

---

# Feature Construction with Version Spaces for Biochemical Applications

---

Stefan Kramer  
Luc De Raedt

SKRAMER@INFORMATIK.UNI-FREIBURG.DE  
DERAEDT@INFORMATIK.UNI-FREIBURG.DE

Institut für Informatik, Albert-Ludwig-University Freiburg, Georges Koehler Allee 79, 79110 Freiburg, Germany

## Abstract

A novel conceptual and practical framework for feature construction is introduced. The user can declaratively specify the features of interest using a conjunctive query. The query may involve various constraints over the frequency of the features (e.g. being frequent on the positives and infrequent on the negatives) and the generality of the features. The solutions to the query are organized in a version space, and are computed using an integration of the version space algorithm and the levelwise algorithm in data mining. The resulting features can then be used by traditional machine learning algorithms.

To experimentally validate the framework, we focus on feature construction within chemoinformatics applications, where the features to be constructed are linear fragments. Linear fragments are sequences of linearly connected atoms.

## 1. Introduction

In the past few years, both machine learning and inductive logic programming have devoted a lot of attention to feature construction (Srinivasan et al., 1996; Kramer, 1999; Dehaspe & Toivonen, 1999; Alphonse & Rouveirol, 2000; Kramer, Lavrač & Flach, 2001). Feature construction in inductive logic programming is important for practical as well as theoretical reasons. Firstly, adequate feature construction techniques may contribute new insights into the relation among relational representations and propositional ones. Secondly, efficient and effective feature construction techniques combined with attribute-value learning may outperform pure inductive logic programming techniques. There exists some recent evidence that this may well be the case. For instance, in the PTE-2 challenge (Srinivasan, King & Bristol, 1999), the best pre-

dictors employed a set of features induced using the Warmr system (Dehaspe & Toivonen, 1999). Warmr basically discovers all frequent Datalog queries within a specified language. These queries can then be employed as boolean features in a propositional learner.

Despite the existence of various feature construction techniques (Srinivasan & King, 1999; Dehaspe & Toivonen, 1999), there exists so far neither a common framework for feature construction, nor a good understanding of the relative advantages of the different feature construction techniques. Indeed, various open questions exist. For instance, should the class information be taken into account (as in (Srinivasan & King, 1999)) or not (as in (Dehaspe & Toivonen, 1999)) during feature construction? How many features should one construct? What if the features are mutually dependent? Some learners, such as e.g. support vector machines, can cope with a large number of features. Others, such as e.g. naive Bayes or logistic regression, have difficulties dealing with a large number of mutually dependent features.

We present a common conceptual framework for studying these issues. It is based on an extension of the version space model that we introduced elsewhere (De Raedt & Kramer, 2001). Within this model, it is possible to declaratively specify the features of interest. The specification takes the form of a set of constraints concerning the frequency of the features on (possibly different) datasets as well as on the generality and syntax of the features. Using our framework, one can e.g. specify that the features should cover at least 5 percent of the positive examples, should cover at most 10 percent of the negatives, and should be more general than a specified feature. The space of all features satisfying the constraints takes the form of a version space. Elsewhere, we have given efficient algorithms for computing the borders of the version space. The algorithms are based on a tight integration of the levelwise algorithm by (Manilla & Toivonen, 1997) and Mellish' description identification algorithm (Mellish,

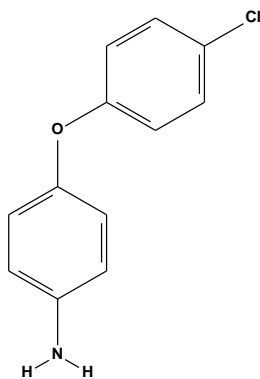


Figure 1. Example compound in a 2-D representation.  $'cl-c \sim c \sim c \sim c - o'$  is an example fragment occurring in the molecule. (Benzene rings consist of six carbon atoms connected by aromatic bonds.)

1990), that extends Mitchell’s version space algorithm (Mitchell, 1982).

In order to validate the framework, we focus on applications in bio- and chemoinformatics. We present an implementation as well as experiments where the constructed features are *molecular fragments*. Molecular fragments are sequences of linearly connected atoms. They are useful and important for the induction of so-called Structure-Activity Relationships (SARs), which are statistical models that relate chemical structure to biological activity. The use of automatically derived fragments in SARs originates from the CASE/MultiCASE systems developed by (Rosenkranz et al., 1999). With more than 150 published references, the CASE/MultiCASE systems are the most extensively used SAR and predictive toxicology systems.

The search for interesting molecular substructures has also been studied within the context of inductive logic programming and knowledge discovery in databases. For instance, Warmr (Dehaspe & Toivonen, 1999) or the approach by Inokuchi, Washio and Motoda (2000) have been used in this context. Warmr is a system discovering frequently succeeding Datalog queries, and thus is not restricted to fragments. The approach by Inokuchi *et al.* (2000) deals with arbitrary frequent subgraphs, and thus is not restricted to linear fragments. Both approaches differ in that the derived features are more expressive. However, finding the features is computationally much more expensive and complex than for linear fragments.<sup>1</sup> Linear fragments were introduced by (Kramer & Frank, 2000). All of these approaches can only deal with a minimum fre-

<sup>1</sup>Testing whether a linear fragment occurs in molecule seems to be polynomial.

quency constraint on a single dataset. In contrast, we also deal with maximum frequency thresholds and multiple data sets.

The paper is organized as follows: in Section 2, we introduce the molecular fragment finding task and the primitives for querying such fragments, in Section 3, we show that the space of solutions to a query is essentially a version space, in Section 4, we discuss how to use the resulting framework for feature construction, in Section 5, we report on various experiments with the constructed features in three biochemical applications, and in Section 6, we conclude.

## 2. Framework

### 2.1 Molecular fragments

A *molecular fragment* is defined as a sequence of linearly connected atoms. For instance,  $'o - s - c'$  is a fragment meaning: “an oxygen atom with a single bond to a sulfur atom with a single bond to a carbon atom”. In such expressions  $'c'$ ,  $'n'$ ,  $'cl'$ , etc. denote elements, and  $'-'$  denotes a single bond,  $'='$  a double bond,  $'\#'$  a triple bond, and  $'\sim'$  an aromatic bond. As common in the literature, we only consider “heavy” (i.e., non-hydrogen) atoms in this paper.

We assume that the system is given a database of example compounds and that each of the example compounds in the database is described using a 2-D representation. The information given there consists of the elements of the atoms of a molecule and the bond orders (single, double, triple, aromatic). An example compound in such a representation is shown in Fig. 1.

A molecular fragment  $f$  covers an example compound  $e$  if and only if  $f$  considered as a graph is a subgraph of example  $e$ . For instance, fragment  $'cl - c \sim c \sim c \sim c - o'$  covers the example compound in Fig. 1.

There are a number of interesting properties of the language of molecular fragments  $\mathcal{M}$ :

- fragments in  $\mathcal{M}$  are partially ordered by the *is more general than* relation; when fragment  $g$  is more general than fragment  $s$  we will write  $g \leq s$ ;
- within this partial order, two syntactically different fragments are equivalent only when they are a reversal of one another; e.g.  $'c - o - s'$  and  $'s - o - c'$  denote the same substructure;
- $g \leq s$  if and only if  $g$  is a subsequence of  $s$  or  $g$  is a subsequence of the reversal of  $s$ ; e.g.  $'c - o' \leq 'c - o - s'$ .

Note that the representation of molecular fragments

is relatively restricted compared to some other representations employed in data mining, such as first-order queries (Dehaspe & Toivonen, 1999) or sub-graphs (Inokuchi, Washio & Motoda, 2000). Although fragments are a relatively restricted representation of chemical structure, it is easy for trained chemists to recognize the functional group(s) that a given fragment occurs in. Thus, the interpretation of a fragment reveals more than meets the eye.

## 2.2 Constraints on fragments

The features that will be constructed can be declaratively specified using a conjunction of primitive constraints  $c_1 \wedge \dots \wedge c_n$ . The primitive constraints  $c_i$  that can be imposed on the unknown target fragments  $f$  are:

- $f \leq p$ ,  $p \leq f$ ,  $\neg(f \leq p)$  and  $\neg(p \leq f)$ : where  $f$  is the unknown target fragment and  $p$  is a specific pattern; this type of primitive constraint denotes that  $f$  should (not) be more specific (general) than the specified fragment  $p$ ; e.g. the constraint ' $c - o' \leq f$ ' specifies that  $f$  should be more specific than ' $c - o'$ ', i.e. that  $f$  should contain ' $c - o'$ ' as a subsequence;
- $freq(f, D)$  denotes the relative frequency of a fragment  $f$  on a set of molecules  $D$ ; the relative frequency of a fragment  $f$  w.r.t. a dataset  $D$  is defined as the percentage of molecules in  $D$  that  $f$  covers;
- $freq(f, D_1) \leq t$ ,  $freq(f, D_2) \geq t$  where  $t$  is a positive real number and  $D_1$  and  $D_2$  are sets of molecules; this constraint denotes that the relative frequency of  $f$  on the dataset  $D_i$  should be larger than (resp. smaller than) or equal to  $t$ ; e.g. the constraint  $freq(f, Pos) \geq 0.95$  denotes that the target fragments  $f$  should have a minimum relative frequency of 95 % on the set of molecules  $Pos$ .

These primitive constraints can now conjunctively be combined in order to declaratively specify the target fragments of interest. Note that the conjunction may specify constraints w.r.t. any number of datasets, e.g. imposing a minimum frequency on a set of active molecules, and a maximum one on a set of inactive ones. E.g. the following constraint :

$$'c - o' \leq f) \wedge \neg(f \leq 'c - o - s - c - o - s')$$

$$freq(f, Act) \geq 0.95 \wedge freq(f, InAct) \leq 0.05$$

queries for all fragments that include the sequence ' $c - o'$ ', are not a subsequence of ' $c - o - s - c - o - s'$ ', have

a frequency on *Act* that is larger than 95 percent and a frequency on *InAct* that is smaller than 5 percent.

## 3. Solving constraints

In this section, we show that the solutionspace  $sol(c_1 \wedge \dots \wedge c_n)$  in  $\mathcal{M}$  for a conjunctive constraint  $c_1 \wedge \dots \wedge c_n$  is a version space and can therefore be represented by its borders.

Due to the fact that the primitive constraints  $c_i$  are independent of one another, it follows that

$$sol(c_1 \wedge \dots \wedge c_n) = sol(c_1) \cap \dots \cap sol(c_n)$$

So, we can find the overall solutions by taking the intersection of the primitive ones.

Secondly, each of the primitive constraints  $c$  is monotonic or anti-monotonic w.r.t. generality (cf. (Manilla & Toivonen, 1997)). A constraint  $c$  is *anti-monotonic* (resp. *monotonic*) w.r.t. generality whenever

$$\forall s, g \in M : (g \leq s) \wedge (s \in sol(c)) \rightarrow (g \in sol(c))$$

(resp.  $(g \in sol(c)) \rightarrow (s \in sol(c))$ ). The basic anti-monotonic constraints in our framework are:  $(f \leq p)$ ,  $freq(f, D) \geq m$ , the basic monotonic ones are  $(p \leq f)$ ,  $freq(f, D) \leq m$ . Furthermore the negation of a monotonic constraint is anti-monotonic and vice versa.

Monotonic and anti-monotonic constraints are important because their solution space is bounded by a border. This fact is well-known in both the data mining literature (cf. (Manilla & Toivonen, 1997)), where the borders are often denoted by  $BD^+$ , as well as the machine learning literature (cf. (Mitchell, 1982)), where the symbols  $S$  and  $G$  are typically used.

To define borders, we need the notions of minimal and maximal elements of a set w.r.t. generality. Let  $F$  be a set of fragments, then define

$$min(F) = \{f \in F \mid \neg \exists q \in F : q \leq f\}$$

$$max(F) = \{f \in F \mid \neg \exists q \in F : f \leq q\}$$

We can now define the borders  $S(c)$  and  $G(c)$ <sup>2</sup> of a primitive constraint  $c$  as

$$G(c) = min(sol(c)) \text{ and } S(c) = max(sol(c))$$

<sup>2</sup>At this point, we will follow Mitchell's terminology, because he works with two dual borders (a set of maximally general solutions  $G$  and a set of maximally specific ones  $S$ ). In data mining, one typically only works with the  $S$ -set.

Anti-monotonic constraints  $c$  will have  $G(c) = \{\top\}$  and for proper constraints  $S(c) \neq \{\perp\}$ ; proper monotonic constraints have  $S(c) = \{\perp\}$  and  $G(c) \neq \{\top\}$ . Furthermore, as in Mitchell’s version space framework we have that

$$sol(c) = \{f \in \mathcal{M} \mid \exists s \in S(c), \exists g \in G(c) : g \leq f \leq s\}$$

This last property implies that  $S(c)$  (resp.  $G(c)$ ) are proper borders for anti-monotone (resp. monotone) constraints.

So, we have that the set of solutions  $sol(c_i)$  to each primitive constraint is a simple version space completely characterized by  $S(c_i)$  and  $G(c_i)$ . Therefore, the set of solutions  $sol(c_1 \wedge \dots \wedge c_n)$  to a conjunctive constraint  $c_1 \wedge \dots \wedge c_n$  will also be completely characterized by the corresponding  $S(c_1 \wedge \dots \wedge c_n)$  and  $G(c_1 \wedge \dots \wedge c_n)$ .

Elsewhere (De Raedt & Kramer, 2001; De Raedt, 2000), we have presented algorithms for computing the  $S$  and  $G$  sets corresponding to the constraints. The algorithm basically integrates the levelwise algorithm by (Manilla & Toivonen, 1997) with Mellish’s description identification algorithm. In principle, one might also employ Hirsh’s version space merging algorithm (Hirsh, 1994).

## 4. A framework for feature construction

The above presented framework for molecular fragment finding will now be used for feature construction. To realize this, one has to specify a conjunctive query. The solutions to the query then form the features of interest. In the remainder of this section, we consider various types of queries, each of which will construct a different set of features. We will assume that the features are constructed with the aim of classification, and we will also assume that there are two sets of examples:  $P$  which corresponds to the positives, and  $N$  which corresponds to the negatives.<sup>3</sup> Furthermore,  $E = P \cup N$ .

### 4.1 A minimum frequency threshold

The minimum frequency threshold query is of the following form  $freq(f, E) \geq \epsilon$  where  $\epsilon$  is a constant. In other words, one is interested in those fragments that frequently occur in the whole dataset. The reason for imposing a minimum frequency threshold is that one is not interested in features that only seldomly occur in the data, because they are likely to be uninteresting.

<sup>3</sup>We wish to stress that it is easy to generalize the techniques for  $n$  classes. Also, some of the techniques can also be generalized for other learning tasks such as e.g. regression.

As one extreme, consider fragments that do not occur in the data. It should also be clear that by increasing  $\epsilon$  one will obtain fewer fragments and that decreasing  $\epsilon$  will result in more fragments. So,  $\epsilon$  allows one to tune the number of constructed features.

The minimum frequency approach has been pursued in the PTE-2 challenge using the Warmr system.

A variant of this type of query would construct features with a minimum frequency on the positives together with those with a minimum frequency on the negatives. This variant will not be pursued any further in this paper.

### 4.2 A minimum and a maximum frequency threshold

One of the problems with imposing only a minimum frequency threshold is that one may obtain features that hold for almost all of the examples. These are likely to be as uninteresting as those that only seldomly occur. Therefore it may well be more appropriate to use a query imposing both a minimum and a maximum frequency. Such queries take the following form  $(freq(f, E) \leq \delta) \wedge (freq(f, E) \geq \epsilon)$ , where  $\epsilon$  and  $\delta$  are real numbers between 0 and 1. It seems reasonable to choose  $\delta = 1 - \epsilon$ .

### 4.3 Using positive and negative examples

One drawback of the previous two settings is that the information about the class of the examples is ignored. Other works such as (Srinivasan & King, 1999) have employed class information during the feature construction task. Class information can also be adopted in our framework using the following type of query  $(freq(f, P) \geq \epsilon) \wedge (freq(f, N) \leq \delta)$ . One then finds those fragments that are frequent on the positives and infrequent on the negatives. Alternatively, one could work with  $(freq(f, P) \leq \epsilon) \wedge (freq(f, N) \geq \delta)$ . Features that are a solution to these types of query are likely to be more predictive with regard to the class information.

The choice of  $\epsilon$  and  $\delta$  is essentially free. E.g. one could choose these parameters in such a way that the resulting fragments are significant w.r.t. the class distributions. For instance, in many chemical applications it can be assumed that interesting fragments have a frequency of less than 25 (here counted in absolute numbers rather than in percentages) in the positive, active compounds. Setting the minimum frequency in the actives to 6, 10, 16 and 20, respectively, one can apply the  $\chi^2$ -Test to a  $2 \times 2$  contingency table with the class as one variable and the occurrence of the

fragment as the other one to determine the maximum allowable frequency in the inactive compounds.

#### 4.4 Mutually dependent features

One drawback of using all features that satisfy a conjunctive query is that many features will be interrelated. Whereas some machine learning systems, such as support vector machines, can cope with such mutually dependent features, this is problematic for many other algorithms such as nearest neighbor, naive bayes, and logistic regression. If the features are to be used by these last class of algorithms, it is best to filter out the resulting fragments. The version space representation actually offers a natural way to realize this. Indeed, if one restricts one’s attention to only the elements of the border sets, then many of the dependencies will automatically be removed. The reason is that within the space of solutions to the constraints, the border elements occupy extreme positions. This is easily seen by taking a chain of fragments  $g \leq f_1 \leq \dots \leq f_n \leq s$  such that  $g \in G$  and  $s \in S$ . Here,  $f_i$  is likely to be quite similar to  $f_{i+1}$  and  $f_{i-1}$ . However, within this chain,  $g$  and  $s$  are maximally different.

Therefore, we will distinguish *extensional* from *intensional* feature construction. The former will employ all features in the version space, the latter will only employ the extreme elements.

## 5. Experimental Results

In order to validate the described approach, we performed experiments in three real-world domains: carcinogenicity prediction (Srinivasan, King & Bristol, 1999) (the PTE-2 dataset), mutagenicity prediction (Srinivasan et al., 1996) and biodegradability prediction (Džeroski et al., 1999). For biodegradation prediction, we used a two-class version (degradable or not) with a half-life time (HLT) threshold of 4 weeks.

In all of the domains, we only used 2-D information and some global properties. The 2-D information includes just the elements and the bond orders (single, double, triple and aromatic). No other information is used. In particular, we did not use any partial charges, atom types, functional groups or the like. In addition to plain 2-D information, we included the results of the Ames test for the PTE data, the LUMO and logP values for mutagenicity, and the molecular weight and the logP for biodegradability.

The general set-up of the experiments is that we first construct features using the feature construction method outlined above. We then feed these features into propositional learners, available in the Weka work-

bench (Witten & Frank, 1999). In particular, we employed C4.5, PART (Frank & Witten, 1998), logistic regression and linear support vector machines.

All the results were obtained using proper 10-fold cross-validation. In other words, the structural features were constructed given all examples but those from the resp. held-out set. Subsequently, we applied the resulting feature definitions to the respective held-out set.

We varied the minimum/maximum frequency thresholds as follows: min = 3 %, max = 97 %; min = 5 %, max = 95 %; min = 10 %, max = 90 % and min = 20 % and max = 80 %. It should be clear that with the first parameter setting, a very large number of features is obtained. Increasing the minimum frequency and decreasing the maximum frequency threshold, we obtain less and less features. Three basic settings for feature construction were compared: using only the minimum frequency threshold *min* (as in Apriori (Agrawal, Imielinsk & Swami, 1993) and Warmr (Dehaspe & Toivonen, 1999)), using both a minimum and a maximum frequency threshold and the extensional version space *VSE*, and finally, using both a minimum and a maximum frequency threshold and the intensional version space *VSI* only.

Last but not least we present the results for class-sensitive feature construction, where we attempted to construct statistically significant features for the prediction task at hand. To implement this, we set the minimum absolute frequency on the positive examples and the maximum absolute frequency on the negative examples as described above (the minimum  $\in \{6, 10, 15, 20\}$  and the maximum set dependent on the class distribution). Given these four settings, we took the union of all resulting features for the learning experiments. For this approach, we present results only for the extensional and intensional version spaces.

In Table 1, we highlighted the best results in boldface. For PTE, the threshold was set to  $\geq 65$  %, for mutagenicity to  $\geq 93$  % and for biodegradability to  $\geq 78$  %.<sup>4</sup> In order not to overload the tables, we left out the standard deviations of the numbers of features. Since these were reasonably small, they would not add much to the presentation of the results.

Regarding the statistical significance of these results, as a rule of thumb, differences of around 5 % are signif-

<sup>4</sup>The best result in the literature for mutagenicity is 93 % (Sebag & Rouveiol, 1997), and the best result so far for the biodegradability domain, among a large number of learning algorithms, subsets of background knowledge and parameter settings, is 78 %. For some reference results, see (Kramer, 1999).

Table 1. Predictive accuracy results in standard ILP benchmark domains (carcinogenicity, mutagenicity and biodegradability prediction). *PART* is a rule learning algorithm (Frank & Witten, 1998), *Log.* means Logistic Regression and *l. SVM* means linear Support Vector Machines. In italics, the numbers of constructed features are given. Interesting results are highlighted in boldface.

min. = 3 %, max. = 97 %				min. = 10 %, max. = 90 %			
	min.	VSE	VSI		min.	VSE	VSI
PTE	<i>355.9</i>	<i>354.9</i>	<i>85.9</i>	PTE	<i>80.5</i>	<i>79.5</i>	<i>24.2</i>
C4.5	59.1	58.8	59.1	C4.5	60.8	57.6	60.8
PART	63.2	62.6	57.9	PART	<b>65.3</b>	<b>67.4</b>	62.8
Log.	55.4	55.8	61.7	Log.	62.9	62.0	60.8
l. SVM	62.9	62.0	59.1	l. SVM.	59.1	56.4	59.6
Mutag	<i>957.9</i>	<i>935.9</i>	<i>136.8</i>	Mutag	<i>255.7</i>	<i>243.7</i>	<i>41.0</i>
C4.5	89.4	89.4	83.5	C4.5	86.7	86.7	80.9
PART	91.0	91.0	87.2	PART	91.0	91.5	85.6
Log.	82.4	86.2	87.8	Log.	91.5	<b>94.7</b>	86.7
l. SVM	<b>93.6</b>	<b>94.7</b>	92.6	l. SVM	87.8	87.8	89.9
Biodeg	<i>212.9</i>	<i>211.9</i>	<i>62.9</i>	Biodeg	<i>52.6</i>	<i>51.6</i>	<i>20.6</i>
C4.5	77.1	77.1	74.0	C4.5	77.7	77.7	74.1
PART	72.6	72.6	72.3	PART	73.2	73.2	75.6
Log.	64.0	64.0	73.4	Log.	<b>80.2</b>	<b>80.2</b>	<b>79.0</b>
l. SVM	<b>79.3</b>	<b>79.3</b>	74.7	l. SVM	<b>80.2</b>	<b>80.2</b>	<b>81.1</b>
min. = 5 %, max. = 95 %				min. = 20 %, max. = 80 %			
PTE	<i>183.1</i>	<i>182.1</i>	<i>53.8</i>	PTE	<i>35.7</i>	<i>34.7</i>	<i>14.9</i>
C4.5	61.7	61.4	59.3	C4.5	64.3	63.2	61.7
PART	62.9	62.6	58.4	PART	63.5	64.7	62.9
Log.	62.6	62.3	58.5	Log.	63.2	63.5	<b>65.6</b>
l. SVM	61.1	60.5	57.6	l. SVM	64.4	62.6	<b>65.3</b>
Mutag	<i>564.0</i>	<i>542.0</i>	<i>88.4</i>	Mutag	<i>105.5</i>	<i>83.5</i>	<i>10.9</i>
C4.5	90.4	90.4	85.1	C4.5	90.4	90.4	88.8
PART	91.0	91.0	83.5	PART	91.0	91.0	91.5
Log.	67.0	69.7	87.8	Log.	90.4	91.5	91.0
l. SVM	<b>93.1</b>	<b>93.1</b>	89.4	l. SVM	89.4	89.4	89.4
Biodeg	<i>129.0</i>	<i>128.0</i>	<i>32.8</i>	Biodeg	<i>23.5</i>	<i>22.5</i>	<i>12.0</i>
C4.5	76.8	76.8	76.5	C4.5	77.1	77.1	75.6
PART	73.8	73.8	75.0	PART	77.1	77.1	75.6
Log.	76.2	76.5	<b>78.0</b>	Log.	77.4	77.4	75.6
l. SVM	<b>79.6</b>	<b>79.9</b>	<b>79.3</b>	l. SVM	75.0	75.0	72.9

icant (according to McNemar’s test) in the PTE and the biodegradability domains. For mutagenicity, significant results usually have a difference of around 7 %. In order to check the stability of the results with respect to different fold partitionings, we repeated a few experiments and found that the variation is relatively small (about one percentage point from partitioning to partitioning).<sup>5</sup>

In the following, we summarize the major observations regarding the techniques and parameter settings used, and regarding the domains.

<sup>5</sup>Note that all algorithms applied in this study are deterministic.

First, looking at the numbers of constructed features, we find that they vary widely from domain to domain. This is due to the heterogeneity of the datasets: Heterogeneous datasets (PTE and biodegradability) contain fewer frequent fragments than homogeneous datasets (the mutagenicity dataset of nitro compounds).

Secondly, when using more features, e.g. with “laissez-faire feature construction” (where min = 3% or min = 5%), we obtain excellent results only for Support Vector Machines (SVMs), since their overfitting protection is based on the notion of maximum margin hyperplanes and not on, e.g., syntactic complexity. Quite generally, the performance of SVMs tends to degrade

if the set of available descriptors is reduced (e.g., from *minimum frequency only* to *VSE*, further to *VSI*). In contrast, the performance of the classical statistical technique, Logistic Regression, tends to improve if the set of available features is getting smaller.

An exception to the above rule is the PTE domain, where “laissez-faire feature construction” does not seem to be a good idea. Once again, this shows the extreme hardness of the domain. As suspected in (Kramer, 1999), the “right” propositionalizations are apparently not yet found in this domain. Employing the Ames test together with a few relevant structural descriptors is the best we can do at the current state of the art. The more coarse-grained the propositionalization and the more focused the information given for learning, the better the predictive accuracy.

Constructing fewer features using the min = 20 %/max = 80% setting harms the performance of the Support Vector Machine algorithm, and does not help except for the PTE domain. As stated above, focusing improves the performance in this domain. Also, it is quite obvious from these figures, that “classical” Machine Learning and statistical algorithms such as C4.5 and PART benefit the most from restricting the number of features.

The latter phenomenon shows particularly in the results for *class-sensitive features construction* (see Table 2). The C4.5 results and especially the PART results are in fact the only competitive ones among those described in this paper. This suggests that class-sensitive feature construction is beneficial in conjunction with classical Machine Learning algorithms. This may be due to the “greedy nature” of many of these algorithms.

Thirdly, from Table 1 it follows that the difference between the extensional version space approach (*VSE*) and the one with minimum frequency only (*min*) is marginal. This is true for the predictive accuracy as well as for the number of derived features.

Fourthly, when comparing the extensional approach to the intensional one, the results for *VSE* are better for learners that can cope with mutually related attributes (such as SVM), whereas *VSI* is better for learners that have difficulties with such attributes. The differences are clearest for logistic regression. This seems to indicate that the intensional approach is well-suited to deal with mutually dependent features.

Finally, many of the above results (especially those highlighted in boldface) are competitive with regard to the state-of-the-art, if not better. This confirms that the presented approach is practically relevant. Fur-

Table 2. Predictive accuracy of class-sensitive feature construction in standard ILP benchmark domains.

class-sensitive	VSE	VSI
PTE	14.1	6.2
C4.5	<b>65.9</b>	<b>65.9</b>
PART	<b>65.9</b>	<b>65.9</b>
Log.	<b>65.3</b>	<b>65.3</b>
l. SVM	64.7	<b>65.0</b>
Mutag	330.2	95.0
C4.5	87.2	87.2
PART	<b>93.1</b>	<b>93.1</b>
Log.	<b>95.7</b>	<b>94.1</b>
l. SVM	92.0	90.1
Biodeg	51.4	35.8
C4.5	76.2	76.5
PART	<b>79.9</b>	<b>79.6</b>
Log.	75.9	75.3
l. SVM	75.9	75.6

thermore, the approach is quite efficient: e.g., a 10-fold cross-validation run for PTE with min. = 5 % and max. = 95 % takes around 30 minutes of CPU time on a LINUX PC with a Pentium III processor (450 MHz).

## 6. Conclusions

We have introduced a novel framework for feature construction. It is based on a querying model in which the relevant features can be specified declaratively. The version space of the resulting features can then efficiently be computed using the levelwise version space algorithm, that we presented elsewhere (De Raedt & Kramer, 2001).

The presented framework for feature construction is more general than those studied so far in the inductive logic programming community. Indeed, important features in this respect include: it can (but need not) take into the class information, it can influence the number of features, it can employ both an intensional and an extensional representation of the set of features.

The presented framework is also competitive in three important applications in the field of biochemistry. In all these applications, results have been obtained that are at least competitive with the best result to date, if not better.

Finally, we wish to stress that – although the technique was presented in the context of molecular fragments – it is possible to adapt it to other types of features.

## References

- E. Alphonse, C. Rouveirol. Lazy propositionalization for relational learning. In *Proceedings of the 14th European Conference on Artificial Intelligence*, IOS Press, 2000.
- R. Agrawal, T. Imielinski, A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of ACM SIGMOD Conference on Management of Data*, 1993.
- L. Dehaspe, H. Toivonen. Discovery of Frequent Datalog Patterns, in *Data Mining and Knowledge Discovery Journal*, Vol. 3, 1999.
- L. De Raedt. A Logical Database Mining Query Language. in *Proceedings of the 10th Inductive Logic Programming Conference*, Lecture Notes in Artificial Intelligence, Vol. 1866, Springer Verlag, 2000.
- L. De Raedt, S. Kramer. The levelwise version space algorithm and its application to molecular fragment finding. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, Morgan Kaufmann, 2001.
- S. Džeroski, H. Blockeel, B. Kompare, S. Kramer, B. Pfahringer, and W. Van Laer. Experiments in predicting biodegradability. in *Proceedings of the 9th International Workshop on Inductive Logic Programming (ILP-99)*, pages 80–91, Berlin Heidelberg New York, 1999. Springer.
- E. Frank, I.H. Witten. Generating Accurate Rule Sets Without Global Optimization. in *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-98)*, Morgan Kaufmann Publishers, San Francisco, CA, 1998.
- H. Hirsh. Generalizing Version Spaces. *Machine Learning*, Vol. 17(1): 5-46 (1994).
- A. Inokuchi, T. Washio, H. Motoda. An Apriori-based algorithm for mining frequent substructures from graph data. in D. Zighed, J. Komorowski, and J. Zyktow (Eds.) *Proceedings of PKDD 2000*, Lecture Notes in Artificial Intelligence, Vol. 1910, Springer-Verlag, 2000.
- S. Kramer. *Relational Learning vs. Propositionalization: Investigations in Inductive Logic Programming and Propositional Machine Learning*, PhD thesis, Vienna University of Technology, Vienna, Austria, 1999. <http://www.informatik.uni-freiburg.de/~skramer/phd.ps.gz>
- S. Kramer and E. Frank. Bottom-Up Propositionalization. in *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, 156-162, 2000.
- S. Kramer, N. Lavrač, P. Flach. Propositionalization Approaches to Relational Data Mining, to appear in: S. Džeroski, N. Lavrač (Eds.): *Relational Data Mining*, Springer Verlag, Berlin Heidelberg New York, 2001.
- H. Mannila and H. Toivonen. Levelwise search and borders of theories in knowledge discovery, *Data Mining and Knowledge Discovery*, Vol. 1, 1997.
- C. Mellish. The description identification algorithm. *Artificial Intelligence*, 1990.
- T. Mitchell. Generalization as Search, *Artificial Intelligence*, 1980.
- H.S. Rosenkranz, A.R. Cunningham, Y.P. Zhang, H.G. Clayhamp, O.T. Macina, N.B. Sussmann, S.G. Grant and G. Klopman. Development, Characterization and Application of Predictive-Toxicology Models. *SAR and QSAR in Environmental Research*, 10:277-298, 1999.
- M. Sebag and C. Rouveirol. Tractable induction and classification in FOL. in *Proc. of IJCAI-97*, 888-892, 1997.
- A. Srinivasan, R.D. King. Feature construction with Inductive Logic Programming: A Study of Quantitative Predictions of Biological Activity Aided by Structural Attributes. *Data Mining and Knowledge Discovery*, Vol. 3, 1999.
- A. Srinivasan, R.D. King and D.W. Bristol. An Assessment of Submissions Made to the Predictive Toxicology Evaluation Challenge. *Proc. of IJCAI-99*, 270-275, 1999.
- A. Srinivasan, S. Muggleton, R.D. King, and M. Sternberg. Theories for mutagenicity: a study of first-order and feature based induction. *Artificial Intelligence*, 85(1-2):277-299, 1996.
- I. Witten, E. Frank. *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.