpart of a knowledge base, which is both understandable (in the sense of 1^{st} -order understandability) and suffices to classify the current example.

Note that 2^{nd} order understandability is implied by 1^{st} order understandability, but that the converse is not true. An example (thus a constructive proof) is given by disjunctive version space. The huge size of the produced knowledge (100 000 selectors in the experiments reported in section 3) forbids its readability, and therefore its 1^{st} understandability.

However the underlying structure of the version space gives means to justify the classifications produced by *DIVS*, as follows. Let E be the example to classify; E belongs to several stars $G(Ex_1), G(Ex_2), \dots$. Now consider the generalization of E with any such Ex_i , noted S_i ; S_i is covered by $G(Ex_i)$ by construction, and can thus be viewed as a subpart of the theory hidden in the disjunctive version space. On the other hand, since all examples covered by $G(Ex_i)$ (and a fortiori by S_i) must be classified in the class of Ex_i , S_i implies that E belongs to the class of Ex_i . The eventual classification of E can then be argued from S_1, S_2, \dots ; this process may even give means for debate : *pro* and *con* justifications of the current classification are built from the stars E belongs to, when E belongs to incompatible stars.

So, the example to classify sets some window on the whole knowledge base. This knowledge is 2^{nd} order understandable if some pieces of knowledge can be extracted

through such windowing, that are understandable, relevant and sufficient to classify the example at hand.

This approach can also be viewed as an understandability post-processing in the line of [dMD94], that extracts from the predictive function, some approximate function that meets the intelligibility requirements of the user.

In the case of neural nets, this understandability post-processing has to do with turning real-valued coefficients into boolean ones (so that numerical functions can be translated into logical ones). In the case of disjunctive version spaces, the post-processing is guided by the example to classify and only proceeds by logical specialization.

5 Conclusion

The main tasks of a learner are the generation of candidate solutions, and the selection of relevant solutions. In the case of weakly-adequate learning, the trouble comes from the selection step. For this reason, this paper is interested in redundant learning, with as few selection of the candidate solutions as possible.

A learner that builds the disjunctive version space including all candidate solutions with polynomial complexity is presented. Experimentations on the waveform problem show that (at least once...) this redundant learner can resist the problem of weak-adequacy, in the sense that it produces redundant classifiers with a good predictive accuracy.

Redundant classifiers are then considered regarding their understandability. We propose to distinguish among the 1st order understandability of the knowledge base, meaning its readability by the expert, and the 2nd order understandability concerned with justifying the classifications inferred from this knowledge base.

It is emphasized that 2nd order understandability is easier to reach than 1st-order one (true for human experts too). The current example to classify (or request to answer) can be used to drive the transformation of the accurate knowledge into understandable knowledge, in the line of [dMD94]. Further research is concerned with characterizing these transformations, as to order the so-obtained justifications from mere sophisms to syllogisms (deductions).

More generally, this work suggests that it is easier to transform an accurate knowledge base into an understandable one, than to go the other way. The history of sciences also gives evidence that the the number of understandable theories is bigger than the number of effective decision-making tools...

Acknowledgments

Thanks to O. Gascuel and A. Guénoche, for their careful proofreading of an earlier draft of this paper.

References

- [BFOS84] L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression by tree*. Wadsworth, Belmont California, 1984.
- [BG90] F. Bergadano and A. Giordana. Guiding induction with domain theories. In Y. Kodratoff and R.S. Michalski, editors, *Machine Learning : an artificial intelligence approach*, volume 3, pages 474–492. Morgan Kaufmann, 1990.
- [BG93] M. Botta and A. Giordana. Smart+ : A multi-strategy learning tool. In *Proceedings of IJCAI-93*, pages 937–943. Morgan Kaufmann, 1993.
- [Bis92] G. Bisson. Learning in fol with a similarity measure. In *Proceedings of 10th AAAI*, 1992.
- [CS93] M.W. Craven and J.W. Shavlik. Learning to represent codons: A challenge problem for constructive induction. In *Proceedings of IJCAI-93*, pages 1319–1324. Morgan Kaufmann, 1993.
- [DH73] R.O. Duda and P.E. Hart. *Pattern Classification and scene analysis*. John Wiley and sons, Menlo Park, CA, 1973.
- [dM93] T. Van de Merckt. Decision trees in numerical attribute spaces. In *Proceedings of IJCAI-93*, pages 1016–1021. Morgan Kaufmann, 1993.
- [dMD94] T. Van de Merckt and C. DeCaestecker. A general framework for concept learning using hybrid systems: The two-functional model. In *TR 93-19, IRIDIA, Université Libre de Bruxelles*, 1994.
- [DTU95] S. Dzeroski, L. Todorovski, and T. Urbancic. Handling real numbers in ILP: a step towards better behavioral clones. In N. Lavrac and S. Wrobel, editors, *Proceedings of ECML-95, European Conference on Machine Learning*. Springer Verlag, 1995.
- [FI93] U.M. Fayyad and B.K. Irani. Multi-interval discretization of continuous valued attributes for classification learning. In *Proceedings of IJCAI-93*, pages 1022–1027. Morgan Kaufmann, 1993.
- [Gam89] M. Gams. New measurements highlight the importance of redundant knowledge. In K. Morik, editor, *Proceedings of EWSL-89*, pages 71–80. Pitman, London, 1989.
- [Gol89] D. E. Goldberg. *Genetic algorithms in search, optimization and machine learning*. Addison Wesley, 1989.
- [Hau88] D. Haussler. Quantifying inductive bias : AI learning algorithms and Valiant’s learning framework. *Artificial Intelligence*, 36:177–221, 1988.
- [Hir92] H. Hirsh. Polynomial-time learning with version spaces. In *Proceedings of National Conference on Artificial Intelligence*, 1992.
- [HKS93] D. Heath, S. Kasif, and S. Salzberg. Induction of oblique decision trees. In *Proceedings of IJCAI-93*, pages 1002–1007. Morgan Kaufmann, 1993.
- [KD91] J. D. Kelly and L. Davis. A hybrid genetic algorithm for classification. In *Proceedings of IJCAI-91*, pages 645–650. Morgan Kaufmann, 1991.
- [Koz94] J. R. Koza. *Genetic Programming II*. MIT Press, Massachusetts, 1994.

- [LSB83] P. Langley, H.A. Simon, and G.L. Bradshaw. Rediscovering chemistry with the BACON system. In R.S Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning : an artificial intelligence approach*, volume 1. Morgan Kaufmann, 1983.
- [Mic83] R.S. Michalski. A theory and methodology of inductive learning. In R.S Michalski, J.G. Carbonell, and T.M. Mitchell, editors, *Machine Learning : an artificial intelligence approach*, volume 1. Morgan Kaufmann, 1983.
- [Mit82] T.M. Mitchell. Generalization as search. *Artificial Intelligence*, 18:203–226, 1982.
- [MMHL86] R.S. Michalski, I. Mozetic, J. Hong, and N. Lavrac. The AQ15 inductive learning system: an overview and experiment. In *Proceedings of IMAL*, 1986.
- [Qui86] J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1:81–106, 1986.
- [Qui90] J.R. Quinlan. Learning logical definition from relations. *Machine Learning*, 5:239–266, 1990.
- [Ram94] M. Ramdani. *Système d'Induction Formelle à base de connaissances imprécises*. PhD thesis, Université Pierre et Marie Curie, Paris VI, 1994.
- [RM86] D.E. Rumelhart and J.L. McClelland. *Parallel Distributed Processing*. MIT Press, 1986.
- [Seb94a] M. Sebag. A constraint-based induction algorithm in FOL. In W. Cohen and H. Hirsh, editors, *Proceedings of ICML-94, International Conference on Machine Learning*. M. Morgan Kaufmann, July 1994.
- [Seb94b] M. Sebag. Using constraints to building version spaces. In L. De Raedt and F. Bergadano, editors, *Proceedings of ECML-94, European Conference on Machine Learning*. Springer Verlag, April 1994.
- [SR94] M. Sebag and C. Rouveirol. Induction of maximally general clauses compatible with integrity constraints. In S. Wrobel, editor, *Proceedings of ILP-94, International Workshop on Inductive Logic Programming*. Springer Verlag, 1994. forthcoming.
- [SYM95] O. Gascuel et al., Symbolic-numerical methods for discrimination, submitted.
- [Tow92] G. Towell. *Symbolic Knowledge and Neural Networks : Insertion, Refinement and Extraction*. PhD thesis, University of Wisconsin, 1992.
- [Ven93] G. Venturini. Sia : A supervised inductive algorithm with genetic search for learning attribute-based concepts. In Bradzil P., editor, *Proceedings of ECML-93, European Conference on Machine Learning*, pages 280–296. Springer Verlag, 1993.
- [WM94] J. Wnek and R.S. Michalski. Discovering representation space transformations for learning concept descriptions combining dnf and m-of-n rules. In T. Fawcett, editor, *Workshop on Constructive Induction, ICML-94*, 1994.
- [Wne93] J. Wnek. *Hypothesis Driven Constructive Induction*. PhD thesis, George Mason University, Fairfax, VA, 1993.