Seeing the World through Homomorphism: An Experimental Study on Reducibility of Examples

Ondřej Kuželka and Filip Železný

Intelligent Data Analysis Research Group
Dept. of Cybernetics, Czech Technical University in Prague
http://ida.felk.cvut.cz
{kuzelo1,zelezny}@fel.cvut.cz

Abstract. We study reducibility of examples in several typical inductive logic programming benchmarks. The notion of reducibility that we use is related to theta-reduction, commonly used to reduce hypotheses in ILP. Whereas examples are usually not reducible on their own, they often become implicitly reducible when language for constructing hypotheses is fixed. We show that number of ground facts in a dataset can be almost halved for some real-world molecular datasets. Furthermore, we study the impact this has on a popular ILP system Aleph.

1 Introduction

In this paper, we are interested in identification of learning examples which are not distinguishable given a fixed hypothesis language. We propose a notion of *safe reduction* and show how it can be exploited in order to obtain smaller examples equivalent to the original ones.

2 Preliminaries

Let us first state some notational conventions used in this paper. The set of literals in a clause C is written as $lits(C)$, $|C| = |lits(C)|$ is the size of C. The set of variables in a clause C is written as $vars(C)$ and the set of all terms by $terms(C)$. A substitution θ is a mapping from variables of a clause C to terms of a clause D.

Let us now define θ-reduction [4], which will be later utilized for reduction of training examples. In order to define θ-reduction, we need the notion of θ-subsumption [5]. Let us denote the set of variables contained in a clause C by $vars(C)$. We say that a clause C θ-subsumes clause D (denoted by $C \preceq_\theta D$) if and only if there is a substitution $\theta : vars(C) \to vars(D)$ such that $C\theta \subseteq D$. It holds that θ-subsumption implies entailment, but not vice-versa, i.e. if $C \preceq_\theta D$ then $C \models D$, but if $C \models D$ then $C \preceq_\theta D$ may or may not be true.

Definition 1 (θ-Reduction). *Let C and D be clauses. We say that C and D are θ-equivalent (denoted by $C \approx_\theta D$) if and only if $C \preceq_\theta D$ and $D \preceq_\theta C$. If there is a clause E such that $C \approx_\theta E$ and $|E| < |C|$ then C is said to be θ-reducible.*

For example, $C = east(T) \leftarrow hasCar(T,C) \wedge hasLoad(C,L1) \wedge hasLoad(C,L2) \wedge box(L2)$ is θ-reducible because $C \approx_\theta E$ where $E = east(T) \leftarrow hasCar(T,C) \wedge hasLoad(C,L) \wedge box(L)$. In this short paper, we will work within the learning from entailment setting [1].

Definition 2 (Learning from Entailment). *Let H and e be clausal theories. Then we say that H covers e under entailment (denoted by $H \preceq_E e$) if and only if $H \models e$.*

For example, let $e = east(t1) \leftarrow hasCar(t1,c1) \wedge hasLoad(c1,l1) \wedge box(l1)$ and let $H = \forall T \forall C : east(T) \leftarrow hasCar(T,C)$. Then we may easily check that H covers e under entailment (i.e. $H \preceq_E e$). In what follows we will refrain from writing the universal quantifiers in clauses.

3 Reducing the Examples

The learning task that we study in this paper is fairly standard. We are given a set of positive and negative examples encoded as first-order clauses and we would like to find a classifier separating the positive examples from the negative examples. This task could be solved by numerous ILP systems. We aim at finding a reduction procedure that would allow us to reduce the number of atoms in the examples while guaranteeing that the coverage of individual hypotheses would not be changed. This is formalized by the next definition which introduces the concept of safe reduction under intepretations.

Definition 3 (Safe Reduction under Entailment). *Let e and $\hat{e}$ be two interpretations and let $\mathcal{L}$ be a language specifying all possible hypotheses. Then $\hat{e}$ is said to be a safe reduction of e under entailment if and only if $\forall H \in \mathcal{L} : (H \preceq_E e) \Leftrightarrow (H \preceq_E \hat{e})$ and $|\hat{e}| < |e|$.*

Clearly, any hypothesis H which splits the positive examples from the negative examples correctly will also split the respective safely reduced examples correctly. Also, when predicting classes of test-set examples, any deterministic classifier that bases its decisions on the queries using solely the covering relation $\preceq_E$ will return the same classification even if we replace some of the examples by their safe reductions[1]. On the other hand, even the classifiers constructed by a deterministic learner on reduced and non-reduced examples may be different. For example, ILP systems that restrict their search space by bottom clauses may return different hypotheses for reduced and non-reduced examples. However, if the search is performed exhaustively, every hypothesis discovered for the reduced data will have to cover the same set of examples as some corresponding hypothesis discovered for the non-reduced data.

Let us now look at possible candidates for safe reduction. For example, θ-reduction $\hat{e}$ of a ground example e is a safe reduction because trivially $\hat{e} = e$.

[1] This is also true for propositionalization approaches that use the $\preceq_E$ relation to construct boolean vectors which are then processed by attribute-value-learners.

Assuming that in practical learning tasks, examples are very often ground, θ-reduction alone does not seem to help us very much. It turns out that we need some additional assumptions in order to be able to reduce even ground examples. Before we present the (admittedly very simple) kind of safe reduction that will be in the center of our interest in this paper, we go through the next simple motivating example.

Example 1. Suppose that we have one positive example $e_1^+ = east(t1) \leftarrow hasCar(t1, c1) \wedge hasLoad(c1, l1) \wedge hasCar(t1, c2)$ and one negative example $e_2^- = east(t2) \leftarrow hasCar(t2, c3)$. Our task is to learn a non-recursive definition of predicate $east/1$. Let the language $\mathcal{L}$ contain all non-recursive Horn clauses free of functions and constants and containing only predicate $east/1$ in the head and predicates $hasCar/2$ and $hasLoad/2$ in the body. Because of the non-recursivity and function-freeness assumptions we can check whether $H \preceq_E e$ is true by testing θ-subsumption between the hypothesis and the example. Let e' be a clause obtained by variabilizing the clause e. Since constants are not allowed in the hypothesis language $\mathcal{L}$, it holds that $H \preceq_\theta e \Leftrightarrow H \preceq_\theta e'$. Next, let $\hat{e}'$ be θ-reduction of e'. It also holds $H \preceq_\theta e \Leftrightarrow H \preceq_\theta \hat{e}'$. Clearly, if $H \preceq_\theta e$, there is a substitution θ such that $H\theta \subseteq e$. If we replace the constants in θ by the respective variables used when variabilizing e and obtain a substitution θ', it will also hold $H\theta' \subseteq e'$. Finally, since there is a substitution θ_{R1} such that $e'\theta_{R1} \subseteq \hat{e}'$ (because $e' \approx \hat{e}'$), it must also hold $H \preceq_\theta \hat{e}'$ because $H\theta'\theta_{R1} \subseteq \hat{e}'$. Similarly, we could justify the implication in the other direction, i.e. $H \preceq_\theta e \Leftarrow H \preceq_\theta \hat{e}'$.

Applied to the example e_1^+, we have that $\hat{e}_1^+ = east(T1) \leftarrow hasCar(T1, C1) \wedge hasLoad(C1, L1)$ is a safe reduction of e_1^+ w.r.t. the language $\mathcal{L}$. Notice that it is important to have a fixed hypothesis language to perform this kind of reductions. For example, if we allowed the constants $c1$, $c2$ and $c3$, $\hat{e}_1^+$ would no longer be a safe reduction of e_1^+ because the hypothesis $H = east(T) \leftarrow hasCar(T, c1) \wedge hasCar(T, c2)$ would cover e_1^+ but not $\hat{e}_1^+$.

We are now ready to describe the reduction method. The method expects the examples encoded as first-order clauses and a description of the hypothesis language on its input. The hypothesis language specification should provide information about which predicates and constants can be used to build the hypotheses. It starts by variabilizing the constants which are not in the hypothesis language but appear in the examples. After that, examples are reduced using θ-reduction and these reductions are output by the algorithm. In order to justify correctness of this procedure, we need the next almost trivial proposition.

Proposition 1. *Let $\mathcal{L}$ be a hypothesis language and let e be a clause. (i) Let $\tilde{e}$ be a clause obtained from e by variabilizing the constants which are not contained in the hypothesis language. Then $(H \preceq_E e) \Leftrightarrow (H \preceq_E \tilde{e})$ for any $H \in \mathcal{L}$ (ii) Let $\hat{e}$ be θ-reduction of $\tilde{e}$. Then $(H \preceq_E e) \Leftrightarrow (H \preceq_E \hat{e})$ for any $H \in \mathcal{L}$.*

The first part of Proposition 1 justifies the step of the reduction, in which some constants are replaced by variables, and the second part justifies the step, in which the examples are reduced using θ-reduction. Optionally, we may perform

also a third step in which the variables introduced in the first step are converted back to constants.

Example 2. Let us have a hypothesis language for the task of predicting mutagenicity of molecules $\mathcal{L}$. Let us have one example

$$e = pos(m) \leftarrow bond(a1, a2, 2) \wedge bond(a2, a1, 2) \wedge a(m, a1, c) \wedge a(m, a2, c) \\ \wedge bond(a1, a3, 1) \wedge bond(a3, a1, 1) \wedge a(m, a3, h) \wedge a(m, a4, h) \wedge bond(a2, a5, 1) \\ \wedge bond(a5, a2, 1) \wedge a(m, a5, h) \wedge bond(a2, a6, 1) \wedge bond(a6, a2, 1) \wedge a(m, a6, h).$$

In the hypothesis language $\mathcal{L}$ we do not consider the names of atoms (i.e. constants $a1$, $a2$ etc). We start by performing the first step of the reduction procedure, i.e. by variabilizing the constants that do not appear in the hypothesis language $\mathcal{L}$. We replace constant $a1$ by variable $A1$, $a2$ by $A2$ and so on and obtain a clause $\tilde{e}$. After this step, we compute the θ-reduction of $\tilde{e}$, which is

$$\hat{e} = pos(m) \leftarrow bond(A1, A2, 2) \wedge bond(A2, A1, 2) \wedge atm(m, A1, c) \wedge \\ \wedge a(m, A2, c) \wedge bond(A1, A3, 1) \wedge bond(A3, A1, 1) \wedge \wedge a(m, A3, h) \wedge . \\ bond(A2, A5, 1) \wedge bond(A5, A2, 1) \wedge a(m, A5, h),$$

The θ-reduction of $\hat{e}$ is witnessed by the substitution $\theta = \{A4/A3, A6/A5\}$. It is easy to check that $e\theta \subseteq \hat{e}$. Clause $\hat{e}$ is a safe reduction of e.

4 Experiments

In this section, we discuss reducibility of examples from real-life datasets using typical hypothesis languages. We show that, in the most extreme case, some examples are reduced to one tenth of their original length. Then we study the effect example reductions have on Aleph. The θ-reduction is performed by an algorithm built upon the θ-subsumption system ReSumEr2 [3]. Reduction runtimes were in all cases under 30 seconds, which is a negligible amount of time compared to time consumed by Aleph as we will see. In all of the experiments we deal only with ground examples and non-recursive clauses. Therefore, for these experiments, we do not need the whole logical formalism. It would suffice to consider only θ-subsumption (homomorphism).

4.1 Mutagenesis

The first set of experiments which we performed was done with the well-known Mutagenesis dataset [6]. In the encoding which we used and which is also usually used in the experiments with most ILP systems, every molecule is described using predicates $atm(M, A, T, T2)$ and $bond(A, B, BT)$. For example, let us have a literal $atm(mol, atom1, c, 22)$. Here, mol is an identifier of the molecule, $atom1$ is an identifier of the atom, the third argument denotes the atomic type, here c means carbon, and the constant 22 in the fourth argument means that $atom1$ is an *aromatic atom*. Similarly, $bond(atom1, atom2, 1)$ denotes a single bond

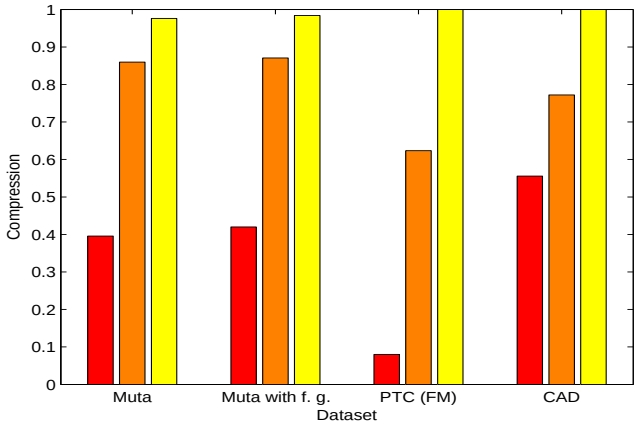

Fig. 1. Compression rates (*number of facts in reduced example / number of facts in original example*) obtained for three real-life datasets (Mutagenesis [6], Mutagenesis with function groups, PTC [2] and CAD [7]). The bars show maximum, average and minimum obtained compression.

between atoms $atom1$ and $atom2$. Each atomic bond is represented by two such $bond/3$ literals. In this case, when testing $H \preceq_E e$ with a single non-recursive clause H, the covering relation $\preceq_E$ corresponds to homomorphism testing (or θ-subsumption testing[2]). We let the hypothesis language $\mathcal{L}$ contain only constants corresponding to atomic types and numbers. After applying the safe reduction procedure, the number of literals in the examples decreased to 86% on average. The most succesful reduction decreased the number of literals in an example to 40% of the original count.

Next, we enriched the examples and the hypothesis language $\mathcal{L}$ by description of function groups such as benzene, phenathrene etc. This corresponds to another frequently used setting. Interestingly, the reduction rates differed only by a very small number from the previous case. The number of literals decreased to 87% on average and, for the most succesful reduction, the number of literals in one example decreased to 42% of the original count of literals.

4.2 Predictive Toxicology Challenge

The next experiment was performed with the Predictive Toxicology Challenge dataset (PTC). PTC dataset consists of molecules marked according to their carcinogenicity for female mice, male mice, female rats and male rats. Since the examples are almost the same for all of the four cases (with the exception

[2] Checking homomorphism of labeled hypergraphs, checking θ-subsumption of clauses and solving constraint satisfaction problems is essentially the same thing.

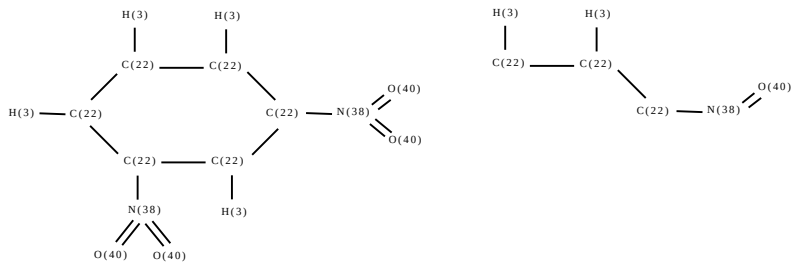

Fig. 2. The example from the Mutagenesis dataset which was compressed most by the reduction procedure. **Left:** The original example. **Right:** The reduced example. Note that every bond is represented by a pair of $bond/3$ literals.

of a few molecules contained in only some of the four datasets), we performed our experiments only for female mice. The atoms in this dataset are described at a cruder level compared to the Mutagenesis dataset. Only the basic types like *carbon* and not *aromatic carbon* are used. A consequence of this is that higher reduction rates are possible. After applying the safe reduction procedure, the number of literals in the examples decreased to 62% on average. The most succesful reduction decreased the number of literals in an example to just **8%** of the original count.

4.3 CAD

The last experiment was performed with the CAD dataset [7]. This dataset contains 96 class-labeled product-structures desings. After applying the safe reduction procedure to the examples from the CAD dataset, the number of literals in them decreased to 77% on average. The most succesful reduction decreased the number of literals in an example to 56% of the original count. The reason for smaller difference between average and maximum reduction rates as compared to the previous two datasets was that the designs contain $next/2$ literals which express a sort of ordering of the elements, which limits applicability of the θ-reduction.

4.4 Aleph with Reduced and Non-reduced Data

Finally, we decided to compare performance of Aleph on reduced and non-reduced versions of the datasets. We set the maximum number of explored nodes to 100000 and the *noise* parameter to five percent of the number of examples in the respective dataset. For both versions of the Mutagenesis dataset (without function groups and with function groups), it took Aleph longer to produce a theory for the reduced data, however, the predictive accuracy was higher for the

Table 1. Accuracy of Aleph (estimated by 10-fold cross-validation) on the reduced and non-reduced datasets. **Muta** is non-reduced Mutagenesis, **Muta-R** is reduced Mutagenesis, **Muta-FG** is Mutagenesis with function groups, **Muta-FG-R** is reduced Mutagenesis with function groups etc.

	Muta	**Muta-R**	**Muta-FG**	**Muta-FG-R**	**PTC**	**PTC-R**	**CAD**	**CAD-R**
Accu.	68.8 ± 6.6	71.6 ± 7.5	73.3 ± 11.3	73.8 ± 10.7	63.3 ± 6.7	63.6 ± 4.7	85.4 ± 9.7	88.7 ± 9.9
Run. [s]	356	378	298	339	17108	6522	244	168

reduced data. In fact, the Mutagenesis dataset was the only dataset where Aleph obtained statistically significant improvement in predictive accuracy (two-sided paired t-test with $\alpha = 0.05$). For PTC dataset, reduction caused a speedup of factor 2.63. For the CAD dataset, Aleph obtained a reasonable speed-up and slightly higher predictive accuracy than for the non-reduced dataset. The reason why not only runtimes but also accuracies were affected by the safe reduction is that through the reduction of examples we also reduce bottom clauses, which, in turn, means smaller search space for Aleph.

Not only can we use the techniques introduced in this paper to reduce the examples, we may also use them to obtain an upper bound on the training set accuracy. Clearly, if we variabilize the constants that do not appear in the hypothesis language $\mathcal{L}$ in two examples e_1 and e_2 and obtain examples $\hat{e}_1$ and $\hat{e}_2$ and if it holds $\hat{e}_1 \preceq_E \hat{e}_2$ and $\hat{e}_2 \preceq_E \hat{e}_1$ then these two examples are indistinguishable using the hypotheses from $\mathcal{L}$. If they have different classification then we have an unremovable term added to the training set error. For example, on the PTC dataset, the calculated upper bound was 98.5%. A more complicated calculation may be used to obtain a much tighter upper bound for the particular case of hypotheses in the form of clausal theories. The bound used in this subsection applies also to propositionalization approaches.

5 Conclusions

We have shown that when a hypothesis language is fixed, even ground examples may become reducible. In other words, we have shown that, in many experiments commonly performed with ILP systems, there is a lot of redundant information in the examples. We have also shown how the underlying principles of the reduction procedure can be used to compute an upper bound on the training-set accuracy.

Acknowledgement

The authors would like to thank the anonymous reviewers of ILP'10 for their valuable comments. The authors have been supported by the Czech Grant Agency through project 103/10/1875 (*Learning from Theories*).

Appendix

Lemma 1 (Plotkin [5]). *Let A and B be clauses. If $A \preceq_\theta B$ then $A \models B$.*

Proposition 1 *Let $\mathcal{L}$ be a hypothesis language and let e be a clause. (i) Let $\tilde{e}$ be a clause obtained from e by variabilizing the constants which are not contained in the hypothesis language. Then $(H \preceq_E e) \Leftrightarrow (H \preceq_E \tilde{e})$ for any $H \in \mathcal{L}$ (ii) Let $\hat{e}$ be θ-reduction of $\tilde{e}$. Then $(H \preceq_E e) \Leftrightarrow (H \preceq_E \hat{e})$ for any $H \in \mathcal{L}$.*

Proof. We will start by showing validity of the implication $(H \models e) \Rightarrow (H \models \tilde{e})$. For contradiction, let us assume that there is a model M of the clausal theory H such that $M \models e$ and $H \not\models \tilde{e}$. Then there must be a substitution θ grounding all variables in $\tilde{e}$ such that[3] $M \models e\theta$ and $M \not\models \tilde{e}\theta$. Now, we will construct another model M' of H in which e will not be satisfied. We take each constant c in e that has been replaced by a variable V in $\tilde{e}$ and update the assignment ϕ of the constants c to objects from the domain of discourse so that $\phi(c) = \phi(V\theta)$. Clearly, we can do this for every constant c since every constant in e has been replaced by exactly one variable. Now, we clearly see that $M' \not\models e$. However, we are not done yet as it might happen that the new model with the modified ϕ would no longer be a model of H. However, this is clearly not the case since none of the constants c appears in H and therefore the change of ϕ has no effect whatsoever on whether or not H is true in M'. So, we have arrived at a contradiction. We have a model M' such that $M' \models H$ and $M' \not\models e$ which contradicts the assumption $H \models e$. The implication $(H \models e) \Leftarrow (H \models \tilde{e})$ follows directly from Lemma 1. We have $\tilde{e} \preceq_\theta e$ therefore also $\tilde{e} \models e$ and finally $H \models \tilde{e} \models e$. (ii) In order to show $(H \preceq_E e) \Rightarrow (H \preceq_E \hat{e})$, it suffices to notice that $H \preceq_E e$ and $e \preceq_\theta \hat{e}$ imply $H \preceq_E \hat{e}$. The implication $(H \preceq_E e) \Rightarrow (H \preceq_E \hat{e})$ may be shown similarly as follows: $H \preceq_E \hat{e}$ and $\hat{e} \preceq_\theta e$ imply $H \preceq_E e$.

References

1. L. De Raedt. Logical settings for concept-learning. *Artif. Intell.*, 95(1):187–201, 1997.
2. C. Helma, R. D. King, S. Kramer, and A. Srinivasan. The predictive toxicology challenge 2000-2001. *Bioinformatics*, 17(1):107–108, 2001.
3. O. Kuželka and F. Železný. A restarted strategy for efficient subsumption testing. *Fundam. Inf.*, 89(1):95–109, 2009.
4. J. Maloberti and E. Suzuki. An efficient algorithm for reducing clauses based on constraint satisfaction techniques. In *ILP*, pages 234–251. Springer, 2004.
5. G. D. Plotkin. A note on inductive generalization. *Machine Intelligence*, 5:153–163, 1970.
6. A. Srinivasan and S. H. Muggleton. Mutagenesis: ILP experiments in a non-determinate biological domain. In *(ILP'04)*, 1994.
7. M. Žáková, F. Železný, J. Garcia-Sedano, C. M. Tissot, N. Lavrač, P. Křemen, and J. Molina. Relational data mining applied to virtual engineering of product designs. In *International Conference on Inductive Logic Programming (ILP '07)*. Springer, 2007.

[3] We are applying θ also to e because e need not be ground.