Demand-Driven Construction of Structural Features in ILP

Stefan Kramer

Institute for Computer Science, Machine Learning Lab
Albert-Ludwigs-University, Georges-Köhler-Allee, Gebäude 079,
D-79110 Freiburg i. Brg., Germany
skramer@informatik.uni-freiburg.de

Abstract. This paper tackles the problem that methods for propositionalization and feature construction in first-order logic to date construct features in a rather unspecific way. That is, they do not construct features "on demand", but rather in advance and without detecting the need for a representation change. Even if structural features are required, current methods do not construct these features in a goal-directed fashion.
In previous work, we presented a method that creates structural features in a class-sensitive manner: We queried the molecular feature miner (MolFea) for features (linear molecular fragments) with a minimum frequency in the positive examples and a maximum frequency in the negative examples, such that they are, statistically significant, over-represented in the positives and under-represented in the negatives. In the present paper, we go one step further. We construct structural features in order to discriminate between those examples from different classes that are particularly problematic to classify. In order to avoid overfitting, this is done in a boosting framework. We are alternating AdaBoost re-weighting episodes and feature construction episodes in order to construct structural features "on demand". In a feature construction episode, we are querying for features with a minimum cumulative weight in the positives and a maximum cumulative weight in the negatives, where the weights stem from the previous AdaBoost iteration. In summary, we propose to construct structural features "on demand" by a combination of AdaBoost and an extension of MolFea to handle weighted learning instances.

1 Introduction

In the past few years, both machine learning and inductive logic programming have devoted a lot of attention to feature construction [21, 13, 4, 3, 15] . Features are constructed either in a class-blind [4] or in a class sensitive manner [21]. However, current methods still construct features in a rather unspecific way. That is, they do not construct features "on demand", but rather in advance and without detecting the need for a representation change. Even if structural features are required, current methods usually do not construct these features in

a goal-directed fashion. In this paper, we investigate the possibility of a "demand-driven" construction of structural features.

Changing the representation on demand in ILP is not new. The system MODELER [23] formed new concepts if the number of exceptions to a rule became unplausibly large. De Raedt [5] proposed to shift the bias towards a more expressive representation language if the current language is not sufficient to express the concept. The present work is also related to other approaches that shift the bias to increasingly expressive representation languages, but do not focus on the demand-driven aspect [2, 1]. The work described in this paper differs in several respects. Firstly, the expressiveness of the representation language is not changed. Secondly, the learning setting is the one of inductive concept learning, where we assume noise in the data. Thirdly, we deal with the demand-driven construction of structural features.

The idea of demand-driven feature construction in ILP is also related to the idea of using ILP methods only where propositional learning fails, or, more precisely, of using an ILP algorithm in order to correct the mistakes made by a propositional learning algorithm. One submission by Srinivasan and co-workers to the PTE-2 challenge [22] based on this idea turned out to be optimal under certain cost functions and/or class distributions according to ROC analysis.

In previous work, we introduced the *Molecular Feature Miner* (MolFea) [7, 16, 17], a domain specific inductive database that is capable of searching for linear molecular fragments (corresponding to features) of interest in large databases of chemical compounds. For instance, one can query the system for features (linear molecular fragments) with a minimum frequency in the positive examples and a maximum frequency in the negative examples, such that they are, statistically significant, over-represented in the positives and under-represented in the negatives. Recent evidence with MolFea [16] showed, that such a class-sensitive feature construction can be beneficial in conjunction with classical Machine Learning systems. In the present paper, we go one step further. We construct structural features in order to discriminate between those examples from different classes that are particularly problematic to classify. In order to avoid overfitting, this is done in a boosting framework. In a feature construction episode, we are querying for features with a mininum cumulative weight in the positives and a maximum cumulative weight in the negatives. This requires an extension of MolFea to handle not only frequencies of instances, but also weights.

This paper is organized as follows. Sections 2 and 3 introduce the molecular feature miner and its extension for handling weighted instances. In Section 4, we present our approach to the demand-driven construction of structural features. After the section on experimental results, we conclude the paper.

2 The Molecular Feature Miner MolFea

In this section, we briefly review the Molecular Feature Miner (MolFea). MolFea is a domain specific inductive database which aims at mining molecu-

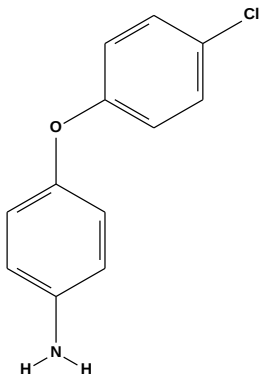

Fig. 1. Example compound in a 2-D representation. '$cl - c \sim c \sim c \sim c - o$' is an example fragment occurring in the molecule.

lar fragments (features) of interest in chemical data. The level-wise version space algorithm (cf. [18]) forms its basis. More information can be found in [7, 16, 17]. Inductive databases follow Mannila and Toivonen's [18] formulation of the general pattern discovery task. Given a database r, a language $\mathcal{L}$ for expressing patterns, and a constraint q, find the theory of r with respect to $\mathcal{L}$ and q, i.e. $Th(\mathcal{L}, r, q) = \{\phi \in \mathcal{L} \mid q(r, \phi)\ is\ true\}$. Viewed in this way $Th(\mathcal{L}, r, q)$ contains all sentences within the pattern language considered that make the constraint q true.

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 [4] or subgraphs [12]. 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 $c_1 \wedge ... \wedge c_n$. 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.

If weights are associated with instances, we can generalize frequency-related constraints to weight-related constraints:

- $sum\_weights(f, D)$ denotes the sum of the weights of the instances in D covered by a fragment f;
- $sum\_weights(f, D_1) \leq t$, $sum\_weights(f, D_2) \geq t$ where t is a positive real number and D_1 and D_2 are sets of molecules; this constraint denotes that the sum of the weights of those examples in D_1 (resp. D_2) covered by f should be larger than (resp. smaller than) or equal to t.

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:

$$('c-o' \leq f) \wedge \neg(f \leq\ 'c-o-s-c-o-s') \wedge$$
$$(freq(f, Act) \geq 0.95) \wedge freq(f, Inact) \leq 0.05)$$

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 $sol(c_1 \wedge ... \wedge c_n)$ in $\mathcal{M}$ for a conjunctive constraint $c_1 \wedge ... \wedge c_n$ is a versionspace 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

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

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. [18]). A constraint c is *monotonic* (resp. anti-monotonic) w.r.t. generality whenever

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

(resp. $(s \in sol(c)) \rightarrow (g \in sol(c))$). The basic anti-monotonic constraints in our framework are: $(f \leq p), (freq(f, D) \geq m), (sum\_weights(f, D) \geq m)$, the basic monotonic ones are $(p \leq f), (freq(f, D) \leq m), (sum\_weights(f, D) \leq m)$. 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. [18]), where the borders are often denoted by BD^+, as well as the machine learning literature (cf. [19]), 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

$min(F) = \{f \in F \mid \neg\exists q \in F : f \leq q\}$
$max(F) = \{f \in F \mid \neg\exists q \in F : q \leq f\}$[1]

[1] Note that *min* gives the minimally general elements, and *max* the maximally general ones. In contrast, $g \leq s$ means that g is more general than s.

We can now define the borders $S(c)$ and $G(c)$[2] of a primitive constraint c as

$$G(c) = max(sol(c)) \; and \; S(c) = min(sol(c))$$

Anti-monotonic constraints c will have $G(c) = \{\top\}$ and for proper constraints $S(c) \neq \{\bot\}$; proper monotonic constraints have $S(c) = \{\bot\}$ and $G(c) \neq \{\top\}$. Furthermore, as in Mitchell's versionspace 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 versionspace completely characterized by $S(c_i)$ and $G(c_i)$. Therefore, the set of solutions $sol(c_1 \wedge ... \wedge c_n)$ to a conjunctive constraint $c_1 \wedge ... \wedge c_n$ will also be completely characterized by the corresponding $S(c_1 \wedge ... \wedge c_n)$ and $G(c_1 \wedge ... \wedge c_n)$.

Elsewhere [7, 6], we have presented algorithms for computing the S and G sets corresponding to the constraints. The algorithm basically integrates the levelwise algorithm by Mannila and Toivonen [18] with Mellish's description identification algorithm. In principle, one might also employ Hirsh's version space merging algorithm [11].

4 Constructing Structural Features On Demand

In the previous section, we summarized the basic ideas underlying our domain specific inductive database MolFea. Using MolFea, we can declaratively specify the features of interest by a set of constraints. The space of all fragments (and corresponding to the fragments, the features) satisfying the constraints takes the form of a version space.

In order to construct structural features on demand, we performed two preliminary experiments: in the first one, we performed iterations of feature construction, where we queried for features that discriminate two individual, misclassified examples from different classes. In the second, we queried for features that discriminate between the false positives (resp. false negatives) and the positives (resp. negatives), either in one or in several iterations. In all of these cases, there were indications of overfitting. Thus, we devised a novel approach in the framework of boosting.

In the novel approach, we interleave AdaBoost [10] re-weighting episodes and feature construction episodes. Another view on the approach would be that we are boosting a weak learner that consists of weight-sensitive feature construction and some propositional learning algorithm.

In Table 1, the pseudo-code of our novel approach is shown. As the pseudo-code indicates, we perform regular AdaBoost iterations. In the first iteration,

[2] 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.

Table 1. Pseudo-code of AdaBoost, as instantiated in the presented approach. The basic AdaBoost procedure repeatedly queries MolFea for features (fragments) that are over-represented in the positive examples E_+ and under-represented in the negative examples E_-. These features are used in a propositional learner. For simplicity of presentation, we included two exit statements in the pseudocode.

---

procedure AdaBoost($E_+, E_-, PropFeatures, MaxIt$)

All training examples in $E_+ \cup E_-$ are weighted by $w_i = 1/N$
$j := 1$;
repeat
  if $j = 1$
  then $Fs_j := PropFeatures$
  else $Fs_j :=$ MolFea(E_+, E_-);
  if $Fs_j = \emptyset$ **then exit**
  $H_j :=$ PropLearner(Fs_j, E_+, E_-);
  $\epsilon_j :=$ weighted error rate of H_j on $E_+ \cup E_-$
  if $\epsilon_j = 0$ **or** $\epsilon_j > 1/2$ **then exit**
  $\beta_j := log \frac{\epsilon_j}{1-\epsilon_j}$
  for each $e_i \in E_+ \cup E_-$ **do**
    if $misclassifies(H_j, e_i)$
    then $w_i := w_i \frac{1-\epsilon_j}{\epsilon_j}$
  renormalize all weights w_i to sum up to 1
until $j = MaxIt$
return hypotheses H_j weighted by β_j

---

we use the initial propositional features $PropFeatures$ that are given as an input argument. In further iterations, MolFea attempts to construct structural features with a high cumulative weight in the positive examples and a low cumulative weight in the negatives. Thus, the feature construction algorithm (like the propositional learner applied subsequently) focuses attention on those examples that are difficult to classify.

The parameters for calling MolFea are determined as follows: We are seeking features (fragments) that are over-represented in the positive examples E_+ and under-represented in the negative examples E_-. If we had a dataset with a weight of one for each instance, we could do the following: We are interested in fragments with a minimum frequency of, say, 6, 10, 15, 20, respectively, and apply the χ^2-Test to a 2×2 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 negative examples. If we multiply the AdaBoost weights of the examples by N, the number of training examples, then we can proceed in

Table 2. Results of the new approach in standard ILP benchmark domains. *It.* denotes the number of iterations, *Acc.* denotes the predictive accuracy of ten-fold cross-validation if the maximum number of iterations was set to the resp. value; # *H.'s* denotes the number of hypotheses from ten-fold cross-validation that are constructed in the resp. iteration (note that the process can stop due to a number of reasons); *Av. C.* denotes the average number of conditions in PART hypotheses of the resp. iteration (also note that PART might just return the empty default theory).

	PTE			Mutag.			Biodeg.		
It.	*Acc.*	# *H.'s*	*Av. C.*	*Acc.*	# *H.'s*	*Av. C.*	*Acc.*	# *H.'s*	*Av. C.*
1	62.9	10	1.0	81.4	10	6.3	59.5	10	2.1
2	62.9	10	7.9	79.8	10	14.8	67.7	10	9.0
3	63.8	10	2.6	88.8	10	25.6	72.9	10	7.9
4	63.8	3	0.3	85.6	10	22.9	73.8	9	4.9
5	63.8	1	0.0	86.7	10	14.8	74.7	5	4.6
6	–	–	–	88.3	10	9.9	74.7	3	0.3
7	–	–	–	88.3	8	2.1	74.7	1	0.0
8	–	–	–	88.8	5	3.6	–	–	–
9	–	–	–	87.8	5	3.0	–	–	–
10	–	–	–	88.3	3	1.0	–	–	–

analogy to the above case in order to determine the values of the minimum and maximum cumulative weights in the queries posed to MolFea.[3]

5 Experimental Results

The goal of the experimentation (as described in this section) was to show

- that it is possible to improve upon the initial propositional representation by the construction of new structural features, and
- that the approach is not prone to overfitting.

We performed experiments in three real-world domains: carcinogenicity prediction [22] (the PTE-2 dataset), mutagenicity prediction [20] and biodegradability prediction [8]. For biodegradation prediction, we used a two-class version (degradable or not) with a half-life time (HLT) threshold of 4 weeks.

The initial propositional features were: the result 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 maximum number of iterations was set to 10. PART [9], one of the best rule learning systems available today, was chosen as the propositional learner in the AdaBoost iterations. The settings for feature construction were as described above: the minimum cumulative weight (multiplied by N) was set to 6, 10, 15 and 20, respectively. The maximum cumulative weight was set dependent on the class distribution (that is actually modified during AdaBoost iterations).

[3] Note that the sum of weights of the training examples is always one in AdaBoostM1.

Table 2 summarizes the results from 10-fold cross- validation: In all three domains, it shows that the approach is able to improve upon the initial representation. As the number of iterations increase, the generalization performance increases as well, and peaks after a few iterations. Thus, we may conclude that the approach indeed improves over the initial propositional representation and that it shows no sign of overfitting. Interestingly, the algorithm stops quite early in several cases, since MolFea cannot find any more statistically significant structural features in the data.

Note that we did not make any attempts to tune the parameters: we just used AdaBoostM1 in conjunction with PART *as is* (the default settings). Besides, as mentioned above, our goal was to show that the overall approach improves upon the initial representation and that it does not overfit the training data.

6 Conclusion

In this paper we tackled the problem of a demand-driven construction of structural features by a combination of weight-related queries posed to the molecular feature miner MolFea and the AdaBoost framework. Although the approach was presented in the context of molecular fragments, it is possible to adapt it to other pattern domains, such as, e.g., first-order Datalog queries. We believe that many extensions of this work are conceivable. Different pattern domains, different machine learning systems and different variants of boosting could be investigated in the future.

References

[1] I. Bournaud, M. Courtine, J.-D. Zucker. Abstractions for knowledge organization of relational descriptions. in: *Proceedings of the 4th International Symposium on Abstraction, Reformulation, and Approximation (SARA-00)* , 87–106, Springer, 2000.

[2] C. Carpineto. Shift of bias without operators. in: *Proceedings of the 10th European Conference on Artificial Intelligence*, 471–473, IOS Press, 1992.

[3] E. Alphonse, C. Rouveirol. Lazy propositionalization for relational learning. in: *Proceedings of the 14th European Conference on Artificial Intelligence*, IOS Press, 2000.

[4] L. Dehaspe, H. Toivonen. Discovery of frequent datalog patterns, *Data Mining and Knowledge Discovery*, 3(1):7–36, 1999.

[5] L. De Raedt. *Interactive Concept Learning.* PhD thesis, Katholieke Universiteit Leuven, Leuven, Belgium, 1991.

[6] L. De Raedt. A logical database mining query language. in: *Proceedings of the 10th Inductive Logic Programming Conference*, 78–92, Lecture Notes in Artificial Intelligence, Vol. 1866, Springer, 2000.

[7] 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)*, 2001.

[8] S. Džeroski, H. Blockeel, B. Kompare, S. Kramer, B. Pfahringer, W. Van Laer. Experiments in predicting biodegradability. in: *Proceedings of the 9th International Workshop on Inductive Logic Programming (ILP-99)*, 80–91, Springer, 1999.

[9] 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.

[10] Y. Freund, R.E. Schapire. A decision theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55:119–139, 1997.

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

[12] A. Inokuchi, T. Washio, H. Motoda. An Apriori-based algorithm for mining frequent substructures from graph data. in: D. Zighed, J. Komorowski, J. Zyktow (eds.), *Proceedings of the European Conference on Principles and Practice of Knowledge Discovery in Databases 2000*, Lecture Notes in Artificial Intelligence, Vol. 1910, Springer, 2000.

[13] 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`

[14] S. Kramer, E. Frank. Bottom-Up Propositionalization. In the *Proceedings of the Work-in-Progress Track at the 10th International Conference on Inductive Logic Programming*, 156–162, 2000.

[15] S. Kramer, N. Lavrač, P. Flach. Propositionalization Approaches to Relational Data Mining. in: S. Džeroski, N. Lavrač (eds.), *Relational Data Mining*, Springer, 2001.

[16] S. Kramer, L. De Raedt. Feature construction with version spaces for biochemical applications. in: *Proceedings of the Eighteenth International Conference on Machine Learning (ICML-01)*, 2001.

[17] S. Kramer, L. De Raedt, C. Helma. Molecular Feature Mining in HIV Data. in: *Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-01)*, 2001.

[18] H. Mannila, H. Toivonen. Level-Wise search and borders of theories in knowledge discovery. *Data Mining and Knowledge Discovery*, 1(3):241–258, 1997.

[19] T. Mitchell. Generalization as search. *Artificial Intelligence*, 18(2), 1982.

[20] A. Srinivasan, S. Muggleton, R. D. King, M. Sternberg. Theories for mutagenicity: a study of first-order and feature based induction. *Artificial Intelligence*, 85(1-2):277–299, 1996.

[21] A. Srinivasan, R. King. Feature construction with inductive logic programming: a study of quantitative predictions of biological activity aided by structural attributes. *Data Mining and Knowledge Discovery*, 3(1):37–57, 1999.

[22] A. Srinivasan, R. D. King, D. W. Bristol. An assessment of submissions made to the predictive toxicology evaluation challenge. in: *Proceedings of the International Joint Conference on Artificial Intelligence 1999*, 270–275, 1999.

[23] S. Wrobel. Demand-driven concept formation. in: K. Morik (ed.) *Knowledge Representation and Organization in Machine Learning*, 289–319, Springer, 1989.