Quantifying the Resilience of Inductive Classification Algorithms

Melanie Hilario and Alexandros Kalousis

CSD - University of Geneva, CH-1211 Geneva 4, Switzerland
hilario|kalousis@cui.unige.ch

Abstract. Selecting the most appropriate learning algorithm for a given task has become a crucial research issue since the advent of multi-paradigm data mining tool suites. To address this issue, researchers have tried to extract dataset characteristics which might provide clues as to the most appropriate learning algorithm. We propose to extend this research by extracting inducer profiles, i.e., sets of metalevel features which characterize their learning algorithms from t of view of their representation and functionality, efficiency, practicality, and resilience. Values for these features can be determined on the basis of author specifications, expert consensus or previous case studies. However, there is a need to characterize learning algorithms in more quantitative terms on the basis of extensive, controlled experiments. This paper illustrates the proposed approach and reports empirical findings on one resilience-related characteristic of learning algorithms for classification, namely their tolerance to irrelevant variables in training data.

1 Background and motivation

It is by now a matter of consensus that there are no universally superior models and methods for induction; the no-free-lunch theorems [19] and the conservation law of generalization performance [18] express basically the same thing, i.e., that no learning algorithm can systematically outperform others across the entire range of application tasks. The key question then is not whether a learning method is superior to others, but under which conditions a particular method can significantly outperform others on a given application problem/dataset. To define these conditions, researchers have attempted to isolate a set of data characteristics which impact the performance of learning algorithms as measured by a given evaluation metric [15][12]. While significant progress has been made on data characterization, the complementary task of extracting inducer characteristics has been relatively neglected. Characterizing a learning algorithm is usually reduced to classifying it as applicable or not to a given dataset [7]. We propose to extend research in this area by building profiles of learning algorithms, i.e., by extracting salient characteristics which allow for meaningful mappings between classes of algorithms and datasets. Rules expressing such mappings can be constructed either manually or automatically (via meta-learning). Such rules can then be used in a prior model selection phase to restrict the space of candidate

D.A. Zighed, J. Komorowski, and J. Zytkow (Eds.): PKDD 2000, LNAI 1910, pp. 106–115, 2000.

learning algorithms. This paper focuses on learning algorithms for classification. Section 2 gives an overview of characteristics that can be used to build algorithm profiles and highlights inadequacies of characterizations gleaned from the literature. In particular, it focuses on characteristics that comprise a learning algorithm's resilience and proposes an experiment-based approach to quantifying these. Section 3 illustrates this approach via a study of ten learning algorithms from the point of view of their sensitivity or tolerance to irrelevant variables. The experimental setup is described and major findings are reported and discussed in the light of related work. Section 4 summarizes and gives a preview of ongoing and future work.

2 Characterizing learning algorithms

Our knowledge of model characteristics comes from three different sources. First, certain characteristics are given explicitly in algorithm specifications; they concern the basic requirements, capabilities or limitations of an algorithm. Examples of such author-specified characteristics are the the types of data supported by the algorithm. A second source is observed consensus of experts in machine learning and data mining; however, when experts disagree or are simply in doubt, one can turn to controlled experimentation. The goal of our work is to complete the first two sources of knowledge by devising experimental strategies for characterizing learning algorithms. For the purposes of this paper, we group these characteristics along four dimensions: representation and functionality, efficiency, robustness and practicality.

2.1 Dimensions of algorithm characteristics

The first dimension along which learning algorithms can be described is representational power and functionality; this subsumes the types of data that can be processed by an algorithm, its incrementality, its ability to handle (mis)classification costs, and its bias-variance profile. Standard statistical methods typically don't handle symbolic representations while certain machine learning methods such as AQ cannot handle real-valued data. Most classifier inducers are non incremental and unable to handle (mis)classification costs, though researchers are actively exploring ways of overcoming these limitations for certain algorithms. The bias/variance profile of a learning method is a rough, qualitative indication of the direction in which the algorithm tends to resolve the trade-off between bias and variance [9]. High-bias learners generate simple, highly constrained models which are quite insensitive to data fluctuations, so that variance is low (e.g., perceptrons, Naive Bayes). Algorithms with a high-variance profile can generate arbitrarily complex models which fit data variations more readily (e.g., decision trees, neural networks). Characteristics which reflect a learning algorithm's efficiency are its average training and execution time as well as space demands; this dimension has been given increased attention with the advent of data mining applications, where scalability of learning algorithms is of primary importance.

An algorithm's practicality is the ease with which a user can use and understand it as well as its results. Examples of characteristics along this dimension are runtime parameter handling, the comprehensibility of the learning method, and the interpretability of the learned classifier. Assessment of an algorithm's practicality depends very much on user preferences and priorities. Most of the characteristics related to practicality can be described only by reporting users' subjective evaluations. The resilience of a learning algorithm refers to its capability of ensuring reliable performance despite variations in training conditions and especially in the training data. Resiliency characteristics express the sensitivity or tolerance of an algorithm to data characteristics or pathologies that are liable to affect performance adversely. Examples are an algorithm's scalability and tolerance to noise, missing values, and irrelevant or redundant features. The rest of this paper will focus on this group of inducer characteristics.

2.2 Quantifying inducer resilience

There have been previous attempts to evaluate, compare, or rank learning algorithms along the feature dimensions described in the preceding section. Often, such characterizations are tentative approximations; they express qualitative generalizations over broad classes of models or algorithms. For instance, while it is generally recognized that neural networks typically take much more time to train than decision trees, little is known about the average magnitude of this difference, or about any potential benefit that might be put in the balance against this added cost. Predictive accuracy has been the dominant metric for evaluating inductive algorithms, and only recently have efficiency-related criteria been taken into account [14]. The Statlog project included an attempt to quantify learning algorithms' practicality, in particular the comprehensibility of the underlying learning principle and the interpretability of results. However, a gap remains to be filled in our understanding of their resilience as defined above. To fill this gap, we have undertaken an extensive experimental study aimed at devising a quantiative scale for evaluating and comparing the resilience-related characteristics of classifier inducers. In the rest of this paper we focus on one characteristic which illustrates most clearly the need for quantitative metrics with precise semantics, as opposed to previous binary or other categorizations or rankings. We studied the impact of irrelevant attributes on the behavior of ten learning algorithms: C5.0-tree, C5.0-rules, C5.0-boost [17], Naive Bayes and instance-based learning from the MLC++ library [10], linear discriminants and Ltree [8], multilayer perceptrons and radial basis function networks as implemented in Clementine [4], and Ripper [5]. For each series of experiments, we used 43 datasets from the UCI Repository [13].

3 Tolerance to irrelevant attributes

3.1 Experimental setup

To evaluate and compare the impact of irrelevant variables on classifier inducers, we adopted the following experimental setup using the ten learning algorithms

and 43 datasets mentioned in Section 2.2. For each dataset, we generated corrupt versions with 10%, 20%, 30%, 40%, and 50% irrelevant variables. To produce a version with p% irrelevant variables from a dataset containing No original variables, we added $N_i = int(.01p * (N_0/(1 - .01p)))$ new variables whose values were generated randomly following a uniform distribution. A couple of precautionary measures were taken for the sake of experimental soundness. First, to ensure that variation in performance is due only to the irrelevant variables and not to side effects such as changes in the balance of variable types, we strove to maintain the original distribution of numeric and symbolic variables when adding irrelevant variables. Second, to mitigate fears that the random-valued features might be serendipitously correlated or associated in any way with the classes, we ascertained that the addition of such features effectively increased the quantity of irrelevant information in the datasets. To do this, we measured the mutual information between the class and each predictive (original or artificially added) variable for each dataset after discretizing all continuous variables using Fayyad and Irani's method [6]. Mutual information measures were averaged over all predictors to produce a single measure for each dataset, then averaged over all datasets with the same percentage of (additional) irrelevant variables to yield a single average measure of mutual information over all datasets at each level of corruption. The resulting curve (Fig. 1) shows that the average mutual information between predictive variables and classes decreases monotonically with the addition of random-valued features, thus confirming their irrelevance. The ten

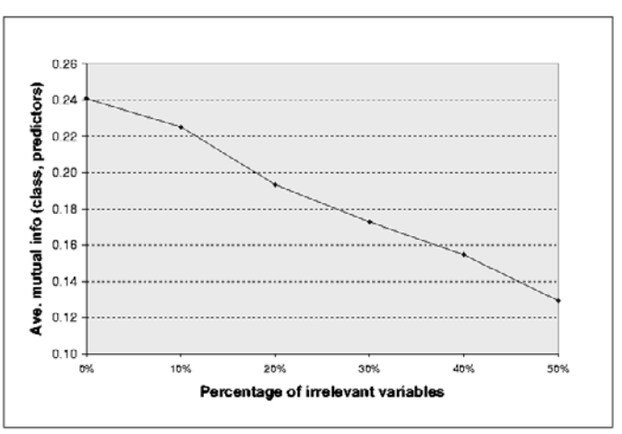

Fig. 1. Average mutual information as a measure of relevance

learning algorithms under study were run on each original dataset and its 5 corrupted variants using stratified 10-fold cross-validation. Each algorithm was run with its default settings. The performance of these algorithms on the original versions provided the baseline against which to study degradation of learning ability as the proportion of irrelevant variables increased. Performance measures used were test-set error and total processing (training plus test) time.

3.2 Results

The degradation in predictive accuracy is summarized in Figure 2.

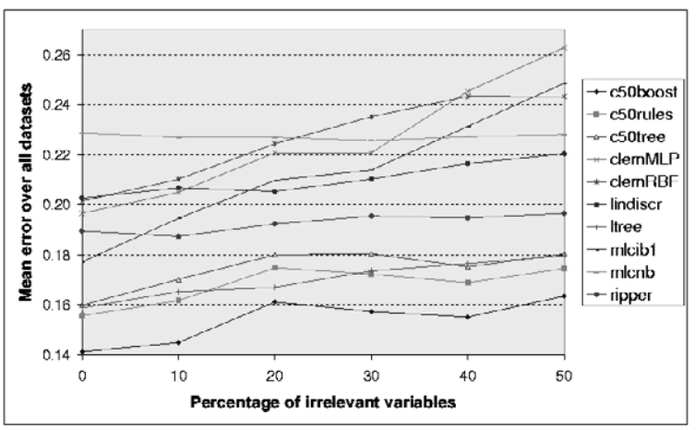

Fig. 2. Generalization error vs percentage of irrelevant variables

As a first cut at interpreting these results, note that an algorithm's predictive accuracy on the given datasets are quite independent of their resilience to irrelevant variables. For instance, boosted C5.0 tree attains the highest predictive accuracy at all levels, but its error curve rises more rapidly than most of the other algorithms in the study. On the other hand, the mean error of Naive Bayes remains among the highest at all levels of irrelevance, yet it seems unaffected by the proportion of irrelevant variables. To quantify sensitivity to irrelevance, it is thus important to decouple the magnitude of the mean error from its variation in response to irrelevant variables. We considered three candidate measures: the error increase $error_i - error_0$ for each level of irrelevance i was eliminated because it does not sufficiently abstract away the magnitude of the error. An alternative is the relative error increase $(error_i - error_0)/error_0$, whose obvious shortcoming is that for equal error increases, the sensitivity score increases with lower values of $error_0$, thus penalizing algorithms with high predictive accuracy. The third measure is simply the slope of the error curve which not only is exempt from the previously mentioned drawbacks but has the additional advantage of a clear semantics: sensitivity to irrelevance is defined quantitatively as the average rate of increase in error (or some other performance metric) with increase in the proportion of irrelevant variables. The resulting error-sensitivity scores and ranks are shown in Table 1(a) and the corresponding regressed curves in Fig. 3.

A closer look at the error sensitivity scores reveals several distinct clusters among the 10 algorithms. Naive Bayes is by far the most resistant to irrelevant variables, maintaining a comfortable distance from Ripper, its closest runner-up.

Algorithm	(a)		(b)	
	Error slope	Rank	Time slope	Rank
Boosted C5.0	0.039	6	33.58	6
C5.0 rules	0.032	3	4.76	3
C5.0 tree	0.034	4	4.26	2
Clementine MLP	0.129	9	378.25	9
Clementine RBF	0.091	8	9595.93	10
Lindiscr	0.035	5	4.92	4
Ltree	0.041	7	14.57	5
MLC++ IB1	0.135	10	56.98	7
MLC++ NBayes	0.000	1	3.64	1
Ripper	0.017	2	98.67	8

Table 1. Degradation of learner performance with irrelevant variables

Decision trees (whether oblique like Ltree or orthogonal like C5.0 in its three variants) and linear discriminants form a cluster with medium tolerance. Finally, neural networks (MLP and RBFN) and IB1 display the highest sensitivity to irrelevance with an average error increase close to or greater than 0.10%.

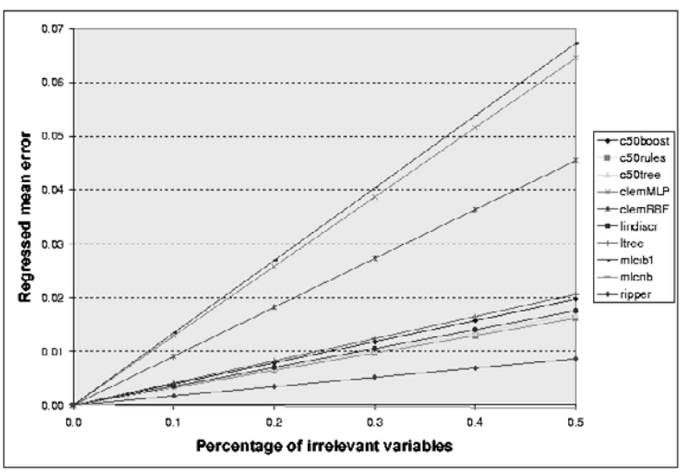

Fig. 3. Regressed error increase with irrelevant variables

The same approach was adopted using total training and test time as the performance criterion. Time measures were standardized across different machines and expressed in Sun Sparc Ultra 10-equivalent CPU seconds (for 124 MB main memory). The derivative of the run time curves was similarly used to quantify the time-sensitivity of learning algorithms to irrelevant variables; the resulting scores are shown in Table 1(b) and the regressed runtime curves in Fig. 4.

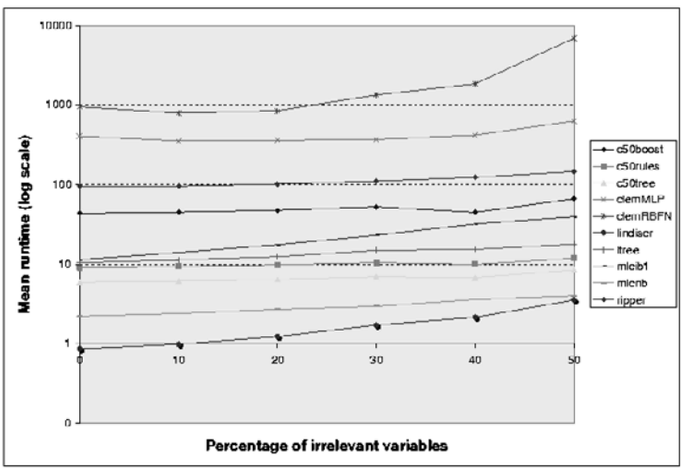

Fig. 4. Regressed run time increase with irrelevant variables

The ranking of the different algorithms undergoes a few changes with the shift to the runtime metric, though a clear overall picture subsists after combining insights based on both accuracy and computational cost. Naive Bayes is still a clear winner with a mean degradation in speed of around 3 CPU seconds. Linear discriminants and single-classifier C5.0 trees and rules trail closely behind, but boosting increases C5.0's speed degradation considerably. Ripper appears to stay close to its baseline accuracy at a much higher computational cost. As with predictive accuracy, the worst degradation is exhibited by the neural networks, MLP and RBF [1]. The runtime curves are represented using a logarithmic scale in Figure 4 and show that Clementine-RBFN's training and test time grows faster by an order of magnitude than Clementine-MLP and by several orders of magnitude than the 8 other algorithms.

3.3 Discussion and related work

Results concerning IB1 confirm expert consensus that instance-based learning (and standard k- NN in general) is highly sensitive to irrelevant variables [1]. Previous work has shown that k-NN's 90% sample complexity (the number of training examples needed to achieve 90% accuracy) grows exponentially with the number of irrelevant features while that of Naive Bayes grows only linearly [11]. This difference can be explained by the underlying inductive principles; instance-based classification identifies nearest neighbors via distance measures based on feature values, so that irrelevant features have as great an effect on classifier as the relevant ones. Thus a dramatic deterioration in both accuracy and

[1] Of the 43 datasets used, the monks problems were left out in computing MLP runtime since a bug in Clementine provokes an indefinite loop in default mode.

speed is understandable in IB1 and all k-NN based systems that use no feature-weighting or other feature selection schemes. At the other extreme, Naive Bayes' prediction criterion uses the class-conditional probabilities of feature values; in the case of irrelevant features, class-conditional probabilities are equal to their marginal probabilities and thus have no impact on the class posterior probabilities. This explains why Naive Bayes suffers no degradation in error and an almost insignificant increase in processing time. C5's fair-to-good resistance to irrelevant features confirms previous results on its predecessors, ID3 and C4.5. Past investigations suggest that, contrary to high expectations raised by its feature selection strategy, ID3's predictive accuracy is seriously hampered by irrelevant attributes [2]. More surprisingly, our experiments reveal that boosting makes C5.0 slightly more sensitive to irrelevance with respect to accuracy and significantly more sensitive with respect to speed. We have yet to find a plausible explanation for this phenomenon; since the number of iterations was kept constant (10 by default), and since the average size of each generated tree turned out to be roughly the same with and without boosting, we eliminated the hypothesis that boosted C5.0 was attempting to overfit to irrelevant features. Another surprise is the extreme sensitivity to irrelevance of the neural networks used in this study. For Clementine-MLP, this contradicts predictions based on the theory. MLPs are expected to be relatively unaffected by irrelevant features since the hidden layer projects inputs into a subspace of much lower dimensionality within which approximation can take place [16]. Concretely, connection weights from irrelevant inputs are expected to gradually tend towards zero as the training process converges; thus the only adverse effect should be a significant increase in training time. Unfortunately our experiments tend to show that accuracy deteriorates significantly with irrelevant inputs despite the dramatic increase in computational costs. For Clementine-MLP, this could be explained by overfitting to irrelevant inputs. In default mode, the number of hidden units is determined automatically, and we observed that the average number of hidden units increased monotonically with the percentage of irrelevant variables, doubling between the baseline and the 50% level (Table 2).

% of irrelevant variables	0%	10%	20%	30%	40%	50%
Number of hidden units	12.4	13.9	15.3	17.2	20.7	24.4

Table 2. Growth of MLP network complexity with irrelevant variables

On the other hand, Clementine-RBFN's sensitivity to irrelevance is predictable from the underlying approach [3]. First, in RBFNs hidden units represent basis functions whose parameters (e.g. centres) are selected using unsupervised training methods such as K- means clustering. This implies reliance on a distance measure which, as in the case of k-NN, gives equal importance to all variables, whether relevant or not. In addition, each hidden unit has a local receptive field, i.e., it influences network output only in the neighborhood of

its center. While predictive accuracy gains from locality when the hidden units reflect actual clusters in the data, it deteriorates significantly when these are built/activated using distance measures distorted by irrelevancies. This degradation could be alleviated by increasing the number of hidden units, which is however set to 20 in Clementine default mode. In short, two essential features of the RBFNs used—unsupervised K-means training and local receptive fields of hidden units plus a particularity of the Clementine default implementation—explain the observed lack of resilience to irrelevant variables. While the ranks of tested algorithms help to visualize overall trends and patterns, we decided to keep the original quantifications of sensitivity to irrelevance in the algorithm profiles. There were several reasons for this. The original sensitivity measures have a precise semantics, as explained in Section 3. Converting them to ranks would, first, result in loss of information and, second, complicate the task of adding a new learning algorithm to the profiled set. Finally, like other (meta-)datasets, algorithm characterizations can be pre-processed and continuous attributes reencoded in ordinal or qualitative form as the need arises. To conclude this section, it should be emphasized that the findings reported on the above algorithms cannot be generalized beyond the specific implementations used, even when they are confirmed by the underlying theory.

4 Summary and future work

This paper argued for the need to build learning algorithm profiles which can be used in prior model selection for data mining. We distinguished four groups of characteristics depending on whether they concern a learning algorithm's representation and functionality, efficiency, resilience, or practicality. We gave a quick overview of each group of characteristics and proposed an experimental setup for quantifying characteristics related to an inductive algorithm's resilience. We illustrated the proposed approach on a study of tolerance to irrelevant variables. Other studies (e.g., on scalability, bias/variance trade-off, resistance to other data idiosyncracies such as missing values and redundant features) have been conducted and will be presented in forthcoming publications. Algorithm profiles built from such studies will be used, together with dataset characterizations, to discover mappings between broad classes of learning tasks/data and models/algorithms. We shall attempt such mappings both manually or automatically, i.e., via (static or batch) meta-learning. In a subsequent phase, the knowledge thus built will be deployed to support the user in the model selection task; in return, feedback from novel applications/ environments will be assimilated via incremental meta-learning to dynamically refine or augment this experimentally cumulated expertise.

Acknowledgements

This work was partially supported by the Swiss Office for Education and Science (OFES) in the framework of ESPRIT IV LTR Project METAL-26357. We cannot

thank Johann Petrak (OeFAI, Vienna) enough for his Perl scripts which greatly facilitated our experimentation.

References

1. D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.
2. H. Almuallim and T. Dietterich. Efficient algorithms for identifying relevant features. In *Proceedings of the Ninth Canadian Conference on Artificial Intelligence*, pages 38–45, Vancouver, BC, 1992. Morgan Kaufmann.
3. C. M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.
4. Clementine. http://www.spss.com.
5. W. W. Cohen. Fast effective rule induction. In A. Prieditis and S. Russell, editors, *Proc. of the 11th International Conference on Machine Learning*, pages 115–123, Tahoe City, CA, 1995. Morgan Kaufmann.
6. U. Fayyad and K. Irani. Multi-interval discretization of continuous-valued attributes for classification learning. In *Proc. of the 13th IJCAI*, pages 1022–1027, Chambéry, France, 1993. Morgan Kaufmann.
7. J. Gama and P. Brazdil. Characterization of classification algorithms. In E. Pinto-Ferreira and N. Mamede, editors, *Progress in Artificial Intelligence. 7th Portuguese Conference on Artificial Intelligence (EPIA-95)*, pages 189–200. Springer-Verlag, 1995.
8. J. Gama and P. Brazdil. Linear tree. *Intelligent Data Analysis*, 3:1–22, 1999.
9. S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4:1–58, 1992.
10. R. Kohavi, D. Sommerfeld, and J. Dougherty. Data mining using mlc++. In *International Conference on Tools with AI*, pages 234–245, 1996.
11. P. Langley and S. Sage. Scaling to domains with irrelevant features. In R. Greiner et al., editor, *Computational Learning Theory and Natural Learning Systems*, volume 4, Cambridge, MA, 1996. MIT Press.
12. D. Michie, D. J. Spiegelhalter, and C. C. Taylor, editors. *Machine learning, neural and statistical classification*. Prentice-Hall, 1994.
13. P. M. Murphy and D.W. Aha. UCI machine learning repository. http://www.ics.uci.edu/ mlearn/MLRepository.html, 1991. Irvine, CA: University of California, Dept. of Information and Computer Science.
14. G. Nakhaeizadeh and A. Schnabl. Development of multi-criteria metrics for evaluation of data mining algorithms. In *Proc. Third International Conference on Knowledge Discovery and Data Mining*, pages 37–42, Newport Beach, CA, 1997. AAAI Press.
15. L. Rendell and E. Cho. Empirical learning as a function of concept character. *Machine Learning*, 5:267–298, 1990.
16. B.D. Ripley. *Pattern Recognition and Neural Networks*. Cambridge U. Press, 1996.
17. http://www.rulequest.com.
18. C. Schaffer. A conservation law for generalization performance. In W. W. Cohen and H. Hirsh, editors, *Proc. of the 11th International Conference on Machine Learning*, pages 259–265, Rutgers, NJ, 1994. Morgan Kaufmann.
19. D. Wolpert. The lack of a priori distinctions between learning algorithms. *Neural Computation*, 8(7):1381–1390, 1996.