Heuristic Rule-Based Regression via Dynamic Reduction to Classification

Frederik Janssen and Johannes Fürnkranz
Knowledge Engineering, TU Darmstadt
Darmstadt, Germany
{janssen,juffi}@ke.tu-darmstadt.de

Abstract

In this paper, we propose a novel approach for learning regression rules by transforming the regression problem into a classification problem. Unlike previous approaches to regression by classification, in our approach the discretization of the class variable is tightly integrated into the rule learning algorithm. The key idea is to dynamically define a region around the target value predicted by the rule, and considering all examples within that region as positive and all examples outside that region as negative. In this way, conventional rule learning heuristics may be used for inducing regression rules. Our results show that our heuristic algorithm outperforms approaches that use a static discretization of the target variable, and performs en par with other comparable rule-based approaches, albeit without reaching the performance of statistical approaches.

1 Introduction

The accurate prediction of a numerical target variable is an important task in machine learning. However, while the statistical learning community has proposed a great variety of algorithms for solving this problem, it has not received so much attention in the data mining and inductive rule learning communities, where a strong focus lies on the comprehensibility of the learned models.

The goal of this work is the design of a heuristic rule learning algorithm that learns regression models in the form of a decision list or rule set consisting of simple regression rules that have constant models in the rule head. This model class is fairly restrictive, but we can show that our approach of a dynamic reduction of regression to classification yields a very good performance within this model class. However, its performance is still below the performance of statistical approaches that incorporate linear models or boosting.

Several strategies to induce a set of regression rules have been proposed in the literature. Some of them rely on the gradient-descent algorithm for finding a rule ensemble that optimizes some loss function. Others convert given trees into sets of rules. We will briefly review work in this area in Section 2. One of the most popular strategies for learning classification rules is the so-called *separate-and-conquer* or *covering* algorithm, which we will briefly recapitulate in Section 3, because, due to its simplicity and its good performance in classification, we decided to use this strategy as the basis for our approach. In Section 4, we introduce our approach for repeatedly converting regression problems into classification problems. Its key idea is to dynamically define a region around the target value predicted by the rule, and considering all examples within that region as positive and all examples outside that region as negative. The experimental setup of our approach and the key results are described in Sections 5 and 6 respectively.

2 Related work

There are only a few published previous attempts that try to adopt separate-and-conquer rule learning to regression. Examples include the FRS system [Demšar, 1999], which is a reimplementation of the earlier FORS algorithm [Karalič and Bratko, 1997], and *predictive clustering rules* (PCR) [Ženko *et al.*, 2005]. Predictive clustering rules are generated by modifying the search heuristic of CN2 [Clark and Niblett, 1989] to use a heuristic that is based on the dispersion of the data. This algorithm also follows a different route by joining clustering approaches with predictive learning. The R^2 system [Torgo, 1995] works to some extent analogously as other separate-and-conquer algorithms by selecting an uncovered region of the input data. But this selection differs from the mechanism used in regular separate-and-conquer learning. However, it also allows for rules to overlap and the rules predict linear models instead of a single target value.

More popular are techniques that learn regression trees [Breiman *et al.*, 1984; Quinlan, 1992; Wang and Witten, 1997], which can then be converted into decision rules as in the M5RULES algorithm [Holmes *et al.*, 1999]. The key idea of these approaches is also to replace the purity heuristic of the decision tree algorithm with a heuristic that measures the reduction in variance.

Other rule-based regression algorithms are based on ensemble techniques. For example, RULEFIT [Friedman and Popescu, 2008] performs a gradient descent optimization, allows the rules to overlap, and the final prediction is calculated by the sum of all predicted values of the covering rules. REGENDER [Dembczyński *et al.*, 2008] uses a forward stagewise additive modeling.

Another popular technique to deal with a continuous target attribute is to discretize the numeric values as a preprocessing step and apply conventional classification algorithms on the discretized target attribute. Research following this path can be found in [Torgo and Gama, 1996; Weiss and Indurkhya, 1995]. The main problem here is that the number of bags for the discretization process is not known in advance. For this reason the performance of this technique strongly depends on the choice of the number of classes. [Langford *et al.*, 2006] convert a regression problem into a set of binary classification problems, where each classifier essentially tests whether an example is above or below a certain threshold.

The key idea of our approach for converting regression to classification is reminiscent of ϵ*-insensitive* loss functions which form the basis of several support-vector regression algorithms [Smola and Schölkopf, 2004]. A key difference is that we adapt the width of the insensitive region dynamically after each refinement step.

3 Separate-and-conquer rule learning

Most inductive rule learning algorithms for classification employ a separate-and-conquer strategy [Fürnkranz, 1999]. Its basic idea is to find a rule that covers a part of the example space that is not explained by any rule yet (the conquer step). After this rule is found, it is added to the current set of rules, and all examples that are covered by the rule are removed from the data set (the separate step). Then, the next rule is searched on the remaining examples. This procedure is repeated until no more (positive) examples are left. In order to prevent overfitting, the two constraints that all examples have to be covered (also called *completeness*) and that no negative example has to be covered in the binary case (*consistency*) can be relaxed so that some positive examples may remain uncovered and/or some negative examples may be covered by the set of rules.

For finding the next rule, most of the algorithms employ a top-down search, which iteratively refines an initially empty rule by successively adding conditions to it. For nominal attributes, candidate conditions are formed by comparing the attribute to all possible values. For numerical attributes, each test ($<$ and $\geq$) that compares the attribute's value to one of the values for this attribute is a candidate condition. The splitpoints are calculated as the mean between two adjacent (previously sorted) values. Among all candidate refinements, the best one is selected using a heuristic function. This rule is then in turn refined until no more negative examples are covered. The best rule encountered in this process is then returned as the final rule.

There are many different heuristics to navigate the search. All of them are trying to maximize the coverage of positive examples (p) and to minimize the negative coverage (n), but differ in the way in which they trade off these two objectives [Fürnkranz and Flach, 2005]. Experimenting with all known heuristics was out of the scope for this paper, so the decision was to use a heuristic that is known to underfit the data (*weighted relative accuracy* [Todorovski *et al.*, 2000]), one that overfits the data (*laplace*), and two that are known to perform well in a large variety of datasets (*correlation* and *relative cost measure*). The latter features a parameter c, which we set to the value of 0.342, which was recommended in [Janssen and Fürnkranz, 2010a]. The definition of these four heuristics is shown in Figure 1.

$$weighted\ relative\ accuracy\ (wra) = \frac{p}{P} - \frac{n}{N}$$

$$laplace\ (lap) = \frac{p+1}{p+n+2}$$

$$correlation = \frac{p \cdot N - n \cdot P}{\sqrt{P \cdot N \cdot (p+n) \cdot (P-p+N-n)}}$$

$$relative\ cost\ measure\ (rcm) = c \cdot \frac{p}{P} - (1-c) \cdot \frac{n}{N}$$

Figure 1: The rule-learning heuristics used in this paper. p and n are the number of covered among the P positive and N negative examples.

In our implementation, we only consider refinements that ensure that a minimum number of examples is covered. For all experiments (cf. Section 6) we fixed the minimum coverage to 3 examples. Missing attribute values are treated as uncovered by any condition. When the class of an instance is missing it is removed from the dataset in a pre-processing step. As a search strategy simple hill-climbing was used.

For classifying unseen examples, we use the learned set of rules as a decision list, i.e, the first rule that covers the example "fires" and predicts the target value that is stated in the head of the rule. If no rule covers the example, the prediction is given by a default rule that usually predicts the majority class in the data.

4 Dynamic Reduction to Classification

There are two principal approaches for adapting heuristic rule learning algorithms to numerical target variables. One way is to change the heuristic function in a way that the goal of discriminating between positive and negative examples is replaced with the goal of reducing the variance in the target value. This approach has been taken by most decision-tree based approaches [Breiman *et al.*, 1984; Quinlan, 1992; Wang and Witten, 1997] and by several regression rule learning approaches [Karalič and Bratko, 1997; Torgo, 1995].

An alternative approach is to derive the necessary statistics, i.e., p and n, the number of positive and negative examples that are covered by the rule, and P and N, the total number of positive and negative examples in the dataset, from numerical data, so that heuristics that are known to work well for classification can be directly used on regression problems. A simple strategy that employs this approach is to discretize the target value using equal-width, equal-frequency, or the search-based approach proposed in [Torgo and Gama, 1996].

However, the use of class-discretization as a pre-processing step makes it harder to fine-tune the class values to the context provided by the examples, just as a pre-discretization of attributes is less flexible than the dynamic discretization that is commonly used in rule learning or decision-tree algorithms. Therefore, we propose an alternative approach that allows to dynamically derive positive and negative examples from regression data.

Each (complete or incomplete) rule $\mathbf{r}$ is associated with a numerical target value $y_{\mathbf{r}}$ which will be predicted for all examples that are covered by this rule. Obvious candidates for determining $y_{\mathbf{r}}$ are the mean and the median of the covered examples. We chose the median because it is more robust against outliers. Our goal is that the covered examples are close to this predicted value. In fact, this is the key motivation behind all approaches that evaluate candidate conditions by the reduction of the variance among the covered examples. We implement this idea in a different way by not directly using the standard deviation as a measure for the rule quality, but by labeling all examples that are within the standard deviation as positive, and all examples that are outside the standard deviation as negative. This allows to evaluate the rule using standard classification heuristics such as those defined in Figure 1 for guiding the learner. More precisely, if the distance between the target value and the predicted value is below a certain threshold, this example is considered to be positive and if it is above the threshold it would be a negative example, i.e.,

$$class(x) = \begin{cases} positive & \text{if } |y - y_{\mathbf{r}}| \leq t_{\mathbf{r}} \\ negative & \text{if } |y - y_{\mathbf{r}}| > t_{\mathbf{r}} \end{cases} \qquad (1)$$

where x is the current example, y is the true value of the example x, $y_{\mathbf{r}}$ is the value predicted by rule $\mathbf{r}$ and $t_{\mathbf{r}}$ is a threshold. Thus, the total number of positive and negative examples is defined as

$$P_{\mathbf{r}} = \sum_{i=1}^{m} \mathbf{1}(|y_i - y_{\mathbf{r}}| \leq t_{\mathbf{r}}); \qquad N_{\mathbf{r}} = m - P_{\mathbf{r}} \qquad (2)$$

where m = number of examples, and $\mathbf{1}(.)$ is the indicator function.

All possible conditions l are then evaluated by forming the refined rule $\mathbf{r}' = \mathbf{r} \cup l$ and counting how many positive ($p_{\mathbf{r},l}$) and negative ($n_{\mathbf{r},l}$) examples are covered by $\mathbf{r}'$. Among them, the condition that maximizes a given rule learning heuristic such as those shown in Figure 1 is chosen.

Note that all counts are indexed with the rule $\mathbf{r}$. This is because both the predicted value $y_{\mathbf{r}}$ and the threshold $t_{\mathbf{r}}$ may change with each refinement step. As a consequence, the labels of the examples and thus any of the above counts may change after $\mathbf{r}$ is replaced with $\mathbf{r}'$. Note, however, that the positive examples provide a focus towards the median of a rule, which makes it more likely that the label of examples changes from positive to negative than in the opposite direction.

One can think of different ways for defining the threshold $t_{\mathbf{r}}$. As motivated above, the standard deviation $\sigma_{\mathbf{r}}$ of the examples that are currently covered by the rule is a natural choice. We also experimented with a factor that slightly reduces or enlarges the standard deviation. Section 6 shows the results for three thresholds, $t_{\mathbf{r}} = 0.95 \cdot \sigma_{\mathbf{r}}$, $t_{\mathbf{r}} = \sigma_{\mathbf{r}}$, and $t_{\mathbf{r}} = 1.05 \cdot \sigma_{\mathbf{r}}$. We also considered asymmetric distributions

name	# nominal	# numeric	# instances	# values
auto-mpg	3	4	398	129
auto-price	1	14	159	145
auto93	6	16	93	81
cloud	2	4	108	94
compressive	0	8	1030	845
concrete	0	9	103	83
cpu	1	6	209	104
diabetes	0	2	43	20
echo-month	3	6	130	53
housing	1	12	506	229
machine	0	6	209	116
meta	2	19	528	436
pyrim	0	27	74	63
strike	1	5	625	358
triazines	0	60	186	102
veteran	4	3	137	101

Table 1: Overview of the databases. For each dataset, we show the number of nominal and numeric attributes, the number of instances, and the number of distinct target values.

where we have two different thresholds, one for $y_{\mathbf{r}} < y$, and one for $y_{\mathbf{r}} > y$, but this did not improve the results.

As in the classification setting, the refinement of a candidate rule continues until no more negative examples are covered, and the best rule encountered in this process is returned as the final rule. Depending on the selected heuristic, this is not necessarily the last rule.

Finally, we have to define a stopping criterion that stops the induction of rules in general. A naive method to do so is to stop the induction of additional rules when a certain amount of examples is covered by rules. We decided to stop learning rules as soon as 90% or more of the examples are covered by rules. While we did not yet perform a thorough evaluation of this parameter, it was chosen according to our experience with a different covering-based regression rule learner [Janssen and Fürnkranz, 2010b]. We only confirmed in a few informal experiments that 90% is a reasonable choice. All remaining examples will be covered by a default rule, which will predict the median value of all uncovered training examples.

5 Experimental setup

In the following, we compare our algorithm to a variety of other algorithms, both state-of-the-art regression rule learners and statistical regression learners, all in their implementations in the WEKA data mining library [Witten and Frank, 2005]. The rule learning algorithms include M5RULES, which applies the separate-and-conquer technique and generates a tree in each iteration and derives regression rules from these model trees or (using the option -R) from regression trees, and REGENDER [Dembczyński *et al.*, 2008] an ensemble-based rule learning algorithm, which we used in its default configuration in WEKA (learning 50 rules) and the setting recommended by its authors (learning 200 rules, different loss function, and different optimization technique). In addition, we also compare our dynamic approach to regression by classification to a static approach. For this purpose, we used the standard approach implemented in WEKA using the class `weka.classifiers.meta.RegressionByDiscretization`. However, we used an equal-frequency instead of the default equal-width discretization of the class variable, because the former seemed to work better. As a learning algorithm we used the same rule learning algorithm upon which our dynamic approach is based, with the same four learning heuristics shown in Figure 1. As representatives for statistical regression algorithms we used *linear regression*, a *multi-layer perceptron* (MLP), and *support-vector regression*.

Table 1 shows the 16 datasets that were used in the experiments. The datasets can be downloaded from the UCI Repository [Frank and Asuncion, 2010] or at Luís Torgo's webpage.[1] The table reports the number of instances and numerical/nominal attributes and also the number of distinct values. For the selection, we focused on datasets that have a reasonable number of different values in the class variable, in order to avoid any unfair advantage that rule-based methods might have on data sets with a numerical data variable with only a few different values.

The primary method to evaluate the algorithms is the *relative root mean squared error (rrmse)* obtained by a 1x10 cross validation implemented in WEKA [Witten and Frank, 2005]. For space restrictions, we cannot report results on individual datasets in this paper. Instead, we report average results over all 16 datasets. We are aware of the problems that come with averaging results over many different domains (i.e., some databases may be outliers with huge variance compared to the majority of the other datasets) and hence also report the average ranks of the algorithms in the tables. Where applicable, we also test for statistical significance using a Friedman-Test with a post-hoc Nemenyi-Test as suggested in [Demšar, 2006]. The resulting CD-charts give insights how good the algorithms perform by evaluating their ranking independently from using average accuracy.

Dynamic Regression by Classification					
factor	heuristic	*rrmse*	rank	# rules	# conds
0.95	wra	0.752	8.63	15.06	38.31
0.95	lap	0.784	11.19	11.25	13.88
0.95	corr	0.726	6.50	10.13	24.63
0.95	rcm	0.780	9.81	19.06	34.25
1.00	wra	0.764	10.06	17.06	47.81
1.00	lap	0.774	10.63	10.19	12.50
1.00	corr	0.753	8.38	9.25	22.06
1.00	rcm	0.767	9.50	19.06	35.75
1.05	wra	0.780	13.13	13.19	34.19
1.05	lap	0.772	10.19	9.69	11.81
1.05	corr	0.796	12.88	10.25	33.31
1.05	rcm	0.775	9.75	19.44	37.56
Static Regression by Classification					
# classes	heuristic	*rrmse*	rank	# rules	# conds
5	wra	0.883	18.25	5.63	20.75
5	lap	0.857	14.75	84.56	197.44
5	corr	0.844	15.13	28.06	84.00
5	rcm	0.852	16.63	22.88	68.00
10	wra	0.930	18.69	6.06	23.13
10	lap	0.872	17.00	138.44	339.25
10	corr	0.864	15.88	49.31	167.25
10	rcm	0.901	17.94	20.75	67.31
20	wra	0.965	20.81	10.06	36.56
20	lap	0.872	18.06	177.44	423.63
20	corr	0.862	17.81	95.13	295.00
20	rcm	0.928	19.13	33.19	102.13
Other Rule-Based Regression algorithms					
algorithm		*rrmse*	rank	# rules	# conds
REGENDER (50)		0.768	9.38	50.00	190.00
M5RULES -R		0.773	10.44	6.19	14.94

Table 2: Evaluation of dynamic regression by classification (top), static regression by classification (bottom), and two other rule-based learning algorithms. Shown is the average *rrmse*, the average rank among these 26 algorithms, and the average number of rules and conditions of the learned concepts.

6 Results

Table 2 shows the results of various parametrizations of our approach in comparison to the static regression by equal-frequency discretization. We tried different variants using 5, 10, and 20 classes. For comparison, we also show the results of two other regression-based rule learning algorithms, namely M5RULES [Holmes *et al.*, 1999] used with the option `-R` so that it predicts constant values in the rule head, and REGENDER [Dembczyński *et al.*, 2008] in its default configuration which learns 50 rules, a number that is still comparable to the number of rules learned by our algorithms. Other settings for these algorithms are evaluated further below.

It can be clearly seen that all dynamic versions outperform all static versions in all configurations. The best static version has an average rank of 14.75, whereas the worst dynamic version has an average rank of 13.13. A Friedman test can reject the null hypothesis that all algorithms are equal with high certainty ($p \ll 0.01$). The post-hoc Nemenyi test shows that the best configuration (dynamic regression using correlation as a search heuristic and a factor of 0.95) is significantly better than all but two static configurations at a significance level of 0.1 ($CD = 9.354$), and better than seven at the significance level of 0.01 ($CD = 11.125$). However, it should be noted that our rule learning algorithm does not produce probability distributions, which could improve its performance.

Somewhat surprisingly, the versions using more classes tend to have a worse performance than the versions with fewer classes. Apparently, the advantage of a finer grained discretization of the target variable is outweighed by the disadvantage that problems with multiple classes are harder to learn for the underlying rule learner. This is also confirmed by the number of rules the algorithms learned. Here, when learning 20 classes the number of rules is the highest among all configurations.

With respect to the different configurations of the dynamic approach, it seems that lower versions of the factor that is applied to the standard deviation seem to yield a somewhat better performance.

Among the four heuristics, *correlation* proves to be the

[1] `http://www.liaad.up.pt/~ltorgo/Regression/DataSets.html`

algorithm	*rrmse*	rank	# rules	# conds
Regular	0.726	7.06	10.13	24.63
Bagged (10)	0.671	5.88	97.94	245.81
Bagged (20)	0.659	4.94	186.75	451.25
Bagged (50)	0.658	4.63	465.88	1146.56
Linear Regression	0.651	4.31	—	—
MLP	0.746	5.88	—	—
SVM Regression	0.673	5.19	—	—
REGENDER (200)	0.679	4.50	200.00	1163.63
M5RULES	0.604	2.63	2.94	5.38

Table 3: Comparison of a bagged version to other types of regression algorithms

best choice. This confirms previous results on classification datasets [Janssen and Fürnkranz, 2010a]. However, it seems that the good performance of the *relative cost measure* does not materialize in this setting (or at least not with the parameter setting that performed well in classification). In this context, a surprising observation is that the complexities of the regression theories do not correspond to their known behavior: while the expectations that *laplace* overfits and *wra* overgeneralizes are clearly met in classification, this cannot be observed in the regression rule sets. In fact, with the dynamic approach, *laplace* finds fewer rules and conditions than *wra*.

Interestingly, the *rrmse* monotonically decreases when the factor attached to the standard deviation is increased for *laplace*, while it increases for *wra*. This means that evaluating rules with *wra* seems to decrease performance with a growing number of positive examples, whereas the performance of *laplace* seems to improve. A possible explanation could be that a narrower range of positive examples may help to prevent overfitting.

Overall, the performance of the dynamic class discretization is comparable to the performance of other rule-based learning algorithms, such as M5RULES with constant predictions or REGENDER with low numbers of rules. With the best-performing heuristic, *correlation*, a slight (but not significant) advantage is noticeable.

However, it must be noted that the overall performance of the algorithm does not match the performance of other standard regression algorithms, which learn more elaborate models than the piece-wise constant functions that are learned by the algorithms tested in Table 2. In order to see this effect, we can compare the performance of M5RULES with the parameter setting `-R`, which predicts constant values in its rule head, to the performance of its default configuration which is able to predict linear models in the rule heads (shown at the bottom of Tables 2 and 3 respectively). In order to maximize predictive performance, an adaptation of our approach for the induction of linear models seems necessary, and is subject to future work.

A different approach to weaken the bias of the piece-wise constant prediction functions is to rely on ensemble techniques, which is the key idea of the approach taken by REGENDER. In order to evaluate the potential of our approach in such a setting, we used bagging with best learner determined in the previous experiment (using correlation as a heuristic and a threshold of $t_{\mathbf{r}} = 0.95 \cdot \sigma$). Table 3 shows the results for learning 10, 20, and 50 rule sets. More iterations did not yield further improvements. We compared this setting to REGENDER learning 200 rules, and to linear regression, support-vector regression, multi-layer perceptrons, and M5RULES. While the linear or piece-wise linear models seem to be somewhat better suited for these problems, no significant difference between the models can be observed.

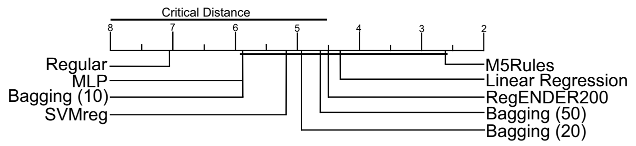

Figure 2: Comparison of the algorithms shown in Table 3 against each other with the Nemenyi test. Groups of algorithms that are not significantly different (at $p = 0.01$) are connected.

The value of the Friedman statistic is 3.879 which exceeds the critical value of 2.663 for $p = 0.01$. The post-hoc Nemenyi test yields critical distance values of 3.476 for $p = 0.01$ and 2.764 for $p = 0.1$ (Figure 2 illustrates this situation for $p = 0.01$). Thus, no significant difference can be shown between the best algorithm M5RULES and the bagged versions of our algorithms with 20 or more theories. Conversely, the test also does not show a significant difference between our regular dynamic regression by classification approach and any of the other regression algorithms except the model-based rule-learning algorithm M5RULES. However, this result should be put into perspective, because in Figure 2 we only included the best-performing parameter setting of Table 2. It seems obvious that the performance of the algorithm without bagging is substantially worse than the performance of the other algorithms.

7 Conclusions and Future Work

In this paper, we proposed a new technique for heuristic learning of regression rules. The key idea is to dynamically transform the regression approach into a classification setting by defining positive examples near the current mean as positive, and those further away as negative. In this way, classification rule heuristics can be directly used for learning regression rules. The algorithm clearly outperforms a static discretization using the same rule learner, and performs at least equally to other state-of-the-art regression rule learning algorithms. While it does not reach the performance of other algorithms that use more expressive models, we also showed that the use of bagging results in an algorithm that is en par with other rule-based regression ensembles and statistical regression techniques.

We expect that the algorithm still can be improved by a more systematic investigation of its parameters such as the factor applied to the standard deviation, or the percentage of examples that need to be covered by rules. The current setting setting of 90% is rather intuitive and perhaps not the best choice. Similarly, a more systematic evaluation of classification rule heuristics in the context of regression tasks could lead to better performance. Other works [Lavrač *et al.*, 1999;

Fürnkranz and Flach, 2005] show that different heuristics are suited for different applications, and it is still not well explored which heuristic is the best choice. In the case of parametrized heuristics, such as the *relative cost metric*, the parameter can also be tuned to optimize the performance on the datasets [Janssen and Fürnkranz, 2010a].

A promising path to optimize the predictive performance of the algorithm would be to replace the constant predictions in the rule heads with linear models. The results of M5RULES, where we included both a version with constant predictions and a version with linear models into the experimental evaluation, show that this should lead to drastic performance improvements. However, much of the interpretability of the rule set would be lost when doing so.

Acknowledgments

This research was supported by the *German Science Foundation (DFG)* under grant no. FU 580/2-2. We would like to thank the reviewers for helpful suggestions and directions for future work.

References

[Frank and Asuncion, 2010] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[Breiman *et al.*, 1984] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth & Brooks, Pacific Grove, CA, 1984.

[Clark and Niblett, 1989] P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.

[Dembczyński *et al.*, 2008] K. Dembczyński, W. Kotłowski, and R. Słowiński. Solving regression by learning an ensemble of decision rules. In *Proc. 9th International Conference on Artificial Intelligence and Soft Computing (ICAISC-08)*, pp. 533–544, Zakopane, Poland, 2008. Springer-Verlag.

[Demšar, 1999] D. Demšar. Obravnavanje numericnih problemov z induktivnim logicnim programiranjem. Master's thesis, Faculty of Computer and Information Science, University of Ljubljana, Slovenia, 1999. In Slovene.

[Demšar, 2006] J. Demšar. Statistical comparisons of classifiers over multiple datasets. *Journal of Machine Learning Research*, 7:1–30, 2006.

[Friedman and Popescu, 2008] J. H. Friedman and B. E. Popescu. Predictive learning via rule ensembles. *Annals Of Applied Statistics*, 2:916, 2008.

[Fürnkranz and Flach, 2005] J. Fürnkranz and P. Flach. ROC 'n' rule learning – Towards a better understanding of covering algorithms. *Machine Learning*, 58(1):39–77, 2005.

[Fürnkranz, 1999] J. Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54, February 1999.

[Holmes *et al.*, 1999] G. Holmes, M. Hall, and E. Frank. Generating rule sets from model trees. In *Proc. 12th Australian Joint Conference on Artificial Intelligence (AI-99)*, pp. 1–12. Springer, 1999.

[Janssen and Fürnkranz, 2010a] F. Janssen and J. Fürnkranz. On the quest for optimal rule learning heuristics. *Machine Learning*, 78(3):343–379, March 2010.

[Janssen and Fürnkranz, 2010b] F. Janssen and J. Fürnkranz. Separate-and-conquer regression. In *Proc. German Workshop on Lernen, Wissen, Adaptivität (LWA-10)*, pp. 81–89, 2010.

[Karalič and Bratko, 1997] A. Karalič and I. Bratko. First order regression. *Machine Learning*, 26(2-3):147–176, 1997.

[Langford *et al.*, 2006] J. Langford, R. Oliveira, and B. Zadrozny. Predicting conditional quantiles via reduction to classification. In *Proc. 22nd Annual Conference on Uncertainty in Artificial Intelligence (UAI-06)*, pp. 257–264, Arlington, Virginia, 2006. AUAI Press.

[Lavrač *et al.*, 1999] N. Lavrač, P. Flach, and B. Zupan. Rule evaluation measures: A unifying view. In *Proc. 9th International Workshop on Inductive Logic Programming (ILP-99)*, pp. 174–185. Springer-Verlag, 1999.

[Quinlan, 1992] J. R. Quinlan. Learning with continuous classes. In *Proc. 5th Australian Joint Conference on Artificial Intelligence (AI-92)*, pp. 343–348, Singapore, 1992. World Scientific.

[Smola and Schölkopf, 2004] A. J. Smola and B. Schölkopf. A tutorial on support vector regression. *Statistics and Computing*, 14:199–222, August 2004.

[Todorovski *et al.*, 2000] L. Todorovski, P. Flach, and N. Lavrač. Predictive performance of weighted relative accuracy. In *Proc. 4th European Symposium on Principles of Data Mining and Knowledge Discovery (PKDD-2000)*, pp. 255–264, Lyon, France, 2000. Springer-Verlag.

[Torgo and Gama, 1996] L. Torgo and J. Gama. Regression by classification. In *Proc. 13th Brazilian Symposium on Artificial Intelligence (SBIA-96)*, pp. 51–60. Curitiba, Brazil, 1996. Springer-Verlag.

[Torgo, 1995] L. Torgo. Data fitting with rule-based regression. In *Proc. 2nd International Workshop on Artificial Intelligence Techniques (AIT'95)*, 1995.

[Ženko *et al.*, 2005] B. Ženko, S. Džeroski, and J. Struyf. Learning predictive clustering rules. In *Proc. 4th International Workshop on Knowledge Discovery in Inductive Databases*, pp. 234–250. Springer, 2005.

[Wang and Witten, 1997] Y. Wang and I. H. Witten. Induction of model trees for predicting continuous classes. In *Poster papers of the 9th European Conference on Machine Learning*, pp. 128–137, Prague, Czech Republic, 1997.

[Weiss and Indurkhya, 1995] S. M. Weiss and N. Indurkhya. Rule-based machine learning methods for functional prediction. *Journal of Artificial Intelligence Research*, 3:383–403, 1995.

[Witten and Frank, 2005] I. H. Witten and E. Frank. *Data Mining — Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers, 2nd edition, 2005.