Hypotheses and Version Spaces

Bernhard Ganter[1] and Sergei O. Kuznetsov[2]

[1] Technische Universität Dresden
[2] VINITI Moscow

Abstract. We consider the relation of a learning model described in terms of Formal Concept Analysis in [3] with a standard model of Machine Learning called version spaces. A version space consists of all possible functions compatible with training data. The overlap and distinctions of these two models are discussed. We give an algorithm how to generate a version space for which the classifiers are closed under conjunction.
As an application we discuss an example from predicitive toxicology. The classifiers here are chemical compounds and their substructures. The example also indicates how the methodology can be applied to Conceptual Graphs.

1 Hypotheses and Implications

Our paper deals with a standard model of Machine Learning called version spaces. But first, to make the paper self-contained, we introduce basic definitions of Formal Concept Analysis (FCA) [5]. We consider a set M of "structural attributes", a set G of objects (or observations) and a relation $I \subseteq G \times M$ such that $(g, m) \in I$ if and only if object g has the attribute m. Instead of $(g, m) \in I$ we also write gIm. Such a triple $\mathbb{K} := (G, M, I)$ is called a *formal context*. Using the *derivation operators*, defined for $A \subseteq G$, $B \subseteq M$ by

$$A' := \{m \in M \mid gIm \text{ for all } g \in A\},$$
$$B' := \{g \in G \mid gIm \text{ for all } m \in B\},$$

we can define a *formal concept* (of the context $\mathbb{K}$) to be a pair (A, B) satisfying $A \subseteq G$, $B \subseteq M$, $A' = B$, and $B' = A$. A is called the *extent* and B is called the *intent* of the concept (A, B). These concepts, ordered by

$$(A_1, B_1) \geq (A_2, B_2) \iff A_1 \supseteq A_2$$

form a complete lattice, called *the concept lattice* of $\mathbb{K} := (G, M, I)$.

Next, we use the FCA setting to describe a learning model from [2]. In addition to the structural attributes of M, we consider (as in [3]) a *goal attribute* $w \notin M$. This divides the set G of all objects into three subsets: The set G_+ of those objects that are known to have the property w (these are the *positive examples*), the set G_- of those objects of which it is known that they do not have w (the *negative examples*) and the set G_τ of *undetermined examples*, i.e.,

A. de Moor, W. Lex, and B. Ganter (Eds.): ICCS 2003, LNAI 2746, pp. 83–95, 2003.

of those objects, of which it is unknown if they have property w or not. This gives three subcontexts of $\mathbb{K} = (G, M, I)$:

$$\mathbb{K}_+ := (G_+, M, I_+), \quad \mathbb{K}_- := (G_-, M, I_-), \quad \text{and } \mathbb{K}_\tau := (G_\tau, M, I_\tau),$$

where for $\varepsilon \in \{+, -, \tau\}$ we have $I_\varepsilon := I \cap (G_\varepsilon \times M)$.

Intents, as defined above, are attribute sets shared by some of the observed objects. In order to form hypotheses about structural causes of the goal attribute w, we are interested in sets of structural attributes that are common to some positive, but to no negative examples. Thus, a *positive hypothesis* for w is an intent of $\mathbb{K}_+$ that is not contained in the intent

$$g^- := \{m \in M \mid (g, m) \in I_-\}$$

of any negative example $g \in G_-$. *Negative hypotheses* are defined accordingly.

These hypotheses can be used as predictors for the undetermined examples. If the intent

$$g' := \{m \in M \mid (g, m) \in I\}$$

of an object $g \in G_\tau$ contains a positive, but no negative hypothesis, then g is *classified positively*. Negative classifications are defined similarly. If g' contains hypotheses of both kinds, or if g' contains no hypothesis at all, then the classification is *contradictory* or *undetermined*, respectively.

In [11] we argued that the consideration can be restricted to *minimal* (w.r.t. inclusion $\subseteq$) hypotheses, positive as well as negative, since an object intent obviously contains a positive hypothesis if and only if it contains a minimal positive hypothesis, etc.

2 Version Spaces

The term "version space" was proposed in [12], [13] and became the name of a certain class of Machine Learning models [14]. Version spaces can be defined in different ways, e.g., Mitchell defined them in terms of sets of maximal and minimal elements [12], [13]. In [20] they are defined in terms of minimal elements and sets of negative examples. They can also be defined in terms of some matching predicate. These representations are equivalent, however transformations from one into another are not always polynomially tractable. We will use the representation with matching predicates.

The next lines describe the basic notions of version spaces. We follow [12], [20], but with slightly modified terms, in order to avoid confusion when merging this approache with that of Formal Concept Analysis. We need

- An *example language* L_e by means of which the instances are described (elsewhere also called *instance language*). For our purposes it is not relevant to discuss how such a language is defined in detail. It suffices to assume that it describes a *set E of examples*.

- A *classifier language* L_c describing the possible classifiers (elsewhere called *concepts*). Again, for us it is not necessary to discuss a precise definition of L_c. We assume that it describes a set C of *classifiers* and a (partial) order $\sqsubseteq$ on C, called the *subsumption order*.
- A *matching predicate* $M(c, e)$ that defines if a classifier c does or does not *match* an example e: We have $M(c, e)$ iff e is an example of classifier c. The matching predicate must be compatible with the subsumption order in the following sense: for $c_1, c_2 \in L_c$,

$$c_1 \sqsupseteq c_2 \Rightarrow \forall_{e \in E}\, M(c_1, e) \rightarrow M(c_2, e).$$

- Sets E_+ and E_- of *positive* and *negative examples* of a *goal attribute* with $E_+ \cap E_- = \emptyset$. The goal attribute is not explicitly given.

On the basis of these data one defines a

- *consistency predicate* $\mathrm{cons}(c)$:
  $\mathrm{cons}(c)$ holds if for every $e \in E_+$ the matching predicate $M(c, e)$ holds and for every $e \in E_-$ the negation $\neg M(c, e)$ holds. The set of all consistent classifiers is called the *version space*

$$\mathrm{VS}(L_c, L_e, M(c, e), E_+, E_-).$$

The learning problem is then defined as follows:

Given $L_c, L_e, M(c, e), E_+, E_-$.
Find the version space $\mathrm{VS}(L_c, L_e, M(c, e), E_+, E_-)$.

In the sequel, we shall usually fix L_c, L_e, and $M(c, e)$, but work with varying sets of positive and negative examples. We therefore sometimes write

$$\mathrm{VS}(E_+, E_-) \quad \text{or even just} \quad \mathrm{VS}$$

for short.

Version spaces are often considered in terms of their *boundary sets*, as proposed in [13]. These can be defined if the language L_c is *admissible*, i.e., if every chain in the subsumption order has a minimal and a maximal element. In this case,

$$G(\mathrm{VS}) := \mathrm{MAX}(\mathrm{VS}) := \{c \in \mathrm{VS} \mid \neg\exists_{c_1 \in \mathrm{VS}}\, c \leq c_1\},$$
$$S(\mathrm{VS}) := \mathrm{MIN}(\mathrm{VS}) := \{c \in \mathrm{VS} \mid \neg\exists_{c_1 \in \mathrm{VS}}\, c_1 \leq c\}.$$

According to [14], the ideal result of learning a goal attribute is the case where the version space consists of a single element. Otherwise, the goal attribute is said to be *learned partially*.

It was said above that the goal attribute is not given explicitely. The elements of the version space can be used as potential classifiers for the goal attribute: A classifier $c \in \mathrm{VS}$ *classifies* an example e positively if c matches e and negatively else. Then, all positive examples are classified positively, all negative examples

are classified negatively, and undetermined examples may be classified either way. If it is assumed that E_+ and E_- carry sufficient information about the goal attribute, we may expect that an undetermined example is likely to have the goal attribute if it is classified positively by a large precentage of the version space (cf. [14]). We say that example e is *α-classified* if no less than $\alpha \cdot |\,\mathrm{VS}\,|$ classifiers classify it positively.

3 Version Spaces in Terms of Galois Connections

We show that basic notions for version spaces can smoothly be expressed in the terminology of Formal Concept Analysis, more precisely, using the derivation operators. Let us remark that these form a Galois connection [5].

Consider the formal context (E, C, I), where E is the set of examples containing the disjoint sets of observed positive and negative examples: $E \supseteq E_+ \cup E_-$, $E_+ \cap E_- = \emptyset$, C is the set of classifiers and relation I corresponds to the matching predicate $M(c, e)$: for $c \in C$, $e \in E$ the relation eIc holds iff $M(c, e) =$ TRUE. The complementary relation, $\bar{I}$, corresponds to the negation: $e\bar{I}c$ holds iff $M(c, e) =$ FALSE.

Proposition 1.

$$\mathrm{VS}(E_+, E_-) = {E_+}^I \cap {E_-}^{\bar{I}}.$$

Proof. The set of classifiers that match all positive examples is ${E_+}^I$. Since $\bar{I} = (E \times C) \setminus I$, the relation $\bar{I}$ corresponds to the predicate "classifier c does not match example e" and the set of classifiers that do not match any negative examples is ${E_-}^{\bar{I}}$. The version space is by definition the set of classifiers that match all positive examples and do not match any negative examples, i.e., is the intersection ${E_+}^I \cap {E_-}^{\bar{I}}$. □

From this Proposition we immediately get a result from [7] about *merging version spaces*:

Corollary 1. *For fixed L_c, L_e, $M(c, e)$ and two sets E_{+1}, E_{-1} and E_{+2}, E_{-2} of positive and negative examples one has*

$$\mathrm{VS}(E_{+1} \cup E_{+2}, E_{-1} \cup E_{-2}) = \mathrm{VS}(E_{+1}, E_{-1}) \cap \mathrm{VS}(E_{+2}, E_{-2}).$$

Proof. This follows from Proposition 1 and the relation $(A \cup B)' = A' \cap B'$, which holds for an arbitrary derivation operator $(\cdot)'$. □

Proposition 2. *The set of all 100%-classified examples defined by the version space* $\mathrm{VS}(E_+, E_-)$ *is given by*

$$({E_+}^I \cap {E_-}^{\bar{I}})^I.$$

Proof. By the previous proposition, the version space given by the set of positive examples E_+ and the set of negative examples E_- is $E_+^I \cap E_-^{\bar{I}}$, therefore the set of all examples that are classified positively by all elements of the version space is $({E_+}^I \cap {E_-}^{\bar{I}})^I$. □

The next proposition explains what it means when the set E_+ of all positive examples is closed with respect to $(\cdot)^{II}$.

Proposition 3. *If* ${E_+}^{II} = E_+$*, then there cannot be any 100%-classified undetermined example, no matter what* E_- *is.*

Proof. By the previous propositions, ${E_+}^I$ is the set of all classifiers that match positive examples and ${E_+}^{II}$ is the set of all examples matched by these classifiers. If ${E_+}^{II} = E_+$, then, independent of any possible examples, these classifers do not match any "new" examples (i.e., those outside E_+). □

Proposition 4. *The set of examples that are classified positively by at least one element of the version space* $\mathrm{VS}(E_+, E_-)$ *is given by*

$$E \setminus ({E_+}^I \cap {E_-}^{\bar{I}})^{\bar{I}}.$$

Proof. By Proposition 1, $({E_+}^I \cap {E_-}^{\bar{I}})$ is the version space given by examples E_+ and E_-. The set of all examples that are not matched by any classifier from this version space is $({E_+}^I \cap {E_-}^{\bar{I}})^{\bar{I}}$. Therefore, the set of examples that are matched by at least one classifier is its complement w.r.t. the set of all possible examples, i.e., $E \setminus ({E_+}^I \cap {E_-}^{\bar{I}})^{\bar{I}}$. □

4 Classifier Semilattices

In the preceding section we have described version spaces in the language of Formal Concept Analysis. But for many examples this description is too simple, because it does not assume any additional structure for the set of classifiers. We have mentioned that for version spaces the classifiers are given in terms of some language L_c, and it is therefore natural to consider the case when the ordered set $(C, \leq)$ of classifiers is a meet-semilattice. Let $\sqcap$ denote the meet operation of this semilattice.

This is not an inept assumption. Classifiers given as logical formulae form a meet semilattice when the set of these formulae is closed under conjunction. Classifiers given as attributes show the same effect if arbitrary attribute combinations are allowed as classifiers, too. This also covers the case of attributes with values. In the setting of [14], for example, each attribute takes one of possible values, either constant or "wildcard" $*$, the latter being the shortcut for universal quantification over constant values of this attribute. Examples are given by

conjunctions of attribute values. A classifier c matches example e if all attribute values of c that do not coincide with the corresponding values of e are wildcards.

Formal Concept Analysis offers *scaling* techniques [5] that reduce the case of many-valued attributes to the case of plain "one-valued" ones. We then may assume that each classifier c corresponds to a subset S_c of a fixed set of attributes, and c matches an example e if and only if e has each of the attributes in S_c. The subsumption order then corresponds to the set inclusion:

$$c_1 \sqsubseteq c_2 \iff S_{c_1} \subseteq S_{c_2}.$$

If $\sqcap$ makes a complete semilattice (e.g., when the semilattice is finite), then the dual join operation $\sqcup$ can be (uniquely) defined. This operation on classifiers corresponds to the intersection of attribute sets: $c_1 \sqcup c_2$ matches exactly those examples that are matched by both c_1 and c_2.

Proposition 5. *If the classifiers, ordered by subsumption, form a complete* $\sqcup$*-semi"-lat"-tice, then the version space is a complete subsemilattice for any sets of examples* E_+ *and* E_-*.*

Proof. Recall that a classifier belongs to the version space iff it matches all positive and no negative examples. If c_1 and c_2 satisfy this condition, then $c_1 \sqcup c_2$, being more specific than c_1 and c_2, does not match any negative examples. On the other hand, the join $c_1 \sqcup c_2$ covers the intersection of sets of examples covered by c_1 and c_2. Since all positive examples are covered by both c_1 and c_2, their join $c_1 \sqcup c_2$ covers all positive examples too. □

In [4] we have introduced a variant of Formal Concept Analysis based on so-called pattern structures. Here we use a slightly restricted form of them. First we assume that the set of all possible classifiers forms a complete semilattice $(C, \sqcap)$, which allows us to define a dual join operation $\sqcup$. A *pattern structure* is a tuple $(E, (C, \sqcap), \delta)$, where E is a set (of "examples"), δ is a mapping

$$\delta : E \to C,$$

$\delta(E) := \{\delta(e) \mid e \in E\}$. The subsumption order can be reconstructed from the semilattice operation:

$$c \sqsubseteq d \iff c \sqcap d = c.$$

For such a pattern structure $(E, (C, \sqcap), \delta)$ we can define derivation operators as we did for formal contexts[1] by

$$A^\diamond := \sqcap_{e \in A} \delta(e) \qquad \text{for } A \subseteq E$$

and

$$c^\diamond := \{e \in E \mid c \sqsubseteq \delta(e)\} \qquad \text{for } c \in C.$$

The pairs (A, c) satisfying

$$A \subseteq E,\ c \in C,\ A^\diamond = c,\ c^\diamond = A$$

[1] In fact, pattern structures can be represented as formal contexts.

are called the *pattern concepts* of $(E, (C, \sqcap), \delta)$, with extent A and *pattern intent* c. The pattern concepts form a complete lattice.

Not all classifieres are pattern intents, and it may be the case that many classifiers describe the same extent A. Only one of them, namely $A^\diamond$, then is a pattern intent. Let us call two classifiers c_1 and c_2 equivalent if $c_1^\diamond = c_2^\diamond$. We then observe that for every classifier c the equivalence class of c contains a unique pattern intent, $c^{\diamond\diamond}$. This is the *representing* pattern intent for c.

If, as above, E_+ and E_- are disjoint subsets of E representing positive and negative examples for some goal attribute, we can define a *positive hypothesis* h as a pattern intent of $(E_+, (C, \sqcap), \delta)$ that is not subsumed by any classifier from $\delta(E_-)$ (for short: not subsumed by any negative example). Formally, $h \in C$ is a positive hypothesis iff

$$h^\diamond \cap E_- = \emptyset \quad \text{and } \exists_{A \subseteq E_+} A^\diamond = h.$$

A disappointing situation is when the version space is empty, i.e., when there are no classifiers at all separating all positive examples from all negative ones. This happens, for example, if there are *hopeless* positive examples, by which we mean elements $e_+ \in E_+$ that have a negative counterpart $e_- \in E_-$ such that every classifier which matches e_+ also matches e_-. An equivalent formulation of the hopelessness of e_+ is that $(e_+)^{\diamond\diamond} \cap E_- \neq \emptyset$.

Theorem 1. *Suppose that the classifiers, ordered by subsumption, form a complete meet-semilattice $(C, \sqcap)$, and let $(E, (C, \sqcap), \delta)$ denote the corresponding pattern structure. Then the following are equivalent:*

1. *The version space* $\mathrm{VS}(E_+, E_-)$ *is not empty.*
2. $(E_+)^{\diamond\diamond} \cap E_- = \emptyset$.
3. *There are no hopeless positive examples and there is a unique minimal positive hypothesis* h_{min}.

In this case, $h_{min} = (E_+)^\diamond$ and the version space is represented by a convex set in the lattice of all pattern intents[2] *with maximal element h_{min}.*

Proof. The version space consists of precisely those classifiers c that satisfy $E_+ \subseteq c^\diamond$ and $c^\diamond \cap E_- = \emptyset$. The join m of all such classifiers has the same property; it therefore is the maximal element of the version space and the only element of $S(\mathrm{VS})$. From $E_+ \subseteq m^\diamond$ we get that $(E_+)^{\diamond\diamond} \subseteq m^\diamond$ and therefore $(E_+)^{\diamond\diamond} \cap E_- = \emptyset$.

Suppose $(E_+)^{\diamond\diamond} \cap E_- = \emptyset$, then $h := (E_+)^\diamond$ is a positive hypothesis subsumed by every positive hypothesis $A^\diamond, A \subseteq E_+$. If $e_+ \in E_+$ the $(e_+)^{\diamond\diamond} \subseteq (E_+)^{\diamond\diamond}$. Thus, if $(E_+)^{\diamond\diamond} \cap E_- = \emptyset$, then there cannot be any hopeless positive examples.

If there is a unique minimal hypothesis, $h_{\min}$, then $E_+ \subseteq A := h_{\min}^\diamond$. Suppose not, then there is some positive example $e \in E_+$, $e \notin A$. Since e is not hopeless, $e^\diamond$ is a hypothesis, and this hypothesis is incomparable to $h_{\min}$, a contradiction. Thus $h_{\min}$ is the maximal element of the version space. □

[2] ordered by subsumption

The theorem gives access to an algorithm for generating the version space. To be more precise: our algorithm will not generate the version space, but the set of representing pattern intents. A well-known algorithm for generating all formal concepts of a formal context can be modified to only generate a convex set of the type described in Theorem 1. This can easily be described here, but without proof. For the latter we refer to [5].

Fix a linear order $\leq$ on the set G of all examples in such a way that all positive examples come first, then all negative ones, and the other elements come last. Let $n_{\max}$ denote the maximal negative example in this order. Also with respect to this order, we reformulate two elementary notations from [5]:

– For $A, B \subseteq G$ and $i \in G$ define

$$A <_i B :\iff i \in B, i \notin A, \text{ and } (j \in A \iff j \in B \text{ for all } j < i).$$

– For $A \subseteq G$ and $i \in G$, $i \notin A$, define

$$A \oplus i := (\{i\} \cup \{j \in A \mid j < i\})^{\diamond\diamond}.$$

Now the pattern intents representing the version space are recursively given as follows:

1. The first element is $h_{\min} = (E_+)^{\diamond}$, provided that $(E_+)^{\diamond\diamond} \cap E_- = \emptyset$. Otherwise, the version space is empty.
2. If an element h of the version space has been computed, then the "next" element h_{next} is computed as follows: Let $A := h^{\diamond}$, and let i be the largest element that is greater than $n_{\max}$ and that satisfies

$$A <_i A \oplus i.$$

Then $h_{\text{next}} = (A \oplus i)^{\diamond}$. If no such i exists, the algorithm terminates.

Corollary 2. *If the classifiers, ordered by subsumption, form a finite meet-semi"-lat"-tice, then a system of representatives for the version space can be computed using the algorithm described above.*

5 Boundary Sets, Hypotheses, and Predictors

According to [5] a subset $A \subseteq M$ is a *proper premise of an attribute* $m \in M$ if $m \notin A$, $m \in A''$ and for any $A_1 \subset A$ one has $m \notin A_1''$. It would be useful to know the *proper premises of the goal attribute* w. But as long as there are undecided examples, i.e., as long as the set E_τ is not empty, our information does not suffice to determine these. We therefore generalize the notion to describe good candidates for such premises:

Definition 1 We call $d \in L_c$ a *positive proper predictor* with respect to example sets E_+, E_- if the following conditions are satisfied:

1. $d^\diamond \cap E_- = \emptyset$,
2. $d^\circ \cap E_+ \neq \emptyset$,
3. if $d \sqsupseteq d_1$ and $d \neq d_1$ then $d_1^\diamond \cap E_- \neq \emptyset$.

Thus, a positive proper predictor matches no negative example (due to Condition 1), at least one positive example (Condition 2), and is a most general classifier with these properties (Condition 3). The set of all such predictors for a pattern structure $\Pi = (E, (C, \sqcap), \delta)$ and example sets E_+ and E_- is denoted by $\mathrm{PP}_+(\Pi, E_+, E_-)$.

Our next proposition describes the interplay between the set of predictors of a pattern structure $\Pi := (E, (C, \sqcap), \delta)$ with example sets E_+ and E_-, the set $\mathrm{H}_+(\Pi, E_+, E_-)$ of its positive hypotheses and the boundary sets $G(\mathrm{VS})$ and $S(\mathrm{VS})$ of the corresponding version space. For a set $X \subseteq C$ of classifiers let $\min_\sqsubseteq(X)$ denote the minimal elements:

$$\min_\sqsubseteq(X) := \{x \in X \mid y \not\sqsubseteq x \text{ for all } y \neq x \text{ in } X\}.$$

Proposition 6. *Let $\Pi = (E, (C, \sqcap), \delta)$ be a pattern structure and $E = E_+ \,\dot\cup\, E_\tau \,\dot\cup\, E_-$. Then*

1. $\mathrm{PP}_+(\Pi, E_+, E_-) = \min_\sqsubseteq \left(\bigcup_{F_+ \subseteq E_+} G(\mathrm{VS}(\Pi, F_+, E_-))\right)$,
2. $\mathrm{H}_+(\Pi, E_+, E_-) = \bigcup_{F \subseteq E_+} S(\mathrm{VS}(\Pi, F_+, E_-))$.

Proof. 1. Consider a positive proper predictor p w.r.t. examples E_+ and E_-. Let $F_+ := \{g \in E_+ \mid p \sqsubseteq g^\diamond\}$. By Condition 3 of Definition 1, no element of the version space $\mathrm{VS}(F_+, E_-)$ can be more general than p. Hence, by the definition of $G(\mathrm{VS})$, we have $p \in G(\mathrm{VS}(\Pi, F_+, E_-))$,

$$PP_+(\Pi, E_+, E_-) \subseteq \bigcup_{F_+ \subseteq E_+} G(\mathrm{VS}(\Pi, F_+, E_-)).$$

Moreover, p should be among the most general elements of

$$\bigcup_{F_+ \subseteq E_+} G(\mathrm{VS}(\Pi, F_+, E_-)),$$

since otherwise, there should have existed a positive classifier more general than p and p could not have been a positive proper predictor w.r.t. E_+, E_-.

In the other direction, let p be a maximal element of the set

$$\bigcup_{F_+ \subseteq E_+} G(\mathrm{VS}(\Pi, F_+, E_-)).$$

Then, by the definition of a version space, for some $F_+ \subseteq E_+$, we have $F_+ \subseteq p^\diamond$, $E_- \not\subseteq p^\diamond$, and hence $p^\diamond \subseteq E_+ \cup E_\tau$, thus Condition 1 of Definition 1 is satisfied. Condition 2 is also satisfied, since for all $g \in F_+ \subseteq E_+$ one has $g \in p^\diamond$. Condition 3 is satisfied, since otherwise, for some F_+ in the version space $VS(\Pi, F_+, E_-)$

there had existed a classifier $p_1 \sqsupseteq p$, which contradicts the assumption that p is maximal w.r.t. subsumption $\sqsubseteq$ in $\bigcup_{F_+ \subseteq E_+} G(\mathrm{VS}(\Pi, F_+, E_-))$. Therefore, p is a positive proper predictor w.r.t. examples E_+, E_-, and E_τ.
2. For each $F_+ \subseteq E_+$, if $\mathrm{VS}(\Pi, F_+, E_-)$ is not empty then, by Theorem 1, the unique element of the set $S(\mathrm{VS}(\Pi, F_+, E_-))$ is a minimal hypothesis for the pattern structure Π and sets of examples F_+, E_-. Thus,

$$H_+(\Pi, E_+, E_-) \supseteq \bigcup_{F \subseteq E_+} S(\mathrm{VS}(\Pi, F_+, E_-)).$$

In the other direction, each positive hypothesis H w.r.t. (Π, E_+, E_-) is a minimal positive hypothesis w.r.t. (Π, F_+, E_-) for $F_+ = H^\diamond \cap E_+$. Hence, H is a unique element of $S(\mathrm{VS}(\Pi, F_+, E_-))$ and

$$H_+(\Pi, E_+, E_-) \subseteq \bigcup_{F \subseteq E_+} S(\mathrm{VS}(\Pi, F_+, E_-)).$$

□

6 An Example

The 12th European Conference on Machine Learning was held jointly with the 5th European Conference on Principles of Knowledge Discovery in Databases. The program included a workshop on Predictive Toxicology Challenge [21], which consisted in a competition of machine learning programs for generation of hypothetical causes of toxicity from positive and negative examples of toxicity. The learning program presented in [1] turned out to be Pareto-optimal among all classification rules generated by learning models participating in the competition in terms of the relative number of false and true positive and negative classifications made by hypotheses generated by a learning program (see [21] for details). Their learning program is based on the JSM-method and generates minimal hypotheses along the lines of definitions in Section 1. It can be viewed as an instance for the methods of this article.

The "examples" in the toxicology challenge were chemical compounds; the goal attribute was toxicity. In the original representation the goal attribute was not binary, however obviously positive examples, i.e., those known to be certainly toxic and negative examples, i.e., those known to be certainly nontoxic, were isolated. In the toxocology challenge we used a standard representation by formal contexts. If we want to represent the chemical substances more properly, we can associate to each chemical compound $g \in G$ as a corresponding pattern $\delta(g)$ its molecular graph. As classifiers we use these graphs and their subgraphs and, more generally, subgraph combinations. The use of subgraph combinations leads to a semilattice of classifiers, as treated in Section 4, which is a subsemilattice of the lattice of order ideals of the ordered (w.r.t. subgraph isomorphism relation, see below) set of labeled graphs. This approach has already been used in [4].

It has to be specified precisely what we mean by a subgraph. A useful definition corresponds to that of the injective specialization relation [16] or injective morphism [15] used for Conceptual Graphs.

Let P be the set of all finite graphs with vertex and edge labels from L, up to isomorphism. A typical such graph is of the form $\Gamma := ((V, l_v), (E, l_e))$, with vertex set V, edge set E, and label assignments l_v, l_e for vertices and edges, respectively. We say that

$$\Gamma_2 := ((V_2, l_{v2}), (E_2, l_{e2}))$$

is **isomorphic to a subgraph** of

$$\Gamma_1 := ((V_1, l_{v1}), (E_1, l_{e1})),$$

or for short, Γ_2 is a **subgraph** of Γ_1, denoted by $\Gamma_2 \leq \Gamma_1$, if there exists a one-to-one mapping $\varphi : V_2 \to V_1$ that (for all $v, w \in V_2$)

- respects edges: $(v, w) \in E_2 \Rightarrow (\varphi(v), \varphi(w)) \in E_1$,
- fits under labels: $l_{v2}(v) = l_{v1}(\varphi(v))$, $l_{e2}(v, w) = l_{e1}(\varphi(v), \varphi(w))$.

Obviously, $\leq$ is an order relation. A more general definition, which takes into account ordering of labels (a label can be less or more "general" than another one) can be found in [9], [10], [4].

Having defined $\leq$ we define a lattice on graph sets along the lines of [9]. In terms of lattice theory, FCA, and pattern structures, this can be described as follows. First, the lattice of order ideals $LI(P, \leq)$ of the order $\leq$ is given by Birkhoff's theorem (see [5], Theorem 39). This gives us infimum and supremum operations $\sqcap$ and $\sqcup$. Then we define a pattern structure $(G, (LI(P, \leq), \sqcup), \delta)$, where δ takes each object $g \in G$ to an element of the lattice of order ideals on graphs described above. Using this pattern structure we generate hypotheses as described in Section 4 above.

The vertex labels in case of molecular graphs corresponding to atom types (e.g., C, N, Cl staying for carbon, nytrogene, chlorine, respectively) and edges are labeled by bond types (single bond, double bond, triple bond, aromatic bond, etc.). An aromatic bond in cyclic components of a molecular graph is usually denoted by a circle inside a cycle.

Consider the disconnected graph in Figure 1. It turns out that this is a minimal hypothesis for toxicity. The connected component to the right belongs to the version space, whereas the connected component to the left is not a premise of the goal attribute at all, since it is a subgraph of some graphs representing negative examples.

7 Discussion and Relation to Other Work

The major drawback of the version spaces, where classifiers (also called "concepts" in some publications on version spaces) are defined syntactically, is the very likely situation when - in case of too restrictive choice of the classifiers -

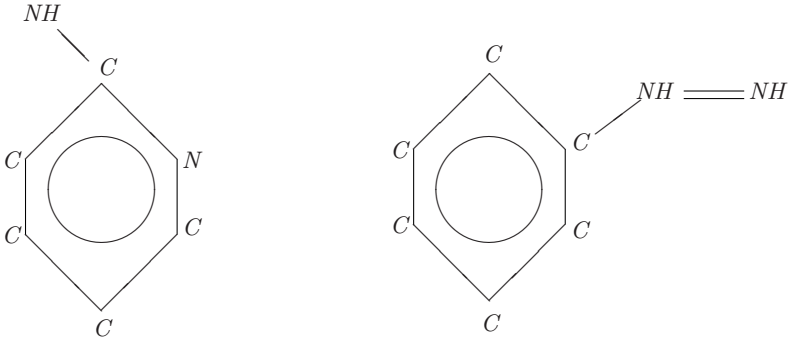

Fig. 1. Two connected components of a hypothesis

there is no classifier that matches all positive examples (so-called "collapse of the version space"). This can easily happen for example when classifiers are just conjunctions of attribute value assignments and "wildcards", a case mentioned above. In other words: The situation discussed in Theorem 1, which presupposes that there are classifiers that match all positive and no negative examples, is too narrow.

If the expressive power is increased syntactically, e.g., by introducing disjunction, then the version space tends to become trivial, while the most specific generalization of positive examples becomes "closer" to or just coincide with the set of positive examples.

As a remedy, we suggest to use sets of hypotheses as we defined them in terms of patterns structures. They offer in fact a sort of "context-restricted" disjunction: not all disjunctions are possible, but only those of minimal hypotheses.

Clearly, the relation of our work with that in Inductive Logic Programming ILP should be realized. In ILP (see, e.g., [17]) the definition of a version space is specified by taking the $\leq$ relation to be the inference relation in logic and the set of classifiers as a subset of the set of logical programming formulas.

As the origin of the $\leq$ order can be arbitrary, one can use the above constructions for learning of "generalized" descriptions from descriptions of positive and negative examples given in a description logic, by conceptual graphs, etc. One should just specify an initial "more general than or equal to" relation $\leq$.

References

1. V.G. Blinova, D.A. Dobrynin, V.K. Finn, S.O. Kuznetsov, and E.S. Pankratova, Toxicology Analysis by Means of the JSM-Method, *Bioinformatics*, 2003.
2. V. K. Finn, Plausible Reasoning in Systems of JSM Type, *Itogi Nauki i Tekhniki, Seriya Informatika*, **15**, 54-101, 1991 [in Russian].
3. B. Ganter and S. Kuznetsov, Formalizing Hypotheses with Concepts, *Proc. 8th Int. Conf. on Conceptual Structures, ICCS'00*, G. Mineau and B. Ganter, Eds., Lecture Notes in Artificial Intelligence, **1867**, 2000, pp. 342-356.

4. B. Ganter and S.O. Kuznetsov, Pattern Structures and Their Projections, *Proc. 9th Int. Conf. on Conceptual Structures, ICCS'01*, G. Stumme and H. Delugach, Eds., Lecture Notes in Artificial Intelligence, **2120**, 2001, pp. 129-142.
5. B. Ganter and R. Wille, *Formal Concept Analysis: Mathematical Foundations*, Springer, 1999.
6. C.A. Gunter, T.-H. Ngair, D. Subramanian, The Common Order-Theoretic Structure of Version Spaces and ATMSs, *Artificial Intelligence* **95**, 357-407, 1997.
7. H. Hirsh, Generalizing Version Spaces, *Machine Learning* **17**, 5-46, 1994.
8. H. Hirsh, N. Mishra, and L. Pitt, Version Spaces Without Boundary Sets, in *Proc. of the 14th National Conference on Artificial Intelligence (AAAI97)*, AAAI Press/MIT Press, 1997.
9. S.O. Kuznetsov, JSM-method as a machine learning method, *Itogi Nauki i Tekhniki, ser. Informatika*, **15**, pp.17-50, 1991 [in Russian].
10. S.O. Kuznetsov, Learning of Simple Conceptual Graphs from Positive and Negative Examples. In: J. Zytkow, J. Rauch (eds.), Proc. *Principles of Data Mining and Knowledge Discovery, Third European Conference, PKDD'99*, Lecture Notes in Artificial Intelligence, **1704**, pp. 384-392, 1999.
11. S.O. Kuznetsov and V.K. Finn, On a model of learning and classification based on similarity operation, *Obozrenie Prikladnoi i Promyshlennoi Matematiki* **3**, no. 1, 66-90, 1996 [in Russian].
12. T. Mitchell, Version Space: An Approach to Concept Learning, PhD thesis, Stanford University, 1978.
13. T. Mitchell, Generalization as Search, *Artificial Intelligence* **18**, no. 2, 1982.
14. T. Mitchell, Machine Learning, The McGraw-Hill Companies, 1997.
15. M.-L. Mugnier and M. Chein, Représenter des connaissances et raisonner avec des graphes, Revue d'Intelligence Artificielle, **10**(1), 1996, pp. 7-56.
16. M.-L. Mugnier, Knowledge Representation and Reasonings Based on Graph Homomorphisms, in *Proc. 8th Int. Conf. on Conceptual Structures, ICCS'2000*, G. Mineau and B. Ganter, Eds., Lecture Notes in Artificial Intelligence, **1867**, 2000, pp. 172-192.
17. S.-H. Nienhuys-Cheng and R. de Wolf, Foundations of Inductive Logic Programming, *Lecture Notes in Artificial Intelligence*, **1228**, 1997.
18. M. Sebag, Using Constraints to Building Version Spaces, in L. de Raedt and F. Bergadano, eds., *Proc. of the European Conference on Machine Learning (ECML-94)*, pp. 257-271, Springer, 1994.
19. M. Sebag, Delaying the Choice of Bias: A Disjunctive Version Space Approach, in L. Saitta ed., *Proc. of the 13th International Conference on Machine Learning*, pp. 444-452, Morgan Kaufmann, 1996.
20. E.N. Smirnov and P.J. Braspenning, Version Space Learning with Instance-Based Boundary Sets, in H. Prade, ed., *Proceedings of the 13th European Conference on Artificial Intelligence*, J. Wiley, Chichester, 460-464, 1998.
21. The web-site on Predictive Toxicology: http://www.predictive-toxicology.org/ptc/