Hyperrelations in version space

Hui Wang
School of Information and Software Engineering
University of Ulster
Newtownabbey, BT 37 0QB, N.Ireland
H.Wang@ulst.ac.uk

Ivo Düntsch & Günther Gediga
Department of Computer Science
Brock University
St. Catherines, Ontario, Canada, L2S 3AI
{duentsch|gediga}@cosc.brocku.ca

Andrzej Skowron
Institute of Mathematics
University of Warsaw
02-097 Warszawa, Poland
skowron@mimuw.edu.pl

Abstract

A version space is a set of all hypotheses consistent with a given set of training examples, delimited by the *specific boundary* and the *general boundary*. In existing studies [5, 6, 8] a hypothesis is a conjunction of attribute-value pairs, which is shown to have limited expressive power [9]. In a more expressive hypothesis space, e.g., disjunction of conjunction of attribute-value pairs, a general version space becomes uninteresting unless some restriction (inductive bias) is imposed [9].

In this paper we investigate version space in a hypothesis space where a hypothesis is a *hyperrelation*, which is in effect a disjunction of conjunctions of disjunctions of attribute-value pairs. Such a hypothesis space is more expressive than the conjunction of attribute-value pairs and the disjunction of conjunction of attribute-value pairs. However, given a dataset, we focus our attention only on those hypotheses which are consistent with given data and are maximal in the sense that the elements in a hypothesis can not be *merged* further. Such a hypothesis is called an *E–set* for the given data, and the set of all E–sets is the version space which is delimited by the least E–set (specific boundary) and the greatest E–set (general boundary).

Based on this version space we propose three classification rules for use in different situations. The first two are based on E–sets, and the third one is based on "degraded" E–sets called *weak hypotheses*, where the maximality constraint is relaxed. We present an algorithm to calculate E–sets, though it is computationally expensive in the worst case. We also present an efficient algorithm to calculate weak hypotheses. The third rule is evaluated using public datasets, and the results compare well with C5.0 decision tree classifier.

1 Introduction

Version space is a useful and powerful concept in the area of concept learning: A version space is the set of all hypotheses in a hypothesis space consistent with a dataset. A version space is delimited by the *specific boundary* and the *general boundary* – the sets of most specific and most general hypotheses respectively, to be explained below.

The ELIMINATE-CANDIDATE algorithm [6, 8] is the flagship algorithm used to construct a version space from a dataset. The hypothesis space used in this algorithm is the conjunction of attribute-value pairs, and it is shown to have limited expressive power [9]. It is also shown by [4] that the size of the general boundary can grow exponentially in the number of training examples, even when the hypothesis space consists of simple conjunctions of attribute-value pairs.

Table 3 was used in [9] to show the limitation of Mitchell's representation. It was shown that the ELIMINATE-CANDIDATE algorithm can not come up with a consistent specific boundary or general boundary for this example due to this limitation (i.e., the chosen hypothesis space). A possible solution, by increasing the expressive power of the representation, is to extend the hypothesis space to disjunctions of conjunctions of attribute-value pairs from the restrictive conjunctions of attribute-value pairs. But it turned out that, without proper inductive bias, the specific boundary would *always* be overly specific (the disjunction of the observed positive examples) and the general boundary would *always* be overly general (the negated disjunction of the observed negative examples) [9], so they do not carry useful information - they are "uninteresting". The desired inductive bias, however, has not been explored.

In this paper we report a study on version space which is aimed at increasing the expressive power of the hypothesis space and, at the same time, keeping the hypotheses "interesting" through introducing an inductive bias. We explore a semilattice structure that exists in the set of all *hypertuples* of a domain, where hypertuples generalise the traditional tuples from value-based to set-based. The semilattice structure can be used as a base for a hypothesis space. We take a hypothesis to be a *hyperrelation*, i.e., a set of hypertuples. A hyperrelation can be interpreted as a disjunction of hypertuples, which can be further interpreted as a conjunction of disjunctions of attribute-value pairs. Such a hypothesis space is much more expressive than the conjunction of attribute-value pairs and the disjunction of conjunction of attribute-value pairs. For a dataset there is a large number of hypertuples which are consistent with the data, some of which can be merged (through the semilattice operation) to form a different consistent hypertuple. We propose to focus on those hypertuples which are consistent with the given data and can not be merged further - they are said to be *maximal*. A hypothesis is then a set of these hypertuples, and the version space is the set of all these hypertuples. Clearly this version space is a subset of the semilattice. An algorithm is presented which is able to construct the version space.

A detailed analysis of Mitchell's version space shows that our version space is indeed a generalisation of Mitchell's and that it is more expressive than Mitchell's.

Classification within our version space is not trivial. We explore ways in which data can be classified using different hypotheses in the version space. We also examine how this whole process can be speeded up for practical benefits. Experimental results are presented to show the effectiveness of this approach.

Most of the theory (and results) of version space can be expressed by the tools offered by Boolean reasoning as investigated, for example, in [10–13, 15]. Unlike some of these publications we will only be concerned with consistent decision systems and exact classification rules.

2 Definitions and notation

For a set U, we let 2^U be the powerset of U. If $\leq$ is a partial ordering on U, and $X \subseteq U$, we let

$$\begin{aligned} \uparrow X &= \{y \in U : (\exists x \in X) x \leq y\}, \\ \downarrow X &= \{y \in U : (\exists x \in X) y \leq x\}. \end{aligned}$$

If $X, Y \subseteq U$, we say that Y *covers* X, written as $X \preccurlyeq Y$, if $X \subseteq \ \downarrow Y$, i.e. if for each $x \in X$ there is some $y \in Y$ with $x \leq y$. We call X *dense for* Y, written $X \trianglelefteq Y$, if for any $y \in Y$ there is some $x \in X$ such that $x \leq y$, i.e. $Y \subseteq \uparrow X$. Note that Y covers X iff Y is dually dense for X.

A *semilattice* is an algebra $\langle A, + \rangle$, such that $+$ is a binary operation on A satisfying

$$\begin{aligned} x + y &= y + x, \\ (x + y) + z &= x + (y + z), \\ x + x &= x. \end{aligned}$$

If $X \subseteq A$, we denote by $[X]$ the sub-semilattice of A generated by X. With some abuse of notation, we also use $+$ for the complex sum, i.e. $X + Y = \{x + y : x \in X,\ y \in Y\}$.

Each semilattice has an intrinsic order defined by

$$x \leq y \iff x + y = y. \tag{2.1}$$

x is called *minimal* (*maximal*) in $X \subseteq U$, if $y \leq x$ ($x \leq y$) implies $x = y$. The set of all minimal (maximal) elements of X is denoted by $\min X$ ($\max X$).

A *decision system* is a tuple $I = \langle U, \Omega, \{V_a : a \in \Omega\}, d, V_d \rangle$, where

1. $U = \{x_0, \ldots, x_N\}$ is a nonempty finite set.
2. $\Omega = \{a_0, \ldots, a_T\}$ is a nonempty finite set of mappings $a_i : U \to V_{a_i}$.
3. $d : U \to V_d$ is a mapping.

Set $\mathbf{V} = \prod_{a \in \Omega} V_a$. We let $I : U \to \mathbf{V}$ be the mapping

$$x \mapsto \langle a_0(x), a_1(x), \ldots, a_T(x) \rangle \tag{2.2}$$

which assigns to each object x its feature vector. $\mathbf{V}$ is called *data space*, and the collection $\mathbf{D} = \{I(x) : x \in U\}$ is called the *training space*. In the sequel $\mathbf{V}$ is assumed finite (consequently $\mathbf{D}$ is finite) unless otherwise stated. The mapping d defines a labelling (or partition) of $\mathbf{D}$ into classes $\mathbf{D}_0, \mathbf{D}_1, \ldots \mathbf{D}_K$, where $I(x)$ and $I(y)$ are in the same class $\mathbf{D}_i$ if and only if $d(x) = d(y)$. If $a \in \Omega, v \in V_a$, then $\langle a, v \rangle$ is called a *descriptor*.

We call

$$\mathcal{L} = \prod_{a \in \Omega} 2^{V_a} \tag{2.3}$$

the *extended data space*, its elements *hypertuples*, and its subsets *hyperrelations*. In contrast we call the elements of $\mathbf{V}$ *simple tuples* and its subsets *simple relations*. If $s \in \mathcal{L}$ and $a \in \Omega$ we let $s(a)$ be the projection of s with respect to a, i.e. the set appearing in the a^{th} component.

$\mathcal{L}$ is a lattice under the following natural order

$$s \leq t \Longleftrightarrow s(a) \subseteq t(a) \text{ for all } a \in \Omega,$$

with the least upper bound $(+)$ and greatest lower bound $(\times)$ operations given by

$$(t+s)(a) = t(a) \cup s(a),$$
$$(t \times s)(a) = t(a) \cap s(a)$$

for all $a \in \Omega$. s and t are said to be *overlapping* if there is some simple tuple d such that $d \leq t \times s$. By identifying a singleton with the element it contains, we may suppose that $\mathbf{V} \subseteq \mathcal{L}$; in particular, we have $\mathbf{D} \subseteq \mathcal{L}$.

Table 1(a) below shows a simple relation (i.e., a set of simple tuples) and Table 1(b) a hyperrelation.

a_1	a_2	y
a	0	α
a	1	α
b	1	α
b	2	β

(a)

a_1	a_2	y
$\{a,b\}$	$\{0,1\}$	α
$\{b\}$	$\{2\}$	β

(b)

Table 1: (a) A set of simple tuples. (b) A set of hypertuples.

For unexplained notation and background reading in lattice theory, we invite the reader to consult [3].

3 Hypertuples and hyperrelations

Suppose that $\mathbf{D}_q$ is a labelled class. The idea of [2, 19] was to collect, if possible, across the elements of $\mathbf{D}_q$ the values of the same attributes without destroying the labelling information. For example, if the system generates the rules

If x is blue, then sell,
if x is green, then sell,

we can identify the values "blue" and "green" and collect these into the single rule

If x is blue or green, then sell.

Our aim is to maximise the number of attribute values on the left hand side of the rule in order to increase the generality of the rule and also increase its base as to guard against random influences [see 1].

This leads to the following definition [18]: We call an element $r \in \mathcal{L}$ *equilabelled* with respect to the class $\mathbf{D}_q$, if

$$\downarrow r \cap \mathbf{D} \neq \emptyset, \tag{3.1}$$

and

$$\downarrow r \cap \mathbf{D} \subseteq \mathbf{D}_q. \tag{3.2}$$

These two conditions express two fundamental principles in machine learning: (3.1) says that r is supported by some $d \in \mathbf{D}$; it is a minimality condition in the sense that we do not want to consider hypertuples that have nothing to do with the training set. Condition (3.2) tells us that r needs to be consistent with the training set: All elements below r which are in the training set $\mathbf{D}$ are in the unique class $\mathbf{D}_q$. We denote the set of all elements equilabelled with respect to $\mathbf{D}_q$ by $\mathcal{E}_q$, and let $\mathcal{E} = \bigcup_{q \leq K} \mathcal{E}_q$ be the set of all equilabelled elements. Note that $\mathbf{D} \subseteq \mathcal{E}$, and that

$$q, r \leq K,\ q \neq r \text{ implies } \mathcal{E}_q \cap \mathcal{E}_r = \emptyset. \tag{3.3}$$

since $\{\mathbf{D}_0, \dots, \mathbf{D}_K\}$ partitions $\mathbf{D}$. Furthermore,

$$\uparrow \mathcal{E}_q \cap \mathcal{E} = \mathcal{E}_q. \tag{3.4}$$

If $\mathbf{D} \subseteq P \subseteq \mathcal{L}$, we let $E(P) = \{t \in \mathcal{L} : t \text{ is maximal in } [P] \cap \mathcal{E}\}$, and set $\mathbf{VSp} = \bigcup\{E(P) : \mathbf{D} \subseteq P \subseteq \mathbf{V}\}$. For each $q \leq K$, let $E_q(P) = E(P) \cap \mathcal{E}_q$.

Each set of the form $E(P)$ is called an *E–set* or a *hypothesis*. We now have

Lemma 3.1. *Let* $\mathbf{D} \subseteq P \subseteq Q \subseteq \mathcal{L}$. *Then,*

1. $E(P) \preccurlyeq E(Q)$.
2. $E(P) \trianglelefteq E(Q)$.

Proof. Both claims follow immediately from the fact that $\emptyset \neq [P] \cap \mathcal{E} \subseteq [Q] \cap \mathcal{E}$, and that $[Q] \cap \mathcal{E}$ is finite. □

Theorem 3.2. $\min \mathbf{VSp} = E(\mathbf{D})$, $\max \mathbf{VSp} = E(\mathbf{V})$.

Proof. We only prove the first part; the second part can be similarly proved.

"$\subseteq$": Suppose that t is minimal in **VSp**.Then, there is some $\mathbf{D} \subseteq P \subseteq \mathbf{V}$ such that $t \in E(P)$. Since $E(\mathbf{D}) \trianglelefteq E(P)$ by Lemma 3.1, there is some $s \in E(D)$ such that $s \leq t$, and the minimality of t implies $s = t$.

"$\supseteq$": If $t \in E(\mathbf{D})$ and $s \leq t$ is minimal in **VSp**, then $s \in E(\mathbf{D})$ as well. Since the elements of $E(\mathbf{D})$ are pairwise incomparable, we have $s = t$. □

In the sequel, we will denote $E(\mathbf{D})$ by $\mathbb{S}$ and $E(\mathbf{V})$ by $\mathbb{G}$, since they correspond, respectively, to the specific and general boundaries of $\mathbf{D}$ in the sense of [7]; we also set $\mathbb{S}_q = E_q(\mathbf{D})$ and $\mathbb{G}_q = E_q(\mathbf{V})$. By Lemma 3.1, $\mathbb{S}$ is the least E–set and $\mathbb{G}$ is the greatest E–set.

An algorithm to find $E(\mathbf{D})$, called the *lattice machine* (LM), has been presented in [18], and it can easily be generalised to find $E(P)$ for each $\mathbf{D} \subseteq P \subseteq \mathbf{V}$:

Algorithm [LM algorithm]

1. $H_0 \overset{\text{def}}{=} P$.
2. $H_{k+1} \overset{\text{def}}{=}$ The set of maximal elements of $[\downarrow (H_k + P)] \cap \mathcal{E}$.
3. Continue until $H_n = H_{n+1}$ for some n.

It was shown in [19] that $H_n = E(P)$. If $|P| = m$, then the complexity of the algorithm is $O(2^m)$ in the worst case.

To illustrate the above notions we consider the following example.

Example A dataset is shown in Figure 1. The small circles and crosses are simple tuples of different classes while the rectangles are hypertuples.

Every hypertuples depicted here are equilabelled since they cover only simple tuples of the same class. All the simple tuples are also equilabelled.

Each hypertuple here is maximal since we cannot move its boundary in any dimension while maintaining it being equilabelled.

The set of all hypertuples in the figure is an E-set or a hypothesis for the underlying concept implied by the dataset, since all hypertuples are equilabelled and maximal, and they together cover all the simple tuples. In fact this E-set is $E(D)$ – the specific boundary. The general boundary is not depicted here as we need knowledge of the domain of each attribute to calculate it. Clearly the E-set does not cover the whole data space; in other word, there are regions that are not covered by the E-set.

Primary elements are those that are covered explicitly by some equilabelled hypertuples; secondary elements are those not covered explicitly but can be covered by extending some equilabelled hypertuple without compromising its equilabelledness. We can easily find primary and secondary elements from this figure. □

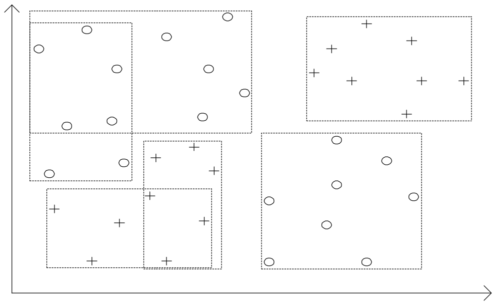

Figure 1: *A dataset and its specific boundary. Each circle or cross is a simple tuple, and each rectangle represents a hypertuple.*

Our next aim is to show that every simple tuple is covered by some equilabelled element:

Lemma 3.3. *For each $t \in \mathbf{V}$ there is some $h \in \mathcal{E}$ such that $t \leq h$.*

Proof. We show that there is some $x \in \mathbf{D}$ such that $\downarrow (t+x) \cap \mathbf{D} = \{x\}$; then, $t \leq t + x \in \mathcal{E}$: Let $t \in V$, and set $M = \{t + x : x \in \mathbf{D}\}$. Suppose that $x \in \mathbf{D}$ such that $t + x$ is minimal in M with respect to $\leq$. Let $y \in \mathbf{D}$ such that $y \leq t+x$, and assume that $y \neq x$; then, there is some $a \in \Omega$ such that $y(a) \neq x(a)$. Since both y and x are simple tuples, we have in fact $y(a) \cap x(a) = \emptyset$. Now, $y \leq t+x$ implies that $y(a) \subseteq t(a) \cup x(a)$, and it follows from $y(a) \cap x(a) = \emptyset$ – and the fact that t is simple – that $y(a) = t(a) \subsetneq t(a) \cup x(a)$. Hence, $t+y \lneq t+x$, contradicting the minimality of $t+x$ in M. □

Since each element of $\mathcal{E}$ is covered by some element of $\mathbb{G}$, and the sets $\mathbb{G}_i$, $i \leq K$, partition $\mathbb{G}$, we obtain

Corollary 3.4. *For each $t \in \mathbf{V}$, there is some $i \leq K$ such that $t \preccurlyeq \mathbb{G}_i$.*

Observe that such $\mathbb{G}_i$ need not be unique, and we need a selection procedure to label t. Our chosen method is majority voting, combined with random selection for tied maxima: Let

$$m'(t,n,\mathbb{G}) = |\{h \in \mathbb{G}_n : t \leq h\}|, \text{ for each } n \leq K, \tag{3.5}$$

$$m(t,\mathbb{G}) = \max\{M(t,n,\mathbb{G}) : n \leq K\}, \tag{3.6}$$

$$M(t,\mathbb{G}) = \{i \leq K : m'(t,i,\mathbb{G}) = m(t,\mathbb{G})\}. \tag{3.7}$$

Rule 1. If $t \in \mathbf{V}$, then label t by a randomly chosen element of $M(t,\mathbb{G})$.

Now we consider an arbitrary hypothesis H. We can replace the $\mathbb{G}$ by H in Eqs. 3.5-3.7, and apply Rule 1 for classification. However, unlike the case where $H = \mathbb{G}$, there is no guarantee that $t \preccurlyeq H$, i.e. that $m(t,H) \neq 0$.

This motivates us to distinguish between different elements in $\mathbf{V}$ given $H = E(P)$: $t \in \mathbf{V}$ is called *primary* if there is H_q with $t \preccurlyeq H_q$, t is *secondary* if there is $h \in H_q$ such that $t+h$ is equilabelled (i.e., $t+h$ does not overlap H_p for $p \neq q$), and t is *tertiary* otherwise. Note that an element can be both primary and secondary. In fact primary data is a subset of secondary data.

Now we let

$$m'_p(t,n,H) = |\{h \in H_n : t \leq h\}|, \text{ for each } n \leq K, \tag{3.8}$$

$$m_p(t,H) = \max\{m'_p(t,n,H) : n \leq K\}, \tag{3.9}$$

$$M_p(t,H) = \{i \leq K : m'_p(t,i,H) = m_p(t,h)\}. \tag{3.10}$$

and

$$m'_s(t,n,H) = |\{h \in H_n : t+h \in \mathcal{E}\}|, \text{ for each } n \leq K, \tag{3.11}$$

$$m_s(t,H) = \max\{m'_s(t,n,H) : n \leq K\}, \tag{3.12}$$

$$M_s(t,H) = \{i \leq K : m'_s(t,i,H) = m_s(t,h)\}. \tag{3.13}$$

Rule 2.

- If t is primary, then label t by a randomly chosen element of $M_p(t,H)$.
- If t is secondary, then label t by a randomly chosen element of $M_s(t,H)$.
- Otherwise, label t as *unclassified*.

Now we explore a generalisation of both primary and secondary data in the following sense:

Lemma 3.5. *If $H = E(P)$, we let $P' = P \cup \{t\}$ and $H' = E(P')$. Then,*

$$t \text{ is secondary for } H \Rightarrow t \text{ is primary for } H'.$$

Proof. Let $h \in H$ such that $t+h$ is equilabelled. Then, $t+h \in [P'] \cap \mathcal{E}$, and thus, $t \leq g$ for some $g \in E(P')$. □

The converse is not true: Let $\mathbf{D} = \{a,b,c,d,e,f\}$ with

$$a = \langle 0,0,0\rangle, b = \langle 1,0,0\rangle, c = \langle 1,0,1\rangle, d = \langle 1,1,1\rangle, e = \langle 0,1,1\rangle, f = \langle 0,1,0\rangle.$$

Suppose that a,b,e are coloured blue and d,e,f are coloured red. Then,

$$E(\mathbf{D}) = \{a+b, e, c+d, f\}.$$

The aim is to classify $t = \langle 0,0,1\rangle$ with respect to the hypothesis $H = E(\mathbf{D})$. Now,

$$a+t = \langle 0,0,01\rangle \text{ is equilabelled blue,}$$
$$e+t = \langle 0,01,1\rangle \text{ is equilabelled blue,}$$
$$c+t = \langle 01,0,1\rangle \text{ is equilabelled red,}$$

while $b+t, d+t, f+t$ are not equilabelled. Furthermore, $q+t$ is not equilabelled for any $q \in E(\mathbf{D})$, and thus, t is not secondary with respect to $E(\mathbf{D})$. If we admit that the knowledge provided by t should be admitted when we try to classify t by $H = E(P)$, then we should classify by using primary data of $E(P \cup \{t\})$. At any rate, admitting t does not cause any inconsistencies, and using primary data of H' is well in line with our aim of maximising consistency.

3.1 Weak hypotheses

In the previous discussion we have introduced two classification rules based on E–sets. Rule 1 can be applied if $\mathbb{G}$ can be practically constructed from $\mathbf{V}$ by the LM algorithm, and Rule 2 can be applied if $\mathbb{S}$ (or $E(P)$) can be practically constructed from $\mathbf{D}$ (for $\mathbf{D} \subseteq P \subset \mathbf{V}$). In both cases we have to construct E–sets for P where $\mathbf{D} \subseteq P \subseteq \mathbf{V}$. From the LM algorithm we know that constructing an E–set is expensive and in the worst case it is exponential. Since the most time is spent finding maximal hypertuples, we make the following definition: A *weak hypothesis* is a set of equilabelled hypertuples which covers $\mathbf{D}$. The following algorithm finds a weak hypothesis H [17]:

```
$H \leftarrow \emptyset$
**for** $q = 0$ to $K$ **do**
  $X \leftarrow D_q$
  **while** $X \neq \emptyset$ **do**
    Order $X$ as $\langle g_0, \ldots, g_{m(X)}\rangle$
    $h \leftarrow g_0$
    **for** $i = 1$ to $m(X)$ **do**
      **if** $h + g_i$ is equilabelled **then**
```

```
            $h \leftarrow h + g_i$
          **end if**
        **end for**
        $X \leftarrow X \setminus \downarrow h$
        $H \leftarrow H \cup \{h\}$
      **end while**
    **end for**
```

The algorithm does not necessarily produce disjoint hypertuples: Let

$$D_0 = \{\langle 0,1,1\rangle, \langle 1,0,1\rangle, \langle 1,1,1\rangle, \langle 1,1,0\rangle, \langle 2,1,1\rangle\},\ D_1 = \{\langle 2,0,1\rangle, \langle 0,0,0\rangle\}.$$

Order $X = D_0$ by

$$\begin{aligned} g_0 &= \langle 0,1,1\rangle, \\ g_1 &= \langle 1,0,1\rangle, \\ g_2 &= \langle 1,1,1\rangle, \\ g_3 &= \langle 1,1,0\rangle, \\ g_4 &= \langle 2,1,1\rangle. \end{aligned}$$

Now, $h_0 = g_0 + g_1 + g_2$ is equilabelled, while $h_0 + g_3$ and $h_0 + g_4$ are not. The next step produces $h_1 = g_3 + g_4$. However, $g_2 \leq h_0$ and $g_2 \leq h_1$.

In the worst case the time complexity for building H_q (the hypothesis for class $\mathbf{D}_q$) is $O(|\mathbf{D}_q|^2)$. Therefore the worst case complexity for building H (the whole hypothesis) is $O(K \times |\mathbf{D}_q|^2)$, where K is the number of classes.

Let $H = \bigcup_{i \leq K} H_i$ be a weak hypothesis for $\mathbf{D}$, where $H_i = \{h_0, \cdots, h_{t(i)}\}$ is a weak hypothesis for class i and h_j is an equilabelled hypertuple. Let $M_p(t,H)$ and $M_s(t,H)$ be defined as in Eq. 3.8 and 3.11. Then we introduce the following rule, which is the same as Rule 2 except that the hypothesis here is weak.

Rule 3.
- If t is primary, then label t by a randomly chosen element of $M_p(t,H)$.
- If t is secondary, then label t by a randomly chosen element of $M_s(t,H)$.
- Otherwise, label t as *unclassified*.

4 Mitchell's version space

The situation in [9] can be described as a special decision system where $d : U \to \{0,1\}$, and it is called the *target concept*. The set of positive examples is denoted by D_1, and that of negative examples by D_0. We will describe the concepts introduced there in our notation and we will follow the explanation in [9, p. 22, Table 2.2].

A *hypothesis* is a hypertuple $t \in \mathcal{L}$ such that for all $a \in \Omega$

$$|t(a)| \leq 1 \text{ or } t(a) = V_a. \tag{4.1}$$

Thus, for each $a \in \Omega$ we have

$$t(a) = \begin{cases} \emptyset, & \text{or} \\ m, & \text{for some } m \in V_a, \text{ or} \\ V_a. \end{cases}$$

The set of all hypotheses is denoted by $\mathcal{H}$. Observe that $\mathcal{H}$ is a hyperrelation, and that $\mathcal{H}$ is partially ordered by $\leq$ as defined by (2.1), and that $\mathbf{V} \subseteq \mathcal{H}$. If $s,t \in \mathcal{H}$, and $s \leq t$, we say that s *is more specific than* t or, equivalently, that t *is more general than* s. We say that s *satisfies hypothesis* t, if

1. s is a simple tuple, i.e. $s \in \mathbf{V}$,
2. $s \leq t$,

and denote the set of all (simple) tuples that satisfy t by $\text{sat}(t)$; observe that $\text{sat}(t) = \downarrow t \cap \mathbf{V}$. More generally, for $A \subseteq \mathcal{L}$ we let

$$\text{sat}(A) = \downarrow A \cap \mathbf{V}.$$

If $t(a) = \emptyset$ for some $a \in \Omega$, then t cannot be satisfied. We interpret $s \in \text{sat}(t)$ as "instance s is classified by hypothesis t". According to [9], t is more general than s, if any instance classified by s is also classified by t; in other words, $\text{sat}(s) \subseteq \text{sat}(t)$. That our notion captures this concept is shown by the following result, the easy proof of which is left to the reader.

Theorem 4.1. *Suppose that* $s,t \in \mathcal{H}$. *Then,*

$$s \leq t \iff \text{sat}(s) \subseteq \text{sat}(t).$$

Since we are interested in hypotheses whose satisfiable observations are within one class of d, we say that $t \in \mathcal{H}$ is *consistent with* $\langle \mathbf{D}, d \rangle$, if

$$\downarrow t \cap \mathbf{D} = D_1. \tag{4.2}$$

Thus, in this case, the training examples satisfying t are exactly the positive ones. The *version space* $\mathbf{VSp}^m$ is now the set of all hypotheses consistent with $\langle \mathbf{D}, d \rangle$. In other words,

$$\mathbf{VSp}^m = \{t \in \mathcal{H} : (\forall s)[s \in \text{sat}(t) \cap \mathbf{D} \iff d(s) = 1]\}.$$

The *general boundary* $\mathbb{G}^m$ is the set of maximal members of $\mathcal{H}$ consistent with $\langle \mathbf{D}, d \rangle$, i.e.

$$\mathbb{G}^m = \max \mathbf{VSp}^m.$$

The *specific boundary* $\mathbb{S}^m$ is the set of minimal members of $\mathcal{H}$ consistent with $\langle \mathbf{D}, d \rangle$, i.e.

$$\mathbb{S}^m = \min \mathbf{VSp}^m.$$

The two boundaries delimit the version space in the sense that for any consistent hypothesis t in the version space there are $g \in \mathbb{G}^m$ and $s \in \mathbb{S}^m$ such that $s \leq t \leq g$.

The example in Table 2 on the following page illustrates the idea of version space. We follow [9] in writing ? in column a, if $a(x) = V_a$.

Table 2: Training data, hypotheses and boundaries [9]

Sky	ATemp	Humid	Wind	Water	FCast	d
Training data **D**						
Sunny	Warm	Normal	Strong	Warm	Same	1
Sunny	Warm	High	Strong	Warm	Same	1
Rainy	Cold	High	Strong	Warm	Change	0
Sunny	Warm	High	Strong	Cool	Change	1
Hypotheses						
Sunny	Warm	?	Strong	?	?	1
Sunny	?	?	Strong	?	?	1
Sunny	Warm	?	?	?	?	1
?	Warm	?	Strong	?	?	1
Sunny	?	?	?	?	?	1
?	Warm	?	?	?	?	1
Specific boundary						
Sunny	Warm	?	Strong	?	?	1
General boundary						
Sunny	?	?	?	?	?	1
?	Warm	?	?	?	?	1

5 Expressive power of version space and Boolean reasoning

A simple tuple t can be regarded as a conjunction of descriptors

$$\langle a_0, t(a_0)\rangle \wedge \langle a_1, t(a_1)\rangle \wedge \cdots \wedge \langle a_T, t(a_T)\rangle.$$

If t is a satisfiable hypothesis, then no $t(a)$ is empty, and $t(a) = V_a$ tells us that any value is allowed in this column. Thus, such a descriptor places no restriction on satisfiability in column a. If $I = \{i \leq T : t(a_i) \neq V_{a_i}\}$, then,

$$s \in \mathrm{sat}(t) \iff (\forall i \in I) s(a_i) = t(a_i),$$

so that we can interpret t as the conjunction

$$\bigwedge_{i \in I} \langle a_i, t(a_i)\rangle.$$

Such an expression is called an *exact template* in [13]. The expressive power of this type of conjunctive hypothesis is limited. An example from [9] illustrating this is shown in Table 3 on the next page. For such a simple dataset, there is no consistent hypothesis in the sense of (4.2).

A hypothesis in $\mathbf{VSp}^m$ is a very special kind of hypertuple, and it is our aim to extend this notion of hypothesis, so that the resulting structures are more expressive while at the same time not so general as to carry no useful information. As suggested by Mitchell, a possible solution is to use arbitrary disjunctions of conjunctions of

Table 3: Limitation of version space

	Sky	ATemp	Humid	Wind	Water	FCast	d
1	Sunny	Warm	Normal	Strong	Cool	Change	1
2	Cloudy	Warm	Normal	Strong	Cool	Change	1
3	Rainy	Warm	Normal	Strong	Cool	Change	0

descriptors as hypothesis representation. It is not hard to see that this is overly general, since any positive Boolean expression can then serve as a hypothesis. In contrast, we suggest to use a specific class of disjunctions of conjunctions which is significantly narrower than the class of all positive Boolean expressions. The building blocks will be the hypertuples contained in the sub-semilattice of $\mathcal{L} = \prod_{a\in\Omega} 2^{V_a}$ generated by $\mathbf{V}$ with $+$ operator – $[\mathbf{V}]$. Since we are only interested in the finite sums of elements of $\mathbf{V}$ we will from now on assume that each V_a is finite.

Suppose that $t = \langle\{m_0^a, m_1^a, \ldots, m_{t(a)}^a\}\rangle_{a\in\Omega}$ is a hypertuple. We interpret t as a conjunction of disjunctions of descriptors

$$\bigwedge_{a\in\Omega} (\langle a, m_0^a\rangle \vee \cdots \vee \langle a, m_{t(a)}^a\rangle). \tag{5.1}$$

By the distributivity of $\wedge$ and $\vee$ this can always be turned into a disjunction of simple tuples, but not every disjunction of simple tuples (considered as a conjunction of descriptors) is equivalent to an expression such as (5.1); consider, for example,

$$(\langle a_0, t_0^0\rangle \wedge \langle a_1, t_1^0\rangle) \vee (\langle a_0, t_0^1\rangle \wedge \langle a_1, t_1^1\rangle).$$

A hypertuple can be viewed as a construction similar to hypercubes which delineate solids in an appropriate space.

Now we compare our hypothesis space with Mitchell's from the perspective of expressive power. Consider a dataset $\mathbf{D}$ as defined earlier. Suppose all attributes x are discrete, and all V_x are finite. In our hypothesis space each attribute x takes on a subset of V_x, so there are $2^{|V_x|}$ different subsets altogether. As a result there are $\prod_{x\in\Omega} 2^{|V_x|}$ different hypertuples. Since a hypothesis is a set of hypertuples (i.e., a hyperrelation), there are $2^{\prod_{x\in\Omega} 2^{|V_x|}}$ distinct hypotheses.

In Mitchell's hypothesis space each attribute x takes on a single value in V_x plus two other special values, "?" and "$\emptyset$". Therefore there are $|V_x| + 2$ different values, and $\prod_{x\in\Omega}(|V_x| + 2)$ different tuples. In his conjunctive hypothesis representation, each hypothesis is a (simple) tuple, so there are $\prod_{x\in\Omega}(|V_x| + 2)$ distinct hypotheses. In his disjunctive hypothesis representation, each hypothesis is a set of (simple) tuples (i.e., simple relation), so there are $2^{\prod_{x\in\Omega}(|V_x|+2)}$ distinct hypotheses.

Clearly our hypothesis space can represent more distinct objects than Mitchell's can. In this sense we say our hypothesis is more expressive than Mitchell's.

Note that $\mathcal{E}$ characterises the eligible hypotheses. So Mitchell's version space can be specialised from our version space in the following way:

- The $\mathcal{E}$ is restricted to $\mathcal{E}^m = \{\gamma(t) : t \in \mathcal{E}, \mathbf{D} \preccurlyeq t\}$, where γ is an operation to turn a hypertuple into a simple tuple in such a way that $\gamma(t) = \langle t_0, t_1, \cdots, t_T \rangle$, where

  $$t_i = \begin{cases} t(x_i), & \text{if } |t(x_i)| = 1, \\ ?, & \text{otherwise.} \end{cases}$$

  Note that $t(x_i)$ is the projection of tuple t onto its x_i attribute.

- Each hypothesis is an element of $\mathcal{E}^m$.

- There are two classes, i.e., $K = 2$;

- The version space is built for only one class.

Given the above restrictions the specific boundary is $\mathbb{S}^m = \{\mathbb{S}_0^m, \mathbb{S}_1^m\}$, where $\mathbb{S}_0^m$ is the specific boundary for the negative class and $\mathbb{S}_1^m$ is the specific boundary for the positive class. Similarly $\mathbb{G}^m = \{\mathbb{G}_0^m, \mathbb{G}_1^m\}$.

6 Experiment

We have evaluated Rule 3 using public datasets. We used 17 public datasets in our evaluation, which are available from UC Irvine Machine Learning Repository. General information about these datasets is shown in the first three columns of Table 4.

We used the CASEEXTRACT algorithm to construct weak hypotheses for the datasets, and applied Rule 3 to classify new data. For presentation purpose we refer to our classification procedure by GLM. The experimental results are shown in the last 5 columns of Table 4. As a comparison the C5.0 results on the same datasets are also shown. It is clear from this table that GLM performs extremely well on primary data, but it accounts for only 76.4% of all data on average. Over all data GLM compares well with C5.0.

Parity problems are well known to be difficult for many machine learning algorithms. We evaluated the GLM algorithm using three well known parity datasets [16] – Monk-1, Monk-2, Monk-3 [1]. Experimental results show that GLM works well for these parity datasets.

7 Discussion and conclusion

Mitchell's classical work on version space has been followed by many. Most notably Hirsh and Sebag. Hirsh [5] discusses how to merge version spaces when a central idea in Mitchell's work is removed – a version space is the set of concepts *strictly consistent* with training data. This merging process can therefore accommodate noise. Sebag [14] presents what she calls a disjunctive version space approach to learning disjunctive concepts from noisy data. A separate version space is learned for each positive training example, then new instances are classified by combining the votes of these different version spaces.

[1] Target Concepts associated to the Monk's problem: Monk-1: ($a1 = a2$) or ($a5 = 1$); Monk-2: exactly two of $a1 = 1$, $a2 = 1$, $a3 = 1$, $a4 = 1$, $a5 = 1$, $a6 = 1$; Monk-3: ($a5 = 3$ and $a4 = 1$) or ($a5 \neq 4$ and $a2 \neq 3$) (5% class noise added to the training set)

Dataset	#Attr.	#Train	#Test	Class. Accuracy (%)				%PP
				C5.0	GLM/SR	GLM/PSR	GLM/SSR	
Annealing	38	798	CV-5	96.6	96.4	98.0	62.9	92.5
Australian	14	690	CV-5	90.6	95.1	95.1	100.0	87.2
Auto	25	205	CV-5	70.7	82.4	87.0	63.0	56.1
Diabetes	8	768	CV-5	72.7	70.7	71.0	40.0	66.8
German	20	1000	CV-5	71.7	72.6	72.6	N/A	65.4
Glass	9	214	CV-5	80.4	86.6	87.6	76.5	79.4
Heart	13	270	CV-5	77.0	81.9	81.9	N/A	61.5
Hepatitis	19	155	CV-5	80.6	82.9	84.1	50.0	69.0
Horse-Colic	22	368	CV-5	85.1	82.7	82.7	N/A	67.7
Iris	4	150	CV-5	94.7	93.1	97.6	66.7	82.7
Monk-1	6	124	432	74.3	100.0	100.0	N/A	100.0
Monk-2	6	169	432	65.1	81.4	87.3	41.5	83.6
Monk-3	6	122	432	97.2	93.8	94.3	89.2	89.1
Sonar	60	208	CV-5	71.6	81.6	81.3	100.0	59.1
TTT	9	958	CV-5	86.2	96.2	96.1	100.0	94.9
Vote	18	232	CV-5	96.5	96.5	98.6	77.3	89.7
Wine	13	178	CV-5	94.3	99.0	99.0	N/A	53.9
Average				82.7	87.8	89.1	72.3	76.4

Table 4: *General information about the datasets and the classification accuracy of C5.0 on all data and of* GLM *on primary and secondary data. The validation method used is either 5 fold cross validation or explicit train/test validation. The acronyms are: SR – overall success ratio of primary and secondary data (i.e., the percentage of successfully classified primary and secondary data tuples over all primary and secondary data tuples), PSR – success ratio of primary data, SSR – success ratio of secondary data, PP –the percentage of primary data, and N/A – not available.*

In this paper we investigate version spaces in a more expressive hypothesis space - disjunction of conjunctions of disjunctions, where each hypothesis is a set of hypertuples. Without a proper inductive bias the version space is uninteresting. We show that, with E-set as an inductive bias, this version space is a generalisation of Mitchell's original version space, which employs a different type of inductive bias.

For classification within the version space we proposed three classification rules for use in different situations. The first two rules are based on E-sets as hypotheses, and they can be applied when the data space (**V**) is finite and small. We showed that constructing E-sets is computationally expensive, so we introduced the third rule which is based on weak hypotheses. We presented an algorithm to construct weak hypotheses efficiently.

Experimental results show that this classification approach performs extremely well on primary data, which account for over 75.0% of all data. Over all data this classification approach is comparable to C5.0.

References

[1] Ivo Düntsch and Günther Gediga. Statistical evaluation of rough set dependency analysis. *International Journal of Human–Computer Studies*, 46:589–604, 1997.

[2] Ivo Düntsch and Günther Gediga. Simple data filtering in rough set systems. *International Journal of Approximate Reasoning*, 18(1–2):93–106, 1998.

[3] G. Grätzer. *General Lattice Theory*. Birkhäuser, Basel, 1978.

[4] D. Haussler. Quantifying inductive bias: Ai learning algorithms and valiant's learning framework. *Artificial Intelligence*, 36:177–221, 1988.

[5] H. Hirsh. Generalizing version spaces. *Machine Learning*, 17(1):5–46, 1994.

[6] T. M. Mitchell. Version spaces: A candidate elimination approach to rule learning. In *Proc. 5th IJCAI*, pages 305–310, 1977.

[7] T. M. Mitchell. *Version spaces: An approach to concept learning*. PhD thesis, Electrical Engineering Dept., Stanford University, Stanford, CA, 1979.

[8] T. M. Mitchell. Generalization as search. *Artificial Intelligence*, 18, 1982.

[9] T. M. Mitchell. *Machine Learning*. The McGraw-Hill Companies, Inc, 1997.

[10] H. S Nguyen. From optimal hyperplanes to otimal decision trees. *Fundamenta Informaticae*, 34:145–174, 1998.

[11] H. S Nguyen and S. H. Nguyen. Pattern extraction from data. *Fundamenta Informaticae*, 34:129–144, 1998.

[12] H.S. Nguyen and A. Skowron. Boolean reasoning for feature extraction problems. In Z.W. Ras and A. Skowron, editors, *Proc. of the 10th International Symposium on Methodologies for Intelligent Systems (ISMIS'97)*, volume 1325 of *Lecture Notes in Artificial Intelligence*, pages 117–126, Berlin, 1997. Springer–Verlag.

[13] Sin Hoa Nguyen, Andrzej Skowron, and Piotr Synak. Discovery of data patterns with applications to decomposition and classification problems. In Lech Polkowski and Andrzej Skowron, editors, *Rough sets in knowledge discovery, Vol. 2*, pages 55–97, Heidelberg, 1998. Physica–Verlag.

[14] M. Sebag. Delaying the choice of bias: A disjunctive version space approach. In *Proc. of the 13th International Conference on Machine Learning*, pages 444–452, San Francisco, 1996. Morgan Kaufmann.

[15] A. Skowron. Extracting laws from decision tables - a rough set approach. *Computational Intelligence*, 11:371–388, 1995.

[16] S.B. Thrun, J. Bala, E. Bloedorn, I. Bratko, B. Cestnik, J. Cheng, K. De Jong, S. Dzeroski, S.E. Fahlman, D. Fisher, R. Hamann, K. Kaufman, S. Keller, I. Kononenko, J. Kreuziger, R.S. Michalski, T. Mitchell,

P. Pachowicz, Y. Reich H. Vafaie, W. Van de Welde, W. Wenzel, J. Wnek, and J. Zhang. The monk's problems - a performance comparison of different learning algorithms. Technical Report CS-CMU-91-197, Carnegie Mellon University, 1991.

[17] H. Wang, W. Dubitzky, I. Düntsch, and D. Bell. A lattice machine approach to automated casebase design: Marrying lazy and eager learning. In *Proc. IJCAI99*, pages 254–259, Stockholm, Sweden, 1999.

[18] Hui Wang, Ivo Düntsch, and David Bell. Data reduction based on hyper relations. In Rakesh Agrawal, Paul Stolorz, and Gregory Piatetsky-Shapiro, editors, *Proceedings of KDD'98*, pages 349–353, New York, 1998.

[19] Hui Wang, Ivo Düntsch, and Günther Gediga. Classificatory filtering in decision systems. *International Journal of Approximate Reasoning*, 23:111–136, 2000.

A Notation

$$\begin{aligned}
X \preccurlyeq Y &\Longleftrightarrow (\forall x \in X)(\exists y \in Y) x \leq y \\
X \trianglelefteq Y &\Longleftrightarrow (\forall y \in Y)(\exists x \in X) x \leq y \\
U &= \{x_0, \ldots, x_N\} \\
\Omega &= \{a_0, \ldots, a_T\} \\
\mathbf{V} &= \prod_{a \in \Omega} V_a \\
\mathcal{L} &= \prod_{a \in \Omega} 2^{V_a} \\
I(x) &= \langle a_0(x), a_1(x), \ldots, a_T(x) \rangle \\
\mathbf{D} &= \{I(x) : x \in U\} = \mathbf{D}_0 \cup \mathbf{D}_1 \cup \cdots \cup \mathbf{D}_K \\
\mathcal{E}_q &= \text{ set of all elements equilabelled with respect to } \mathbf{D}_q \\
\mathcal{E} &= \bigcup_{q \leq K} \mathcal{E}_q \\
E(P) &= \{t \in \mathcal{L} : t \text{ is maximal in } [P] \cap \mathcal{E}\} \\
E_q(P) &= E(P) \cap \mathcal{E}_q \\
\mathbb{S} &= E(\mathbf{D}) \\
\mathbb{S}_q &= E_q(\mathbf{D}) \\
\mathbb{G} &= E(\mathbf{V}) \\
\mathbb{G}_q &= E_q(\mathbf{V}) \\
\mathbf{VSp} &= \bigcup \{E(P) : \mathbf{D} \subseteq P \subseteq \mathbf{V}\}
\end{aligned}$$

B Acknowledgements

Co–operation for this work was partially supported by EU COST Action 274 “Theory and Applications of Relational Structures as Knowledge Instruments” (TARSKI). The work of Hui Wang was partly supported by the European Commission project ICONS, project no. IST-2001-32429. Ivo Düntsch gratefully acknowledges support from the National Sciences and Engineering Research Council of Canada (NSERC).