Cost-sensitive Feature Reduction Applied to a Hybrid Genetic Algorithm

Lavrac, N.; Gamberger, D.; Turney, Peter

Cost-sensitive feature reduction applied to a hybrid genetic algorithm

Nada Lavrač[1], Dragan Gamberger[2], Peter Turney[3]

[1] Jožef Stefan Institute, Jamova 39, 1001 Ljubljana, Slovenia
[2] Rudjer Bošković Institute, Bijenička 54, 10000 Zagreb, Croatia
[3] Institute for Information Technology, National Research Council Canada, M-50 Montreal Road, Ottawa, Ontario, Canada, K1A 0R6

Abstract. This study is concerned with whether it is possible to detect what information contained in the training data and background knowledge is relevant for solving the learning problem, and whether irrelevant information can be eliminated in preprocessing before starting the learning process. A case study of data preprocessing for a hybrid genetic algorithm shows that the elimination of irrelevant features can substantially improve the efficiency of learning. In addition, cost-sensitive feature elimination can be effective for reducing costs of induced hypotheses.

1 Introduction

The problem of relevance was addressed in early research on inductive concept learning [10]. Recently, this problem has also attracted much attention in the context of feature selection in attribute-value learning [1, 4, 12]. Basically one can say that all learners are concerned with the selection of 'good' literals which will be used to construct the hypothesis.

This study is concerned with whether it is possible to detect what information contained in the training data and background knowledge is relevant for solving the learning problem, and whether irrelevant information can be eliminated in preprocessing before starting the learning process. An important difference between our approach and most other approaches is that, when deciding about the relevance of literals, we are concerned with finding 'globally relevant' literals w.r.t. the entire set of training examples, as opposed to finding the 'good literals' in the given local training set.

This paper presents a case study of data preprocessing for a hybrid genetic algorithm which shows that the elimination of irrelevant features can substantially improve the efficiency of learning. In addition, cost-sensitive feature elimination can be effective for reducing costs of induced hypotheses. The paper is organized as follows: Section 2 introduces the representational formalism, the so-called p/n pairs of examples, gives the definition of irrelevant literals and presents the theorem which is the basis for literal elimination. Section 3 presents the cost-sensitive literal elimination algorithm REDUCE. Section 4 introduces the problem domain, the 20 and the 24 trains East-West Challenges, and presents the results of our experiments that show that the performance of a hybrid genetic

algorithm RL-ICET [14] can be significantly improved by applying REDUCE in preprocessing of the dataset.

2 Relevance of literals

The representation formalism. In REDUCE, the basic language elements are literals of the form $Attribute = Value$ for discrete attributes, and $Attribute \leq Value$ and $Attribute > Value$ for continuous attributes, as well as logical negations of these literals (the so-called *negative literals*). Training examples are bitstrings (tuples) of truth-values of these literals (1 - true, 0 - false).

In our experiments we are dealing only with discrete attributes, therefore only literals $Attribute = Value$ and $\neg(Attribute = Value)$ (i.e., $Attribute \neq Value$) will be considered. To illustrate this representation, consider two discrete attributes A and B, with respective sets of values $typeA = \{a_1, a_2, a_3\}$ and $typeB = \{b_1, b_2\}$. In this representational framework, there are five positive literals ($A = a_1$, $A = a_2$, $A = a_3$, $B = b_1$, $B = b_2$) and five negative literals. Suppose that there are two training examples e_1 and e_2. In the selected formalism, the training example $e_1 = (a_2, b_2)$ is represented by the bitstring 0100110110, and $e_2 = (a_3, b_1)$ by 0011011001.

The p/n pairs of examples and relevance of literals. We assume that the set of training examples E is represented by a two-dimensional matrix with columns corresponding to the set of positive and negative literals L, and rows that are bitstrings of truth-values of literals, corresponding to training examples e_i. The matrix is divided in two parts: P corresponds to the positive examples, and N to the negative examples.

To enable a formal discussion of the relevance of literals we the following definitions are introduced [3].

Definition 1. A p/n pair is a pair of training examples where $p \in P$ and $n \in N$.

Definition 2. Literal $l \in L$ covers a p/n pair if in column l of the matrix E of training examples the positive example p has value 1 and the negative example n has value 0. In other words, l covers a p/n pair if the value of literal l is true for p and false for n. The set of all p/n pairs covered by literal l will be denoted by $E(l)$.

Definition 3. Literal l covers literal l' if $E(l') \subseteq E(l)$.

Assume that literals are assigned costs. In our study, cost is a measure of complexity – the more complex is the literal, the higher is its cost. Let $c(l)$ denote the cost of literal $l \in L$.

Definition 4. Literal l' is irrelevant if there exists a literal $l \in L$ such that l covers l' ($E(l') \subseteq E(l)$) and the cost of l is lower than the cost of l' ($c(l) \leq c(l')$).

Our claim is that irrelevant literals can be eliminated in preprocessing. This claim is based on the following theorem, which assumes that the hypothesis language $\mathcal{L}$ is rich enough to allow for a complete and consistent hypothesis H to be induced from the set of training examples E.[4]

Theorem 1. Let L be a set of literals, and $L' \subseteq L$. A complete and consistent hypothesis H can be found using only literals from the set L' if and only if for each possible p/n pair from the training set E there exists at least one literal $l \in L'$ that covers the p/n pair.

The proof of this theorem can be found in [3]. The importance of the theorem is manifold. First, it points out that when deciding about the relevance of literals it will be significant to detect which p/n pairs are covered by the literal. Second, the theorem enables us to directly detect useless literals that do not cover any p/n pair. This theorem is the basis of the REDUCE algorithm for literal elimination.

3 A cost-sensitive literal elimination algorithm

Algorithm 1 implements the cost-sensitive literal elimination algorithm, initially developed within the ILLM learner [2]. This algorithm is the core of REDUCE.

Algorithm 1. Cost-sensitive literal elimination

Given: CL – costs of literals in L
Input: P, N – tables of positive and negative examples, L – set of literals
$RP \leftarrow P$, $RN \leftarrow N$, $RL \leftarrow L$
for $\forall\ l_i \in RL$, $i \in [1, |L|]$ **do**
if l_i has value 0 (*false*) for all rows of RP **then**
eliminate l_i from RL
eliminate column l_i from RP and RN tables
if l_i has value 1 (*true*) for all rows of RN **then**
eliminate l_i from RL
eliminate column l_i from RP and RN tables
if l_i is covered by any $l_j \in RL$ for which $c(l_j) \leq c(l_i)$ **then**
eliminate l_i from RL
eliminate column l_i from RP and RN tables
endfor
Output: RP, RN – reduced tables of positive and negative examples, RL – reduced set of literals

The complexity of Algorithm 1 is $\mathcal{O}(|L|^2 \times |E|)$, where $|L|$ is the number of literals and $|E|$ is the number of examples. The algorithm can be efficiently implemented using simple bitstring manipulation on the matrix of training examples. Moreover, this algorithm can be easily transformed into an iterative algorithm that can be used during the process of generation of literals (see [6]).

[4] Hypothesis H is complete if it covers all the positive examples $p \in P$. Hypothesis H is consistent if it does not cover any negative example $n \in N$.

4 Experimental results

4.1 The East-West Challenge and RL-ICET

Michie et al. [11] issued a "challenge to the international computing community" to discover low size-complexity Prolog programs for classifying trains as Eastbound or Westbound. The challenge was inspired by a problem posed by Ryszard Michalski [9].

The original challenge issued by Michie et al. [11] included three separate tasks. Donald Michie later issued a second challenge, involving a fourth task. Our experiments described here involve the first and fourth tasks. The first task was to discover a simple rule for distinguishing 20 trains, 10 Eastbound and 10 Westbound, whereas the fourth task involved 24 trains, 12 Eastbound and 12 Westbound. For both tasks, the winner was decided by representing the given rule as a Prolog program and measuring its size-complexity. The size-complexity of the Prolog program was calculated as the sum of the number of clause occurrences, the number of term occurrences, and the number of atom occurrences.

A cost-sensitive algorithm ICET was developed for generating low-cost decision trees [13]. ICET is a hybrid of a genetic algorithm and a decision tree induction algorithm: it takes feature vectors as input and generates decision trees as output. The algorithm is sensitive to both the cost of features (attributes) and the cost of classification errors. For the East-West Challenge, ICET was extended to handle Prolog input. The decision tree output was converted to Prolog manually. This algorithm is called RL-ICET (Relational Learning with ICET) [14]. RL-ICET is similar to the LINUS learning system [5]. RL-ICET uses a three-part strategy. First, a preprocessor translates the Prolog relations and predicates into a feature vector format. The preprocessor in RL-ICET was designed specially for the East-West Challenge, whereas LINUS has a general-purpose preprocessor. Second, an attribute-value learner applies a decision tree induction algorithm (ICET) to the feature vectors. Each feature is assigned a cost, based on the size of the fragment of Prolog code that represents the corresponding predicate or relation. A decision tree that has a low cost corresponds (roughly) to a Prolog program that has a low size-complexity. When it searches for a low cost decision tree, ICET is in effect searching for a low size-complexity Prolog program. Third, a postprocessor translates the decision tree into a Prolog program. Postprocessing with RL-ICET is done manually.

RL-ICET was the winning algorithm for the second task in the first East-West Challenge and it performed very well in the other three tasks. Much of the success of RL-ICET may be attributed to its preprocessor which translates the Prolog descriptions of the trains into a feature vector representation. The relatively compact Prolog descriptions were translated into rather large feature vectors of 1199 elements. The large vectors were required to ensure that all the features that were potentially interesting for the final solution would be available for ICET.

Although this approach can be recommended also for other applications of

inductive learning methods, one should be aware of the main limiting factor of the transformation approach which is that the number of generated features grows rapidly with the complexity of the application. This potentially results in a space complexity that cannot be handled by standard inductive learners. Furthermore, the idea of using a genetic algorithm for the selection of significant features (as in ICET) is interesting but it suffers from time complexity with large initial feature sets.

4.2 The experimental setting and results of experiments

The objective of the experiments was to show the utility of the literal elimination algorithm REDUCE. Two experiments were performed separately for the 20 and 24 trains problems [7, 8]. In both experiments, the RL-ICET preprocessor was used to generate the appropriate features and to transform the training examples into a feature vector format. This resulted in two training sets of 20 and 24 examples each, described by 1199 features.

In order to apply the REDUCE algorithm descibed in Section 3 we first have to convert the starting feature vector of 1199 elements to the corresponding literal vector which has twice as many elements, containing 1199 features generated by the RL-ICET preprocessor (positive literals) as well as their negated counterparts (1199 negative literals). After that, we eliminate the irrelevant literals and, in the third phase, we construct the reduced set of features which includes all the features which have at least one of their literals in the reduced literal set. The reasons for this three-step procedure are explained in [8].

The experimental setup, designed to test the utility of REDUCE, was as follows. First, 10 runs of the ICET algorithm were performed on the set of training examples described with 1199 features. Second, 10 runs of ICET were performed on the training examples described with the reduced set of features selected by REDUCE. The results were compared with respect to costs of decision trees and execution times.[5] Ten runs were needed because of the stochastic nature of the ICET algorithm: each time it runs, it yields a different result (assuming that the random number seed is changed). If we compared one single run of ICET on 1199 features to one run of ICET on the reduced feature set, the outcome of the comparison could be due to chance.

The results of the experiment are summarized in Table 1. The average results of 10 runs of RL-ICET were compared with respect to the costs of decision trees and execution times. Notice that all the experiments are independent of each other, e.g., results of experiment 4 should not be compared to the results of experiment 14. Only average results are relevant for the comparison.

[5] The performance in previous experiments by RL-ICET was measured by the cost of decision trees induced by ICET, as well as the complexity of Prolog programs after the RL-ICET transformation of decision trees into the Prolog program form [14]. Here we skip the latter, since the transformation into the Prolog form is currently manual and sub-optimal, which means that a tree with lowest cost found by ICET is not necessarily transformed into a Prolog program with lowest complexity.

20 *trains*						24 *trains*					
86 *features*			1199 *features*			86 *features*			1199 *features*		
Trial	*Time* t_1	*Cost* c_1	*Trial*	*Time* t_2	*Cost* c_2	*Trial*	*Time* t_1	*Cost* c_1	*Trial*	*Time* t_2	*Cost* c_2
1	11 : 05	18	11	2 : 21 : 32	24	1	14 : 35	20	11	1 : 54 : 15	27
2	11 : 19	21	12	2 : 21 : 34	21	2	14 : 26	18	12	1 : 55 : 29	21
3	12 : 55	18	13	2 : 19 : 15	20	3	14 : 59	18	13	2 : 00 : 25	26
4	11 : 35	18	14	2 : 19 : 32	20	4	14 : 17	21	14	1 : 56 : 31	25
5	15 : 16	18	15	2 : 16 : 20	18	5	13 : 32	18	15	1 : 56 : 47	25
6	11 : 35	18	16	2 : 23 : 52	22	6	13 : 31	22	16	1 : 57 : 14	24
7	11 : 32	18	17	2 : 24 : 09	21	7	14 : 29	18	17	1 : 56 : 52	28
8	11 : 38	18	18	2 : 18 : 41	16	8	13 : 54	23	18	1 : 56 : 33	23
9	11 : 28	18	19	2 : 16 : 58	18	9	13 : 51	23	19	1 : 49 : 08	27
10	11 : 18	21	20	2 : 23 : 09	20	10	14 : 30	18	20	1 : 47 : 46	28
Sum	119 : 41	186	*Sum*	23 : 25 : 02	200	*Sum*	2 : 22 : 04	199	*Sum*	19 : 11 : 00	254
Mean	11 : 57	18.6	*Mean*	2 : 16 : 54	20	*Mean*	14 : 12	19.9	*Mean*	1 : 55 : 05	25.4

Table 1. Results of the experiments.

Results of the 20 trains experiment. With the 20 train data, REDUCE cut the original set of 1199 features down to 86 features. In this way, the complexity of the learning problem was reduced to about 7% (86/1199) of the initial learning problem [7]. Results of 10 runs of ICET on the 1199 feature set are the results reported in [14], whereas results of 10 runs of ICET on the training examples described with 86 features are new.

The results show that the efficiency of learning significantly increased. In the initial problem with 1199 features, the average time per experiment was about 2 hours and 17 minutes, whereas in the reduced problem setting with 86 features the average time per experiment was about 12 minutes. The difference between times t_1 and t_2 is significant at the 99.99% confidence level. This shows the utility of literal reduction for genetic algorithms which are typically greedy regarding CPU time.

The average cost of descriptions induced from the 86 feature set has decreased (from 20 to 18.6), but the difference between decision tree costs c_1 and c_2 is not significant. In addition, the variance (or the standard deviation) of the costs was reduced, i.e., the costs of the decision trees generated from 1199 features vary more than the costs of the trees generated from 86 features: $var(c_1) = 1.6$ ($sd(c_1) = 1.3$) and $var(c_2) = 5.1$ ($sd(c_2) = 2.3$).

Results of the 24 trains experiment. In this experiment, REDUCE decreased the number of features from 1199 to 116. In this way, the complexity of the learning problem was reduced to about 10% (116/1199) of the initial learning problem [8]. The results show that the efficiency of learning significantly increased. In the initial problem with 1199 features, the average time per exper-

iment was nearly two hours, whereas in the reduced problem setting with 116 features the average time per experiment was about 14 minutes. The difference between times t_1 and t_2 is significant at the 99.99% confidence level.

The average cost of decision trees induced from the 116 feature set has also decreased. The difference between decision tree costs c_1 and c_2 is significant at the 99.99% confidence level. Our hypothesis that variance (standard deviation) of the output of RL-ICET can be reduced is only weakly supported since the inequality of variance is insignificant: $var(c_1) = 4.8$ ($sd(c_1) = 2.2$) and $var(c_2) = 5.2$ ($sd(c_2) = 2.3$).

5 Discussion and further work

This paper presents a case study of data preprocessing which shows that cost-sensitive elimination of irrelevant features can substantially improve the efficiency of learning and can reduce the costs of induced hypotheses. This study, using a hybrid genetic decision tree induction algorithm RL-ICET on two East-West Challenge problems, together with the previous results in feature reduction [2, 3] confirm the usefulness of feature reduction in preprocessing.

Some other experiments were performed and further experimentation is planned along these lines. In order to evaluate the effects of feature reduction, we have compared the results of ICET (with and without feature reduction) with the results achieved using C4.5 [15]. In both experiments, feature reduction (reduction to 86 and 116 features, respectively) helped ICET to outperform C4.5 when comparing costs of decision trees, both in terms of minimal and average costs [8]. On the other hand, the application of REDUCE did not help C4.5 itself to induce a lower cost solution from examples described with fewer features.

This study is also a step towards a better understanding of the notion of relevance for inductive concept learning. We are aware of some assumptions and simplifications which need to be elaborated in further work since they may hinder the application of the proposed approach in real-life applications. For example, we do not consider missing values of training examples. On the other hand, some of the important practical aspects are taken into account, such as the costs of literals.

Acknowledgements. This research was supported by the Slovenian Ministry of Science and Technology, the Croatian Ministry of Science, the National Reseach Council of Canada and the ESPRIT LTR 20237 Inductive Logic Programming II. The authors are grateful to Donald Michie for his stimulative interest in this work, and to Sašo Džeroski for his involvement in earlier experiments.

References

1. R. Caruana and D. Freitag, Greedy attribute selection, In *Proceedings of the 11th International Conference on Machine Learning*, Morgan Kaufmann (1994) 28–36.

2. D. Gamberger, A minimization approach to propositional inductive learning, In *Proceedings of the 8th European Conference on Machine Learning (ECML-95)*, Springer (1995) 151–160.
3. D. Gamberger and N. Lavrač, Towards a theory of relevance in inductive concept learning. Technical report IJS-DP-7310, J. Stefan Institute, Ljubljana (1995).
4. G.H. John, R. Kohavi and K. Pfleger, Irrelevant features and the subset selection problem, In *Proceedings of the 11th International Conference on Machine Learning*, Morgan Kaufmann (1994) 190–198.
5. N. Lavrač and S. Džeroski. *Inductive Logic Programming: Techniques and Applications*. Ellis Horwood (1994).
6. N. Lavrač, D. Gamberger and S. Džeroski. An approach to dimensionality reduction in learning from deductive databases. In *Proceedings of the 5th International Workshop on Inductive Logic Programming*, 337–354, (1995).
7. N. Lavrač, D. Gamberger, and P. Turney. Reduction of the number of features in the East-West Challenge, Technical Report IJS-DP-7347, J. Stefan Institute, Ljubljana (1996).
8. N. Lavrač, D. Gamberger, and P. Turney. Feature reduction in the 24 trains East-West Challenge, Technical Report IJS-DP-7372, J. Stefan Institute, Ljubljana (1996).
9. R.S. Michalski and J.B. Larson. Inductive inference of VL decision rules. Paper presented at Workshop in Pattern-Directed Inference Systems, Hawaii, 1977. SIGART Newsletter, ACM 63 (1977) 38–44.
10. R.S. Michalski, A theory and methodology of inductive learning, In: R. Michalski, J. Carbonell and T. Mitchell (eds.) *Machine Learning: An Artificial Intelligence Approach*, Tioga (1983) 83–134.
11. D. Michie, S. Muggleton, D. Page, and A. Srinivasan. To the international computing community: A new East-West challenge. Oxford University Computing Laboratory, Oxford (1994). (URL `ftp://ftp.comlab.ox.ac.uk/pub/Packages/ILP/trains.tar.Z`.).
URL `ftp://ftp.ics.uci.edu/pub/machine-learning-databases/trains/` contains information on both the 20 and 24 train challenges.
URL `http://www.gmd.de/ml-archive/ILP/public/data/east_west/` contains information on the 20 train challenge.
12. D. Skalak. Prototype and feature selection by sampling and random mutation hill climbing algorithms, In *Proceedings of the 11th International Conference on Machine Learning*, Morgan Kaufmann (1994) 293–301.
13. P. Turney. Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm.
Journal of Artificial Intelligence Research 2 (1995) 369–409. [Available on the Internet at URL http://www.cs.washington.edu/research/jair/home.html.]
14. P. Turney. Low size-complexity inductive logic programming: The East-West Challenge as a problem in cost-sensitive classification. In *Advances in Inductive Logic Programming*, IOS Press (1996) 308–321.
15. J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann (1993).

This article was processed using the LaTeX macro package with LLNCS style