

Adaptable Boundary Sets

E.N. Smirnov H.J. van den Herik I.G. Sprinkhuizen-Kuyper

IKAT/Infonomics,
Department of Computer Science,
Maastricht University,
P.O.Box 616, 6200 MD Maastricht,
The Netherlands

e-mail: {smirnov, herik, kuyper}@cs.unimaas.nl

Abstract

This paper proposes adaptable boundary sets as a new version-space representation. It is shown that adaptable boundary sets can be adjusted to a version-space representation between boundary sets [5, 6, 7] and instance-based boundary sets [11, 12, 13]. This allows the choice of proper version-space representations to be realised dynamically during the learning phase.

1 Introduction

Version spaces are an approach to concept learning [5, 6, 7]. They are defined as sets of descriptions in concept languages that correctly classify training instances. When concept languages are partially ordered, version spaces can be represented by boundary sets. Boundary sets are sets of minimal and maximal descriptions in version spaces. It was proven that they correctly represent version spaces [5, 11].

An analysis of boundary sets shows that their size can grow exponentially in the number of training instances [1]. To overcome this problem alternative version-space representations were introduced in [2, 3, 4, 8, 9, 10, 11, 12, 15]. They extend the scope of concept languages for which version spaces are efficiently applicable.

One of the main difficulties with the alternative version-space representations is that in some cases they are less computationally efficient than boundary sets although their size is still polynomial in the number of training instances. This makes the choice of version-space representations a serious implementation problem. To solve this problem we propose a new version-space representation called adaptable boundary sets. Depending on the memory requirements the representation can be adjusted to a version-space representation between boundary sets and instance-based boundary sets [11, 12]. Thus, the choice of proper version-space representations can be realised dynamically during the learning phase.

The paper is organised as follows¹. Section 2 provides a formalisation of concept learning. The new version-space representation is introduced in section 3. Its learning, merging and other useful algorithms are given in sections 4, 5, and 6, respectively. Section 7 presents an example of the representation application. Finally, in section 8 conclusions are given.

2 Formalisation

Formalisation starts by determining the elements of concept learning. Let I be a set of descriptions of all possible entities in some domain. A concept C is defined as a subset of I . Concepts are represented

¹The paper is a short version of [14] where the omitted proofs can be found.

in a concept language Lc . The language Lc is defined as a set of descriptions c s.t. (1) every concept is represented by at most one description c , and (2) every description c represents exactly one concept.

The elements of concepts are called instances. They are related to concept descriptions by a cover relation M . The cover relation $M(c, i)$ holds for $c \in Lc$ and $i \in I$ if and only if the instance i is a member of the concept represented by c . A description $c \in Lc$ is said to cover an instance $i \in I$ if and only if the cover relation $M(c, i)$ holds.

As a rule any target concept C is incompletely defined by sets $I^+ \subseteq I$ and $I^- \subseteq I$ of positive and negative training instances s.t. $I^+ \subseteq C$ and $I^- \cap C = \emptyset$. Hence, the concept-learning task in this case is to find descriptions of C in Lc .

To find the descriptions of a target concept, we specify them by the consistency criterion: a concept description c is consistent if and only if c correctly classifies training instances. The set of all the consistent descriptions in a concept language is called the version space [5].

Definition 1 (Version Space) *Given a set I of all possible entities, a concept language Lc , a cover relation M , and sets I^+ and I^- , the version space $VS(I^+, I^-)$ is defined as follows:*

$$VS(I^+, I^-) = \{c \in Lc \mid (\forall i \in I^+)M(c, i) \wedge (\forall i \in I^-)\neg M(c, i)\}.$$

In order to learn version spaces, they have to be represented. The key to find a good version-space representation is to observe that concept languages can be ordered. The order can be based on a relation “more specific” [5, 7].

Definition 2 (Relation “more specific” (\leq)) *Consider a set I of all possible entities, a concept language Lc , and a cover relation M . Then:*

$$(\forall c_1, c_2 \in Lc)((c_1 \leq c_2) \leftrightarrow (\forall i \in I)(M(c_1, i) \rightarrow M(c_2, i))).$$

When the relation “ \leq ” is defined on a concept language Lc , a description $c_1 \in Lc$ is said to be more specific than a description $c_2 \in Lc$ if and only if $c_1 \leq c_2$. By duality, a description $c_1 \in Lc$ is said to be more general than a description $c_2 \in Lc$ if and only if $c_2 \leq c_1$.

The relation “ \leq ” is a partial order [5]. Hence, if “ \leq ” is defined on a concept language Lc , then Lc is partially-ordered. One class of partially ordered languages was extensively used for defining version-space representations. This is the class of admissible concept languages. It is introduced using the notion of bounded sets. Bounded sets are given after we state what minimal and maximal sets are.

Definition 3 (Minimal/Maximal Sets) *If C is a partially-ordered set, then:*

$$\begin{aligned} MIN(C) &= \{c \in C \mid (\forall c' \in C)((c' \leq c) \rightarrow (c' = c))\} \\ MAX(C) &= \{c \in C \mid (\forall c' \in C)((c \leq c') \rightarrow (c' = c))\}. \end{aligned}$$

The minimal and maximal sets of version spaces are known as minimal and maximal boundary sets [5, 7]. To refer to them we introduce the following notation.

Notation 4 $MIN(VS(I^+, I^-))$ is denoted by $S(I^+, I^-)$; $MAX(VS(I^+, I^-))$ is denoted by $G(I^+, I^-)$.

Definition 5 (Bounded Sets) *A partially ordered set C is bounded if and only if:*

$$(\forall c \in C)((\exists s \in MIN(C))(s \leq c) \wedge (\exists g \in MAX(C))(c \leq g)).$$

Definition 6 (Admissible Concept Languages) A partially-ordered concept language Lc is admissible if and only if every non-empty subset $C \subseteq Lc$ is bounded.

Most admissible concept languages, used in practice, exhibit one of the two dual properties below.

Property 1 Each non-empty set $C \subseteq Lc$ has greatest lower bound $glb(C)$ s.t.:

$$(\forall i \in I)((\forall c \in C)M(c, i) \leftrightarrow M(glb(C), i)).$$

Property 2 Each non-empty set $C \subseteq Lc$ has least upper bound $lub(C)$ s.t.:

$$(\forall i \in I)((\exists c \in C)M(c, i) \leftrightarrow M(lub(C), i)).$$

An admissible concept language Lc exhibits property 1 when for each nonempty subset $C \subseteq Lc$ there exists the greatest lower bound $glb(C)$ such that an instance $i \in I$ is covered by all the concept descriptions $c \in C$ if and only if i is covered by $glb(C)$. It was proven in [11] that when property 1 holds the size of the set $S(I^+, I^-)$ is equal to one for all $I^+ \subseteq I$ and $I^- \subseteq I$.

By duality we can explain when the language exhibits property 2.

3 Adaptable Boundary Sets

Adaptable boundary sets are a new version-space representation with the important property that its size can be adjusted. The formal definition of the representation is based on the notion of a special covering of a set, given below.

Definition 7 (Special Covering) P is a special covering of a set S if and only if:

- (1) $P = \{\emptyset\}$ if $S = \emptyset$, and
- (2) P is a covering of S if $S \neq \emptyset$.

Notation 8 The set of all special coverings of a set S is denoted as $SP(S)$.

Definition 9 (Adaptable Boundary Sets (ABSs)) If Lc is an admissible concept language, then adaptable boundary sets of a version space $VS(I^+, I^-)$ are an ordered pair of indexed families of sets $\langle \{S(I_p^+, I^-)\}_{I_p^+ \in P^+}, \{G(I^+, I_p^-)\}_{I_p^- \in P^-} \rangle$, where $P^+ \in SP(I^+)$ and $P^- \in SP(I^-)$.

To prove that ABSs are a correct version-space representation for the class of admissible concept languages, we consider two dependencies:

- if $c \in Lc$ is more general than an element of the set $S(I_p^+, I^-)$ for each training set $I_p^+ \in Q^+$, then c is consistent with each set I_p^+ , where Q^+ is a nonempty set and $Q^+ \subseteq P^+$;
- if $c \in Lc$ is more specific than an element of the set $G(I^+, I_p^-)$ for each training set $I_p^- \in Q^-$, then c is consistent with each set I_p^- , where Q^- is a nonempty set and $Q^- \subseteq P^-$.

The dependencies are reformulated in lemmas 10 and 11.

Lemma 10 Consider a nonempty set $Q^+ \subseteq P^+$. If Lc is an admissible concept language, then:

$$(\forall c \in Lc)((\forall I_p^+ \in Q^+)(\exists s \in S(I_p^+, I^-))(s \leq c) \rightarrow (\forall I_p^+ \in Q^+)(\forall i \in I_p^+)M(c, i)).$$

Lemma 11 Consider a nonempty set $Q^- \subseteq P^-$. If Lc is an admissible concept language, then:

$$(\forall c \in Lc)((\forall I_p^- \in Q^-)(\exists g \in G(I^+, I_p^-))(c \leq g) \rightarrow (\forall I_p^- \in Q^-)(\forall i \in I_p^-)\neg M(c, i)).$$

Theorem 12 (Correctness of Adaptable Boundary Sets) Consider a version space $VS(I^+, I^-)$ given by ABSs: $\langle \{S(I_p^+, I^-)\}_{I_p^+ \in P^+}, \{G(I^+, I_p^-)\}_{I_p^- \in P^-} \rangle$. If the concept language Lc is admissible, then:

$$(\forall c \in Lc)((c \in VS(I^+, I^-)) \leftrightarrow ((\forall I_p^+ \in P^+)(\exists s \in S(I_p^+, I^-))(s \leq c) \wedge (\forall I_p^- \in P^-)(\exists g \in G(I^+, I_p^-))(c \leq g))).$$

Theorem 12 states that ABSs correctly represent each version space $VS(I^+, I^-)$ for the class of admissible concept languages. The descriptions in $VS(I^+, I^-)$ are exactly those that are (1) more general than an element of each set $S(I_p^+, I^-)$, and (2) more specific than an element of each set $G(I^+, I_p^-)$. Thus, all the elements of $VS(I^+, I^-)$ are considered and are not explicitly enumerated.

Given training sets I^+ and I^- , all the special coverings $P^+ \in SP(I^+)$ and $P^- \in SP(I^-)$ determine a set $R(I^+, I^-)$ of all possible ABSs of $VS(I^+, I^-)$. Two elements in $R(I^+, I^-)$ are well-known in machine learning. They are as follows:

- (1) if $P^+ = \{I^+\}$ and $P^- = \{I^-\}$, then ABSs are defined as $\langle \{S(I^+, I^-)\}, \{G(I^+, I^-)\} \rangle$. Thus, according to [5, 6, 7] they are equal to boundary sets.
- (2) if $(\forall I_p^+ \in P^+)(\exists i \in I^+)(I_p^+ = \{i\})$ and $(\forall I_p^- \in P^-)(\exists i \in I^-)(I_p^- = \{i\})$, ABSs are defined as $\langle \{S(\{p\}, I^-)\}_{p \in I^+}, \{G(I^+, \{n\})\}_{n \in I^-} \rangle$. Thus, according to [11, 12, 13] they are equal to instance-based boundary sets.

Boundary sets and instance-based boundary sets are boundaries of the set $R(I^+, I^-)$ if it is ordered. Hence, all the adaptable boundary sets in the set $R(I^+, I^-)$ can be considered between these two version-space representations.

4 Learning Algorithm

The learning algorithm updates the ABSs of a version space, when a new training instance is given. It is proposed for the class of admissible concept languages and consists of two parts.

The first part of the algorithm computes the new special covering of one of the training sets, given a new training instance $i \in I$. More precisely, if the instance i is positive, the algorithm adds i to the training set I^+ . The special covering $P^{+'}$ of the updated set $I^{+'}$ ($I^{+'} = I^+ \cup \{i\}$) is formed from the special covering P^+ of the set I^+ s.t. the instance i is added to some elements (training subsets) of P^+ and/or the set $\{i\}$ is added to P^+ . If the instance i is negative, the formation of the updated set $I^{-'}$ and its special covering $P^{-'}$ is realised analogously.

Formally, the special coverings $P^{+'}$ and $P^{-'}$ are called extensions of the special coverings P^+ and P^- . The notion of extension is given below.

Definition 13 (Extension of a Special Covering) Let P and P' be special coverings of sets S and $S \cup \{i\}$. P' is an extension of P if and only if:

$$(\forall I'_p \in P')((\exists I_p \in P)(I'_p = I_p) \vee (\exists I_p \in P)(I'_p = I_p \cup \{i\}) \vee (I'_p = \{i\})).$$

Note that the first part of the algorithm is not completely specified. In practice it is determined s.t. the size of the ABSs, based on the new covering, is minimised.

The second part of the learning algorithm computes the ABSs of the version space that is consistent with the processed training data and the new training instance i . More precisely, if the instance i is positive, the algorithm computes the ABSs: $\langle \{S(I_p^{+'}, I^-)\}_{I_p^{+'} \in P^{+'}}, \{G(I^{+'}, I_p^-)\}_{I_p^- \in P^-} \rangle$ ² of the version space $VS(I^{+'}, I^-)$, given the ABSs: $\langle \{S(I_p^+, I^-)\}_{I_p^+ \in P^+}, \{G(I^+, I_p^-)\}_{I_p^- \in P^-} \rangle$ of the version space $VS(I^+, I^-)$ and the special covering $P^{+'}$ of the training set $I^{+'}$. If the instance i is negative, the input/output of this part of the algorithm is analogously described.

The second part of the algorithm is based on theorems 14 and 15 given below.

Theorem 14 Consider a version space $VS(I^+, I^-)$ given by $\langle \{S(I_p^+, I^-)\}_{I_p^+ \in P^+}, \{G(I^+, I_p^-)\}_{I_p^- \in P^-} \rangle$, and a version space $VS(I^{+'}, I^-)$ given by $\langle \{S(I_p^{+'}, I^-)\}_{I_p^{+'} \in P^{+'}}, \{G(I^{+'}, I_p^-)\}_{I_p^- \in P^-} \rangle$, where $I^{+'} = I^+ \cup \{i\}$, $P^{+'} \in SP(I^{+'})$, and $P^{+'}$ is an extension of P^+ . If L_c is an admissible concept language, then:

$$\begin{aligned} G(I^{+'}, I_p^-) &= \{g \in G(I^+, I_p^-) \mid M(g, i)\} \text{ for } I_p^- \in P^- \\ S(I_p^{+'}, I^-) &= S(I_p^+, I^-) \text{ for } I_p^{+'} \in P^{+'} \text{ if } (\exists I_p^+ \in P^+)(I_p^{+'} = I_p^+) \\ S(I_p^{+'}, I^-) &= \text{MIN}(\{c \in VS(I_p^+, I^-) \mid M(c, i)\}) \text{ for } I_p^{+'} \in P^{+'} \text{ if } (\exists I_p^+ \in P^+)(I_p^{+'} = I_p^+ \cup \{i\}) \\ S(\{i\}, I^-) &= \text{MIN}(VS(\{i\}, I^-)) \text{ if } \{i\} \in P^{+'}. \end{aligned}$$

Theorem 15 Consider a version space $VS(I^+, I^-)$ given by $\langle \{S(I_p^+, I^-)\}_{I_p^+ \in P^+}, \{G(I^+, I_p^-)\}_{I_p^- \in P^-} \rangle$, and a version space $VS(I^+, I^{-'})$ given by $\langle \{S(I_p^+, I^-)\}_{I_p^+ \in P^+}, \{G(I^+, I_p^{-'})\}_{I_p^{-'} \in P^{-'}} \rangle$, where $I^{-'} = I^- \cup \{i\}$, $P^{-'} \in SP(I^{-'})$, and $P^{-'}$ is an extension of P^- . If L_c is an admissible concept language, then:

$$\begin{aligned} S(I_p^+, I^{-'}) &= \{s \in S(I_p^+, I^-) \mid \neg M(s, i)\} \text{ for } I_p^+ \in P^+ \\ G(I^+, I_p^{-'}) &= G(I^+, I_p^-) \text{ for } I_p^{-'} \in P^{-'} \text{ if } (\exists I_p^- \in P^-)(I_p^{-'} = I_p^-) \\ G(I^+, I_p^{-'}) &= \text{MAX}(\{c \in VS(I^+, I_p^-) \mid \neg M(c, i)\}) \text{ for } I_p^{-'} \in P^{-'} \text{ if } (\exists I_p^- \in P^-)(I_p^{-'} = I_p^- \cup \{i\}) \\ G(I^+, \{i\}) &= \text{MAX}(VS(I^+, \{i\})) \text{ if } \{i\} \in P^{-'}. \end{aligned}$$

We describe the second part of the learning algorithm. Given a new positive instance $i \in I$, the ABSs $\langle \{S(I_p^+, I^-)\}_{I_p^+ \in P^+}, \{G(I^+, I_p^-)\}_{I_p^- \in P^-} \rangle$ of a version space $VS(I^+, I^-)$, and a special covering $P^{+'}$ of the training set $I^{+'} = I^+ \cup \{i\}$, the algorithm executes four steps. In the first step for each $I_p^- \in P^-$ the algorithm forms the set $G(I^{+'}, I_p^-)$. The set is formed from those elements of the set $G(I^+, I_p^-)$ that cover the instance i . In the second step for each set $I_p^{+'} \in P^{+'}$ for which there exists a set $I_p^+ \in P^+$ s.t. $I_p^{+'} = I_p^+$ the algorithm takes the set $S(I_p^{+'}, I^-)$ from the ABSs $\langle \{S(I_p^+, I^-)\}_{I_p^+ \in P^+}, \{G(I^+, I_p^-)\}_{I_p^- \in P^-} \rangle$. In the third step for each set $I_p^{+'} \in P^{+'}$ for which there exists a set $I_p^+ \in P^+$ s.t. $I_p^{+'} = I_p^+ \cup \{i\}$, the algorithm forms the set $S(I_p^{+'}, I^-)$. The set is formed from the minimal elements of the version space $VS(I_p^+, I^-)$ that cover the instance i . In the fourth step the algorithm generates the set $S(\{i\}, I^-)$, if the set $\{i\}$ is in $P^{+'}$. After execution of all the four steps the ABSs $\langle \{S(I_p^{+'}, I^-)\}_{I_p^{+'} \in P^{+'}}, \{G(I^{+'}, I_p^-)\}_{I_p^- \in P^-} \rangle$ of the version space $VS(I^{+'}, I^-)$ is returned.

The algorithm's second-part performance for a negative instance is dual to that for a positive instance; hence, it is analogous in form.

5 Merging Algorithms

The merging algorithms change ABSs by merging some elements (training subsets) in the special coverings they are based on. In this section we consider several merging algorithms that differ in their concept-language applicability.

²Note that $I^{+'} = I^+ \cup \{i\}$ and $P^{+'}$ is a special covering of $I^{+'}$.

5.1 General Merging Algorithm

The general merging algorithm is proposed for the class of admissible concept languages. It consists of two parts. To describe the first part of the algorithm we introduce the notion of merged regroupment.

Definition 16 (Merged Regroupment) *Let P and P' be special coverings of a set S . P' is a merged regroupment of P if and only if:*

$$(\forall I'_p \in P')(\exists Q \subseteq P)(I'_p = \cup_{I_p \in Q} I_p).$$

Given the ABSs $\langle \{S(I_p^+, I^-)\}_{I_p^+ \in P^+}, \{G(I^+, I_p^-)\}_{I_p^- \in P^-} \rangle$ of a version space $VS(I^+, I^-)$, the first part of the algorithm computes regroupments $P^{+'}$ and $P^{-'}$ of the special coverings P^+ and P^- , respectively. Note that this computation is not specified. In practice it is determined s.t. the size of the merged ABSs, based on the new coverings, is minimised.

The algorithm's second part computes the merged ABSs $\langle \{S(I_p^{+'}, I^-)\}_{I_p^{+'} \in P^{+'}}, \{G(I^+, I_p^{-'})\}_{I_p^{-'} \in P^{-'}} \rangle$ of the version space $VS(I^+, I^-)$, given the ABSs $\langle \{S(I_p^+, I^-)\}_{I_p^+ \in P^+}, \{G(I^+, I_p^-)\}_{I_p^- \in P^-} \rangle$ of the same version space, and the special coverings $P^{+'}$ and $P^{-'}$. This part of the algorithm is based on theorem 17 given below.

Theorem 17 *Consider a version space $VS(I^+, I^-)$ given by $\langle \{S(I_p^+, I^-)\}_{I_p^+ \in P^+}, \{G(I^+, I_p^-)\}_{I_p^- \in P^-} \rangle$ and $\langle \{S(I_p^{+'}, I^-)\}_{I_p^{+'} \in P^{+'}}, \{G(I^+, I_p^{-'})\}_{I_p^{-'} \in P^{-'}} \rangle$ such that $P^{+'}$ is a merged regroupment of P^+ and $P^{-'}$ is a merged regroupment of P^- . If Lc is an admissible concept language, then:*

$$\begin{aligned} S(I_p^{+'}, I^-) &= \{c \in MG(\{S(I_p^+, I^-)\}_{I_p^+ \in Q^+}) \mid (\forall n \in I^-) \neg M(c, n)\} \text{ for } I_p^{+'} \in P^{+'} \\ G(I^+, I_p^{-'}) &= \{c \in MS(\{G(I^+, I_p^-)\}_{I_p^- \in Q^-}) \mid (\forall p \in I^+) M(c, p)\} \text{ for } I_p^{-'} \in P^{-'} \end{aligned}$$

where:

$$\begin{aligned} Q^+ &\in SP(I_p^{+'}) \text{ and } Q^+ \subseteq P^+ \\ Q^- &\in SP(I_p^{-'}) \text{ and } Q^- \subseteq P^- \\ MG(\{S(I_p^+, I^-)\}_{I_p^+ \in Q^+}) &= MIN(\{c \in Lc \mid (\forall I_p^+ \in Q^+) (\exists s \in S(I_p^+, I^-)) (s \leq c)\}) \\ MS(\{G(I^+, I_p^-)\}_{I_p^- \in Q^-}) &= MAX(\{c \in Lc \mid (\forall I_p^- \in Q^-) (\exists g \in G(I^+, I_p^-)) (c \leq g)\}). \end{aligned}$$

We describe the second part of the general merging algorithm. Given the ABSs $\langle \{S(I_p^+, I^-)\}_{I_p^+ \in P^+}, \{G(I^+, I_p^-)\}_{I_p^- \in P^-} \rangle$ of a version space $VS(I^+, I^-)$, and the special coverings $P^{+'}$ and $P^{-'}$, the algorithm executes two steps. In the first step for each set $I_p^{+'} \in P^{+'}$ the algorithm forms the set $S(I_p^{+'}, I^-)$. If $I_p^{+'} \in P^+$, the set $S(I_p^{+'}, I^-)$ is taken from the ABSs $\langle \{S(I_p^+, I^-)\}_{I_p^+ \in P^+}, \{G(I^+, I_p^-)\}_{I_p^- \in P^-} \rangle$ ³. If $I_p^{+'} \notin P^+$, the algorithm first computes the set $MG(\{S(I_p^+, I^-)\}_{I_p^+ \in Q^+})$ using the nonempty set $Q^+ \subseteq P^+$ s.t. $I_p^{+'} = \bigcup_{I_p^+ \in Q^+} I_p^+$. The set $MG(\{S(I_p^+, I^-)\}_{I_p^+ \in Q^+})$ is computed as a set of minimal generalisations s.t. each generalisation is more general than at least one element of each set $S(I_p^+, I^-)$ for all $I_p^+ \in Q^+$. Then the algorithm forms the set $S(I_p^{+'}, I^-)$ from those elements of the set $MG(\{S(I_p^+, I^-)\}_{I_p^+ \in Q^+})$ that do not cover any negative instance $n \in I^-$. In the second step for each set $I_p^{-'} \in P^{-'}$, the algorithm forms the set $G(I^+, I_p^{-'})$. If $I_p^{-'} \in P^-$, the set $G(I^+, I_p^{-'})$ is taken from the ABSs $\langle \{S(I_p^+, I^-)\}_{I_p^+ \in P^+}, \{G(I^+, I_p^-)\}_{I_p^- \in P^-} \rangle$. If $I_p^{-'} \notin P^-$, the algorithm first computes the set $MS(\{G(I^+, I_p^-)\}_{I_p^- \in Q^-})$ using the nonempty set $Q^- \subseteq P^-$ s.t. $I_p^{-'} = \bigcup_{I_p^- \in Q^-} I_p^-$. The set $MS(\{G(I^+, I_p^-)\}_{I_p^- \in Q^-})$ is computed as a set of minimal specialisations s.t. each specialisation is more

³Note that this sub-step does not follow theorem 17 for efficiency reasons.

specific than at least one element of each set $G(I^+, I_p^-)$ for all $I_p^- \in Q^-$. Then the algorithm forms the set $G(I^+, I_p^-)$ from those elements of the set $MS(\{G(I^+, I_p^-)\}_{I_p^- \in Q^-})$ that do cover all the positive instances $p \in I^+$. After the execution of both steps, the merged ABSs $\langle \{S(I_p^{+'}, I^-)\}_{I_p^{+'} \in P^{+'}}, \{G(I^+, I_p^-)\}_{I_p^- \in P^{-'}} \rangle$ of the version space $VS(I^+, I^-)$ is returned.

5.2 Specialised Merging Algorithms

In this subsection we propose two specialised merging algorithms applicable for the class of admissible concept languages when properties 1 and 2 hold. We show that by investigating the properties, the parts of the algorithms that determine coverings of the merged ABSs can be completely specified. Moreover, we show that computing the merged ABSs is simplified.

5.2.1 Specialised Merging Algorithm: the Case of Property 1

The specialised merging algorithm, that exploits property 1, is based on theorem 18. Given a nonempty set $Q^- \subseteq P^-$ s.t. $(\forall I_p^- \in Q^-) |G(I^+, I_p^-)| = 1$, the theorem determines how to compute the set $G(I^+, \cup_{I_p^- \in Q^-} I_p^-)$.

Theorem 18 *Consider a nonempty set $Q^- \subseteq P^-$ s.t. $(\forall I_p^- \in Q^-) |G(I^+, I_p^-)| = 1$. If Lc is an admissible concept language and property 1 holds, then:*

$$G(I^+, \cup_{I_p^- \in Q^-} I_p^-) = \{glb(\cup_{I_p^- \in Q^-} G(I^+, I_p^-))\}.$$

We describe the specialised merging algorithm under the assumption that the set $S(I^+, I^-)$ of the version space $VS(I^+, I^-)$ to be learned consists of only one element. Note that the assumption is plausible since the size of the set is equal to one if property 1 holds. Thus, the input ABSs is $\langle \{S(I^+, I^-)\}, \{G(I^+, I_p^-)\}_{I_p^- \in P^-} \rangle$ of $VS(I^+, I^-)$.

The first part of the algorithm forms a new regroupment $P^{-'}$ of the special covering P^- in two steps. In the first step all the sets $G(I^+, I_p^-)$, which size is equal to one, are determined. All the subsets I_p^- are put in a set Q^- . In the second step the set $I_p^{-'} = \cup_{I_p^- \in Q^-} I_p^-$ is created. $P^{-'}$ is formed from P^- by excluding all the subsets $I_p^- \in Q^-$ and adding the set $I_p^{-'}$.

The second part of the algorithm computes the set $G(I^+, I_p^{-'})$. By theorem 18 the set is computed consisting of the greatest lower bound of the set $\cup_{I_p^- \in Q^-} G(I^+, I_p^-)$. After this computation the merged ABSs $\langle \{S(I^+, I^-)\}, \{G(I^+, I_p^{-'})\}_{I_p^{-'} \in P^{-'}} \rangle$ of the version space $VS(I^+, I^-)$ is returned.

5.2.2 Specialised Merging Algorithm: the Case of Property 2

The specialised merging algorithm, that exploits property 2, is based on theorem 19. The theorem is dual to theorem 18 and can be analogously explained.

Theorem 19 *Consider a nonempty set $Q^+ \subseteq P^+$ s.t. $(\forall I_p^+ \in Q^+) |S(I_p^+, I^-)| = 1$. If Lc is an admissible concept language and property 2 holds, then:*

$$S(\cup_{I_p^+ \in Q^+} I_p^+, I^-) = \{lub(\cup_{I_p^+ \in Q^+} S(I_p^+, I^-))\}.$$

Since property 2 is dual to property 1, the algorithm can be explained analogously to the specialised merging algorithm from the previous subsection.

6 Other Useful Algorithms

In this section we propose two useful algorithms of ABSs. The first algorithm checks whether version spaces represented by ABSs are empty. It can be applied for the class of admissible concept languages when one of properties 1 and 2 holds. The algorithm is based on theorems 20 and 21.

Theorem 20 *If Lc is an admissible concept language and property 1 holds, then:*

$$(VS(I^+, I^-) \neq \emptyset) \leftrightarrow (\forall I_p^- \in P^-)(VS(I^+, I_p^-) \neq \emptyset).$$

Theorem 21 *If Lc is an admissible concept language and property 2 holds, then:*

$$(VS(I^+, I^-) \neq \emptyset) \leftrightarrow (\forall I_p^+ \in P^+)(VS(I_p^+, I^-) \neq \emptyset).$$

The algorithm considered is described as follows. Assume that the version space $VS(I^+, I^-)$ is given by ABSs $\langle \{S(I_p^+, I^-)\}_{I_p^+ \in P^+}, \{G(I^+, I_p^-)\}_{I_p^- \in P^-} \rangle$. The algorithm checks whether $VS(I^+, I^-) = \emptyset$ by testing the sets $S(I_p^+, I^-)$ and $G(I^+, I_p^-)$. If at least one of these sets is empty, then by theorem 20 or 21 $VS(I^+, I^-) = \emptyset$. Hence, the algorithm returns true; otherwise, it returns false.

The second useful algorithm is the classification algorithm of ABSs. It implements the rule of unanimous voting [5, 6, 7]: an instance is classified if all the descriptions in a version space agree on an instance classification. The algorithm is based on theorems 22 and 23.

Theorem 22 $(\forall i \in I)((\forall c \in VS(I^+, I^-))M(c, i) \leftrightarrow VS(I^+, I^- \cup \{i\}) = \emptyset)$

Theorem 23 $(\forall i \in I)((\forall c \in VS(I^+, I^-))\neg M(c, i) \leftrightarrow VS(I^+ \cup \{i\}, I^-) = \emptyset).$

When an instance $i \in I$ has to be classified, the classification algorithm starts by forming the ABSs of the version space $VS(I^+, I^- \cup \{i\})$. If $VS(I^+, I^- \cup \{i\})$ is empty, by theorem 22 all the descriptions in $VS(I^+, I^-)$ cover the instance i . Hence, the instance i is positive w.r.t. $VS(I^+, I^-)$. If $VS(I^+, I^- \cup \{i\})$ is not empty, the algorithm forms the ABSs of the version space $VS(I^+ \cup \{i\}, I^-)$. If $VS(I^+ \cup \{i\}, I^-)$ is empty, by theorem 23 all the descriptions in $VS(I^+, I^-)$ do not cover the instance i . Hence, the instance i is negative w.r.t. $VS(I^+, I^-)$.

The classification algorithm uses the learning algorithm and the algorithm that checks whether version spaces are empty. Since the latter is applicable when one of properties 1 and 2 holds, the classification algorithm is used only for these cases.

7 Example of Adaptable Boundary Sets

Consider a concept-learning task s.t. the set I and the language Lc are 1-CNF languages with 8 Boolean attributes and $I \subseteq Lc$ ⁴. The training instances are given in the left column of table 1. They are chosen s.t. $|G(\{i_1^+\}, \{i_2^-, i_3^-, i_4^-, i_5^-\})| = 2^{|\{i_2^-, i_3^-, i_4^-, i_5^-\}|} = 16$; i.e., the size of the set is exponential in the number of negative instances.

To avoid this exponential dependence, the learning algorithm is tuned to form new special coverings $P^{+'}$ and $P^{-'}$ s.t. the size of ABSs is less or equal to the size of instance-based boundary sets. This requirement is imposed since the size of instance-based boundary sets is polynomial in the size of training data for 1-CNF languages with Boolean attributes [11, 12].

The ABSs of the target version space is initialised s.t. $S(\emptyset, \emptyset) = \{(\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)\}$ and $G(\emptyset, \emptyset) = \{(\?, \?, \?, \?, \?, \?, \?, \?)\}$. The trace of the learning algorithm is given in the right column of table 1. It is explained as follows. Before the instance i_4^- the ABSs are equal to boundary sets, since the sizes of boundary sets and instance-based boundary sets are equal in this case. When the instance i_4^- is

⁴Note that property 1 holds for 1-CNF languages with Boolean attributes.

| Training Instance | Algorithm Trace |
|---------------------------------|---|
| 1). $i_1^+ = (1,1,1,1,1,1,1,1)$ | $S(\{i_1^+\}, \emptyset) = \{(1, 1, 1, 1, 1, 1, 1, 1)\}$ $G(\{i_1^+\}, \emptyset) = \{(?, ?, ?, ?, ?, ?, ?, ?)\}$ |
| 2). $i_2^- = (0,0,1,1,1,1,1,1)$ | $S(\{i_1^+\}, \{i_2^-\}) = \{(1, 1, 1, 1, 1, 1, 1, 1)\}$ $G(\{i_1^+\}, \{i_2^-\}) = \{(1, ?, ?, ?, ?, ?, ?, ?), (? , 1, ?, ?, ?, ?, ?)\}$ |
| 3). $i_3^- = (1,1,0,0,1,1,1,1)$ | $S(\{i_1^+\}, \{i_2^-, i_3^-\}) = \{(1, 1, 1, 1, 1, 1, 1, 1)\}$ $G(\{i_1^+\}, \{i_2^-, i_3^-\}) = \{(1, ?, 1, ?, ?, ?, ?, ?), (1, ?, ?, 1, ?, ?, ?, ?),$ $(?, 1, 1, ?, ?, ?, ?, ?), (? , 1, ?, 1, ?, ?, ?, ?)\}$ |
| 4). $i_4^- = (1,1,1,1,0,0,1,1)$ | $S(\{i_1^+\}, \{i_2^-, i_3^-, i_4^-\}) = \{(1, 1, 1, 1, 1, 1, 1, 1)\}$ $G(\{i_1^+\}, \{i_2^-, i_3^-\}) = \{(1, ?, 1, ?, ?, ?, ?, ?), (1, ?, ?, 1, ?, ?, ?, ?),$ $(?, 1, 1, ?, ?, ?, ?, ?), (? , 1, ?, 1, ?, ?, ?, ?)\}$ $G(\{i_1^+\}, \{i_4^-\}) = \{(?, ?, ?, ?, 1, ?, ?, ?), (? , ?, ?, ?, 1, ?, ?)\}$ |
| 5). $i_5^- = (1,1,1,1,1,1,0,0)$ | $S(\{i_1^+\}, \{i_2^-, i_3^-, i_4^-, i_5^-\}) = \{(1, 1, 1, 1, 1, 1, 1, 1)\}$ $G(\{i_1^+\}, \{i_2^-, i_3^-\}) = \{(1, ?, 1, ?, ?, ?, ?, ?), (1, ?, ?, 1, ?, ?, ?, ?),$ $(?, 1, 1, ?, ?, ?, ?, ?), (? , 1, ?, 1, ?, ?, ?, ?)\}$ $G(\{i_1^+\}, \{i_4^-, i_5^-\}) = \{(?, ?, ?, ?, 1, ?, 1, ?), (? , ?, ?, ?, 1, ?, ? , 1),$ $(?, ?, ?, ?, 1, 1, ?), (? , ?, ?, ?, 1, ?, 1)\}$ |
| 6). $i_6^+ = (1,0,1,0,1,0,1,0)$ | $S(\{i_1^+, i_6^+\}, \{i_2^-, i_3^-, i_4^-, i_5^-\}) = \{(1, ?, 1, ?, 1, ?, 1, ?)\}$ $G(\{i_1^+, i_6^+\}, \{i_2^-, i_3^-\}) = \{(1, ?, 1, ?, ?, ?, ?)\}$ $G(\{i_1^+, i_6^+\}, \{i_4^-, i_5^-\}) = \{(?, ?, ?, ?, 1, ?, 1, ?)\}$ |

Table 1: Training Instances and the Trace of the Learning Algorithm.

processed the size of boundary sets becomes larger. Hence, the algorithm creates a new set $G(\{i_1^+\}, \{i_4^-\})$ and later updates it by the next instance i_5^- (see set $G(\{i_1^+\}, \{i_4^-, i_5^-\})$). Note that $|G(\{i_1^+\}, \{i_2^-, i_3^-\})| + |G(\{i_1^+\}, \{i_4^-, i_5^-\})| = 8$; i.e., the size of the G part of the ABSs is not exponential in the number of the negative instances. When the positive instance i_6^+ is processed, the sets $G(\{i_1^+, i_6^+\}, \{i_2^-, i_3^-\})$ and $G(\{i_1^+, i_6^+\}, \{i_4^-, i_5^-\})$ are pruned to contain only one element.

The learning algorithm forms the S -part of the ABSs consisting of the set $S(\{i_1^+, i_6^+\}, \{i_2^-, i_3^-, i_4^-, i_5^-\})$ only, since the size of the set is equal to one [11].

The resulting ABSs are processed by the specialised merging algorithm, since G sets contain just one element. Thus, the final ABSs are given by $S(\{i_1^+, i_6^+\}, \{i_2^-, i_3^-, i_4^-, i_5^-\}) = \{(1, ?, 1, ?, 1, ?, 1, ?)\}$ and $G(\{i_1^+, i_6^+\}, \{i_2^-, i_3^-, i_4^-, i_5^-\}) = \{(1, ?, 1, ?, 1, ?, 1, ?)\}$.

8 Conclusions

In this paper we presented ABSs as a new version-space representation. We showed that ABSs can be adjusted to each version-space representation between boundary sets and instance-based boundary sets. Hence, the choice of version-space representations can be done dynamically during the learning phase.

We consider ABSs as a first attempt unifying version-space representations ⁵. This allows different representations to benefit from each other. For example:

- if the size of boundary sets becomes exponential in the number of training instances, they can be considered as adaptable and learning can continue by the learning algorithm from section 4;
- if the comprehensibility of instance-based boundary sets is not sufficient, they can be considered as adaptable and learning can continue after their optimisation by one of the merging algorithms from section 5.

Future research will focus on unifying other existing version-space representations (cf. [2, 3, 4, 8, 15]). This will allow a development of a complete practical theory of version-space representations.

⁵Note that boundary sets and instance-based boundary sets are special cases of ABSs.

References

- [1] D. Haussler. Quantifying inductive bias: AI learning algorithms and valiant's learning framework. *Artificial Intelligence*, 36(2):177–221, 1988.
- [2] H. Hirsh. Polynomial-time learning with version spaces. In *Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92)*, pages 117–122, Menlo Park, CA, 1992. AAAI Press.
- [3] H. Hirsh, N. Mishra, and L. Pitt. Version spaces without boundary sets. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 491–496, Menlo Park, CA, 1997. AAAI Press.
- [4] P. Idemstam-Almquist. Demand networks: an alternative representation of version spaces. Master's thesis, Department of Computer Science and Systems Sciences, The Royal Institute of Technology and Stockholm University, Stockholm, Sweden, 1990.
- [5] T.M. Mitchell. *Version spaces: an approach to concept learning*. PhD thesis, Electrical Engineering Dept., Stanford University, Stanford, CA, 1978.
- [6] T.M. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2):203–226, 1982.
- [7] T.M. Mitchell. *Machine learning*. McGraw-Hill, New York, NY, 1997.
- [8] G. Sablon. *Iterative versionspaces with an application in inductive logic programming*. PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium, 1995.
- [9] M. Sebag and C. Rouveirol. Tractable induction and classification in first order logic via stochastic matching. In *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence (IJCAI-97)*, pages 888–893, San Francisco, CA, 1997. Morgan Kaufmann.
- [10] M. Sebag and C. Rouveirol. Resource-bounded relational reasoning: induction and deduction through stochastic matching. *Machine Learning*, 38(1-2):41–62, 2000.
- [11] E.N. Smirnov. *Conjunctive and disjunctive version spaces with instance-based boundary sets*. PhD thesis, Department of Computer Science, Maastricht Univeristy, Maastricht, The Netherlands, 2001.
- [12] E.N. Smirnov and P.J. Braspenning. Version space learning with instance-based boundary sets. In *Proceedings of the Thirteenth European Conference on Artificial Intelligence (ECAI-98)*, pages 460–464, Chichester, UK, (The Best Paper Prize), 1998. John Wiley and Sons.
- [13] E.N. Smirnov and H.J. van den Herik. Applying preference biases to conjunctive and disjunctive version spaces. In *Proceedings of the Ninth International Conference on Artificial Intelligence: Methodology, Systems, and Applications (AIMSA-2000)*, LNAI 1904, pages 321–330, Berlin, Germany, 2000. Springer.
- [14] E.N. Smirnov, H.J. van den Herik, and I.G. Sprinkhuizen-Kuyper. Generalising boundary sets. Technical report, CS 01-08, Department of Computer Science, Maastricht University, Maastricht, The Netherlands, 2001. <http://www.cs.unimaas.nl/~Kuyper/papers/TRCS0108ps.gz>.
- [15] B.D. Smith and P.S. Rosenbloom. Incremental non-backtracking focusing: a polynomially bounded generalization algorithm for version spaces. In *Proceedings of the Eight National Conference on Artificial Intelligence (AAAI-90)*, pages 848–853. MIT Press, 1990.