Concept Learning with Approximation: Rough Version Spaces

Vincent Dubois and Mohamed Quafafou

IRIN, Université de Nantes, France

Abstract. The concept learning problem is a general framework for learning concept consistent with available data. Version Spaces theory and methods are build in this framework. However, it is not designated to handle noisy (possibly inconsistent) data. In this paper, we use rough set theory to improve this framework. Firstly, we introduce a rough consistency. Secondly, we define an approximative concept learning problem. Thirdly, we present a Rough Version Space theory and related methods to address the approximative concept learning problem. Using a didactic example, we put these methods into use. An overview of possible extension of this work concludes this article.

Keywords. Approximation, Concept Learning, Rough Sets, Version spaces

1 Introduction

The concept learning problem is a well-known and fruitful framework, as numerous developments in the Version Space field attests, but is ill-suited to real-word, inconsistent data. This come from the harsh consistency property it is build upon: either a concept is consistent with the data, either it is not. We alleviate this problem by using rough set theory to soften the consistency requirement. As a consequence, newly defined approximative concept learning framework and in particular Rough Version Space can handle inconsistent data, as shown in our example.

This paper is organized as follow: section 2 introduces the Concept Learning Task and version spaces and their properties. Section 3 presents an approximative version of concepts developed in section 2. Section 4 proposes algorithms and methods to handle the approximative concept learning problem. Section 5 puts them in action on a didactic example. Section 6 concludes this paper.

2 Concept Learning

2.1 Notations

We note $^{\complement}A$ the complementary of A in the space it is defined. We note segment $[A, B]$ the set of all element x such that $A \subset x \subset B$. If $f : A \mapsto B$, and $C \subset A$, we note $f(C) = \{c \in B | \exists a \in C, f(a) = c\}$.

J.J. Alpigini et al. (Eds.): RSCTC 2002, LNAI 2475, pp. 239–246, 2002.
© Springer-Verlag Berlin Heidelberg 2002

We note I the set of all instances. The set 2^I of all concepts is denoted C. Let L_I and L_C denotes language on I and C. The function $R_I : L_I \mapsto I$ (resp. $R_C : L_C \mapsto C$) maps a instance (resp. a concept) representation to the instance (resp. a concept) it represents. R_C and R_I are not necessarily injective nor surjective, i.e. some concept or instances may have many or no representation in the chosen language.

Definition 1. *A concept $c \in C$ is said to* cover *an instance $i \in I$ iff $i \in c$. This definition extends to the language by the following way:* $\forall i \in L_I, \forall c \in L_C, c$ cover i $\Leftrightarrow \mathrm{R_I(i)} \in \mathrm{R_C(c)}$

Definition 2. *A concept c is said to be* consistent *with a set of positive instances P and a set of negative instances iff $P \subset c$ and $c \cap N = \emptyset$*

Definition 3. *The* concept class of representable concept *is the set of all concepts having a representation in the concept language and is defined by $C_{R_C} = R_C(L_C)$*

2.2 Version Spaces

Mitchell [1] introduced the theoretical framework of versions spaces. He defines Version Space as the set of all concepts in L_C that are consistent with the sets of positive and negative examples P and N.

$$VS_{L_C}(P,N) = \{c \in L_C | R_C(c) \subset R_I(P), R_C(c) \cap R_I(N) = \emptyset\} \quad (1)$$

$$= R_C^{-1}([R_I(P), \complement R_I(N)]) \quad (2)$$

Mitchell proposes to represent version spaces by the mean of their boundaries. Both the Candidate Elimination algorithm and the Description Identification [2] Algorithms use maximally specific and maximally general concepts sets (resp. S and G) to represent the version space, and effectively compute them. The main flaw of this approach is the size of these boundaries. Haussler [3] has pointed out that the boundary size may grow exponentially in the number of examples. Thus, alternative representations have been proposed for the versions spaces.

Hirsh proposes to replace the set of maximally general concept G by the list of negative examples [4], and later to replace both S and G by the list of examples [5]. He also highlighted properties of intersecting version space: the intersection of two version spaces is the version space built by using $P_1 \cup P_2$ and $N_1 \cup N_2$ as positive and negative example sets. Another approach is to keep only one element of each boundary set, together with backtrack information [6]. This prevent exponential growth of the boundary set at reasonable computational cost. In order to address inconsistency problem, M. Sebag proposes to compute [7] simple version spaces (one positive example and all negative consistent ones) for each positive example and perform a vote among them.

3 Approximation and Version Spaces

Rough Set Theory (RST) efficiently deals with sets approximation [8]. Given a relation $\sim$ on C, it defines the dual operators H and L on 2^C such that $H(A) = \{a \in S/\exists b \in A, a \sim b\}$ and $L(A) = \{a \in S/\forall b, a \sim b \Rightarrow b \in A\}$ These operator were initially defined by Pawlak, with an equivalence relation $\sim$. However, requiring an equivalence relation is a strong constraint, and it has been relaxed, the definition of H and L being applied on any relation. Of course, many of the original RST properties are lost (see [9] for correspondence between $\sim$ properties and H and L ones).

3.1 Rough Consistency

Definition 4. *A concept $c \in C$ (resp. $\in L_C$) is said to be roughly consistent with positive and negative example sets P, N iff c (resp. $R_C(c)$) is similar to a concept consistent with P, N.*

Proposition 1. *A concept c roughly consistent with the example sets $(P_1 \cup N_1, P_2 \cup N_2)$ is necessarily roughly consistent with (P_1, N_1) and roughly consistent with (P_2, N_2). The converse proposition does not necessarily holds.*

Proof. $\exists c'$ consistent with $(P_1 \cup N_1, P_2 \cup N_2)$ such that $c \sim c'$. c' is consistent with both (P_1, N_1) and (P_2, N_2). Thus c is roughly consistent with both. The converse is not necessarily true because the concept consistent with (P_1, N_1) and (P_2, N_2) is not necessarily the same.

Definition 5. *We call a partition of the set of example (P, N) a* strategy *and note it Pa. The family of positives and negatives examples are noted $(P_{Pa})_i$ and $(N_{Pa})_i$ respectively. Whenever there is no ambiguity about Pa, it is omitted.*

We now express the approximative concept learning problem:
Given:

- a language L_I of instance representation and a language L_C of concept representation;
- a relation *cover* $\subset L_I \times L_c$
- a similarity relation $\sim$ on concepts $\subset C \times C$
- a set of positive example $P \subset L_I$ and a set of negative example $N \subset L_I$
- a strategy Pa based on (P, N)

Find: a concept $c \in L_C$ such that $R_C(c)$ is roughly consistent with each (P_i, N_i).

The approximative learning problem definition introduces two new element: the similarity relation and the strategy. The relation defines how to perform approximation. The strategy defines which examples are to be processed simultaneously.

Proposition 2. *Let us consider two different approximative concept learning problem ($ACLP$ and $ACLP'$) that are alike in respect to all parameters except the strategy (Pa and Pa'). If the strategy Pa' is finer than the strategy Pa, then every solution to $ACLP$ is a solution to $ACLP'$.*

Proof. This comes from proposition 1: if a concept is consistent with a set of examples, it is consistent with any of its subset. Each (P'_i, N'_i) in Pa' is a subset of some P_j, N_j in Pa. Thus, any concept roughly consistent with each P_j, N_j is also roughly consistent with each (P'_i, N'_i). As L_C is the same, the proposition holds.

3.2 RVS Definition

Definition 6. *Given an instance space I, an instance language L_I and its representation function R_I, a concept language L_C and its representation function R_C, a similarity relation $\sim$ on concepts and the associated approximation operator H and L, a set of positive examples P, a set of negatives examples N, a partition Pa of (P, N), the* rough version spaces *$RVS_H(Pa)$ is defined by:*

$$RVS_H(Pa) = \bigcap_i R_C^{-1}(H([R_I(P_i), {}^{\complement}R_I(N_i)])) \tag{3}$$

Although H and L are dual operator, RVS_H and RVS_L do not share the same properties. We will show that RVS_H is the most important and useful in this pair. This work is focused on RVS_H, given its nice properties in handling inconsistency.

Proposition 3. *$RVS_H(Pa)$ is exactly the set of all solutions to the approximate concept learning problem*

Proof. Using H definition, we expand $H([R_I(P_i), {}^{\complement}R_I(N_i)])$: $\{c \in C / \exists c' \in [R_I(P_i), {}^{\complement}R_I(N_i)], c \sim c'\}$, i.e. the set of all concepts similar to any concept consistent with (P_i, N_i). It holds that $R_C^{-1}(H([R_I(P_i), {}^{\complement}R_I(N_i)])$ is the set of all concept in L_C that are roughly consistent with (P_i, N_i). Hence, the set of concepts in L_C that is consistent with all (P_i, N_i) is the intersection of these set, $RVS_H(Pa)$.

We now define a notation for rough version space in the particular case where the strategy is a singleton.

Definition 7. *We call* simple RVS *a RVS defined using a strategy $Pa = \{(P, N)\}$ that is singleton. It is convenient to note it by the following way:*

$$RVS_H(P, N) = RVS_H(\{(P, N)\} = R_C^{-1}(H([R_I(P), {}^{\complement}R_I(N)])) \tag{4}$$

Property 1. Any RVS can be expressed by using only simple RVS: $RVS_H(Pa) = \bigcap_i RVS_H(P_i, N_i)$

Most result can be established on simple RVS and generalized to any RVS by using this fundamental property.

Property 2. Hirsh property on intersection does not hold with simple RVS. $RVS_H(P_1 \cup P_2, N_1 \cup N_2) \subset RVS_H(P_1, N_1) \cap RVS_H(P_2, N_2)$, and equality does not hold in general.

Proof. Using property 3, we translate directly property 1 in RVS terms.

Property 3. If we use the equality relation to build our upper operator (i.e. identity operator Id), we get: $VS_{L_C}(P,N) = RVS_{Id}(P,N)$

Any classical VS is particular case of a simple RVS.

3.3 Properties

The main bias in version space is the language. Only representable concept can be learn trough version space. Rough Version Spaces add a new bias to the language: the upper approximation operator H.

Definition 8. *The* induced concept class C_H *associated with the upper approximation operator* H *is defined by* $C_H = H(C)$

Property 4. The induced concept class C_H is directly related to the relation $\sim$:

$$C_H = \{c | \exists c' \in C, c \sim c'\} \tag{5}$$

This property is useful when creating a relation with a given induced concept class. It is also possible to restrict $\sim$ on $C_{R_C} \times C$. We get $C_H \subset C_{R_C}$ without changing $RVS_H(Pa)$.

Proposition 4. *If* Pa *is finer than* Pa'*, then* $RVS_H(Pa') \subset RVS_H(Pa)$

Proof. As $RVS_H(Pa)$ is the set of all solution to the approximate concept learning, this proposition is a direct application of proposition 2.

Proposition 5. *If* $\sim$ *restriction on* $C_{R_C} \times C$ *is reflexive, then*

$$VS_{L_C}(P,N) \subset RVS_H(P,N) \subset RVS_H(Pa) \tag{6}$$

Proof. First part: Let c be a concept in $VS_{L_C}(P,N)$. By definition of VS, $R_C(c) \in [R_I(P), \complement R_I(N)]$ and $R_C(c) \in C_{R_C}$. Thus, $R_C(c) \sim R_C(c)$ and $R_C(c) \in H([R_I(P), \complement R_I(N)])$. It proves that $c \in RVS_H(P,N)$. The second part is the previous proposition.

4 Algorithms and Implementation

The method proposed to deal with inconsistency when using RVS is to search a strategy Pa such that $RVS(Pa)$ does not collapse. The algorithm is the following : starting with the partition $Pa = \{(P,N)\}$, we refine it by dividing its larger part in two until $RVS(Pa)$ does not collapse. This ensures that if there exists Pa such that $RVS(Pa)$ does not collapse, then we will find one.

Computing RVS extensively is not affordable, except on toy problems. Thus, we propose to find a bounding segment to the simple $RVS(P,N)$.

Proposition 6. *Given positive and negative example sets P, N, we search for P', N' such that:*

$$P' = \{x \in L_I | \forall c \in RVS_H(P,N), c \text{ cover x}\} \tag{7}$$

$$N' = \{x \in L_I | \forall c \in RVS_H(P,N), \neg c \text{ cover x}\} \tag{8}$$

We have the following VS bound for $RVS_H(P,N)$

$$RVS_H(P,N) \subset VS_{L_C}(P',N') \tag{9}$$

Proof. Any concept c in $RVS_H(P,N)$ cover all example in P' and no example in N', and is in L_C. Therefore, c is in $VS_{L_C}(P',N')$

Property 5. If $\sim$ restriction on C_{R_C} is reflexive, then:

$$VS_{L_C}(P,N) \subset RVS_H(P,N) \subset VS_{L_C}(P',N') \tag{10}$$

Corollary 1. *If $\sim$ restriction on C_{R_C} is reflexive, then $N' \subset N$ and $P' \subset P$.*

Proof. The previous property gives $VS_{L_C}(P,N) \subset VS_{L_C}(P',N')$. Equivalently, $(P',N') \subset (P,N)$: the more example, the smaller the VS.

Computing P' and N' may be a hard task if H is only known extensively. However, we expect that in case where H is defined using logic proposition, figuring out P' and N' is affordable (it is possible do define H to fit our needs). If any representable concept is similar to itself, then the corollary hold. In this case, we only need to test each example.

4.1 RVS Approximation Combination

RVS can be approximated by using simple RVS approximation and Hirsh property on VS intersection:

$$RVS_H(Pa) \subset VS_{L_C}(\bigcup_i P'_i, \bigcup_i N'_i) \tag{11}$$

Using this property, we approximate RVS by using only classical VS tools. VS use and computation is a well known problem, and any results on VS apply here.

4.2 Refining RVS Approximation

The RVS approximation by VS may be considered as a sufficient result: it allows concept learning in case where VS collapsed by using VS as a tool. but it may be interesting to give a glimpse at the "real" RVS. As RVS set is not a segment (in fact, it is not a convex set), it can only be bounded by a pair of boundary sets S and G, not fully described. The previous approximation provides such bounds, namely the approximation VS bounds, but there is no guarantee that any element in S or G is actually in RVS. So the idea is to refine these bound by using each simple RVS in turn until it collapse or an element in RVS is found.

Table 1. Positive and negative examples

P/N							P'/N'
P_1	sun	warm	normal	strong	warm	same	P'_1
P_1	sun	warm	high	strong	warm	same	P'_1
P_1	sun	warm	high	strong	cool	change	P'_1
P_1	sun	warm	normal	strong	cool	change	P'_1
P_1	rain	cold	normal	weak	warm	same	
P_2	sun	warm	normal	weak	warm	same	
P_2	sun	cold	normal	strong	warm	same	
N_1	rain	cold	high	strong	warm	change	N'_1
N_1	rain	cold	normal	strong	warm	same	N'_1
N_1	rain	cold	normal	weak	cool	change	N'_1
N_1	rain	cold	normal	weak	cool	same	N'_1
N_1	sun	cold	normal	weak	warm	same	N'_1
N_1	rain	warm	normal	weak	cool	same	N'_1
N_2	rain	cold	high	weak	cool	same	N'_2
N_2	rain	cold	high	strong	warm	change	N'_2
N_2	rain	cold	normal	strong	cool	same	
N_2	sun	cold	normal	strong	warm	change	

5 RVS by Hand

This example is strongly inspired by Mitchell EnjoySport's one in [10]. It explains the extend if empty algorithm on data for a case where $RVS_H(P, N)$ collapse, and show how it is possible to find a successful strategy. $L_I = (sun, rain) \times (warm, cold) \times (normal, high)$
$\times (strong, weak) \times (warm, cool) \times (same, change)$

$L_C = (sun, rain, \star) \times (warm, cold, \star) \times (normal, high, \star)$
$\times (strong, weak, \star) \times (warm, cool, \star) \times (same, change, \star) \cup \{\emptyset\}$

Examples are given in table 5 ($N = N_1 \cup N_2$ and $P = P_1 \cup P_2$) $H(A) = \{a \in C_{R_C}/\exists a' \in A, |a \Delta\ a'| \leq 1\}$

This states that approximately valid concepts are defined by the following properties: They are representable and they classify all examples as some valid concept, except at most one.

Remarks that $c \in C_{R_C} \Rightarrow c \in [c]$, so $\sim$ is reflexive on C_{R_C}. Hence, we have $P' \subset P$ and $N' \subset N$ (Corollary 1).

If p is in P but not in P', it means that it exists $c \in H([P, {}^{\complement}N])$ such that $p \notin c$, i.e. $c \in C_{R_C} \cap [P - \{p\}, {}^{\complement}N - \{p\}]$. This state that $VS_{L_C}(P - \{p\}, N \cup \{p\})$ does not collapse. Conversely, if it does not collapse, p is not in P'. We find dual property for N and N'. By using this on P, N, we have that $P' = P$ and $N = N'$, and then $RVS_H(P, N) = VS_H(P, N) = \emptyset$. Thus, we need another strategy. Let us try $Pa = \{(P_1, N_1), (P_2, N_2)\}$.

By using collapse tests, we get P'_1, P'_2 and N'_1, N'_2 given in table 5. Using these result and the approximation property of RVS, we found that: $RVS_H(Pa) \leq VS_{L_C}(P'_1 \cup P'_2, N'_1 \cup N'_2)$.

We search S and G bounding set for $VS_{L_C}(P'_1 \cup P'_2, N'_1 \cup N'_2)$: $S = \{(sun, warm, \star, strong, \star, \star)\}$

$G = \{(sun, warm, \star, \star, \star, \star), (\star, warm, \star, strong, \star, \star), (sun, \star, \star, strong, \star, \star)\}$

Actually, $VS = S \cup G$. Let us now search a concept in $RVS_H(Pa)$. The concept in S is not valid according for $RVS_H(P_2, S_2)$. The first concept in G is the only one valid.

So we have $RVS_H(Pa) = \{(sun, warm, \star, \star, \star, \star)\}$.

If it had been empty, we would have try to refine our strategy. Obviously, if each example is isolated, we get a non empty RVS ($\emptyset$ is in this RVS). We are sure to always find a valid strategy.

6 Conclusion

In this paper, we have proposed an approximative extension of the classical consistency property by using rough set theory. This allowed us to build an approximative concept learning framework, and an improved Version Space, namely the Rough Version Space. Taking advantage of some useful properties of this Rough Version Space, we proposed some general methods and algorithms to compute this Rough Version Space. These have been applied on an example where classical version space would have collapsed, and our methods give a meaningful result.

As previously stated here, methods and algorithms proposed are generic. It is possible to take advantage of particular approximation operator to develop more efficient specific approach to handle RVS.

References

1. Mitchell, T.M.: Version Spaces: An approach to Concept Learning. PhD thesis, Stanford University (1978)
2. Mellish, C.: The description identification problem. Artificial Intelligence **52** (1991) 151–167
3. Haussler, D.: Quantifying inductive bias: AI learning algorithms and valiant's learning framework. Artificial Intelligence **36** (1988) 177–221
4. Hirsh, H.: Polynomial-time learning with version spaces. In : National Conference on Artificial Intelligence (1992) 117–122
5. Hirsh, H., Mishra, N., Pitt, L.: Version spaces without boundary sets. In: AAAI/IAAI. (1997) 491–496
6. Sablon, G., De Raedt, L., Bruynooghe, L.: Iterative versionspaces. Artificial Intelligence **69** (1994) 393–409
7. Sebag, M.: Delaying the choice of bias: A disjunctive version space approach. In: International Conference on Machine Learning.(1996) 444–452
8. Pawlak, Z.: Rough Sets: Theorical Aspects of Reasoning About Data. Kluwer Academic Publishers, Dordrecht, Netherlands. (1991)
9. Yao, Y.Y.: Constructive and algebraic approaches for generalized rough set models. (In: Bulletin of International Rough Set Society)
10. Mitchell, T.M.. Machine Learning. McGraw-Hill. (1997)