Applied Intelligence 7, 39–55 (1997)
© 1997 Kluwer Academic Publishers. Manufactured in The Netherlands.

Overcoming the Myopia of Inductive Learning Algorithms with RELIEFF

IGOR KONONENKO, EDVARD ŠIMEC AND MARKO ROBNIK-ŠIKONJA
University of Ljubljana, Faculty of Computer and Information Science, Tržaška 25, SI-1001 Ljubljana, Slovenia
igor.kononenko@fri.uni-lj.si

Abstract. Current inductive machine learning algorithms typically use greedy search with limited lookahead. This prevents them to detect significant conditional dependencies between the attributes that describe training objects. Instead of myopic impurity functions and lookahead, we propose to use RELIEFF, an extension of RELIEF developed by Kira and Rendell [10, 11], for heuristic guidance of inductive learning algorithms. We have reimplemented Assistant, a system for top down induction of decision trees, using RELIEFF as an estimator of attributes at each selection step. The algorithm is tested on several artificial and several real world problems and the results are compared with some other well known machine learning algorithms. Excellent results on artificial data sets and two real world problems show the advantage of the presented approach to inductive learning.

Keywords: learning from examples, estimating attributes, impurity function, RELIEFF, empirical evaluation

1. Introduction

Inductive learning algorithms typically use a greedy search strategy to overcome the combinatorial explosion during the search for good hypotheses. The heuristic function that estimates the potential successors of the current state in the search space has a major role in the greedy search. Current inductive learning algorithms use variants of impurity functions like information gain, gain ratio [25], gini-index [1], distance measure [16], *j*-measure [30], and MDL [14]. However, all these measures assume that attributes are conditionally independent given the class and therefore in domains with strong conditional dependencies between attributes the greedy search has poor chances of revealing a good hypothesis.

Kira and Rendell [10, 11] developed an algorithm called RELIEF, which seems to be very powerful in estimating the quality of attributes. For example, in the parity problems of various degrees with a significant number of irrelevant (random) additional attributes RELIEF is able to correctly estimate the relevance of all attributes in a time proportional to the number of attributes and the square of the number of training instances (this can be further reduced by limiting the number of iterations in RELIEF). While the original RELIEF can deal with discrete and continuous attributes, it cannot deal with incomplete data and is limited to two-class problems only. We developed an extension of RELIEF called RELIEFF that improves the original algorithm by estimating probabilities more reliably and extends it to handle incomplete and multi-class data sets while the complexity remains the same.

RELIEFF seems to be a promising heuristic function that may overcome the myopia of current inductive learning algorithms. Kira and Rendell used RELIEF as a preprocessor to eliminate irrelevant attributes from data description before learning. RELIEFF is general, relatively efficient, and reliable enough to guide the search in the learning process. In this paper a reimplementation of Assistant learning algorithm for top down induction of decision trees [4] is described, named Assistant-*R*. Instead of information gain, Assistant-*R* uses RELIEFF as a heuristic function for estimating the attributes' quality at each step during the tree generation. Experiments on a series of artificial and real-world data sets are described and the results obtained using RELIEFF as a selection criterion are compared to results of some other approaches. The following approaches are compared:

- the use of information gain as a selection criterion;
- LFC [27, 28] that tries to overcome the myopia of information gain with a limited lookahead;
- the naive Bayesian classifier, that assumes conditional independence of attributes;
- the *k*-nearest neighbors algorithm.

The paper is organized as follows. In the next section, the original RELIEF is briefly described along with its interpretation and its extended version RELIEFF. In Section 3, we present the reimplementation of Assistant called Assistant-*R*. In Section 4.1 we briefly describe the other algorithms used in our experiments. In Section 4.2 we describe the experimental methodology. Section 5 describes experiments, and compares the results of the different algorithms. We show that Assistant-*R* performs at least as well as Assistant-*I* and sometimes much better. In conclusion, the potential breakthroughs are discussed on the basis of the excellent results on artificial data sets. Finally, integration of the compared algorithms is proposed.

2. RELIEFF

2.1. RELIEF

The key idea of RELIEF is to estimate attributes according to how well their values distinguish among the instances that are near to each other. For that purpose, given an instance, RELIEF searches for its two nearest neighbors: one from the same class (called *nearest hit*) and the other from a different class (called *nearest miss*). The original algorithm of RELIEF [10, 11] randomly selects *n* training instances, where *n* is the user-defined parameter. The algorithm is given in Fig. 1.

Function diff(Attribute, Instance1, Instance2) calculates the difference between the values of Attribute for two instances. For discrete attributes the difference is either 1 (the values are different) or 0 (the values are equal), while for continuous attributes the difference is the actual difference normalized to the interval [0, 1]. Normalization with *n* guarantees all weights $W[A]$ to be in the interval $[-1, 1]$, however, normalization with *n* is an unnecessary step if $W[A]$ is to be used for relative comparison among attributes.

```
1. set all weights W[A] := 0.0;
2. for i := 1 to n do
3. begin
4.     randomly select an instance R;
5.     find nearest hit H and nearest miss M;
6.     for A := 1 to #all_attributes do
7.         W[A] := W[A] - diff(A,R,H)/n
8.                      + diff(A,R,M)/n;
9. end;
```

Figure 1. The basic algorithm of RELIEF.

The weights are estimates of the quality of attributes. The rationale of the formula for updating the weights is that a good attribute should have the same value for instances from the same class (subtracting the difference diff(A, R, H)) and should differentiate between instances from different classes (adding the difference diff(A, R, M)).

The function diff is used also for calculating the distance between instances to find the nearest neighbors. The total distance is simply the sum of differences over all attributes. In fact original RELIEF uses the squared difference, which for discrete attributes is equivalent to diff. In all our experiments, there was no significant difference between results using diff or squared difference. If N is the number of all training instances then the complexity of the above algorithm is $O(n \times N \times \#\text{all\_attributes})$.

2.2. Interpretation of RELIEF's Estimates

The following derivation shows that RELIEF's estimates are strongly related to impurity functions. It is obvious that RELIEF's estimate $W[A]$ of attribute A is an approximation of the following difference of probabilities:

$$\begin{aligned} W[A] = {} & P(\text{different value of } A \mid \text{nearest instance from different class}) \\ & - P(\text{different value of } A \mid \text{nearest instance from same class}) \end{aligned} \tag{1}$$

If we eliminate from (1) the requirement that the selected instance is *the nearest*, the formula becomes:

$$\begin{aligned} W'[A] = {} & P(\text{different value of } A \mid \text{different class}) \\ & - P(\text{different value of } A \mid \text{same class}) \end{aligned} \tag{2}$$

If we rewrite

$$\begin{aligned} P_{\text{eqval}} &= P(\text{equal value of } A) \\ P_{\text{samecl}} &= P(\text{same class}) \\ P_{\text{samecl}|\text{eqval}} &= P(\text{same class} \mid \text{equal value of } A) \end{aligned}$$

we obtain using Bayes rule:

$$W'[A] = \frac{P_{\text{samecl}|\text{eqval}} P_{\text{eqval}}}{P_{\text{samecl}}} - \frac{(1 - P_{\text{samecl}|\text{eqval}}) P_{\text{eqval}}}{1 - P_{\text{samecl}}}$$

For sampling with replacement in strict sense the following equalities hold:

$$P_{\text{samecl}} = \sum_C P(C)^2$$

$$P_{\text{samecl}|\text{eqval}} = \sum_V \left(\frac{P(V)^2}{\sum_V P(V)^2} \times \sum_C P(C \mid V)^2 \right)$$

Using the above equalities we obtain:

$$\begin{aligned} W'[A] &= \frac{P_{\text{eqval}} \times \text{Ginigain}'(A)}{P_{\text{samecl}}(1 - P_{\text{samecl}})} \\ &= \text{const} \times \sum_V P(V)^2 \times \text{Ginigain}'(A) \qquad (3) \end{aligned}$$

where

$$\begin{aligned} \text{Ginigain}'(A) = \sum_V \left(\frac{P(V)^2}{\sum_V P(V)^2} \times \sum_C P(C \mid V)^2 \right) \\ - \sum_C P(C)^2 \qquad (4) \end{aligned}$$

is highly correlated with the gini-index gain [1] for classes C and values V of attribute A. The difference is that instead of factor

$$\frac{P(V)^2}{\sum_V P(V)^2}$$

the gini-index gain uses

$$\frac{P(V)}{\sum_V P(V)} = P(V)$$

Equation (3) shows strong relation of RELIEF's weights with the gini-index gain. The probability $\sum_V P(V)^2$ that two instances have the same value of attribute A in Eq. (3) is a kind of normalization factor for multi-valued attributes. Impurity functions tend to overestimate multi-valued attributes and various normalization heuristics are needed to avoid this tendency (e.g., gain ratio [25], distance measure [16], and binarization of attributes [4]). Equation (3) shows that RELIEF exhibits an implicit normalization effect.

Another deficiency of gini-index gain is that its values tend to decrease with the increasing number of classes [14]. Denominator which is constant factor in Eq. (3) for a given attribute again serves as a kind of normalization and therefore RELIEF's estimates do not exhibit such strange behavior as gini-index gain does.

The above derivation eliminated the "*nearest instance*" condition from the probabilities. If we put it back we can interpret RELIEF's estimates as the average over local estimates in smaller parts of the instance space. This enables RELIEF to take into account the context of other attributes, i.e., the conditional dependencies between attributes given the class value which can be detected in the context of locality. From the global point of view, these dependencies are hidden due to the effect of averaging over all training instances, and exactly this makes impurity functions myopic. Impurity functions use correlation between the attribute and the class disregarding the context of other attributes. This is the same as using the global point of view and disregarding the local peculiarities.

The example data set given in Table 1 illustrates the difference between myopic estimation functions and RELIEF. We have three attributes and eight training instances. The class value is determined with XOR function on attributes $A1$ and $A2$, while the third attribute $A3$ is randomly generated. RELIEF (Eq. (1)) correctly estimates that attributes $A1$ and $A2$ are the most important while the contribution of attribute $A3$ is poor. On the other hand, $W'[A]$ (Eq. (3)), Ginigain$'$ (Eq. (4)), original gini-index gain [1], information gain [9], gain ratio [25], and distance measure [16] estimate that the contribution of $A3$ is the highest while attributes $A1$ and $A2$ are estimated as completely irrelevant.

Hong [8] developed a procedure similar to RELIEF for estimating the quality of attributes, where he directly emphasizes the use of contextual information. The difference to RELIEF is that his approach uses only information from nearest misses and ignores nearest

Table 1. Example data set and the estimated quality of attributes.

Function	$A1$	$A2$	$A3$	Class
	1	0	1	1
	1	0	0	1
	0	1	0	1
	0	1	1	1
	0	0	0	0
	0	0	0	0
	1	1	1	0
	1	1	0	0
RELIEF $= W[A]$ (1)	0.542	0.458	-0.750	
$W'[A]$ (3)	0.000	0.000	0.063	
Ginigain$'$ (4)	0.000	0.000	0.029	
gini index gain [1]	0.000	0.000	0.033	
information gain [9]	0.000	0.000	0.049	
gain-ratio [25]	0.000	0.000	0.051	
distance [16]	0.000	0.000	0.026	

hits. Besides, Hong uses the normalization to penalize the contribution of nearest misses that are far away from a given instance.

2.3. *Extensions of RELIEF*

The original RELIEF can deal with discrete and continuous attributes. However, it cannot deal with incomplete data and is limited to two-class problems only. Equation (1) is of crucial importance for any extensions of RELIEF. It turned out that the extensions of RELIEF are not straightforward unless we realized that RELIEF in fact approximates probabilities. The extensions should be designed in such a way that those probabilities are reliably approximated. We developed an extension of RELIEF, called RELIEFF, that improves the original algorithm by estimating probabilities more reliably and extends it to deal with incomplete and multi-class data sets. A brief description of the extensions follows.

Reliable Probability Approximation. The parameter n in the algorithm RELIEF, described in Section 2.1, represents the number of instances for approximating probabilities in Eq. (1). The larger n implies more reliable approximation. The obvious choice, adopted in RELIEFF for relatively small number of training instances (up to one thousand), is to run the outer loop of RELIEF over all available training instances.

The selection of the nearest neighbors is of crucial importance in RELIEF. The purpose is to find the nearest neighbors with respect to important attributes. Redundant and noisy attributes may strongly affect the selection of the nearest neighbors and therefore the estimation of probabilities with noisy data becomes unreliable. To increase the reliability of the probability approximation RELIEFF searches for k nearest hits/misses instead of only one near hit/miss and averages the contribution of all k nearest hits/misses. It was shown that this extension significantly improves the reliability of estimates of attributes' qualities [13]. To overcome the problem of parameter tuning, in all our experiments k was set to 10 which, empirically, gives satisfactory results. In some problems significantly better results can be obtained with tuning (as is typical for the majority of machine learning algorithms).

Incomplete Data. To enable RELIEF to deal with incomplete data sets, the function diff(Attribute, Instance1, Instance2) in RELIEFF is extended to missing values of attributes by calculating the probability that two given instances have different values for the given attribute:

- if one instance (e.g., $I1$) has unknown value:

$$\text{diff}(A, I1, I2) = 1 - P(\text{value}(A, I2) \mid \text{class}(I1))$$

- if both instances have unknown value:

$$\text{diff}(A, I1, I2) = 1 - \sum_{V}^{\#\text{values}(A)} (P(V \mid \text{class}(I1)) \times P(V \mid \text{class}(I2)))$$

The conditional probabilities are approximated with relative frequencies from the training set.

This approach assumes that conditional probabilities of attribute-values given the class are applicable without the context of any other attribute. This may in some cases be too naive, however including the context of other atributes is far too inefficient.

Multi-Class Problems. Kira and Rendell [10, 11] claim that RELIEF can be used to estimate the attributes' qualities in data sets with more than two classes by splitting the problem into a series of

2-class problems. This solution seems unsatisfactory (in Section 4.1 we discuss the performance of this approach and compare it with the extension described below). To use it in practice, RELIEF should be able to deal with multi-class problems without any prior changes in the knowledge representation that could affect the final outcomes.

Instead of finding one near miss M from a different class, RELIEFF searches for k near misses $M_i(C)$, $i = 1 \cdots k$ for each different class C and averages their contribution for updating the estimate $W[A]$. The average is weighted with the prior probability of each class:

$$\begin{aligned} W[A] &:= W[A] \\ &\quad - \sum_{i=1}^{k} \frac{\mathrm{diff}(A, R, H_i)}{n \times k} + \sum_{C \neq \mathrm{class}(R)} \sum_{i=1}^{k} \\ &\quad \times \left[\frac{P(C)}{1 - P(\mathrm{class}(R))} \times \frac{\mathrm{diff}(A, R, M_i(C))}{n \times k} \right] \end{aligned}$$

The idea is that the algorithm should estimate the ability of attributes to separate each pair of classes regardless of which two classes are closest to each other. The normalization of prior probabilities of classes is necessary as k near misses from each different class would tend to exaggerate the influence of classes with small number of cases.

Note that the time complexity of RELIEFF is $O(N^2 \times \#\text{attributes})$, where N is the number of training instances.

2.4. RELIEFF's Estimates and Attribute's Quality

To estimate the contribution of parameter k (# nearest hits/misses) on RELIEFF's estimates of attribute's quality Kononenko [13] compared the *intended information gain* of attributes with the estimates, generated by RELIEFF, by calculating the standard linear *correlation coefficient*. The correlation coefficient can show how are the intended quality and the estimated quality of attributes related.

A typical graph for data sets with conditionally independent attributes and with strongly dependent attributes (parity problems of various degrees) is shown in Fig. 2. For conditionally independent attributes, the quality of the estimate monotonically increases with the number of nearest neighbors. For conditionaly dependent attributes, the quality increases up to a maximum but later decreases as the number of nearest neighbors exceeds the number of instances that belong to the same peak in the distribution space for a given class.

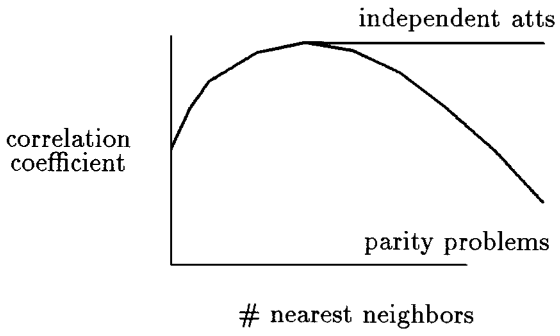


Figure 2. The correlation of the RELIEFF's estimates with the intended quality of attributes on data sets with conditionally independent and strongly dependent attributes.

Note that, if attributes were evaluated with the myopic impurity functions, like the gini-index and the information gain, the quality of the estimates would be high for conditionally independent attributes and poor for strongly dependent attributes. This corresponds to the estimates by RELIEFF with very large number of nearest hits/misses.

To test the effect of the normalization factor in Eq. (3) we run RELIEFF also on one well known medical data set, "primary tumor", described in Section 5.3. The major difference between the estimates by impurity functions and the estimates by RELIEFF in the "primary tumor" problem is in the estimates of two most significant attributes. Information gain and gini-index overestimate one attribute with 3 values (by the opinion of physicians specialists). RELIEFF and normalized versions of impurity functions correctly estimate this attribute as less important.

3. Assistant-*R*

Assistant-*R* is a reimplementation of the Assistant learning system for top down induction of decision trees [4]. The basic algorithm goes back to CLS (Concept Learning System) developed by Hunt et al. [9] and reimplemented by several authors (see [25] for an overview). In the following we describe the main features of Assistant.

Binarization of Attributes. The algorithm generates binary decision trees. At each decision step the binarized version of each attribute is selected that maximizes the information gain of the attribute. For continuous attributes a decision point is selected that maximizes the attribute's information gain. For discrete attributes a heuristic greedy algorithm is used to find the locally best split of attribute's values into two subsets. The purpose of the binarization is to reduce the replication problem and to strengthen the statistical support for generated rules.

Decision Tree Pruning. Prepruning and postpruning techniques are used for pruning off unreliable parts of decision trees. For prepruning, three user-defined thresholds are provided: minimal number of training instances, minimal attributes information gain and maximal probability of majority class in the current node. For postpruning, the method developed by Niblett and Bratko [22] is used that uses Laplace's law of succession for estimating the expected classification error of the current node commited by pruning/not pruning its subtree.

Incomplete Data Handling. During learning, training instances with a missing value of the selected attribute are weighted with probabilities of each attribute's value conditioned with a class label. During classification, instances with missing values are weighted with unconditional probabilities of attribute's values.

Naive Bayesian Classifier. For each internal node in a decision tree eventually a third successor appears labeled with attribute's values for which no training instances are available. For such "null leaves", the naive Bayesian formula is used to calculate the probability distribution in the leaf by using only attributes that appear in the path from the root to the leaf:

$$P(C \mid A_{\text{root}} \cdots A_{\text{leaf}}) = P(C) \prod_A \frac{P(C \mid A)}{P(C)} \tag{5}$$

Note that this calculation is done off-line, i.e., during the learning phase. For classification, the "null" leaves are already labeled with the calculated class probability distribution and are used for classification in the same manner as ordinary leaves.

The main difference between Assistant and its reimplementation Assistant-*R* is that RELIEFF is used for attribute selection. In addition, wherever appropriate, instead of the relative frequency, Assistant-*R* uses the *m*-estimate of probabilities, which was shown to often significantly increase the performance of machine learning algorithms [2, 3]. For prior probabilities Laplace's law of succession is used:

$$P_a(X) = \frac{N(X) + 1}{N + \#\_\text{of}\_\text{possible}\_\text{outcomes}} \tag{6}$$

where N is the number of all trials and $N(X)$ the number of trials with the outcome X. These prior probabilities are then used in the *m*-estimate of conditional probabilities:

$$P(X \mid Y) = \frac{N(X\&Y) + m \times P_a(X)}{N(Y) + m} \tag{7}$$

The parameter m trades off between the contributions of the relative frequency and the prior probability.

In our experiments, the parameter m was set to 2 (this setting is usually used as default and, empirically, gives satisfactory results [2, 3] although with tuning in some problem domains better results may be expected). The *m*-estimate is used in the naive Bayesian formula (5), for postpruning instead of Laplace's law of succession as proposed by Cestnik and Bratko [3], and for RELIEFF's estimates of probabilities. In Eq. (1) we can use probabilities from the root of the tree as an estimate of prior probabilities for a lower internal node t with $n(t)$ corresponding training instances:

$$\begin{aligned} W[A] &\\ = \Bigg(&\frac{n(t)}{n(t)+m} \times P(\text{diff. val. of } A \mid \text{nearest miss}, t) \\ &+ \frac{m}{n(t)+m} P(\text{diff. val. of } A \mid \text{nearest miss}, \text{root})\Bigg) \\ - \Bigg(&\frac{n(t)}{n(t)+m} \times P(\text{diff. val. of } A \mid \text{nearest hit}, t) \\ &+ \frac{m}{n(t)+m} \times P(\text{diff. val. of } A \mid \text{nearest hit}, \text{root})\Bigg) \end{aligned} \tag{8}$$

4. Experimental Environment

4.1. Algorithms for Comparison

We performed a series of experiments with Assistant-*R* and compared its performance to the following algorithms:

Assistant-*I*: A variant of Assistant-*R* that instead of RELIEFF uses information gain for the selection criterion, as does Assistant. However, the other differences to Assistant remain (*m*-estimate of probabilities). This algorithm enables us to evaluate the contribution of RELIEFF. The parameters for Assistant-I and Assistant-R were fixed throughout the experiments (no prepruning, postpruning with $m = 2$).

LFC: Ragavan et al. [27, 28] use limited lookahead in their LFC (Lookahead Feature Construction) algorithm for top down induction of decision trees to detect significant conditional dependencies between attributes for constructive induction. They show interesting results on some data sets. We reimplemented their algorithm [29] and tested its performance. Our results, presented in this paper, show some drawbacks of the experimental comparison described by Ragavan and Rendell and confirm the advantage of the limited lookahead for constructive induction.

LFC generates binary decision trees. At each node, the algorithm constructs new binary attributes from the original attributes, using logical operators (conjunction, disjunction, and negation). From the constructed binary attributes, the best attribute is selected and the process is recursively repeated on two subsets of training instances, corresponding to the two values of the selected attribute. For constructive induction a limited lookahead is used. The space of possible useful constructs is restricted, due to the geometrical representation of the conditional entropy which is the estimator of the attributes' quality. To further reduce the search space, the algorithm also limits the breadth and the depth of search.

AS LFC uses lookahead it is less myopic than the greedy algorithm of Assistant. The comparison of results may show the performance of the greedy search in combination with RELIEFF versus the lookahead strategy. To make results comparable to Assistant-R we equipped LFC with pruning and probability estimation facilities as described in Section 3. All tests were performed with a default set of parameters (depth of the lookahead 3, beam size 20), although in some domains better results may be obtained by parameter tuning. However, higher values of the parameters may combinatorially increase the search space of LFC, which makes the algorithm impractical.

Naive Bayesian Classifier: A classifier that uses the naive Bayesian formula (5) to calculate the probability of each class given the values of all attributes and assuming the conditional independence of the attributes. A new instance is classified into the class with maximal calculated probability. The m-estimate of probabilities was used and the parameter m was set to 2 in all experiments. The performance of the naive Bayesian classifier can serve as an estimate of the conditional independence of attributes.

***k*-NN:** The k-nearest neighbor algorithm. For a given new instance the algorithm searches for k nearest training instances and classifies the instance into the most frequent class of these k instances. For the k-NN algorithm the same distance measure was used as for RELIEFF (see Section 2.1).

The presented results were obtained with Manhattan-distance. The results using Euclidian distance are practically the same. The best results with respect to parameter k are presented, although for fair comparison such parameter tuning should be allowed only on the training and not the testing sets.

We selected the naive Bayesian classifier and the k-NN algorithm for comparison because they are both well known, simple, and they both perform well in many real-world problems. The performance of these two algorithms may show the nature of the classification problems.

4.2. Experimental Methodology

Each experiment on each data set was performed 30 times by randomly selecting 70% of instances for learning and 30% for testing and the results were averaged. Each system used the same subsets of instances for learning and for testing in order to provide the same experimental conditions. To verify the significance of differences we used the *one-tailed t-test* with $\alpha = 0.0005$ (99.95% confidence level) and the null hypothesis stating that the difference is zero [5]. All the differences in results having the value of statistic t above the threshold ($t > 3.66$) are considered significant.

The exception from the above methodology were the experiments in the finite element mesh design problem, where the experimental methodology was dictated by previous published results, as described in Section 5.4.

Besides the classification accuracy, we measured also the average information score [15]. This measure eliminates the influence of prior probabilities and appropriately treats probabilistic answers of the classifier. The average information score is defined as:

$$\mathrm{Inf} = \frac{\sum_{i=1}^{\#\text{testing instances}} \mathrm{Inf}_i}{\#\text{testing instances}} \tag{9}$$

where the information score of the classification of i-th testing instance is defined by:

$$\mathrm{Inf}_i = -\log_2 P(\mathrm{Cl}_i) + \log_2 P'(\mathrm{Cl}_i) \quad \text{if } P'(\mathrm{Cl}_i) \geq P(\mathrm{Cl}_i),$$

and

$$\mathrm{Inf}_i = -(-\log_2(1 - P(\mathrm{Cl}_i)) + \log_2(1 - P'(\mathrm{Cl}_i))) \quad \text{if } P'(\mathrm{Cl}_i) < P(\mathrm{Cl}_i).$$

Cl_i is the class of the i-th testing instance, $P(\mathrm{Cl})$ is the prior probability of class Cl and $P'(\mathrm{Cl})$ the probability returned by a classifier. If the returned probability of the correct class is greater than the prior probability the information score is positive, as the obtained information is correct. It can be interpreted as the prior information necessary for correct classification minus the posterior information necessary for correct classification. If the returned probability of the correct class is lower than the prior probability the information score is negative, as the obtained information is wrong. It can be interpreted as the prior information necessary for incorrect classification minus the posterior information necessary for incorrect classification.

The main difference between the classification accuracy and the information score can be illustrated with the following example. Let the prior distribution of classes be $P(C_1) = 0.2$ and $P(C_2) = 0.8$ and let the posterior distribution returned by the classifier be $P(C_1) = 0.4$ and $P(C_2) = 0.6$. If the correct class is C_1 then the information score is positive while the classification accuracy treats the given posterior distribution as wrong answer. If the correct class is C_2 then the information score is negative while the classification accuracy treats the given posterior distribution as correct answer.

Classification accuracy may in some special cases exhibit high variance while information score is much more stable. In a very special case where we have a data set with irrelevant attributes and exactly 50% of instances from one class and 50% of instances from the other class, the leave-one-out testing for a probabilistic classifier would give the approximate accuracy of 50%, while for the "default" classifier, that classifies every instance into the majority class, the accuracy would be 0%. A slight modification of the distribution of training instances would drastically change the latter accuracy to approximately 50%. A more drastic modification of the distribution, say 80% of cases for one class and 20% for the other, would increase the accuracy of the "default" classifier to 80%, while the accuracy of the probabilistic classifier would be approximately $0.8 \times 0.8 + 0.2 \times 0.2 = 68\%$. However, for both classifiers the information score would in all scenarios remain approximately 0 bits which would indicate, that both classifiers are unable to extract any useful information from attributes.

5. Experimental Results

In this section we give results on several artificial and real-world data sets. The presentation of the experiments is divided into four parts according to four groups of data sets: artificial data sets with the controlled conditional dependency between attributes, some other benchmark artificial data sets, medical data sets, and other real-world data sets. For each group we give a brief description of data sets followed by the results. The results in tables include averages over several runs and standard errors.

5.1. *Artificial Data Sets*

We generated several data sets in order to compare the performance of various algorithms:

INF1: Domain with three conditionally independent informative binary attributes for each of the three classes and with three random binary attributes. The learner should detect which three attributes are informative which is a relatively easy task. All five algorithms should be able to solve this problem.

INF2: Domain obtained from INF1 by replacing each informative attribute with two attributes whose values define the value of the original attribute with XOR relation. For this problem, the learner should detect six important attributes and the fact that attributes are pairwise strongly conditionally dependent. This is a fairly complex problem and cannot be solved with the myopic heuristics. This data set should show the advantage of LFC and Assistant-R.

TREE: Domain whose instances were generated from a decision tree with 6 internal nodes, each containing a different binary attribute. 5 random binary attributes were added to the description of instances. This problem should be easy for greedy decision tree learning algorithms while other approaches may have difficulties due to an inappropriate knowledge representation of the target concept.

PAR2: Parity problem with two significant binary attributes and 10 random binary attributes. 5% of randomly selected instances were labeled with wrong class. This problem is hard as there is a lot of attributes with equal score when evaluated with a myopic evaluation function, such as information gain.

PAR3: Same as PAR2 except that there were three significant attributes for the parity relation which makes the problem harder.

PAR4: Same as PAR2 except that there were four significant attributes for the parity relation which makes the problem the hardest among the parity problems used in our experiments.

The basic characteristics of the artificial data sets are listed in Table 2. Characteristics include the percentage of the majority class (which can be interpreted as “default accuracy”) and the class entropy which gives an impression of the complexity of the classification problem.

The results of the learning algorithms LFC, Assistant-I and Assistant-R, as well as the naive Bayesian classifier and the k-NN algorithm, are given in Table 3 (classification accuracy) and Table 4 (information score). The results are as expected and show that:

- All classifiers perform well in a (relatively simple) domain with conditionally independent attributes (INF1).
- Both versions of Assistant perform well in the problem of the reconstruction of a decision tree (TREE), while the other classifiers are significantly worse.
- Only Assistant-R and LFC are able to successfully solve the problems with strong conditional dependencies between attributes (INF2, PAR2-4).

Table 2. Basic description of artificial data sets.

Domain	#Class	#Atts.	#Val/att.	# Instances	Maj. class (%)	Entropy (bit)
INF1	3	12	2.0	200	36	1.58
INF2	3	21	2.0	200	36	1.58
TREE	2	11	2.0	200	57	0.99
PAR2	2	12	2.0	200	54	0.99
PAR3	2	13	2.0	200	54	0.99
PAR4	2	14	2.0	400	50	1.00

Table 3. Classification accuracy of the learning systems on artificial data sets.

Domain	LFC	Assistant-I	Assistant-R	Naive Bayes	k-NN
INF1	86.0 ± 5.1	90.1 ± 3.5	88.8 ± 3.8	91.6 ± 3.1	89.0 ± 3.6
INF2	67.1 ± 6.3	55.4 ± 9.8	68.7 ± 7.8	32.1 ± 4.5	56.8 ± 6.3
TREE	75.8 ± 5.4	79.2 ± 5.7	78.8 ± 6.2	69.0 ± 5.9	68.2 ± 5.3
PAR2	93.6 ± 3.3	74.9 ± 7.9	95.7 ± 2.8	56.7 ± 5.7	79.4 ± 4.3
PAR3	84.1 ± 10.1	$65.6 \pm 11.$	95.7 ± 2.1	55.5 ± 5.2	60.4 ± 6.7
PAR4	69.4 ± 13.8	59.3 ± 6.3	94.8 ± 1.6	55.1 ± 3.4	61.9 ± 3.8

Table 4. Average information score of the learning systems on artificial data sets.

Domain	LFC	Assistant-I	Assistant-R	Naive Bayes	k-NN
INF1	1.24 ± 0.10	1.24 ± 0.06	1.22 ± 0.07	1.35 ± 0.06	0.94 ± 0.04
INF2	0.86 ± 0.14	0.50 ± 0.18	0.76 ± 0.13	0.07 ± 0.03	0.33 ± 0.06
TREE	0.47 ± 0.11	0.47 ± 0.08	0.47 ± 0.09	0.21 ± 0.04	0.17 ± 0.04
PAR2	0.84 ± 0.06	0.36 ± 0.12	0.83 ± 0.03	0.08 ± 0.04	0.28 ± 0.03
PAR3	0.67 ± 0.19	0.22 ± 0.17	0.80 ± 0.03	0.05 ± 0.03	0.10 ± 0.03
PAR4	0.38 ± 0.27	0.13 ± 0.09	0.79 ± 0.03	0.05 ± 0.02	0.10 ± 0.03

However, of these two, Assistant-R performs better, especially in the case of the hardest problem (PAR4). Note that LFC can solve PAR4 if the depth of the lookahead is increased, however, the time complexity of the lookahead increases exponentially with its depth. On the other hand, Assistant-R solves all parity problems equally quickly.

- The information score of the naive Bayesian classifier in the problems with strong conditional dependencies between attributes is poor which indicates that this classifier failed to find any regularity in these data sets.

5.2. Benchmark Artificial Data Sets

Besides the artificial data sets from the previous subsection, we used also the following benchmark artificial data sets used by other authors (note that results of other authors can not be directly compared to our results as experimental conditions (training/testing splits) were not the same):

BOOL: Boolean function defined on 6 attributes with 10% of class noise (optimal recognition rate is 90%). The target function is:

$$Y = (X_1 \oplus X_2) \vee (X_3 \wedge X_4) \vee (X_5 \wedge X_6)$$

This data set was used by Smyth et al. [31] and they report 67.2 ± 1.7% of the classification accuracy for naive Bayes, 82.5 ± 1.1% for backpropagation, and 85.9 ± 0.9% for their rule-based classifier.

LED: LED-digits problem with 10% of noise in attribute values. The optimal recognition rate is estimated to be 74%. Smyth et al. [31] report 68.1 ± 1.7% of the classification accuracy for naive Bayes, 64.6 ± 3.5 for backpropagation, and 72.7 ± 1.3 for their rule-based classifier. This data set can be obtained from Irvine database [21].

KRK1: The problem of legality of King-Rook-King chess endgame positions. The attributes describe the relevant relations between pieces, such as "same_rank" and "adjacent_file". Originally the data included five sets of 1000 examples (1000 for learning and 4000 for testing) and was used to test Inductive Logic Programming algorithms [7]. The reported classification accuracy is 99.7 ± 0.1%. We used only one set of 1000 examples (i.e., 700 instances for training).

KRK2: Same as KRK1 except that the only available attributes are the coordinates of pieces. The same data set was used by Mladenič [19]. The reported results are about 69% accuracy for her ATRIS system and 64% for Assistant.

The basic description of data sets is provided in Table 5 and results are given in Tables 6 and 7.

It is interesting that in the LED domain, the naive Bayesian classifier and the k-NN algorithm reach the estimated upper bound of the classification accuracy. This suggests that all attributes should be considered for optimal classification in this domain. In this problem the attributes are conditionally independent given the class, therefore the good performance of the naive Bayesian classifier is not surprising. However, in the

Table 5. Basic description of some benchmark artificial data sets.

Domain	#Class	#Atts.	#Val/att.	#Instances	Maj. class (%)	Entropy (bit)
BOOL	2	6	2.0	640	67	0.91
LED	10	7	2.0	1000	11	3.33
KRK1	2	18	2.0	1000	67	0.92
KRK2	2	6	8.0	1000	67	0.92

Table 6. Classification accuracy of the learning systems on artificial data sets.

Domain	LFC	Assistant-I	Assistant-R	Naive Bayes	k-NN
BOOL	89.8 ± 1.6	89.8 ± 1.6	89.8 ± 1.6	66.6 ± 2.5	89.8 ± 1.6
LED	70.8 ± 2.3	71.1 ± 2.4	71.7 ± 2.2	73.9 ± 2.1	73.9 ± 2.1
KRK1	98.7 ± 1.2	98.6 ± 1.2	98.6 ± 1.2	91.6 ± 1.4	92.2 ± 1.9
KRK2	86.0 ± 2.1	66.6 ± 3.1	70.1 ± 3.3	64.8 ± 2.1	70.7 ± 1.7

Table 7. Average information score of the learning systems on artificial data sets.

Domain	LFC	Assistant-I	Assistant-R	Naive Bayes	k-NN
BOOL	0.57 ± 0.03	0.51 ± 0.03	0.53 ± 0.03	0.07 ± 0.02	0.50 ± 0.03
LED	2.13 ± 0.07	2.11 ± 0.06	2.12 ± 0.07	2.33 ± 0.05	2.22 ± 0.05
KRK1	0.87 ± 0.04	0.84 ± 0.03	0.84 ± 0.03	0.60 ± 0.02	0.32 ± 0.01
KRK2	0.59 ± 0.05	0.12 ± 0.05	0.19 ± 0.04	-0.03 ± 0.02	0.12 ± 0.02

Table 8. Basic description of the medical data sets.

Domain	#Class	#Atts.	#Val/att.	#Instances	Maj. class (%)	Entropy (bit)
PRIM	22	17	2.2	339	25	3.89
BREA	2	10	2.7	288	80	0.73
LYMP	4	18	3.3	148	55	1.28
RHEU	6	32	9.1	355	66	1.73
HEPA	2	19	3.8	155	79	0.74
DIAB	2	8	8.8	768	65	0.93
HEART	2	13	5.0	270	56	0.99

other three domains the performance of the naive Bayesian classifier is poor, due to the strong conditional dependencies between attributes. The information score (see Table 7) shows that the naive Bayesian classifier provides (on the average) no information in the BOOL and KRK2 domains.

The performance of the different variants of Assistant is almost the same, except for the KRK2 domain, where the performance of Assistant-I is poor (note that the default accuracy in KRK2 is 67%). The performance of Assistant-R and the k-NN algorithm is significantly better (99.95% confidence level). However, the information score shows that both, Assistant-R and k-NN, are not very successful in this problem. As expected, without constructive induction it is not possible to reveal regularities in the chess positions described only with the coordinates of pieces. LFC is able to construct important attributes in this domain, which enables it to achieve significantly better results than the other algorithms.

5.3. *Medical Data Sets*

We compared the performance of the algorithms on several medical data sets:

- Data sets obtained from University Medical Center in Ljubljana, Slovenia: the problem of locating of primary tumour in patients with metastases (PRIM), the problem of predicting the recurrence of breast cancer five years after the removal of the tumour (BREA), the problem of determining the type of the cancer in lymphography (LYMP), and diagnosis in rheumatology (RHEU).
- HEPA: prognostics of survival for patients suffering from hepatitis. The data was provided by Gail Gong from Carnegie-Mellon University.
- Data sets obtained from the StatLog database [18]: diagnosis of diabetes (DIAB) and diagnosis of heart diseases (HEART). For the DIAB data set, Ragavan and Rendell [27] report 78.8% classification accuracy with their LFC algorithm. They also report poor performance of several other algorithms without constructive induction (up to 58%). However, our results (see below) and results of the StatLog project [18] show that the poor results of the other algorithms in this domain are not due to the lack of constructive induction. In our experiments, on DIAB dataset, all classifiers perform equally well, with the exception of the naive Bayesian classifier which is significantly better.

The basic characteristics of the above medical data sets are given in Table 8. The results of experiments on these data sets are provided in Tables 9 and 10.

In medical data sets, attributes are typically conditionally independent given the class. Therefore, it is not surprising that the naive Bayesian classifier shows

Table 9. Classification accuracy of the learning systems on medical data sets.

Domain	LFC	Assistant-I	Assistant-R	Naive Bayes	k-NN
PRIM	37.1 ± 4.9	40.8 ± 5.1	38.9 ± 4.7	48.6 ± 4.1	42.1 ± 5.0
BREA	76.1 ± 4.3	76.8 ± 4.6	78.5 ± 3.9	78.7 ± 4.5	79.5 ± 2.7
LYMP	82.4 ± 5.2	77.0 ± 5.5	77.0 ± 5.9	84.7 ± 4.2	82.6 ± 5.7
RHEU	60.6 ± 4.7	64.8 ± 4.0	63.8 ± 4.9	66.5 ± 4.0	66.0 ± 3.6
HEPA	79.0 ± 5.3	77.2 ± 5.3	82.3 ± 5.4	86.1 ± 3.9	82.6 ± 4.9
DIAB	69.2 ± 3.0	71.1 ± 2.8	71.5 ± 2.6	76.3 ± 2.4	73.9 ± 2.5
HEART	77.3 ± 5.2	75.4 ± 4.0	77.6 ± 4.5	84.5 ± 3.0	82.9 ± 3.7

Table 10. Average information score of the learning systems on medical data sets.

Domain	LFC	Assistant-I	Assistant-R	Naive Bayes	k-NN
PRIM	1.02 ± 0.14	1.19 ± 0.11	1.07 ± 0.11	1.60 ± 0.14	1.15 ± 0.11
BREA	0.01 ± 0.09	0.02 ± 0.08	0.05 ± 0.06	0.08 ± 0.07	0.02 ± 0.02
LYMP	0.79 ± 0.10	0.62 ± 0.09	0.61 ± 0.09	0.79 ± 0.09	0.53 ± 0.08
RHEU	0.41 ± 0.10	0.43 ± 0.08	0.41 ± 0.08	0.52 ± 0.07	0.43 ± 0.05
HEPA	0.19 ± 0.14	0.13 ± 0.09	0.22 ± 0.11	0.37 ± 0.11	0.21 ± 0.05
DIAB	0.26 ± 0.06	0.26 ± 0.04	0.27 ± 0.04	0.37 ± 0.04	0.24 ± 0.02
HEART	0.52 ± 0.10	0.45 ± 0.07	0.46 ± 0.07	0.64 ± 0.06	0.46 ± 0.04

clear advantage on these data sets [12]. It is interesting that the performance of the k-NN algorithm is good in these domains, although worse than the performance of the naive Bayesian classifier.

The information score (Table 10) for BREA data set indicates that no learning algorithm was able to solve this problem. This suggests that the attributes are not relevant.

Both versions of Assistant have similar performance, except in the HEPA domain where, Assistant-R has significantly better performance (99.95% confidence level). A detailed analysis showed that in this problem RELIEFF discovered a significant conditional interdependency between two attributes given the class. These two attributes score poorly when considered independently. That is why Assistant-I was not able to discover this regularity in data.

On the other hand, other (redundant) attributes are available that contain similar information as these two attributes together. This is the reason why the naive Bayesian classifier performs better. We tried to provide the naive Bayesian classifier with an additional attribute by joining the two conditionally dependent attributes. However, the performance remained the same.

LFC achieved significantly better results than the other two inductive algorithms in the LYMP domain, where constructive induction seems to be useful. However, LFC performed significantly worse in the RHEU domain while in the other domains the three inductive algorithms perform equally well.

5.4. *Non-Medical Real-World Data Sets*

We compared the performance of the algorithms also on the following non-medical real world data sets (SOYB, IRIS, and VOTE are obtained from the Irvine database [21], SAT is obtained from the StatLog database [18]):

SOYB: The famous soybean data set used by Michalski and Chilausky [17].

IRIS: The well known Fisher's problem of determining the type of iris flower.

MESH3, MESH15: The problem of determining the number of elements for each of the edges of an object in the finite element mesh design problem [6]. There are five objects for which experts have constructed appropriate meshes. In each of five experiments one object is used for testing and the other four for learning and the results are averaged. The results reported by Džeroski [7] for various ILP systems are 12% classification accuracy for FOIL, 22% for mFOIL and 29% for GOLEM and the result reported by Pompe et al. [24] is 28% for SFOIL. The description of the MESH problem is appropriate for ILP systems. For attribute learners only relations with arity 1 (i.e., attributes) can be used to describe the problem. Note that in this domain the training/testing splits are the same for all algorithms. The testing methodology is a special case of leave-one-out, therefore, the results in the tables for this problem have no standard deviations.

Quinlan [26] reports results of some ILP systems that achieved over 90% in that domain testing on positive and negative instances. However, those results are misleading. Each positive instance has ten negative instances on average. Therefore we have 11 copies of the same instance and any classification of this instance is correct at least for 9 out of 11 copies which gives 82% classification accuracy for a classifier that always classifies into wrong class.

MESH3 contains the three basic attributes from the original database and ignores the relational description of objects. Therefore, in the MESH3 domain attribute learners are given less information than ILP learners.

MESH15 contains, besides the 3 original attributes, 12 attributes derived from the relational background knowledge. In this problem, attribute learners have advantage as they are already provided with additional attributes. The provided description of objects for ILP learners is actually more informative. In principle, the same attributes and a number of additional attributes could be derived by (extremely cleaver) ILP learners from the relational description of the background knowledge. However, this is a fairly complex task. Therefore attribute learners with MESH15 data set have better chances than ILP learners to reveal a good hypothesis.

SAT: The database consists of multi-class spectral values of pixels in 3×3 neighborhoods in a satellite image, and the classification of the central pixel in each neighborhood. The results of the StatLog project [18] are 90.6% classification accuracy for the k-NN algorithm, 86.1% for backpropagation, 85.0% for C4.5, 84.8% for CN2 and 69.3% for the naive Bayesian classifier (using relative frequencies and not the m-estimate of probabilities).

VOTE: The voting records are from a session of the 1984 United States Congress. Smyth et al. [31] report 88.9% of classification accuracy for the naive Bayesian classifier, 93.0% for backpropagation and 94.9% for their rule-based classifier.

The basic characteristics of non-medical real world data sets are presented in Table 11. Tables 12 and 13

Table 11. Basic description of the non-medical real-world data sets.

Domain	#Class	#Atts.	#Val/att.	#Instances	Maj. class (%)	Entropy (bit)
SOYB	15	35	2.9	630	15	3.62
IRIS	3	4	6.0	150	33	1.59
MESH3	13	3	7.0	278	26	3.02
MESH15	13	15	7.1	278	26	3.02
SAT	6	36	10.0	6435	24	1.96
VOTE	2	16	2.0	435	61	0.96

Table 12. Classification accuracy of the learning systems on non-medical real-world data sets.

Domain	LFC	Assistant-I	Assistant-R	Naive Bayes	k-NN
SOYB	83.6 ± 4.8	91.9 ± 2.5	89.6 ± 2.7	89.4 ± 1.6	84.0 ± 1.9
IRIS	95.0 ± 3.8	95.7 ± 3.7	95.2 ± 2.6	96.6 ± 2.6	97.0 ± 2.1
MESH3	27.4%	32.7%	32.0%	33.5%	33.8%
MESH15	39.2%	41.0%	42.4%	34.5%	35.3%
SAT	81.9 ± 0.9	77.8 ± 0.7	81.5 ± 0.9	80.1 ± 0.6	90.5 ± 0.6
VOTE	93.2 ± 2.6	95.9 ± 1.5	95.5 ± 1.5	90.1 ± 1.8	92.5 ± 2.0

Table 13. Average information score of the learning systems on non-medical real-world data sets.

Domain	LFC	Assistant-I	Assistant-R	Naive Bayes	k-NN
SOYB	3.01 ± 0.13	3.10 ± 0.10	3.01 ± 0.10	3.27 ± 0.08	2.68 ± 0.08
IRIS	1.46 ± 0.08	1.41 ± 0.06	1.43 ± 0.06	1.49 ± 0.05	1.32 ± 0.04
MESH3	0.51 bit	0.57 bit	0.56 bit	0.65 bit	0.39 bit
MESH15	0.67 bit	0.74 bit	0.70 bit	0.56 bit	0.54 bit
SAT	1.92 ± 0.02	1.68 ± 0.02	1.84 ± 0.02	1.93 ± 0.08	1.86 ± 0.01
VOTE	0.81 ± 0.05	0.84 ± 0.03	0.83 ± 0.03	0.75 ± 0.04	0.67 ± 0.03

give the results. On SOYB and IRIS data sets, all classifiers perform equally well. The results of the naive Bayesian classifier indicate that the attributes are conditionally relatively independent in these data sets, which is in agreement with previously published results.

On the SAT data set, k-NN significantly outperforms other algorithms which is in agreement with the results of the StatLog project [18]. However, the naive Bayesian classifier with the m-estimate of probabilities reaches the classification accuracy of inductive learning algorithms. The results of the naive Bayesian classifier used in the StatLog project are much worse. Cestnik [2] has shown that the m-estimate significantly increases the performance of the naive Bayesian classifier which is also confirmed with our experiments.

Both versions of Assistant perform the same on all data sets except on the SAT data set where Assistant-R and LFC achieve significantly better result (99.95% confidence level). This result confirms that RELIEFF estimates the quality of attributes better than the information gain.

On the VOTE data set the naive Bayesian classifier is the worst, while both versions of Assistant are comparable to the rule based classifier by Smyth et al. [31].

The most interesting results appear in the MESH domains. Although attribute learners in MESH3 have less information than ILP systems, they all outperform the results by ILP systems as reported by Džeroski [7] and Pompe et al. [24]. With 12 additional attributes in MESH15, the results of inductive learners are significantly improved. All inductive learning systems significantly outperform the naive Bayesian classifier and the k-NN algorithm.

A detailed analysis showed that this excellent result by both versions of Assistant is due to the use of the naive Bayesian formula to calculate the class probability distribution in "null" leaves (see Section 3). Namely, for this problem it often happens that testing instances fall into a "null" leaf because there are no training instances that have the same values of significant attributes as the testing instances. The naive Bayesian classifier efficiently solves this problem.

LFC generates no "null" leaves as all constructed attributes are strictly binary with values *true* and *false*. Therefore, the classification of objects with a different value of the original attribute than all training instances always proceeds to the branch labeled *false*. The effect of this strategy is that, for a given testing instance, the corresponding leaf contains training instances with same or similar values for most of the attributes that appear on the path from the root to the leaf. This strategy also works well in MESH problems.

6. Discussion

Note that the null leaves of both versions of Assistant had no influence on the performance on artificial data sets as there is no missing values in the data. Also, in MESH15 problem the performance of LFC is good although it does not generate null leaves. Therefore, the use of null leaves is not the crucial difference between Assistant and LFC.

Equation (3) shows an interesting relation between the RELIEF's estimates and impurity function. RELIEF can efficiently estimate continuous and discrete attributes. The implicit normalization in Eq. (3) enables RELIEF to appropriately deal with multivalued attributes. However, if Assistant-I would use Eq. (3) instead of the information gain, it would still be myopic. For example, in PAR2-4 problems, Eq. (3) would estimate all attributes as equally non-important.

Therefore, the reason of the success of Assistant-R is in the "nearest instances" heuristic which influences the estimation of probabilities. This heuristic enables RELIEF to detect strong conditional dependencies between the attributes which would be overlooked if the estimates of probabilities would be done on randomly selected instances instead of the nearest instances.

RELIEFF is an efficient heuristic estimator of attribute quality that is able to deal with data sets with conditionally dependent and independent attributes. The extensions in RELIEFF enable it to deal with noisy, incomplete, and multi-class data sets. With increasing the number (k) of nearest hits/misses the correlation of RELIEFF's estimates with other impurity functions also increases unless k is greater than the number of instances in the same peak of the instance space. The study reported in [14] showed that RELIEFF has an acceptable bias with respect to other measures when estimating attributes with different number of values.

The myopia of current inductive learning systems can be partially overcome by replacing the existing heuristic functions with RELIEFF. Assistant-R, a variant of top down induction of decision trees algorithms that uses RELIEFF for estimating the quality of the attributes, significantly outperforms other classifiers in domains with strong conditional dependencies between attributes. The myopia of other inductive learners may cause them to overlook significant relations. While this can be easily demonstrated with artificial data sets, it was also shown in two real world problems: HEPA and SAT. In these data sets RELIEFF detected significant conditional interdependencies between attributes, that resulted in a significantly better result by Assistant-R than the result by Assistant-I.

One feature of RELIEF not addressed in this paper is that if the same attribute is replicated in a data set, all replications will get the same estimate. With the increasing number of replications the quality of estimates will descrease as the replicated attribute affects the distances between instances.

For constructive induction LFC uses a limited lookahead to detect significant conditional dependencies between the attributes. LFC shows similar advantage over other algorithms as Assistant-R does. In one artificial problem (KRK2) and one real world problem (LYMP) LFC performs significantly better due to constructive induction. However, in some cases the constructive induction may spoil the results as is the case with RHEU data set. LFC performs well in most of the problems, which suggests that the limited lookahead is a good search strategy in most real-world problems. The lookahead, however, should have a reasonable limit as the time complexity exponentialy increases with the lookahead depth.

Although RELIEFF may overcome the myopia, it is useless in Assistant-R when the change of representation is required. In such cases the constructive induction should be applied. For example, in the KRK2 problem, Assistant-R achieves good result which can not be further improved without constructive induction. A good idea for constructive induction may be to use RELIEFF instead of or in the combination with the lookahead.

The naive Bayesian classifier has obvious advantage in domains with conditionally relatively independent attributes, such as medical diagnostic problems. In such domains, the naive Bayesian classifier is able to reliably estimate the conditional probabilities and is also able to use all attributes, i.e., all available information. It would be interesting to appropriately combine the power of RELIEFF and the naive Bayesian classifier.

Current ILP systems [20] are not able to use the attributes appropriately. This was demonstrated in the MESH3 domain where all attribute learners outperformed existing ILP systems. To enable ILP systems to deal with the attribute-value representation, a combination with the (semi) naive Bayesian classifier could be useful. On the other hand, current ILP systems use greedy search techniques and the heuristics that guide the search are myopic. Pompe and Kononenko [23] implemented an adapted version of RELIEFF in the FOIL like ILP system called ILP-R and preleminary experiments show similar advantages of this system over other ILP systems as Assistant-R has over Assistant-I.

7. Conclusion

RELIEFF is an efficient heuristic estimator of attribute quality that is able to deal with data sets with conditionally dependent and independent attributes, with noisy, incomplete, and multi-class data sets. The myopia of current inductive learning systems can be partially overcome by replacing the existing heuristic functions with RELIEFF. The acceptable increase in computational complexity may in certain domains payoff with eventual discovery of strong conditional dependencies between attributes, which cannot be detected using the myopic impurity measure to guide the greedy search.

The experimental results indicate that in the majority of real world problems the myopia has no or only marginal effect. One may wonder whether myopia is really worth much attention at all. However, when faced with a new data set it is unreasonable to try only myopic algorithm unless it is know in advance that in the data set there are no strong conditional dependencies between attributes. Any serious application of machine learning on new data should try to discover as much regularities in the data as possible. Therefore, non-myopic approaches, such as one described in this paper, should be used as indispensable tools for analysing the data.

Acknowledgments

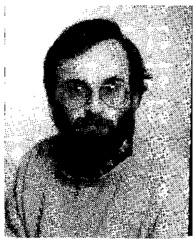
The use of m-estimate in Eq. (1) was proposed by Bojan Cestnik. We thank Matjaž Zwitter for the PRIM and BREA data sets, Milan Soklič for LYMP, Gail Gong for HEPA, Padhraic Smyth for BOOL and LED, Sašo Džeroski for KRK1 and MESH, Bob Henery for the DIAB, HEART, and SAT data sets from the StatLog database at Strathclyde University, and Patrick Murphy and David Aha for the data sets from the Irvine database. We are grateful to our colleagues Sašo Džeroski, Matevž Kovačič, Matjaž Kukar, Uroš Pompe, and Tanja Urbančič and to anonymous reviewers for their comments on earlier drafts that significantly improved the paper. This work was supported by the Slovenian Ministry of Science and Technology.

References

1. L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone, *Classification and Regression Trees*, Wadsworth International Group, 1984.
2. B. Cestnik, "Estimating probabilities: A crucial task in machine learning," *Proc. European Conference on Artificial Intelligence*, Stockholm, Aug. 1990, pp.147–149.
3. B. Cestnik and I. Bratko, "On estimating probabilities in tree pruning," *Proc. European Working Session on Learning*, edited by Y. Kodratoff, Springer-Verlag: Porto, March 1991, pp. 138–150.
4. B. Cestnik, I. Kononenko, and I. Bratko, "ASSISTANT 86: A knowledge elicitation tool for sophisticated users," in *Progress in Machine Learning*, edited by I. Bratko and N. Lavrač, Sigma Press: Wilmslow, England, 1987.
5. W. Chase and F. Brown, *General Statistics*, John Wiley & Sons, 1986.
6. B. Dolšak and S. Muggleton, "The application of inductive logic programming to finite element mesh design," in *Inductive Logic Programming*, edited by S. Muggleton, Academic Press, 1992.
7. S. Džeroski, "Handling noise in inductive logic programming," M.Sc. Thesis, University of Ljubljana, Faculty of Electrical Engineering & Computer Science, Ljubljana, Slovenia, 1991.
8. S.J. Hong, "Use of contextual information for feature ranking and discretization," Technical Report, IBM RC19664, 7/94, 1994 (to appear in IEEE Trans. on Knowledge and Data Engineering).
9. E. Hunt, J. Martin, and P. Stone, *Experiments in Induction*, Academic Press: New York, 1966.
10. K. Kira and L. Rendell, "A practical approach to feature selection," *Proc. Intern. Conf. on Machine Learning*, edited by D. Sleeman and P. Edwards, Morgan Kaufmann: Aberdeen, July 1992, pp. 249–256.
11. K. Kira and L. Rendell, "The feature selection problem: Traditional methods and new algorithm," *Proc. AAAI'92*, San Jose, CA, July 1992.
12. I. Kononenko, "Inductive and Bayesian learning in medical diagnosis," *Applied Artificial Intelligence*, vol. 7, pp. 317–337, 1993.
13. I. Kononenko, "Estimating attributes: Analysis and extensions of RELIEF," *Proc. European Conf. on Machine Learning*, edited by L. De Raedt and F. Bergadano, Springer-Verlag: Catania, April 1994, pp. 171–182.
14. I. Kononenko, "On biases when estimating multivalued attributes," *Proc. IJCAI-95*, edited by C. Mellish, Morgan Kaufmann: Montreal, Aug. 1995, pp. 1034–1040.
15. I. Kononenko and I. Bratko, "Information based evaluation criterion for classifier's performance," *Machine Learning*, vol. 6, pp. 67–80, 1991.
16. R.L. Mantaras, "ID3 Revisited: A distance based criterion for attribute selection," *Proc. Int. Symp. Methodologies for Intelligent Systems*, Charlotte, North Carolina, U.S.A., Oct. 1989.
17. R.S. Michalski and R.L. Chilausky, "Learning by being told and learning from examples: An experimental comparison of the two methods of knowledge acquisition in the context of developing an expert system for soybean disease diagnosis," *International Journal of Policy Analysis and Information Systems*, vol. 4, pp. 125–161, 1980.
18. D. Michie, D.J. Spiegelhalter, and C.C. Taylor (eds.), *Machine Learning, Neural and Statistical Classification*, Ellis Horwood Limited, 1994.
19. D. Mladenič, "Combinatorial optimization in inductive concept learning," *Proc. 10th Intern. Conf. on Machine Learning*, Morgan Kaufmann: Amherst, June 1993, pp. 205–211.
20. S. Muggleton (ed.), *Inductive Logic Programming*, Academic Press, 1992.
21. P.M. Murphy and D.W. Aha, *UCI Repository of Machine Learning Databases* [Machine-readable data repository], Irvine, CA, University of California, Department of Information and Computer Science, 1991.
22. T. Niblett and I. Bratko, "Learning decision rules in noisy domains," *Proc. Expert Systems 86*, Brighton, UK, Dec. 1986.
23. U. Pompe and I. Kononenko, "Linear space induction in first order logic with RELIEFF," in *Mathematical and Statistical Methods in Artificial Intelligence*, edited by G. Della Riccia, R. Kruse, and R. Viertl, CISM Lecture Notes, Springer-Verlag, 1995.
24. U. Pompe, M. Kovačič, and I. Kononenko, "SFOIL: Stochastic approach to inductive logic programming," *Proc. Slovenian Conf. on Electrical Engineering and Computer Science*, Portorož, Slovenia, Sept. 1993, pp. 189–192.
25. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.
26. R. Quinlan, "The minimum description length principle and categorical theories," *Proc. 11th Int. Conf. on Machine Learning*

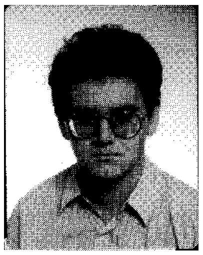
edited by W. Cohen and H. Hirsh, Morgan Kaufmann: Ruthers University, New Brunswick, July 1994, pp. 233–241.

27. H. Ragavan and L. Rendell, “Lookahead feature construction for learning hard concepts,” *Proc. 10th Intern. Conf. on Machine Learning*, Morgan Kaufmann: Amherst, June 1993, pp. 252–259.
28. H. Ragavan, L. Rendell, M. Shaw, and A. Tessmer, “Learning complex real-world concepts through feature construction,” Technical Report UIUC-BI-AI-93-03, The Beckman Institute, University of Illinois, 1993.
29. M. Robnik, “Constructive induction with decision trees,” B.Sc. Thesis (in Slovene), University of Ljubljana, Faculty of Electrical Engineering & Computer Science, Ljubljana, Slovenia, 1993.
30. P. Smyth and R.M. Goodman, “Rule induction using information theory,” in *Knowledge Discovery in Databases*, edited by G. Piatetsky-Shapiro and W. Frawley, MIT Press, 1990.
31. P. Smyth, R.M. Goodman, and C. Higgins, “A hybrid rule-based Bayesian classifier,” *Proc. European Conf. on Artificial Intelligence*, Stockholm, Aug. 1990, pp. 610–615.



Igor Kononenko received his B.Sc. in 1982, M.Sc. in 1985, and Ph.D. in 1990 in computer science from University of Ljubljana, Slovenia. He is an assistant professor at the Faculty of Computer and Information Science in Ljubljana and the member of Artificial Intelligence Laboratory at the same Faculty. His research interests include artificial intelligence, machine learning and neural networks. He is the (co)author of about 60 publications in these fields. He is the member of the program committee of the European Conference on Machine Learning and the member of the editorial board of Applied Intelligence Journal.

Edvard Šimec received his B.Sc. in 1994 from University of Ljubljana, Slovenia. He is currently working as a software engineer at ASTER, the Silicon Graphics distribution company for Slovenia, in Ljubljana.



Marko Robnik-Šikonja received his B.Sc. in computer science in 1993 from University of Ljubljana, Slovenia. He is currently working as a researcher in the Artificial Intelligence Laboratory at the Faculty of Computer and Information Science in Ljubljana. He is working on his M.Sc. thesis in the fields of constructive induction and regression in machine learning.