Hyperrelations in version space

Hui Wang a,*, Ivo Düntsch b, Günther Gediga c, Andrzej Skowron d

a School of Information and Software Engineering, University of Ulster, Faculty of Informatics, Newtownabbey, Co. Antrim BT 37 0QB, Ireland
b Department of Computer Science, Brock University, St. Catharines, Ont., Canada L2S 3A1
c Institut für Evaluation und Marktanalysen, Brinkstr. 19, 49143 Jüchen, Germany
d Institute of Mathematics, University of Warsaw, 02-097 Warszawa, Poland

Received 1 October 2002; accepted 1 October 2003

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 [Machine Learning 17(1) (1994) 5; Proc. 5th IJCAI (1977) 305; Artificial Intelligence 18 (1982)] a hypothesis is a conjunction of attribute-value pairs, which is shown to have limited expressive power [Machine Learning, The McGraw-Hill Companies, Inc (1997)]. 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 [Machine Learning, The McGraw-Hill Companies, Inc (1997)].

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 cannot 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).

*Corresponding author. Tel.: +44-1232-368981; fax: 44-1232-366068.
E-mail addresses: h.wang@ulst.ac.uk (H. Wang), duentsch@cosc.brocku.ca (I. Düntsch), gediga@eval-institut.de (G. Gediga), skowron@mimuw.edu.pl (A. Skowron).

0888-613X/S - see front matter © 2003 Published by Elsevier Inc.
doi:10.1016/j.ijar.2003.10.007
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.

© 2003 Published by Elsevier Inc.

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 cannot 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 cannot 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
\[
\uparrow X = \{y \in U : (\exists x \in X) x \leq y\},
\]
\[
\downarrow X = \{y \in U : (\exists x \in X) y \leq x\}.
\]

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

A semilattice is an algebra $\langle A, + \rangle$, such that $+$ is a binary operation on $A$
satisfying
\[
x + y = y + x,
\]
\[
(x + y) + z = x + (y + z),
\]
\[
x + x = x.
\]
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
\[ x \leq y \iff x + y = y. \quad (2.1) \]

$x$ 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 $\text{min} X$ ($\text{max} X$).

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 $a_i : U \rightarrow V_{a_i}$.
3. $d : U \rightarrow V_d$ is a mapping.

Set $V = \prod_{a \in \Omega} V_a$. We let $I : U \rightarrow V$ be the mapping
\[ x \mapsto \langle a_0(x), a_1(x), \ldots, a_T(x) \rangle \quad (2.2) \]

which assigns to each object $x$ its feature vector. $V$ is called data space, and the collection $D = \{I(x) : x \in U\}$ is called the training space. In the sequel $V$ is assumed finite (consequently $D$ is finite) unless otherwise stated. The mapping $d$ defines a labelling (or partition) of $D$ into classes $D_0, D_1, \ldots, D_K$, where $I(x)$ and $I(y)$ are in the same class $D_i$ 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
\[ L = \prod_{a \in \Omega} 2^{V_a} \quad (2.3) \]

the extended data space, its elements hypertuples, and its subsets hyperrelations.

In contrast we call the elements of $V$ simple tuples and its subsets simple relations. If $s \in L$ 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.

$L$ is a lattice under the following natural order
\[ s \leq t \iff s(a) \subseteq t(a) \text{ for all } a \in \Omega, \]

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 $V \subseteq L$; in particular, we have $D \subseteq L$. 


Table 1

<table>
<thead>
<tr>
<th></th>
<th>a₁</th>
<th>a₂</th>
<th>y</th>
</tr>
</thead>
<tbody>
<tr>
<td>(a) A set of simple tuples</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>0</td>
<td></td>
<td>z</td>
</tr>
<tr>
<td>a</td>
<td>1</td>
<td></td>
<td>z</td>
</tr>
<tr>
<td>b</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>2</td>
<td></td>
<td>β</td>
</tr>
<tr>
<td>(b) A set of hypertuples</td>
<td>{0,1}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{a,b}</td>
<td></td>
<td>z</td>
<td></td>
</tr>
<tr>
<td>{b}</td>
<td>{2}</td>
<td></td>
<td>β</td>
</tr>
</tbody>
</table>

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

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

3. Hypertuples and hyperrelations

Suppose that $D_q$ is a labelled class. The idea of [2,19] was to collect, if possible, across the elements of $D_q$ 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 [1].

This leads to the following definition [18]: We call an element $r \in \mathcal{L}$ equi-labelled with respect to the class $D_q$, if

$$\downarrow r \cap D \neq \emptyset$$  \hspace{1cm} (3.1)

and

$$\downarrow r \cap D \subseteq D_q.$$  \hspace{1cm} (3.2)

These two conditions express two fundamental principles in machine learning: (3.1) says that $r$ is supported by some $d \in D$; 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 \( D \) are in the
unique class \( D_q \). We denote the set of all elements equilabelled with respect to
\( D_q \) by \( \mathcal{E}_q \), and let \( \mathcal{E} = \bigcup_{q \leq K} \mathcal{E}_q \) be the set of all equilabelled elements. Note that
\( D \subseteq \mathcal{E} \), and that
\[ q, r \leq K, q \neq r \implies \mathcal{E}_q \cap \mathcal{E}_r = \emptyset, \quad (3.3) \]
since \( \{D_0, \ldots, D_K\} \) partitions \( D \). Furthermore,
\[ \mathcal{E}_q \cap \mathcal{E} = \mathcal{E}_q. \quad (3.4) \]

If \( D \subseteq P \subseteq \mathcal{L} \), we let \( E(P) = \{ t \in \mathcal{L} : t \text{ is maximal in } [P] \cap \mathcal{E} \} \), and set
\( VSP = \bigcup \{ E(P) : D \subseteq P \subseteq V \} \). 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 \( D \subseteq P \subseteq Q \subseteq \mathcal{L} \). Then,

1. \( E(P) \leq E(Q) \).
2. \( E(P) \leq E(Q) \).

**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. \( \square \)

**Theorem 3.2.** \( \min VSP = E(D) \), \( \max VSP = E(V) \).

**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 \( D \subseteq P \subseteq V \) such
that \( t \in E(P) \). Since \( E(D) \leq E(P) \) 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 \( t \in E(D) \) and \( s \leq t \) is minimal in \( VSP \), then \( s \in E(D) \) as well. Since the
elements of \( E(D) \) are pairwise incomparable, we have \( s = t \). \( \square \)

In the sequel, we will denote \( E(D) \) by \( \mathcal{S} \), and \( E(V) \) by \( \mathcal{G} \), since they corre-
pond, respectively, to the specific and general boundaries of \( D \) in the sense of
\( \mathcal{T} \); we also set \( \mathcal{S}_q = E_q(D) \) and \( \mathcal{G}_q = E_q(V) \). By Lemma 3.1, \( \mathcal{S} \) is the least \( E \)-set
and \( \mathcal{G} \) is the greatest \( E \)-set.

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

**Algorithm (LM algorithm).**

1. \( H_0 \overset{\text{def}}{=} P \).
2. \( H_{k+1} \overset{\text{def}}{=} \text{The set of maximal elements of } [\downarrow (H_k + P)] \cap \mathcal{E}. \)
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 Fig. 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.

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

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

Fig. 1. A dataset and its specific boundary. Each circle or cross is a simple tuple, and each rectangle represents a hypertuple.
Proof. We show that there is some \( x \in D \) such that \( \downarrow (t + x) \cap D = \{ x \} \); then, \( t + t + x \in \delta \). Let \( t \in V \), and set \( M = \{ t + x : x \in D \} \). Suppose that \( x \in D \) such that \( t + x \) is minimal in \( M \) with respect to \( \leq \). Let \( y \in D \) 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 \( y(a) = t(a) \subseteq t(a) \cup x(a) \). Hence, \( t + y \leq t + x \), contradicting the minimality of \( t + x \) in \( M \). \[ \square \]

Since each element of \( \delta \) is covered by some element of \( G \), and the sets \( G_i \), \( i \leq K \), partition \( G \), we obtain

Corollary 3.4. For each \( t \in V \), there is some \( i \leq K \) such that \( t \leq G_i \).

Observe that such \( G_i \) 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

\[
m'(t, n, G) = |\{ h \in G_n : t \leq h \}| \quad \text{for each } n \leq K, \tag{3.5}\]

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

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

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

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

This motivates us to distinguish between different elements in \( V \) given \( H = E(P) : t \in V \) is called primary if there is \( H_q \) with \( t \leq H_q \), \( t \) 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

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

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

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

and
\[ m'_i(t, n, H) = \{|h \in H_n : t + h \in \mathcal{E}\}| \quad \text{for each } n \leq K, \quad (3.11) \]
\[ m_s(t, H) = \max\{m'_i(t, n, H) : n \leq K\}, \quad (3.12) \]
\[ M_i(t, H) = \{i \leq K : m'_i(t, i, H) = m_s(t, H)\}. \quad (3.13) \]

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 \( D = \{a, b, c, d, e, f\} \) with
\[ a = \langle 0, 0, 0 \rangle, \quad b = \langle 1, 0, 0 \rangle, \quad c = \langle 1, 0, 1 \rangle, \quad d = \langle 1, 1, 1 \rangle, \]
\[ e = \langle 0, 1, 1 \rangle, \quad f = \langle 0, 1, 0 \rangle. \]

Suppose that \( a, b, e \) are coloured blue and \( d, e, f \) are coloured red. Then,
\[ E(D) = \{a + b, e, c + d, f\}. \]

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

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 \( q \in E(D) \), and thus, \( t \) is not secondary with respect to \( E(D) \). 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 G can be practically constructed from V by the LM algorithm, and Rule 2 can be applied if S (or E(P)) can be practically constructed from D (for D ⊆ P ⊆ V). In both cases we have to construct E-sets for P where D/C18 P/C18 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 D. The following algorithm finds a weak hypothesis H [17]:

\[
\begin{align*}
H & \leftarrow \emptyset \\
\text{for } q = 0 \text{ to } K \text{ do} \\
\quad X & \leftarrow D_q \\
\quad \text{while } X \neq \emptyset \text{ do} \\
\quad \quad \text{Order } X \text{ as } \langle g_0, \ldots, g_{m(X)} \rangle \\
\quad \quad h & \leftarrow g_0 \\
\quad \quad \text{for } i = 1 \text{ to } m(X) \text{ do} \\
\quad \quad \quad \text{if } h + g_i \text{ is equilabelled then} \\
\quad \quad \quad \quad h & \leftarrow h + g_i \\
\quad \quad \text{end if} \\
\quad \quad \text{end for} \\
\quad X & \leftarrow X \setminus \downarrow h \\
\quad H & \leftarrow H \cup \{h\} \\
\text{end while} \\
\text{end for}
\end{align*}
\]

The algorithm does not necessarily produce disjoint hypertuples: Let

\[
\begin{align*}
D_0 & = \{\langle 0, 1, 1 \rangle, \langle 1, 0, 1 \rangle, \langle 1, 1, 1 \rangle, \langle 1, 1, 0 \rangle, \langle 2, 1, 1 \rangle\}, \\
D_1 & = \{\langle 2, 0, 1 \rangle, \langle 0, 0, 0 \rangle\}.
\end{align*}
\]

Order X = D_0 by

\[
\begin{align*}
g_0 & = \langle 0, 1, 1 \rangle, \\
g_1 & = \langle 1, 0, 1 \rangle, \\
g_2 & = \langle 1, 1, 1 \rangle, \\
g_3 & = \langle 1, 1, 0 \rangle, \\
g_4 & = \langle 2, 1, 1 \rangle.
\end{align*}
\]

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 $D_q$) is $O(|D_q|^2)$. Therefore the worst case complexity for building $H$ (the whole hypothesis) is $O(K \times |D_q|^2)$, where $K$ is the number of classes.

Let $H = \bigcup_{i \leq K} H_i$ be a weak hypothesis for $D$, where $H_i = \{h_0, \ldots, h_{n(i)}\}$ 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 Eqs. (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. Mitchells version space

The situation in [9] can be described as a special decision system where $d : U \rightarrow \{0, 1\}$, 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 $a \in \Omega$

$$|t(a)| \leq 1 \text{ or } t(a) = V_a.$$  \hspace{1cm} (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 $V \subseteq \mathcal{H}$. 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. $s \in V$,
2. $s \leq t$,

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

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

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
333 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.

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

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

338 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 \(\langle D, d \rangle\), if

\[
\downarrow t \cap D = D_1.
\]  

(4.2)

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

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

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

\[
\mathcal{G}^m = \max \text{VSp}^m.
\]

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

\[
\mathcal{S}^m = \min \text{VSp}^m.
\]

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

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

5. **Expressive power of version space and boolean reasoning**

357 A simple tuple \(t\) can be regarded as a conjunction of descriptors

\[
\langle a_0, t(a_0) \rangle \land \langle a_1, t(a_1) \rangle \land \cdots \land \langle a_T, t(a_T) \rangle.
\]

359 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,

\[
s \in \text{sat}(t) \iff (\forall i \in I) s(a_i) = t(a_i),
\]

363 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. For such a simple dataset, there is no consistent hypothesis in the sense of (4.2).

A hypothesis in $V^m$ 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 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 $V$ with $+$ oper-

<table>
<thead>
<tr>
<th>Sky</th>
<th>ATemp</th>
<th>Humid</th>
<th>Wind</th>
<th>Water</th>
<th>FCast</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Warm</td>
<td>Normal</td>
<td>Strong</td>
<td>Warm</td>
<td>Same</td>
<td>1</td>
</tr>
<tr>
<td>Sunny</td>
<td>Warm</td>
<td>High</td>
<td>Strong</td>
<td>Warm</td>
<td>Same</td>
<td>1</td>
</tr>
<tr>
<td>Rainy</td>
<td>Cold</td>
<td>High</td>
<td>Strong</td>
<td>Warm</td>
<td>Change</td>
<td>0</td>
</tr>
<tr>
<td>Sunny</td>
<td>Warm</td>
<td>High</td>
<td>Strong</td>
<td>Cool</td>
<td>Change</td>
<td>1</td>
</tr>
</tbody>
</table>

Table 3
Limitation of version space

<table>
<thead>
<tr>
<th>Sky</th>
<th>ATemp</th>
<th>Humid</th>
<th>Wind</th>
<th>Water</th>
<th>FCast</th>
<th>d</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sunny</td>
<td>Warm</td>
<td>Normal</td>
<td>Strong</td>
<td>Cool</td>
<td>Change</td>
<td>1</td>
</tr>
<tr>
<td>Cloudy</td>
<td>Warm</td>
<td>Normal</td>
<td>Strong</td>
<td>Cool</td>
<td>Change</td>
<td>1</td>
</tr>
<tr>
<td>Rainy</td>
<td>Warm</td>
<td>Normal</td>
<td>Strong</td>
<td>Cool</td>
<td>Change</td>
<td>0</td>
</tr>
</tbody>
</table>
ator \in \mathcal{V}. Since we are only interested in the finite sums of elements of \mathcal{V} we will from now on assume that each \( V_a \) is finite.

Suppose that 
\[
t = \{ (m^a_0, m^a_1, \cdots, m^a_t(a)) \}_{a \in \Omega}
\]

is a hypertuple. We interpret \( t \) as a conjunction of disjunctions of descriptors

\[
\bigwedge_{a \in \Omega} \left( \langle a, m^a_0 \rangle \lor \cdots \lor \langle a, m^a_t(a) \rangle \right).
\]  

By the distributivity of \( \land \) and \( \lor \) 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,

\[
\langle (a_0, t^0_0) \land (a_1, t^0_1) \rangle \lor \langle (a_0, t^1_0) \land (a_1, t^1_1) \rangle.
\]

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 \( D \) 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 \( 2^{\mid V_x \mid} \) different subsets altogether. As a result there are \( \prod_{x \in \Omega} 2^{\mid V_x \mid} \) different hypertuples. Since a hypothesis is a set of hypertuples (i.e., a hyperrelation), there are \( 2^{\prod_{x \in \Omega} 2^{\mid V_x \mid}} \) 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 \( \mid V_x \mid + 2 \) different values, and \( \prod_{x \in \Omega} (\mid V_x \mid + 2) \) different tuples. In his conjunctive hypothesis representation, each hypothesis is a (simple) tuple, so there are \( \prod_{x \in \Omega} (\mid V_x \mid + 2) \) distinct hypotheses. In his disjunctive hypothesis representation, each hypothesis is a set of (simple) tuples (i.e., simple relation), so there are \( 2^{\prod_{x \in \Omega} (\mid V_x \mid + 2)} \) 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 \( \mathcal{E}^m = \{ \gamma(t) : t \in \mathcal{E}, D \models t \} \), where \( \gamma \) is an operation to turn a hypertuple into a simple tuple in such a way that

\[
\gamma(t) = \langle t_0, t_1, \ldots, t_T \rangle, \text{ where }\]

\[
t_i = \begin{cases} t(x_i), & \text{if } |t(x_i)| = 1, \\ ?, & \text{otherwise.} \end{cases}
\]
Note that \( t(x_i) \) is the projection of tuple \( t \) onto its \( x_i \) attribute.

- Each hypothesis is an element of \( E^m \).
- 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 \( S^m = \{ S^m_0, S^m_1 \} \), where \( S^m_0 \) is the specific boundary for the negative class and \( S^m_1 \) is the specific boundary for the positive class. Similarly \( G^m = \{ G^m_0, G^m_1 \} \).

6. Experimental

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 caseextraction 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

\(^1\) 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).
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

<table>
<thead>
<tr>
<th>Dataset</th>
<th>#Attr.</th>
<th>#Train</th>
<th>#Test</th>
<th>Class. Accuracy (%)</th>
<th>%PP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>C5.0</td>
<td>GLM/SR</td>
</tr>
<tr>
<td>Annealing</td>
<td>38</td>
<td>798</td>
<td>CV-5</td>
<td>96.6</td>
<td>96.4</td>
</tr>
<tr>
<td>Australian</td>
<td>14</td>
<td>690</td>
<td>CV-5</td>
<td>90.6</td>
<td>95.1</td>
</tr>
<tr>
<td>Auto</td>
<td>25</td>
<td>205</td>
<td>CV-5</td>
<td>70.7</td>
<td>82.4</td>
</tr>
<tr>
<td>Diabetes</td>
<td>8</td>
<td>768</td>
<td>CV-5</td>
<td>72.7</td>
<td>70.7</td>
</tr>
<tr>
<td>German</td>
<td>20</td>
<td>1000</td>
<td>CV-5</td>
<td>71.7</td>
<td>72.6</td>
</tr>
<tr>
<td>Glass</td>
<td>9</td>
<td>214</td>
<td>CV-5</td>
<td>80.4</td>
<td>86.6</td>
</tr>
<tr>
<td>Heart</td>
<td>13</td>
<td>270</td>
<td>CV-5</td>
<td>77.0</td>
<td>81.9</td>
</tr>
<tr>
<td>Hepatitis</td>
<td>19</td>
<td>155</td>
<td>CV-5</td>
<td>80.6</td>
<td>82.9</td>
</tr>
<tr>
<td>Horse-Colic</td>
<td>22</td>
<td>368</td>
<td>CV-5</td>
<td>85.1</td>
<td>82.7</td>
</tr>
<tr>
<td>Iris</td>
<td>4</td>
<td>150</td>
<td>CV-5</td>
<td>94.7</td>
<td>93.1</td>
</tr>
<tr>
<td>Monk-1</td>
<td>6</td>
<td>124</td>
<td>432</td>
<td>74.3</td>
<td>100.0</td>
</tr>
<tr>
<td>Monk-2</td>
<td>6</td>
<td>169</td>
<td>432</td>
<td>65.1</td>
<td>81.4</td>
</tr>
<tr>
<td>Monk-3</td>
<td>6</td>
<td>122</td>
<td>432</td>
<td>97.2</td>
<td>93.8</td>
</tr>
<tr>
<td>Sonar</td>
<td>60</td>
<td>208</td>
<td>CV-5</td>
<td>71.6</td>
<td>81.6</td>
</tr>
<tr>
<td>TTT</td>
<td>9</td>
<td>958</td>
<td>CV-5</td>
<td>86.2</td>
<td>96.2</td>
</tr>
<tr>
<td>Vote</td>
<td>18</td>
<td>232</td>
<td>CV-5</td>
<td>96.5</td>
<td>96.5</td>
</tr>
<tr>
<td>Wine</td>
<td>13</td>
<td>178</td>
<td>CV-5</td>
<td>94.3</td>
<td>99.0</td>
</tr>
<tr>
<td>Average</td>
<td></td>
<td></td>
<td></td>
<td>82.7</td>
<td>87.8</td>
</tr>
</tbody>
</table>

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.
443 separate version space is learned for each positive training example, then new
444 instances are classified by combining the votes of these different version spaces.
445 In this paper we investigate version spaces in a more expressive hypothesis
446 space—disjunction of conjunctions of disjunctions, where each hypothesis is a
447 set of hypertuples. Without a proper inductive bias the version space is unin-
448 teresting. We show that, with $E$-set as an inductive bias, this version space is a
449 generalisation of Mitchell’s original version space, which employs a different
450 type of inductive bias.
451 For classification within the version space we proposed three classification
452 rules for use in different situations. The first two rules are based on $E$-sets as
453 hypotheses, and they can be applied when the data space ($V$) is finite and small.
454 We showed that constructing $E$-sets is computationally expensive, so we
455 introduced the third rule which is based on weak hypotheses. We presented an
456 algorithm to construct weak hypotheses efficiently.
457 Experimental results show that this classification approach performs ex-
458 tremely well on primary data, which account for over 75.0% of all data. Over
459 all data this classification approach is comparable to C5.0.

460 8. Uncited reference

461 [7]

462 Acknowledgements

463 The work of Hui Wang was partly supported by the European Commission
464 project ICONS, project no. IST-2001-32429.

465 Appendix A. Notation

466 $X \leq Y \iff (\forall x \in X)(\exists y \in Y) x \leq y$
467 $X \leq Y \iff (\forall y \in Y)(\exists x \in X) x \leq y$
468 $U = \{x_0, \ldots, x_N\}$
469 $\Omega = \{a_0, \ldots, a_T\}$
470 $V = \prod_{\alpha \in \Omega} V_\alpha$
471 $\mathcal{L} = \prod_{\alpha \in \Omega} 2^{\mathcal{V}_\alpha}$
472 $I(x) = \langle a_0(x), a_1(x), \ldots, a_T(x) \rangle$
473 $D = \{I(x) : x \in U\} = D_0 \cup D_1 \cup \ldots \cup D_K$
474 $\mathcal{E}_q = \text{set of all elements equilabelled with respect to } D_q$
475 $\mathcal{E} = \bigcup_{q \leq K} \mathcal{E}_q$
\[ E(P) = \{ t \in \mathcal{L} : t \text{ is maximal in } [P] \cap \mathcal{E} \} \]
\[ E_q(P) = E(P) \cap \mathcal{E}_q \]
\[ \mathcal{S} = E(D) \]
\[ \mathcal{G}_q = E_q(D) \]
\[ \mathcal{G} = E(V) \]
\[ \mathcal{G}_q = E_q(V) \]
\[ \text{VSp} = \bigcup \{ E(P) : D \subseteq P \subseteq V \} \]

### References
