Machine Learning, 53, 23–69, 2003

Theoretical and Empirical Analysis of ReliefF and RReliefF

MARKO ROBNIK-ŠIKONJA Marko.Robnik@fri.uni-lj.si
IGOR KONONENKO Igor.Kononenko@fri.uni-lj.si
University of Ljubljana, Faculty of Computer and Information Science, Tržaška 25, 1001 Ljubljana, Slovenia

Editor: Raul Valdes-Perez

Abstract. Relief algorithms are general and successful attribute estimators. They are able to detect conditional dependencies between attributes and provide a unified view on the attribute estimation in regression and classification. In addition, their quality estimates have a natural interpretation. While they have commonly been viewed as feature subset selection methods that are applied in prepossessing step before a model is learned, they have actually been used successfully in a variety of settings, e.g., to select splits or to guide constructive induction in the building phase of decision or regression tree learning, as the attribute weighting method and also in the inductive logic programming.

A broad spectrum of successful uses calls for especially careful investigation of various features Relief algorithms have. In this paper we theoretically and empirically investigate and discuss how and why they work, their theoretical and practical properties, their parameters, what kind of dependencies they detect, how do they scale up to large number of examples and features, how to sample data for them, how robust are they regarding the noise, how irrelevant and redundant attributes influence their output and how different metrics influences them.

Keywords: attribute evaluation, feature selection, Relief algorithm, classification, regression

1. Introduction

A problem of estimation of the quality of attributes (features) is an important issue in the machine learning. There are several important tasks in the process of machine learning e.g., feature subset selection, constructive induction, decision and regression tree building, which contain the attribute estimation procedure as their (crucial) ingredient.

In many learning problems there are hundreds or thousands of potential features describing each input object. Majority of learning methods do not behave well in this circumstances because, from a statistical point of view, examples with many irrelevant, but noisy, features provide very little information. A feature subset selection is a task of choosing a small subset of features that ideally is necessary and sufficient to describe the target concept. To make a decision which features to retain and which to discard we need a reliable and practically efficient method of estimating their relevance to the target concept.

In the constructive induction we face a similar problem. In order to enhance the power of the representation language and construct a new knowledge we introduce new features. Typically many candidate features are generated and again we need to decide which features to retain and which to discard. To estimate the relevance of the features to the target concept is certainly one of the major components of such a decision procedure.

Decision and regression trees are popular description languages for representing knowledge in the machine learning. While constructing a tree the learning algorithm at each interior node selects the splitting rule (feature) which divides the problem space into two separate subspaces. To select an appropriate splitting rule the learning algorithm has to evaluate several possibilities and decide which would partition the given (sub)problem most appropriately. The estimation of the quality of the splitting rules seems to be of the principal importance.

The problem of feature (attribute) estimation has received much attention in the literature. There are several measures for estimating attributes' quality. If the target concept is a discrete variable (the classification problem) these are e.g., information gain (Hunt, Martin, & Stone, 1966), Gini index (Breiman et al., 1984), distance measure (Mantaras, 1989), j-measure (Smyth & Goodman, 1990), Relief (Kira & Rendell, 1992b), ReliefF (Kononenko, 1994), MDL (Kononenko, 1995), and also χ^2 and G statistics are used. If the target concept is presented as a real valued function (numeric class and the regression problem) then the estimation heuristics are e.g., the mean squared and the mean absolute error (Breiman et al., 1984), and RReliefF (Robnik Šikonja & Kononenko, 1997).

The majority of the heuristic measures for estimating the quality of the attributes assume the conditional (upon the target variable) independence of the attributes and are therefore less appropriate in problems which possibly involve much feature interaction. Relief algorithms (Relief, ReliefF and RReliefF) do not make this assumption. They are efficient, aware of the contextual information, and can correctly estimate the quality of attributes in problems with strong dependencies between attributes.

While Relief algorithms have commonly been viewed as feature subset selection methods that are applied in a prepossessing step before the model is learned (Kira & Rendell, 1992b) and are one of the most successful preprocessing algorithms to date (Dietterich, 1997), they are actually general feature estimators and have been used successfully in a variety of settings: to select splits in the building phase of decision tree learning (Kononenko, Šimec, & Robnik-Šikonja, 1997), to select splits and guide the constructive induction in learning of the regression trees (Robnik Šikonja & Kononenko, 1997), as attribute weighting method (Wettschereck, Aha, & Mohri, 1997) and also in inductive logic programming (Pompe & Kononenko, 1995).

The broad spectrum of successful uses calls for especially careful investigation of various features Relief algorithms have: how and why they work, what kind of dependencies they detect, how do they scale up to large number of examples and features, how to sample data for them, how robust are they regarding the noise, how irrelevant and duplicate attributes influence their output and what effect different metrics have.

In this work we address these questions as well as some other more theoretical issues regarding the attribute estimation with Relief algorithms. In Section 2 we present the Relief algorithms and discuss some theoretical issues. We conduct some experiments to illustrate these issues. We then turn (Section 3) to the practical issues on the use of ReliefF and try to answer the above questions (Section 4). Section 5 discusses applicability of Relief algorithms for various tasks. In Section 6 we conclude with open problems on both empirical and theoretical fronts.

We assume that examples $I_1, I_2, \ldots, I_n$ in the instance space are described by a vector of attributes A_i, $i = 1, \ldots, a$, where a is the number of explanatory attributes, and are labelled with the target value τ_j. The examples are therefore points in the a dimensional space. If the target value is categorical we call the modelling task classification and if it is numerical we call the modelling task regression.

2. Relief family of algorithms

In this section we describe the Relief algorithms and discuss their similarities and differences. First we present the original Relief algorithm (Kira & Rendell, 1992b) which was limited to classification problems with two classes. We give account on how and why it works. We discuss its extension ReliefF (Kononenko, 1994) which can deal with multiclass problems. The improved algorithm is more robust and also able to deal with incomplete and noisy data. Then we show how ReliefF was adapted for continuous class (regression) problems and describe the resulting RReliefF algorithm (Robnik Šikonja & Kononenko, 1997). After the presentation of the algorithms we tackle some theoretical issues about what Relief output actually is.

2.1. Relief—basic ideas

A key idea of the original Relief algorithm (Kira & Rendell, 1992b), given in figure 1, is to estimate the quality of attributes according to how well their values distinguish between instances that are near to each other. For that purpose, given a randomly selected instance R_i (line 3), Relief searches for its two nearest neighbors: one from the same class, called *nearest hit* H, and the other from the different class, called *nearest miss* M (line 4). It updates the quality estimation $W[A]$ for all attributes A depending on their values for R_i, M, and H (lines 5 and 6). If instances R_i and H have different values of the attribute A then the attribute A separates two instances with the same class which is not desirable so we decrease the quality estimation $W[A]$. On the other hand if instances R_i and M have different values of the attribute A then the attribute A separates two instances with different class values which is desirable so we increase the quality estimation $W[A]$. The whole process is repeated for m times, where m is a user-defined parameter.

Algorithm Relief
Input: for each training instance a vector of attribute values and the class value
Output: the vector W of estimations of the qualities of attributes

1. set all weights $W[A] := 0.0$;
2. **for** i := 1 **to** m **do begin**
3. randomly select an instance R_i;
4. find nearest hit H and nearest miss M;
5. **for** $A := 1$ **to** a **do**
6. $W[A] := W[A] - \text{diff}(A, R_i, H)/m + \text{diff}(A, R_i, M)/m$;
7. **end**;

Figure 1. Pseudo code of the basic Relief algorithm.

Function diff(A, I_1, I_2) calculates the difference between the values of the attribute A for two instances I_1 and I_2. For nominal attributes it was originally defined as:

$$\text{diff}(A, I_1, I_2) = \begin{cases} 0; & value(A, I_1) = value(A, I_2) \\ 1; & otherwise \end{cases} \tag{1}$$

and for numerical attributes as:

$$\text{diff}(A, I_1, I_2) = \frac{|value(A, I_1) - value(A, I_2)|}{max(A) - min(A)} \tag{2}$$

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 distances over all attributes (Manhattan distance).

The original Relief can deal with nominal and numerical attributes. However, it cannot deal with incomplete data and is limited to two-class problems. Its extension, which solves these and other problems, is called ReliefF.

2.2. *ReliefF—extension*

The ReliefF (Relief-F) algorithm (Kononenko, 1994) (see figure 2) is not limited to two class problems, is more robust and can deal with incomplete and noisy data. Similarly to Relief, ReliefF randomly selects an instance R_i (line 3), but then searches for k of its nearest neighbors from the same class, called nearest hits H_j (line 4), and also k nearest neighbors from each of the different classes, called nearest misses $M_j(C)$ (lines 5 and 6). It updates the quality estimation $W[A]$ for all attributes A depending on their values for R_i, hits H_j

Algorithm ReliefF
Input: for each training instance a vector of attribute values and the class value
Output: the vector W of estimations of the qualities of attributes

1. set all weights $W[A] := 0.0$;
2. **for** i := 1 **to** m **do begin**
3. randomly select an instance R_i;
4. find k nearest hits H_j;
5. **for** each class $C \neq class(R_i)$ **do**
6. from class C find k nearest misses $M_j(C)$;
7. **for** $A := 1$ **to** a **do**
8. $W[A] := W[A] - \sum_{j=1}^{k} \text{diff}(A, R_i, H_j)/(m \cdot k) +$
9. $\sum_{C \neq class(R_i)} [\frac{P(C)}{1-P(class(R_i))} \sum_{j=1}^{k} \text{diff}(A, R_i, M_j(C))]/(m \cdot k)$;
10. **end**;

Figure 2. Pseudo code of ReliefF algorithm.

and misses $M_j(C)$ (lines 7, 8 and 9). The update formula is similar to that of Relief (lines 5 and 6 on figure 1), except that we average the contribution of all the hits and all the misses. The contribution for each class of the misses is weighted with the prior probability of that class $P(C)$ (estimated from the training set). Since we want the contributions of hits and misses in each step to be in [0, 1] and also symmetric (we explain reasons for that below) we have to ensure that misses' probability weights sum to 1. As the class of hits is missing in the sum we have to divide each probability weight with factor $1 - P(class(R_i))$ (which represents the sum of probabilities for the misses' classes). The process is repeated for m times.

Selection of k hits and misses is the basic difference to Relief and ensures greater robustness of the algorithm concerning noise. User-defined parameter k controls the locality of the estimates. For most purposes it can be safely set to 10 (see Kononenko, 1994 and discussion below).

To deal with incomplete data we change the diff function. Missing values of attributes are treated probabilistically. We calculate the probability that two given instances have different values for given attribute conditioned over class value:

– if one instance (e.g., I_1) has unknown value:

$$\text{diff}(A, I_1, I_2) = 1 - P(value(A, I_2) \mid class(I_1)) \tag{3}$$

– if both instances have unknown value:

$$\text{diff}(A, I_1, I_2) = 1 - \sum_{V}^{\#values(A)} (P(V \mid class(I_1)) \times P(V \mid class(I_2))) \tag{4}$$

Conditional probabilities are approximated with relative frequencies from the training set.

2.3. RReliefF—in regression

We finish the description of the algorithmic family with RReliefF (Regressional ReliefF) (Robnik Šikonja & Kononenko, 1997). First we theoretically explain what Relief algorithm actually computes.

Relief's estimate $W[A]$ of the quality of attribute A is an approximation of the following difference of probabilities (Kononenko, 1994):

$$\begin{aligned} W[A] = {} & P(\text{diff. value of } A \mid \text{nearest inst. from diff. class}) \\ & - P(\text{diff. value of } A \mid \text{nearest inst. from same class}) \end{aligned} \tag{5}$$

The positive updates of the weights (line 6 in figure 1 and line 9 in figure 2) are actually forming the estimate of probability that the attribute discriminates between the instances with different class values, while the negative updates (line 6 in figure 1 and line 8 in figure 2) are forming the probability that the attribute separates the instances with the same class value.

In regression problems the predicted value $\tau(\cdot)$ is continuous, therefore (nearest) hits and misses cannot be used. To solve this difficulty, instead of requiring the exact knowledge of whether two instances belong to the same class or not, a kind of probability that the predicted values of two instances are different is introduced. This probability can be modelled with the relative distance between the predicted (class) values of two instances.

Still, to estimate $W[A]$ in (5), information about the sign of each contributed term is missing (where do hits end and misses start). In the following derivation Eq. (5) is reformulated, so that it can be directly evaluated using the probability that predicted values of two instances are different. If we rewrite

$$P_{\text{diff}A} = P(\text{different value of } A \mid \text{nearest instances}) \tag{6}$$

$$P_{\text{diff}C} = P(\text{different prediction} \mid \text{nearest instances}) \tag{7}$$

and

$$P_{\text{diff}C|\text{diff}A} = P(\text{diff. prediction} \mid \text{diff. value of } A \text{ and nearest instances}) \tag{8}$$

we obtain from (5) using Bayes' rule:

$$W[A] = \frac{P_{\text{diff}C|\text{diff}A} P_{\text{diff}A}}{P_{\text{diff}C}} - \frac{(1 - P_{\text{diff}C|\text{diff}A}) P_{\text{diff}A}}{1 - P_{\text{diff}C}} \tag{9}$$

Therefore, we can estimate $W[A]$ by approximating terms defined by Eqs. (6)–(8). This can be done by the algorithm on figure 3.

Algorithm RReliefF
Input: for each training instance a vector of attribute values **x** and predicted value $\tau(\mathbf{x})$
Output: vector W of estimations of the qualities of attributes

```
1.  set all N_dC, N_dA[A], N_dC&dA[A], W[A] to 0;
2.  for i := 1 to m do begin
3.      randomly select instance R_i;
4.      select k instances I_j nearest to R_i;
5.      for j := 1 to k do begin
6.          N_dC := N_dC + diff(τ(·), R_i, I_j) · d(i, j);
7.          for A := 1 to a do begin
8.              N_dA[A] := N_dA[A] + diff(A, R_i, I_j) · d(i, j);
9.              N_dC&dA[A] := N_dC&dA[A] + diff(τ(·), R_i, I_j)·
10.                                       diff(A, R_i, I_j) · d(i, j);
11.         end;
12.     end;
13. end;
14. for A := 1 to a do
15.     W[A] := N_dC&dA[A]/N_dC - (N_dA[A] − N_dC&dA[A])/(m − N_dC);
```

Figure 3. Pseudo code of RReliefF algorithm.

Similarly to ReliefF we select random instance R_i (line 3) and its k nearest instances I_j (line 4). The weights for different prediction value $\tau(\cdot)$ (line 6), different attribute (line 8), and different prediction & different attribute (lines 9 and 10) are collected in N_{dC}, $N_{dA}[A]$, and $N_{dC\&dA}[A]$, respectively. The final estimation of each attribute $W[A]$ (Eq. (9)) is computed in lines 14 and 15.

The term $d(i, j)$ in figure 3 (lines 6, 8 and 10) takes into account the distance between the two instances R_i and I_j. Rationale is that closer instances should have greater influence, so we exponentially decrease the influence of the instance I_j with the distance from the given instance R_i:

$$d(i, j) = \frac{d_1(i, j)}{\sum_{l=1}^{k} d_1(i, l)} \tag{10}$$

and

$$d_1(i, j) = e^{-\left(\frac{\text{rank}(R_i, I_j)}{\sigma}\right)^2} \tag{11}$$

where $\text{rank}(R_i, I_j)$ is the rank of the instance I_j in a sequence of instances ordered by the distance from R_i and σ is a user defined parameter controlling the influence of the distance. Since we want to stick to the probabilistic interpretation of the results we normalize the contribution of each of k nearest instances by dividing it with the sum of all k contributions. The reason for using ranks instead of actual distances is that actual distances are problem dependent while by using ranks we assure that the nearest (and subsequent as well) instance always has the same impact on the weights.

ReliefF was using a constant influence of all k nearest instances I_j from the instance R_i. For this we should define $d_1(i, j) = 1/k$.

Discussion about different distance functions can be found in following sections.

2.4. Computational complexity

For n training instances and a attributes Relief (figure 1) makes $O(m \cdot n \cdot a)$ operations. The most complex operation is selection of the nearest hit and miss as we have to compute the distances between R and all the other instances which takes $O(n \cdot a)$ comparisons.

Although ReliefF (figure 2) and RReliefF (figure 3) look more complicated their asymptotical complexity is the same as that of original Relief, i.e., $O(m \cdot n \cdot a)$. The most complex operation within the main **for** loop is selection of **k** nearest instances. For it we have to compute distances from all the instances to R, which can be done in $O(n \cdot a)$ steps for n instances. This is the most complex operation, since $O(n)$ is needed to build a heap, from which k nearest instances are extracted in $O(k \log n)$ steps, but this is less than $O(n \cdot a)$.

Data structure k-d (k-dimensional) tree (Bentley, 1975; Sefgewick, 1990) is a generalization of the binary search tree, which instead of one key uses k keys (dimensions). The root of the tree contains all the instances. Each interior node has two successors and splits instances recursively into two groups according to one of k dimensions. The recursive splitting stops when there are less than a predefined number of instances in a node. For n instances we can build the tree where split on each dimension maximizes the variance in that dimension

and instances are divided into groups of approximately the same size in time proportional to $O(k \cdot n \cdot \log n)$. With such tree called optimized k-d tree we can find t nearest instances to the given instance in $O(\log n)$ steps (Friedman, Bentley, & Finkel, 1975).

If we use k-d tree to implement the search for nearest instances we can reduce the complexity of all three algorithms to $O(a \cdot n \cdot \log n)$ (Robnik Šikonja, 1998). For Relief we first build the optimized k-d tree (outside the main loop) in $O(a \cdot n \cdot \log n)$ steps so we need only $O(m \cdot a)$ steps in the loop and the total complexity of the algorithm is now the complexity of the preprocessing which is $O(a \cdot n \cdot \log n)$. The required sample size m is related to the problem complexity (and not to the number of instances) and is typically much more than $\log n$ so asymptotically we have reduced the complexity of the algorithm. Also it does not make sense to use sample size m larger than the number of instances n.

The computational complexity of ReliefF and RReliefF using k-d trees is the same as that of Relief. They need $O(a \cdot n \cdot \log n)$ steps to build k-d tree, and in the main loop they select t nearest neighbors in $\log n$ steps, update weights in $O(t \cdot a)$ but $O(m(t \cdot a + \log n))$ is asymptotically less than the preprocessing which means that the complexity has reduced to $O(a \cdot n \cdot \log n)$. This analysis shows that ReliefF family of algorithms is actually in the same order of complexity as multikey sort algorithms.

Several authors have observed that the use of k-d trees becomes inefficient with increasing number of attributes (Friedman, Bentley, & Finkel, 1975; Deng & Moore, 1995; Moore, Schneider, & Deng, 1997) and this was confirmed for Relief family of algorithms as well (Robnik Šikonja, 1998).

Kira and Rendell (1992b) consider m an arbitrary chosen constant and claim that the complexity of Relief is $O(a \cdot n)$. If we accept their argument than the complexity of ReliefF and RReliefF is also $O(a \cdot n)$, and the above analysis using k-d trees is useless. However, if we want to obtain sensible and reliable results with Relief algorithms then the required sample size m is related to the problem complexity and is not constant as we will show below.

2.5. *General framework of Relief algorithms*

By rewriting Eq. (5) into a form suitable also for regression

$$\begin{aligned} W[A] = {} & P(\text{diff. value of } A \mid \text{near inst. with diff. prediction}) \\ & - P(\text{diff. value of } A \mid \text{near inst. with same prediction}) \end{aligned} \tag{12}$$

we see that we are actually dealing with (dis)similarities of attributes and prediction values (of near instances). A generalization of Relief algorithms would take into account the similarity of the predictions τ and of the attributes A and combine them into a generalized weight:

$$W_G[A] = \sum_{I_1, I_2 \in \mathcal{I}} \text{similarity}(\tau, I1, I2) \cdot \text{similarity}(A, I_1, I_2) \tag{13}$$

where I_1 and I_2 were appropriate samples drawn from the instance population $\mathcal{I}$. If we use [0, 1] normalized similarity function (like e.g., diff) than with these weights we can model the following probabilities:

- $P(\text{similar } A \mid \text{similar } \tau)$, $P(\text{dissimilar} A \mid \text{similar } \tau)$, and
- $P(\text{similar } A \mid \text{dissimilar } \tau)$, $P(\text{dissimilar } A \mid \text{dissimilar } \tau)$.

In the probabilistic framework we can write:

$$P(\text{similar } A \mid \text{similar } \tau) + P(\text{dissimilar } A \mid \text{similar } \tau) = 1 \tag{14}$$

$$P(\text{similar } A \mid \text{dissimilar } \tau) + P(\text{dissimilar } A \mid \text{dissimilar } \tau) = 1 \tag{15}$$

so it is sufficient to compute one of the pair of probabilities from above and still to get all the information.

Let us think for a moment what we intuitively expect from a good attribute estimator. In our opinion good attributes separate instances with different prediction values and do not separate instances with close prediction values. These considerations are fulfilled by taking one term from each group of probabilities from above and combine them in a sensible way. If we rewrite Relief's weight from Eq. (12):

$$\begin{aligned} W[A] &= 1 - P(\text{similar } A \mid \text{dissimilar } \tau) - 1 + P(\text{similar } A \mid \text{similar } \tau) \\ &= P(\text{similar } A \mid \text{similar } \tau) - P(\text{similar } A \mid \text{dissimilar } \tau) \end{aligned}$$

we see that this is actually what Relief algorithms do: they reward attribute for not separating similar prediction values and punish it for not separating different prediction values.

The similarity function used by Relief algorithms is

$$\text{similarity}(A, I_1, I_2) = -\text{diff}(A, I_1, I_2)$$

which enables intuitive probability based interpretation of results. We could get variations of Relief estimator by taking different similarity functions and by combining the computed probabilities in a different way. For example, the Contextual Merit (CM) algorithm (Hong, 1997) uses only the instances with different prediction values and therefore it takes only the first term of Eq. (12) into account. As a result CM only rewards attribute if it separates different prediction values and ignores additional information, which the similar prediction values offer. Consequently CM is less sensitive than Relief algorithms are, e.g., in parity problems with three important attributes CM separates important from unimportant attributes for a factor of 2 to 5 and only 1.05–1.19 with numerical attributes while with ReliefF under the same conditions this factor is over 100. Part of troubles CM has with numerical attributes also comes from the fact that it does not take the second term into account, namely it does not punish attributes for separating similar prediction values. As numerical attributes are very likely to do that CM has to use other techniques to confront this effect.

2.6. Relief and impurity functions

Estimations of Relief algorithms are strongly related to impurity functions (Kononenko, 1994). When the number of nearest neighbors increases i.e., when we eliminate the

requirement that the selected instance is the nearest, Eq. (5) becomes

$$W'[A] = P(\text{different value of } A \mid \text{different class}) - P(\text{different value of } A \mid \text{same class}) \tag{16}$$

If we rewrite

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

we obtain using Bayes' rule:

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

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

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

Using the above equalities we obtain:

$$W'[A] = \frac{P_{eqval} \times Ginigain'(A)}{P_{samecl}(1 - P_{samecl})} \tag{17}$$

where

$$Ginigain'(A) = \sum_V \left(\frac{P(V)^2}{\sum_V P(V)^2} \times \sum_C P(C \mid V)^2 \right) - \sum_C P(C)^2 \tag{18}$$

is highly correlated with the Gini-index gain (Breiman et al., 1984) 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 (17) (which we call myopic ReliefF), shows strong correlation of Relief's weights with the Gini-index gain. The probability $P_{equal} = \sum_V P(V)^2$ that two instances have the same value of attribute A in Eq. (17) 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 (Quinlan, 1986), distance measure (Mantaras, 1989), and binarization of attributes (Cestnik, Kononenko, & Bratko, 1987)). Equation (17) 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. Denominator, which is constant factor in Eq. (17) 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. This normalization effect remains even if Eq. (17) is used as (myopic) attribute estimator. The detailed bias analysis of various attribute estimation algorithms including Gini-index gain and myopic ReliefF can be found in Kononenko (1995).

The above derivation eliminated from the probabilities the condition that the instances are the nearest. 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 the attributes given the predicted 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 the impurity functions myopic. The 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 local peculiarities. The power of Relief is its ability to exploit information locally, taking the context into account, but still to provide the global view.

We illustrate this in figure 4 which shows dependency of ReliefF's estimate to the number of nearest neighbors taken into account. The estimates are for the parity problem with two informative, 10 random attributes, and 200 examples. The dotted line shows how the ReliefF's estimate of one of informative attributes is becoming more and more myopic with

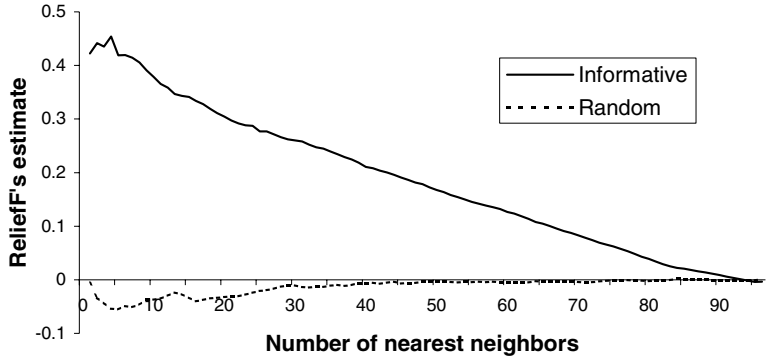

Figure 4. ReliefF's estimates of informative attribute are deteriorating with increasing number of nearest neighbors in parity domain.

the increasing number of the nearest neighbors and how the informative attribute eventually becomes indistinguishable from the unimportant attributes. The negative estimate of random attributes with small numbers of neighbors is a consequence of slight asymmetry between hits and misses. Recall that Relief algorithm (figure 1) randomly selects an instance R and its nearest instance from the same class H and from different class M. Random attributes with different values at R and H get negative update. Random attributes with different values at R and M get positive update. With larger number of nearest neighbors the positive and negative updates are equiprobable and the quality estimates of random attributes is zero. The miss has different class value therefore there has to be at least some difference also in the values of the important attributes. The sum of the differences in the values of attributes forms the distance, therefore if there is a difference in the values of the important attribute and also in the values of some random attributes, such instances are less likely to be in the nearest neighborhood. This is especially so when we are considering only a small number of nearest instances. The positive update of random attributes is therefore less likely than the negative update and the total sum of all updates is slightly negative.

2.7. Relief's weights as the portion of explained concept changes

We analyze the behavior of Relief when the number of the examples approaches infinity i.e., when the problem space is densely covered with the examples. We present the necessary definitions and prove that Relief's quality estimates can be interpreted as the ratio between the number of the explained changes in the concept and the number of examined instances. The exact form of this property differs between Relief, ReliefF and RReliefF.

We start with Relief and claim that in a classification problem as the number of examples goes to infinity Relief's weights for each attribute converge to the ratio between the number of class label changes the attribute is responsible for and the number of examined instances. If a certain change can be explained in several different ways, all the ways share the credit for it in the quality estimate. If several attributes are involved in one way of the explanation all of them get the credit in their quality estimate. We formally present the definitions and the property.

Definition 2.1. Let $B(I)$ be the set of instances from $\mathcal{I}$ nearest to the instance $I \in \mathcal{I}$ which have different prediction value τ than I:

$$B(I) = \left\{ Y \in \mathcal{I}; \mathrm{diff}(\tau, I, Y) > 0 \wedge Y = \arg\min_{Y \in \mathcal{I}} \delta(I, Y) \right\} \tag{19}$$

Let $b(I)$ be a single instance from the set $B(I)$ and $p(b(I))$ a probability that it is randomly chosen from $B(I)$. Let $A(I, b(I))$ be a set of attributes with different values at instances I and $b(I)$.

$$A(I, b(I)) = \{A \in \mathcal{A};\ b(I) \in B(I) \wedge\ \mathrm{diff}(A, I, b(I)) > 0\} \tag{20}$$

We say that attributes $A \in A(I, b(I))$ are responsible for the change of the predicted value of the instance I to the predicted value of $b(I)$ as the change of their values is one of the

minimal number of changes required for changing the predicted value of I to $b(I)$. If the sets $A(I, b(I))$ are different we say that there are different ways to explain the changes of the predicted value of I to the predicted value $b(I)$. The probability of certain way is equal to the probability that $b(I)$ is selected from $B(I)$.

Let $A(I)$ be a union of sets $A(I, b(I))$:

$$A(I) = \bigcup_{b(I) \in B(I)} A(I, b(I)) \tag{21}$$

We say that the attributes $A \in A(I)$ are responsible for the change of the predicted value of the instance I as the change of their values is the minimal necessary change of the attributes' values of I required to change its predicted value. Let the quantity of this responsibility take into account the change of the predicted value and the change of the attribute:

$$r_A(I, b(I)) = p(b(I)) \cdot \text{diff}(\tau, I, b(I)) \cdot \text{diff}(A, I, b(I)) \tag{22}$$

The ratio between the responsibility of the attribute A for the predicted values of the set of cases $\mathcal{S}$ and the cardinality m of that set is therefore:

$$R_A = \frac{1}{m} \sum_{I \in \mathcal{S}} r_A(I, b(I)) \tag{23}$$

Property 2.1. *Let the concept be described with the attributes $A \in \mathcal{A}$ and n noiseless instances $I \in \mathcal{I}$; let $\mathcal{S} \subseteq \mathcal{I}$ be the set of randomly selected instances used by Relief (line 3 on figure 1) and let m be the cardinality of that set. If Relief randomly selects the nearest instances from all possible nearest instances then for its quality estimate $W[A]$ the following property holds*:

$$\lim_{n \to \infty} W[A] = R_A \tag{24}$$

The quality estimate of the attribute can therefore be explained as the ratio of the predicted value changes the attribute is responsible for to the number of the examined instances.

Proof: Equation (24) can be explained if we look into the spatial representation. There are a number of different characteristic regions of the problem space which we usually call peaks. The Relief algorithms selects an instance R from $\mathcal{S}$ and compares the value of the attribute and the predicted value of its nearest instances selected from the set $\mathcal{I}$ (line 6 on figure 1), and than updates the quality estimates according to these values. For Relief this mean: $W[A] := W[A] + \text{diff}(A, R, M)/m - \text{diff}(A, R, H)/m$, where M is the nearest instance from the different class and H is the nearest instance from the same class.

When the number of the examples is sufficient ($n \to \infty$), H must be from the same characteristic region as R and its values of the attributes converge to the values of the instance R. The contribution of the term $-\text{diff}(A, R, H)$ to $W[A]$ in the limit is therefore 0.

Only terms $\text{diff}(A, R, M)$ contribute to $W[A]$. The instance M is randomly selected nearest instance with different prediction than R, therefore in noiseless problems there must be at least some difference in the values of the attributes and M is therefore an

Table 1. Tabular description of the concept $\tau = (A_1 \wedge A_2) \vee (A_1 \wedge A_3)$ and the responsibility of the attributes for the change of the predicted value.

Line	A_1	A_2	A_3	τ	Responsible attributes
1	1	1	1	1	A_1
2	1	1	0	1	A_1 or A_2
3	1	0	1	1	A_1 or A_3
4	1	0	0	0	A_2 or A_3
5	0	1	1	0	A_1
6	0	1	0	0	A_1
7	0	0	1	0	A_1
8	0	0	0	0	(A_1, A_2) or (A_1, A_3)

instance of $b(R)$ selected with probability $p(M)$. As M has different prediction value than R the value $\text{diff}(\tau, R, M) = 1$. The attributes with different values at R and M constitute the set $A(R, M)$. The contribution of M to $W[A]$ for the attributes from the $A(R, M)$ equals $\text{diff}(A, R, M)/m = \text{diff}(\tau, R, M)) \cdot \text{diff}(A, R, M))/m$ with probability $p(M)$).

Relief selects m instances $I \in \mathcal{S}$ and for each I randomly selects its nearest miss $b(I)$ with probability $p(b(I))$. The sum of updates of $W[A]$ for each attribute is therefore: $\sum_{I \in \mathcal{S}} p(b(I))\text{diff}(\tau, R, b(I))\text{diff}(A, R, b(I))/m) = R_A$, and this concludes the proof. □

Let us show an example, which illustrates the idea. We have a Boolean problem where the class value is defined as $\tau = (A_1 \wedge A_2) \vee (A_1 \wedge A_3)$. Table 1 gives a tabular description of the problem. The right most column shows which of the attributes is responsible for the change of the predicted value.

In line 1 we say that A_1 is responsible for the class assignment because changing its value to 0 would change τ to 0, while changing only one of A_2 or A_3 would leave τ unchanged. In line 2 changing any of A_1 or A_2 would change τ too, so A_1 and A_2 represent two manners how to change τ and also share the responsibility. Similarly we explain lines 3 to 7, while in line 8 changing only one attribute is not enough for τ to change. However, changing A_1 and A_2 or A_1 and A_3 changes τ. Therefore the minimal number of required changes is 2 and the credit (and updates in the algorithm) goes to both A_1 and A_2 or A_1 and A_3. There are 8 peaks in this problem which are equiprobable so A_1 gets the estimate $\frac{4+2\cdot\frac{1}{2}+2\cdot\frac{1}{2}}{8} = \frac{3}{4} = 0.75$ (it is alone responsible for lines 1, 5, 6, and 7, shares the credit for lines 2 and 3 and cooperates in both credits for line 8). A_2 (and similarly A_3) gets estimate $\frac{2\cdot\frac{1}{2}+\frac{1}{2}}{8} = \frac{3}{16} = 0.1875$ (it shares the responsibility for lines 2 and 4 and cooperates in one half of line 8).

Figure 5 shows the estimates of the quality of the attributes for this problem for Relief (and also ReliefF). As we wanted to scatter the concept we added besides three important attributes also five random binary attributes to the problem description. We can observe that as we increase the number of the examples the estimate for A_1 is converging to 0.75, while the estimates for A_2 and A_3 are converging to 0.1875 as we expected. The reason for rather slow convergence is in the random sampling of the examples, so we need much more examples than the complete description of the problem (256 examples).

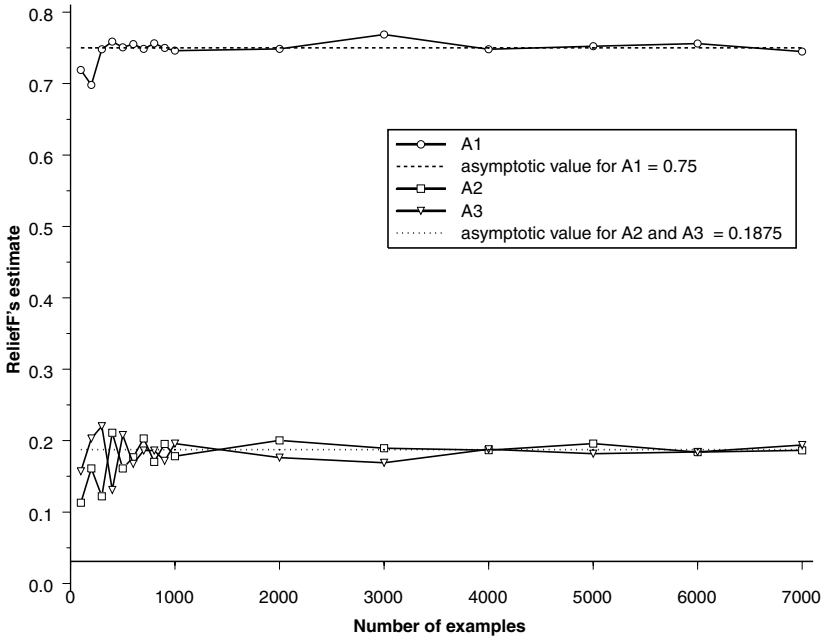

Figure 5. The estimates of the attributes by Relief and ReliefF are converging to the ratio between class labels they explain and the number of examined instances.

For ReliefF this property is somehow different. Recall that in this algorithm we search nearest misses from each of the classes and weight their contributions with prior probabilities of the classes (line 9). We have to define the responsibility for the change of the class value c_i to the class value c_j.

Definition 2.2. Let $B_j(I)$ be a set of instances from $\mathcal{I}$ nearest to the instance $I \in \mathcal{I}$, $\tau(I) = c_i$ with prediction value c_j, $c_j \neq c_i$:

$$B_j(I) = \{Y \in \mathcal{I}; \tau(Y) = c_j \wedge Y = \arg\min_{Y \in \mathcal{I}} \delta(I, Y)\} \tag{25}$$

Let $b_j(I)$ be a single instance from the set $B_j(I)$ and $p(b_j(I))$ a probability that it is randomly chosen from $B_j(I)$. Let $A(I, b_j(I))$ be a set of attributes with different values at instances I and $b_j(I)$.

$$A(I, b_j(I)) = \{A \in \mathcal{A}; b_j(I) \in B_j(I) \wedge \text{diff}(A, I, b_j(I)) > 0\} \tag{26}$$

We say that attributes $A \in A(I, b_j(I))$ are responsible for the change of the predicted value of the instance I to the predicted value of $b_j(I)$ as the change of their values is one of the minimal number of changes required for changing the predicted value of I to $b_j(I)$. If the sets $A(I, b_j(I))$ are different we say that there are different ways to explain the changes of

the predicted value of I to the predicted value $b_j(I)$. The probability of certain way is equal to the probability that $b_j(I)$ is selected from $B_j(I)$.

Let $A_j(I)$ be a union of sets $A(I, b_j(I))$:

$$A_j(I) = \bigcup_{b_j(I) \in B_j(I)} A(I, b_j(I)) \tag{27}$$

We say that the attributes $A \in A_j(I)$ are responsible for the change of predicted value c_i of the instance I to the value $c_j \neq c_i$ as the change of their values is the minimal necessary change of the attributes' values of I required to change its predicted value to c_j. Let the quantity of this responsibility take into account the change of the predicted value and the change of the attribute:

$$r_{A_j}(I, b(I)) = p(b_j(I)) \cdot \mathrm{diff}(\tau, I, b_j(I)) \cdot \mathrm{diff}(A, I, b_j(I)) \tag{28}$$

The ratio between the responsibility of the attribute A for the change of predicted values from c_i to c_j for the set of cases $\mathcal{S}$ and the cardinality m of that set is thus:

$$R_A(i, j) = \frac{1}{m} \sum_{I \in \mathcal{S}} r_{A_j}(I, b_j(I)) \tag{29}$$

Property 2.2. *Let $p(c_i)$ represent the prior probability of the class c_i. Under the conditions of Property* 2.1, *algorithm ReliefF behaves as*:

$$\lim_{n \to \infty} W[A] = \sum_{i=1}^{c} \sum_{\substack{j=1 \\ j \neq i}}^{c} \frac{p(c_i)p(c_j)}{1 - p(c_i)} R_A(i, j) \tag{30}$$

We can therefore explain the quality estimates as the ratio of class values changes the attribute is responsible for to the number of the examined instances weighted with the prior probabilities of class values.

Proof: is similar to the proof of Property 2.1. Algorithm ReliefF selects an instance $R \in \mathcal{S}$ (line 3 on figure 2). Probability of R being labelled with class value c_i is equal to prior probability of that value $p(c_i)$. The algorithm than searches for k nearest instances from the same class and k nearest instances from each of the other classes (line 6 on figure 2) and and than updates the quality estimates $W[A]$ according to these values (lines 8 and 9 on figure 2).

As the number of the examples is sufficient ($n \to \infty$), instances H_j must be from the same characteristic region as R and their values of the attributes converge to the values of the attributes of the instance R. The contribution of nearest hits to $W[A]$ in the limit is therefore 0.

Only nearest misses contribute to $W[A]$. The instances M_j are randomly selected nearest instances with different prediction than R, therefore in the noiseless problems there must be at least some difference in the values of the attributes and all M_j are therefore instances of $b_j(R)$ selected with probability $p_j(M)$. The contributions of k instances are weighted with the probabilities $p(b_j(R))$.

As M_j have different prediction value than R the value $\text{diff}(\tau, R, M_j) = 1$. The attributes with different values at R and M_j constitute the set $A_j(R, b_j(R))$. The contribution of the instances M_j to $W[A]$ for the attributes from the $A_j(R, b_j(R))$ equals $\sum_{\substack{j=1 \\ j\neq i}}^{c} \frac{p(c_j)}{1-p(c_i)} p(b_j(R))\text{diff}(A, R_i, b_j(R))/m$.

ReliefF selects m instances $I \in \mathcal{S}$ where $p(c_i) \cdot m$ of them are labelled with c_i. For each I it randomly selects its nearest misses $b_j(I)$, $j \neq i$ with probabilities $p(b_j(I))$. The sum of updates of $W[A]$ for each attribute is therefore:

$$W[A] = \sum_{I\in\mathcal{S}} \sum_{\substack{j=1 \\ j\neq i}}^{c} \frac{p(c_j)}{1-p(c_i)} p(b_j(I))\text{diff}(C, I, b_j(I))\text{diff}(A, I, b_j(I))/m$$

$$= \sum_{I\in\mathcal{S}} \sum_{\substack{j=1 \\ j\neq i}}^{c} \frac{p(c_j)}{1-p(c_i)} r_{A_j}(I, b(I))/m,$$

This can be rewritten as the sum over the class values as only the instances with class value c_i contribute to the $R_A(i, j)$:

$$= \sum_{i=1}^{c} p(c_i) \sum_{\substack{j=1 \\ j\neq i}}^{c} \frac{p(c_j)}{1-p(c_i)} R_A(i, j)$$

$$= \sum_{i=1}^{c} \sum_{\substack{j=1 \\ j\neq i}}^{c} \frac{p(c_i)p(c_j)}{1-p(c_i)} R_A(i, j),$$

which we wanted to prove. □

Corollary 2.3. *In two class problems where* diff *function is symmetric*: $\text{diff}(A, I_1, I_2) = \text{diff}(A, I_2, I_1)$ *Property* 2.2 *is equivalent to Property* 2.1.

Proof: As diff is symmetric also the responsibility is symmetric $R_A(i, j) = R_A(j, i)$. Let us rewrite Equation (30) with $p(c_1) = p$ and $p(c_2) = 1 - p$. By taking into account that we are dealing with only two classes we get $\lim_{n\to\infty} W[A] = R_A(1, 2) = R_A$. □

In a Boolean noiseless case as in the example presented above the fastest convergence would be with only 1 nearest neighbor. With more nearest neighbors (default with the algorithms ReliefF and RReliefF) we need more examples to see this effect as all of them has to be from the same/nearest peak.

The interpretation of the quality estimates with the ratio of the explained changes in the concept is true for RReliefF as well, as it also computes Eq. (12), however, the updates are proportional to the size of the difference in the prediction value. The exact formulation and proof remain for further work.

Note that the sum of the expressions (24) and (30) for all attributes is usually greater than 1. In certain peaks there are more than one attribute responsible for the class assignment i.e., the minimal number of attribute changes required for changing the value of the class is greater than 1 (e.g., line 8 in Table 1). The total number of the explanations is therefore greater (or equal) than the number of the inspected instances. As Relief algorithms normalize the

weights with the number of the inspected instances m and not the total number of possible explanations, the quality estimations are not proportional to the attributes's responsibility but present rather a portion of the explained changes. For the estimates to represent the proportions we would have to change the algorithm and thereby lose the probabilistic interpretation of attributes' weights.

When we omit the assumption of the sufficient number of the examples then the estimates of the attributes can be greater than their asymptotic values because the instances more distant than the minimal number of required changes might be selected into the set of nearest instances and the attributes might be positively updated also when they are responsible for the changes which are more distant than the minimal number of required changes.

The behavior of the Relief algorithm in the limit (Eq. (2.1)) is the same as the the asymptotic behavior of the algorithm Contextual Merit (CM) (Hong, 1997) which uses only the contribution of nearest misses. In multi class problems CM searches for nearest instances from different class disregarding the actual class they belong, while ReliefF selects equal number of instances from each of the different classes and normalizes their contribution with their prior probabilities. The idea is that the algorithm should estimate the ability of attributes to separate each pair of the classes regardless of which two classes are closest to each other. It was shown (Kononenko, 1994) that this approach is superior and the same normalization factors occur also in asymptotic behavior of ReliefF given by Equation (2.2).

Based solely on the asymptotic properties one could come to, in our opinion, the wrong conclusion that it is sufficient for estimation algorithms to consider only nearest instances with different prediction value. While the nearest instances with the same prediction have no effect when the number of the instances is unlimited they nevertheless play an important role in problems of practical sizes.

Clearly the interpretation of Relief's weights as the ratio of explained concept changes is more comprehensible than the interpretation with the difference of two probabilities. The responsibility for the explained changes of the predicted value is intuitively clear. Equation (24) is non probabilistic, unconditional, and contains a simple ratio, which can be understood taking the unlimited number of the examples into account. The actual quality estimates of the attributes in given problem are therefore approximations of these ideal estimates which occur only with abundance of data.

3. Parameters of ReliefF and RReliefF

In this section we address different parameters of ReliefF and RReliefF: the impact of different distance measures, the use of numerical attributes, how distance can be taken into account, the number of nearest neighbors used and the number of iterations.

The datasets not defined in the text and used in our demonstrations and tests are briefly described in the Appendix.

3.1. Metrics

The diff(A_i, I_1, I_2) function calculates the difference between the values of the attribute A_i for two instances I_1 and I_2. Sum of differences over all attributes is used to determine the

distance between two instances in the nearest neighbors calculation.

$$\delta(I_1, I_2) = \sum_{i=1}^{a} \text{diff}(A_i, I_2, I_2) \tag{31}$$

This looks quite simple and parameterless, however, in instance based learning there are a number of feature weighting schemes which assign different weights to the attributes in the total sum:

$$\delta(I_1, I_2) = \sum_{i=1}^{a} w(A_i)\text{diff}(A_i, I_1, I_2) \tag{32}$$

ReliefF's estimates of attributes' quality can be successfully used as such weights (Wettschereck, Aha, & Mohri, 1997).

Another possibility is to form a metric in a different way:

$$\delta(I_1, I_2) = \left(\sum_{i=1}^{a} \text{diff}(A_i, I_2, I_2)^p \right)^{\frac{1}{p}} \tag{33}$$

which for $p = 1$ gives Manhattan distance and for $p = 2$ Euclidean distance. In our use of Relief algorithms we never noticed any significant difference in the estimations using these two metrics. For example, on the regression problems from the UCI repository (Murphy & Aha, 1995) (8 tasks: Abalone, Auto-mpg, Autoprice, CPU, Housing, PWlinear, Servo, and Wisconsin breast cancer) the average (linear) correlation coefficient is 0.998 and (Spearman's) rank correlation coefficient is 0.990.

3.2. Numerical attributes

If we use diff function as defined by (1) and (2) we run into the problem of underestimating numerical attributes. Let us illustrate this by taking two instances with 2 and 5 being their values of attribute A_i, respectively. If A_i is the nominal attribute, the value of $\text{diff}(A_i, 2, 5) = 1$, since the two categorical values are different. If A_i is the numerical attribute, $\text{diff}(A_i, 2, 5) = \frac{|2-5|}{7} \approx 0.43$. Relief algorithms use results of diff function to update their weights therefore with this form of diff numerical attributes are underestimated.

Estimations of the attributes in Modulo-8-2 data set (see definition by Eq. (43)) by RReliefF in left hand side of Table 2 illustrate this effect. Values of each of 10 attributes are integers in the range 0–7. Half of the attributes are treated as nominal and half as numerical; each numerical attribute is exact match of one of the nominal attributes. The predicted value is the sum of 2 important attributes by modulo 8: $\tau = (I_1 + I_2) \bmod 8$. We can see that nominal attributes get approximately double score of their numerical counterparts. This causes that not only important numerical attributes are underestimated but also numerical random attributes are overestimated which reduces the separability of the two groups of attributes.

Table 2. Estimations of attributes in Modulo-8-2 dataset assigned by RReliefF. Left hand estimations are for diff function defined by Eqs. (1) and (2), while the right hand estimations are for diff function using thresholds (Eq. (34)).

Attribute	No ramp	Ramp
Important-1, nominal	0.193	0.436
Important-2, nominal	0.196	0.430
Random-1, nominal	−0.100	−0.200
Random-2, nominal	−0.105	−0.207
Random-3, nominal	−0.106	−0.198
Important-1, numerical	0.096	0.436
Important-2, numerical	0.094	0.430
Random-1, numerical	−0.042	−0.200
Random-2, numerical	−0.044	−0.207
Random-3, numerical	−0.043	−0.198

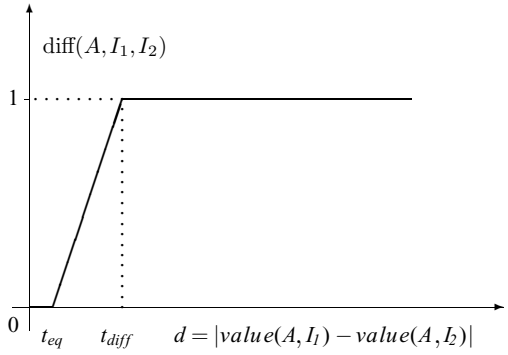

Figure 6. Ramp function.

We can overcome this problem with the ramp function as proposed by Hong (1994, 1997). It can be defined as a generalization of diff function for the numerical attributes (see figure 6):

$$\text{diff}(A, I_1, I_2) = \begin{cases} 0; & d \leq t_{\text{eq}} \\ 1; & d > t_{\text{diff}} \\ \dfrac{d - t_{\text{eq}}}{t_{\text{diff}} - t_{\text{eq}}}; & t_{\text{eq}} < d \leq t_{\text{diff}} \end{cases} \tag{34}$$

where $d = |value(A, I_1) - value(A, I_2)|$ presents the distance between attribute values of two instances, and t_{eq} and t_{diff} are two user definable threshold values; t_{eq} is the maximum distance between two attribute values to still consider them equal, and t_{diff} is the minimum

distance between attribute values to still consider them different. If we set $t_{\text{eq}} = 0$ and $t_{\text{diff}} = \max(A) - \min(A)$ we obtain (2) again.

Estimations of attributes in Modulo-8-2 data set by RReliefF using the ramp function are in the right hand side of Table 2. The thresholds are set to their default values: 5% and 10% of the length of the attribute's value interval for t_{eq} and t_{diff}, respectively. We can see that estimates for nominal attributes and their numerical counterparts are identical.

The threshold values can be set by the user for each attribute individually, which is especially appropriate when we are dealing with measured attributes. Thresholds can be learned in advance considering the context (Ricci & Avesani, 1995) or automatically set to sensible defaults (Domingos, 1997). The sigmoidal function could also be used, but its parameters do not have such straightforward interpretation. In general if the user has some additional information about the character of a certain attribute she/he can supply the appropriate diff function to (R)ReliefF.

We use the ramp function in results reported throughout this work.

3.3. Taking distance into account

In instance based learning it is often considered useful to give more impact to the near instances than to the far ones i.e., to weight their impact inversely proportional to their distance from the query point.

RReliefF is already taking the distance into account through Eqs. (10) and (11). By default we are using 70 nearest neighbors and exponentially decrease their influence with increasing distance from the query point. ReliefF originally used constant influence of k nearest neighbors with k set to some small number (usually 10). We believe that the former approach is less risky (as it turned out in a real world application (Dalaka et al., 2000)) because as we are taking more near neighbors we reduce the risk of the following pathological case: we have a large number of instances and a mix of nominal and numerical attributes where numerical attributes prevail; it is possible that all the nearest neighbors are closer than 1 so that there are no nearest neighbors with differences in values of a certain nominal attribute. If this happens in a large part of the problem space this attribute gets zero weight (or at least small and unreliable one). By taking more nearest neighbors with appropriately weighted influence we eliminate this problem.

ReliefF can be adjusted to take distance into account by changing the way it updates it weights (lines 8 and 9 in figure 2):

$$W[A] := W[A] - \frac{1}{m}\sum_{j=1}^{k} \text{diff}(A, R, H_j)d(R, H_j) + \frac{1}{m}\sum_{C \neq class(R)} \frac{P(C)}{1 - P(class(R))}\sum_{j=1}^{k} \text{diff}(A, R, M_j(C))d(R, M_j(C)) \quad (35)$$

The distance factor of two instances $d(I_1, I_2)$ is defined with Eqs. (10) and (11).

The actual influence of the near instances is normalized: as we want probabilistic interpretation of results each random query point should give equal contribution. Therefore we

normalize contributions of each of its k nearest instances by dividing it with the sum of all k contributions in Eq. (10).

However, by using ranks instead of actual distances we might lose the intrinsic self normalization contained in the distances between instances of the given problem. If we wish to use the actual distances we only change Eq. (11):

$$d_1(i, j) = \frac{1}{\sum_{l=1}^{a} \text{diff}(A_l, R_i, I_j)} \tag{36}$$

We might use also some other decreasing function of the distance, e.g., square of the sum in the above expression, if we wish to emphasize the influence of the distance:

$$d_1(i, j) = \frac{1}{\left(\sum_{l=1}^{a} \text{diff}(A_l, R_i, I_j)\right)^2} \tag{37}$$

The differences in estimations can be substantial although the average correlation coefficients between estimations and ranks over regression datasets from UCI obtained with RReliefF are high as shown in Table 3.

The reason for substantial deviation in Auto-mpg problem is sensibility of the algorithm concerning the number of nearest neighbors when using actual distances. While with expression (11) we exponentially decreases influence according to the number of nearest neighbors, Eqs. (36) and (37) use inverse of the distance and also instances at a greater distance may have a substantial influence. With actual distances and 70 nearest instances in this problem we get myopic estimate which is uncorrelated to non-myopic estimate. So, if we are using actual distances we have to use a moderate number of the nearest neighbors or test several settings for it.

Table 3. Linear correlation coefficients between estimations and ranks over 8 UCI regression datasets. We compare RReliefF using Eqs. (11), (36) and (37).

	Eqs. (11) and (36)		Eqs. (11) and (37)		Eqs. (36) and (37)	
Problem	ρ	r	ρ	r	ρ	r
Abalone	0.969	0.974	0.991	0.881	0.929	0.952
Auto-mpg	−0.486	0.174	0.389	−0.321	0.143	0.357
Autoprice	0.844	0.775	0.933	0.819	0.749	0.945
CPU	0.999	0.990	0.990	0.943	1.000	0.943
Housing	0.959	0.830	0.937	0.341	0.181	0.769
Servo	0.988	0.999	0.985	0.800	1.000	0.800
Wisconsin	0.778	0.842	0.987	0.645	0.743	0.961
Average	0.721	0.798	0.888	0.587	0.678	0.818

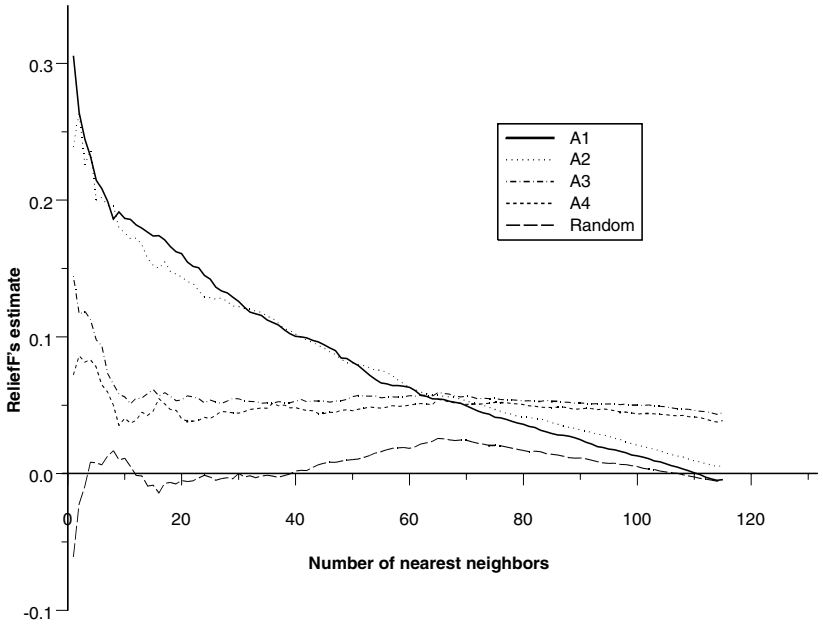

Figure 7. ReliefF's estimates and the number of nearest neighbors.

3.4. Number of nearest neighbors

While the number of nearest neighbors used is related to the distance as described above there are still some other issues to be discussed, namely how sensitive Relief algorithms are to the number of nearest neighbors used (lines 4, 5, and 6 in figure 2 and line 4 in figure 3). The optimal number of the nearest neighbors used is problem dependent as we illustrate in figure 7 which shows ReliefF's estimates for four important and one of the random attributes in Boolean domain defined as:

$$\text{Bool-Simple:} \quad C = (A_1 \oplus A_2) \vee (A_3 \wedge A_4). \tag{38}$$

We know that A_1 and A_2 are more important for determination of the class value than A_3 and A_4 (the attributes' values are equiprobable). ReliefF recognizes this with up to 60 nearest neighbors (there are 200 instances). With more that 80 nearest neighbors used the global view prevails and the strong conditional dependency between A_1 and A_2 is no longer detected. If we increase the number of instances from 200 to 900 we obtain similar picture as in figure 7, except that the crossing point moves from 70 to 250 nearest neighbors.

The above example is quite illustrative: it shows that ReliefF is robust in the number of nearest neighbors as long as it remains relatively small. If it is too small it may not be robust enough, especially with more complex or noisy concepts. If the influence of all neighbors is equal disregarding their distance to the query point the proposed default value is 10 (Kononenko, 1994). If we do take distance into account we use 70 nearest neighbors with

exponentially decreasing influence ($\sigma = 20$ in Eq. (11). In a similar problem with CM algorithm (Hong, 1997) it is suggested that using $\log n$ nearest neighbors gives satisfactory results in practice. However we have to emphasize that this is problem dependent and especially related to the problem complexity, the amount of noise and the number of available instances.

Another solution to this problem is to compute estimates for all possible numbers of nearest neighbors and take the highest estimate of each attribute as its final result. In this way we avoid the danger of accidentally missing an important attribute. Because all attributes receive somewhat higher score we risk that some differences would be blurred and, we increase the computational complexity. The former risk can be resolved later on in the process of investigating the domain by producing a graph similar to figure 7 showing dependencies of ReliefF's estimates on the number of nearest neighbors. The computational complexity increases from $O(m \cdot n \cdot a)$ to $O(m \cdot n \cdot (a + \log n))$ due to sorting of the instances with decreasing distance. In the algorithm we have to do also some additional bookkeeping, e.g., keep the score for each attribute and each number of nearest instances.

3.5. Sample size and number of iterations

Estimates of Relief algorithms are actually statistical estimates i.e., the algorithms collect the evidence for (non)separation of similar instances by the attribute across the problem space. To provide reliable estimates the coverage of the problem space must be appropriate. The sample has to cover enough representative boundaries between the prediction values. There is an obvious trade off between the use of more instances and the efficiency of computation. Wherever we have large datasets sampling is one of possible solutions to make problem tractable. If a dataset is reasonably large and we want to speed-up computations we suggest selection of all the available instances (n in complexity calculations), and rather to control the number of iterations with parameter m (line 2 in figures 2 and 3). As it is non trivial to select a representative sample of the unknown problem space our decision is in favor of the (possibly) sparse coverage of the more representative space rather than the dense coverage of the (possibly) non-representative sample.

Figure 8 illustrates the behavior of RReliefF's estimates changing with the number of iterations on Cosinus-Lin dataset with 10 random attributes and 1000 examples. We see that after the initial variation at around 20 iterations the estimates settle to stable values, except for difficulties at detecting differences e.g., the quality difference between A_1 and A_3 is not resolved until around 300 iterations (A_1 within the cosine function controls the sign of the expression, while A_3 with the coefficient 3 controls the amplitude of the function). We should note that this is quite typical behavior and usually we get stable estimates after 20–50 iterations. However if we want to refine the estimates we have to iterate further on. The question of how much more iterations we need is problem dependent. We try to answer this question for some chosen problems in Section 4.5.

4. Analysis of performance

In this Section we investigate some practical issues on the use of ReliefF and RReliefF: what kind of dependencies they detect, how do they scale up to large number of examples

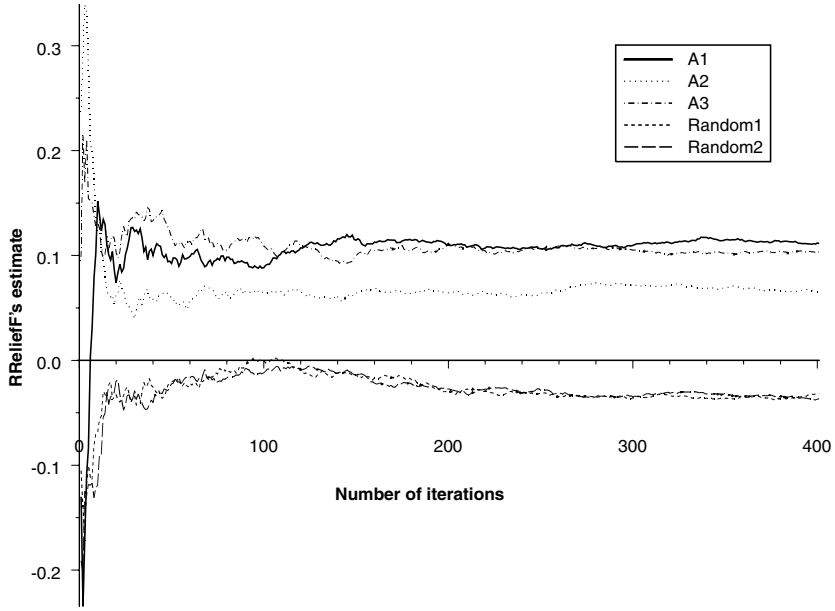

Figure 8. RReliefF's estimates and the number of iterations (m) on Cosinus-Lin dataset.

and features, how many iterations we need for reliable estimation, how robust are they regarding the noise, and how irrelevant and duplicate attributes influence their output. For ReliefF some of these questions have been tackled in a limited scope in Kononenko (1994) and Kononenko, Šimec, and Robnik-Šikonja (1997) and for RReliefF in Robnik Šikonja and Kononenko (1997).

Before we analyze these issues we have to define some useful measures and concepts for performance analysis, comparison and explanation.

4.1. Useful definitions and concepts

4.1.1. Concept variation. We are going to examine abilities of ReliefF and ReliefF to recognize and rank important attributes for a set of different problems. As Relief algorithms are based on the nearest neighbor paradigm they are capable to detect classes of problems for which this paradigm holds. Therefore we will define a measure of concept difficulty, called concept variation, based on the nearest neighbor principle.

Concept variation (Rendell & Seshu, 1990) is a measure of problem difficulty based on the nearest neighbor paradigm. If many pairs of neighboring examples do not belong to the same class, then the variation is high and the problem is difficult (Perèz & Rendell, 1996; Vilalta, 1999). It is defined on a-dimensional Boolean concepts as

$$V_a = \frac{1}{a2^a} \sum_{i=1}^{a} \sum_{\text{neigh}(X,Y,i)} \text{diff}(C, X, Y) \tag{39}$$

where the inner summation is taken over all 2^a pairs of the neighboring instances X and Y that differ only in their i-th attribute. Division by $a2^a$ converts double sum to average and normalizes the variation to [0, 1] range. The two constant concepts (which have all class values 0 and 1, respectively) have variation 0, and the two parity concepts of order a have variation 1. Random concepts have variation around 0.5, because the neighbors of any given example are approximately evenly split between the two classes. The variation around 0.5 can be thus considered as high and the problem as difficult.

This definition of the concept variation is limited to Boolean problems and demands the description of the whole problem space. The modified definition by (Vilalta, 1999) encompasses the same idea but is defined also for numeric and non-binary attributes. It uses only a sample of examples, which makes variation possible to compute in the real world problems as well. Instead of all pairs of the nearest examples it uses a distance metric to weight the contribution of each example to the concept variation. The problem with this definition is that it uses the contributions of all instances and thereby looses the information on locality. We propose another definition which borrows from both mentioned above.

$$V = \frac{1}{m \cdot a} \sum_{i=1}^{m} \sum_{j=1}^{a} \text{diff}(C, \text{neigh}(X_i, Y, j)) \tag{40}$$

where Y has to be the nearest instance different from X_i in attribute j. Similarly to the original definition we are counting the number of differences in each dimension but we are doing so only on a sample of m examples available. We are using diff function which allows handling of multi valued and numerical attributes. Note that in the case of a dimensional Boolean concept with all 2^a instances available Eq. (40) is equivalent to Eq. (39).

We present behaviors of different definitions of the concept variation in figure 9. We have measured the concept variations on the parity concepts with orders 2 to 9 on dataset with 9 attributes and 512 randomly sampled examples. By the original definition (Rendell & Seshu, 1990) the concept variation is linearly increasing with the parity order. For modified definition (Vilalta, 1999) the concept variation remains almost constant (0.5), while our definition (V with 1 nearest) exhibits similar behavior as the original. If we were not sampling the examples but rather generated the whole problem space (which is what the original definition uses) V behaved exactly as the original definition. If in our definition (Eq. (40)) we averaged the contribution of 10 nearest instances that differed from the instance X_i in the j-th attribute (V with 10 nearest on figure 9) the behavior becomes more similar to that of (Vilalta, 1999). This indicates that locality is crucial in sensible definitions of the concept variation.

Note that we are using diff function and if prediction values are used instead of class values, this definition can be used for regression problems as well.

4.1.2. Performance measures. In our experimental scenario below we run ReliefF and RReliefF on a number of different problems and observe

– if their estimates distinguish between important attributes (conveying some information about the concept) and unimportant attributes and

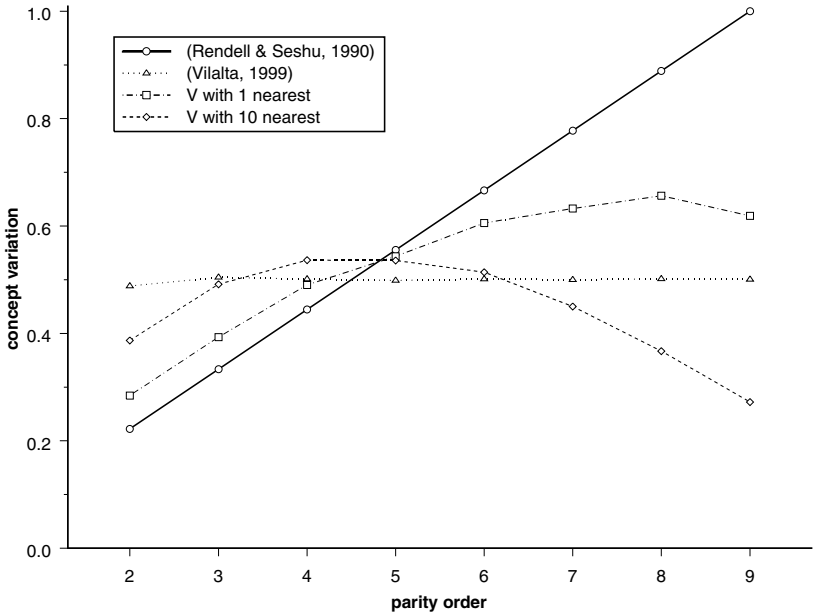

Figure 9. Concept variation by different definitions on parity concepts of orders 2 to 9 and 512 examples.

– if their estimates rank important attributes correctly (attributes which have stronger influence on prediction values should be ranked higher).

In estimating the success of Relief algorithms we use the following measures:

Separability s is the difference between the lowest estimate of the important attributes and the highest estimate of the unimportant attributes.

$$s = W_{I_{\text{worst}}} - W_{R_{\text{best}}} \tag{41}$$

We say that a heuristics is successful in separating between the important and unimportant attributes if $s > 0$.

Usability u is the difference between the highest estimates of the important and unimportant attributes.

$$u = W_{I_{\text{best}}} - W_{R_{\text{best}}} \tag{42}$$

We say that estimates are useful if u is greater than 0 (we are getting at least some information from the estimates e.g., the best important attribute could be used as the split in tree based model). It holds that $u \geq s$.

4.1.3. Other attribute estimators for comparison. For some problems we want to compare the performance of ReliefF and RReliefF with other attribute estimation heuristics. We

have chosen the most widely used. For classification this is the gain ratio (used in e.g., C4.5 (Quinlan, 1993)) and for regression it is the mean squared error (MSE) of average prediction value (used in e.g., CART (Breiman et al., 1984)).

Note that MSE, unlike Relief algorithms and gain ratio, assigns lower weights to better attributes. To make s and u curves comparable to that of RReliefF we are actually reporting separability and usability with the sign reversed.

4.2. Some typical problems and dependencies

We use artificial datasets in the empirical analysis because we want to control the environment: in real-world datasets we do not fully understand the problem and the relation of the attributes to the target variable. Therefore we do not know what a correct output of the feature estimation should be and we cannot evaluate the quality estimates of the algorithms. We mostly use variants of parity-like problems because these are the most difficult problems within the nearest neighbor paradigm. We try to control difficulty of the concepts (which we measure with the concept variation) and therefore we introduce many variants with various degrees of the concept variation. We use also some non-parity like problems and demonstrate performances of Relief algorithms on them. We did not find another conceptually different class of problems on which the Relief algorithms would exhibit significantly different behavior.

4.2.1. Sum by modulo concepts. We start our presentation of abilities of ReliefF and RReliefF with the concepts based on summation by modulo. Sum by modulo p problems are integer generalizations of parity concept, which is a special case where attributes are Boolean and the class is defined by modulo 2. In general, each Modulo-p-I problem is described by a set of attributes with integer values in the range $[0, p)$. The predicted value $\tau(X)$ is the sum of I important attributes by modulo p.

$$\text{Modulo-}p\text{-}I\text{:} \quad \tau(X) = \left(\sum_{i=1}^{I} X_i \right) \bmod p \tag{43}$$

Let us start with the base case i.e., Boolean problems ($p = 2$). As an illustrative example we will show problems with parity of 2–8 attributes ($I \in [2, 8]$) on the data set described with 9 attributes and 512 examples (a complete description of the domain). Figure 10 shows s curve for this problem (u curve is identical as we have a complete description of a domain). In this and all figures below each point on the graph is an average of 10 runs.

We can see that separability of the attributes is decreasing with increasing difficulty of the problem for parity orders of 2, 3, and 4. At order 5 when more than half of the attributes are important the separability becomes negative i.e., we are no longer capable of separating the important from unimportant attributes. The reason is that we are using more than one nearest neighbor (one nearest neighbor would always produce positive s curve on this noiseless problem) and as the number of peaks in the problem increases with 2^I, and the number of examples remains constant (512) we are having less and less examples per peak. At $I = 5$ when we get negative s the number of nearest examples from the neighboring peaks with

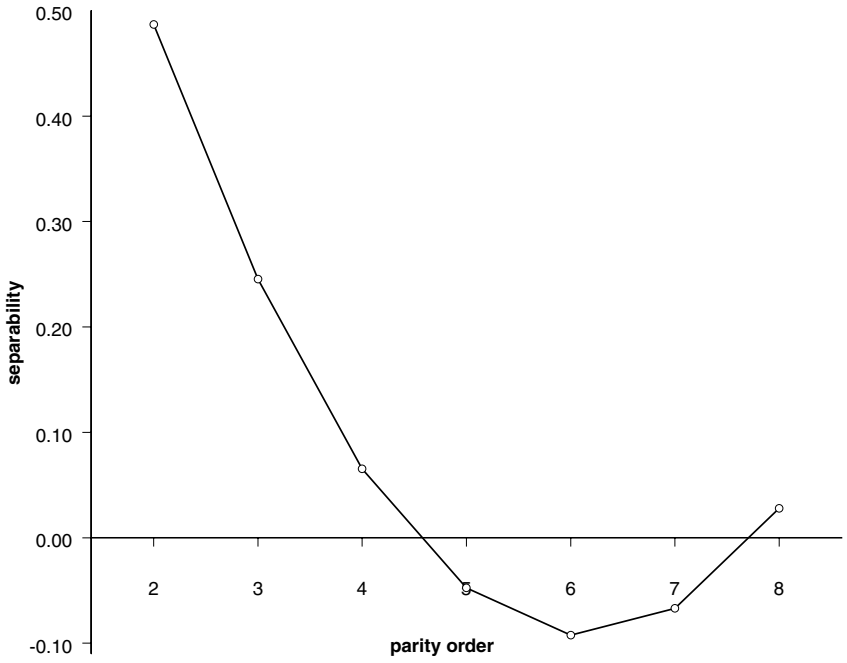

Figure 10. Separability on parity concepts of orders 2 to 8 and all 512 examples.

distance 1 (different parity) surpasses the number of nearest examples from the target peak. An interesting point to note is when $I = 8$ (there is only 1 random attribute left) and s becomes positive again. The reason for this is that the number of nearest examples from the target peak and neighboring peaks with distance 2 (with the same parity!) surpasses the number of nearest examples from neighboring peaks with distance 1 (different parity).

A sufficient number of examples per peak is crucial for reliable estimations with ReliefF as we show in figure 11. The bottom s and u curves show exactly the same problem as above (in figure 10) but in this case the problem is not described with all 512 examples but rather with 512 randomly generated examples. The s scores are slightly lower than in figure 10 as we have in effect decreased the number of different examples (to 63.2% of the total). The top s and u curves show the same problem but with 8 times more examples (4096). We can observe that with that many examples the separability for all problems is positive.

In the next problem p increases while the number of important attributes and the number of examples are fixed (to 2 and 512, respectively).

Two curves at the bottom of figure 12 show separability (usability is very similar and is omitted due to clarity) for the classification problem (there are p classes) and thus we can see the performance of ReliefF. The attributes can be treated as nominal or numerical, however, the two curves show similar behavior i.e., separability is decreasing with increasing modulo. This is expected as the complexity of problems is increasing with the number of classes, attribute values, and peaks. The number of attributes values and classes is increasing with p, while the number of peaks is increasing with p^I (polynomial increase).

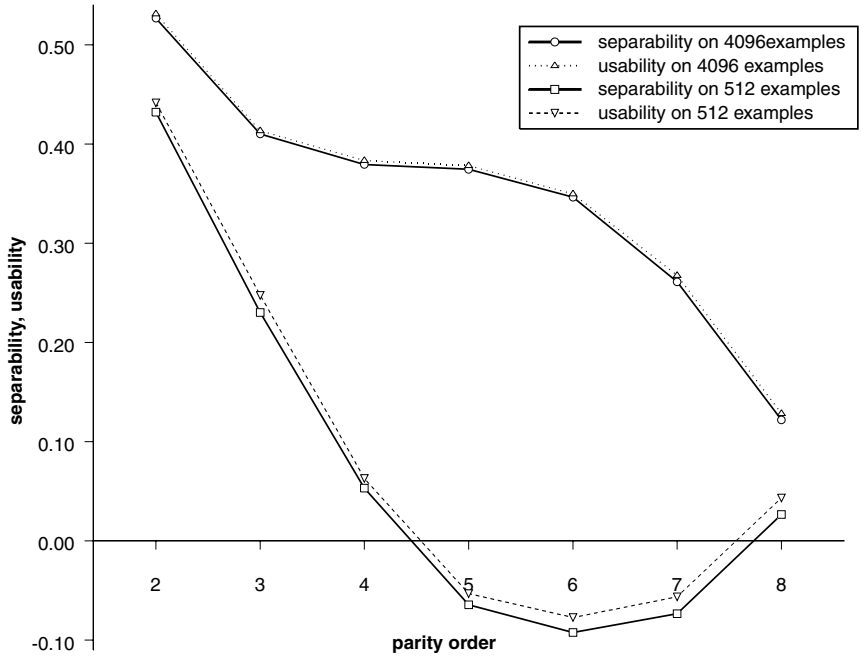

Figure 11. Separability and usability on parity concepts of orders 2 to 8 and randomly sampled 512 or 4096 examples.

Again, more examples would shift positive values of s further to the right. A slight but important difference between separability for nominal and numerical attributes shows that numerical attributes convey more information in this problem. Function diff is 1 for any two different nominal attributes while for numerical attributes diff returns the relative numerical difference which is more informative.

The same modulo problems can be viewed as regression problems and the attributes can be again interpreted as nominal or numerical. Two curves at the top of figure 12 shows separability for the modulo problem formulated as regression problem (RReliefF is used). We get positive s values for larger modulo compared to the classification problem and if the attributes are treated as numerical the separability is not decreasing with modulo at all. The reason is that classification problems were actually more difficult. We tried to predict p separate classes (e.g., results 2 and 3 are completely different in classification) while in regression we model numerical values (2 and 3 are different relatively to the scale).

Another interesting problem arises if we fix modulo to a small number (e.g., $p = 5$) and vary the number of important attributes. Figure 13 shows s curves for 4096 examples and 10 random attributes. At modulo 5 there are no visible differences in the performance for nominal and numerical attributes therefore we give curves for nominal attributes only. The s curves are decreasing rapidly with increasing I. Note that the problem complexity (number of peaks) is increasing with p^I (exponentially).

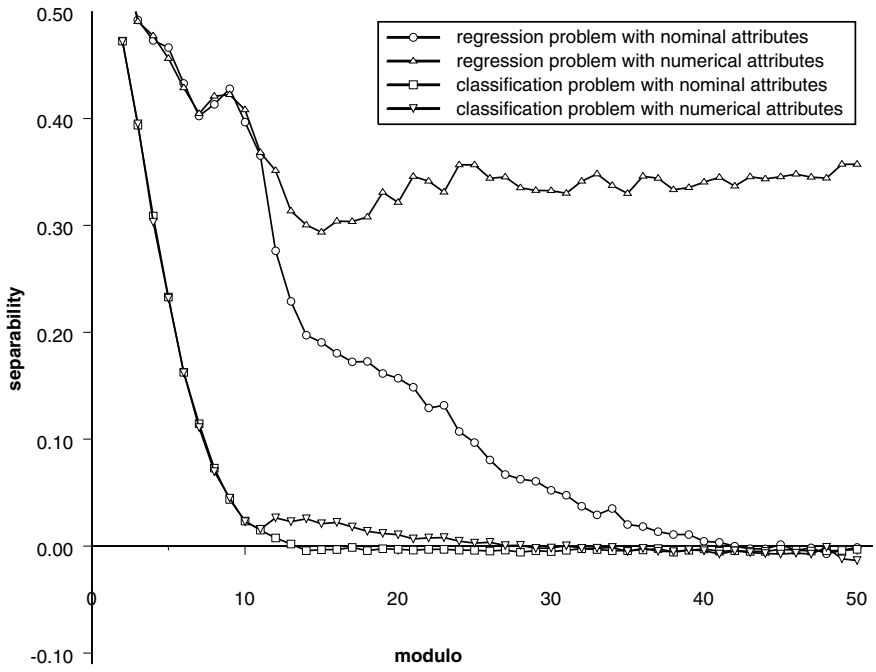

Figure 12. Separability for ReliefF and RReliefF on modulo classification and regression problems with changing modulo.

Modulo problems are examples of difficult problems in the sense that their concept variation is high. Note that impurity-based measures such as Gain ratio, are not capable of separating important from random attributes for any of the above described problems.

4.2.2. MONK's problems. We present results of attribute estimation on well known and popular MONK's problems (Thrun et al., 1991) which consist of three binary classification problems based on common description by six attributes. A_1, A_2, and A_4 can take the values of 1, 2, or 3, A_3 and A_6 can take the values 1 or 2, and A_5 can take one of the values 1, 2, 3, or 4. Altogether there are 432 examples but we randomly generated training subsets of the original size to estimate the attributes in each of the tasks, respectively.

- *Problem* M_1: 124 examples for the problem: $(A_1 = A_2) \vee (A_5 = 1)$
- *Problem* M_2: 169 examples for the problem: exactly two attributes have value 1
- *Problem* M_3: 122 examples with 5% noise (misclassifications) for the problem: $(A_5 = 3 \wedge A_4 = 1) \vee (A_5 \neq 4 \wedge A_2 \neq 3)$.

We generated 10 random samples of specified size for each of the problems and compared estimates of ReliefF and Gain ratio. Table 4 reports results.

Table 4. Estimations of attributes in three MONK's databases for ReliefF and Gain ratio. The results are averages over 10 runs.

	M_1		M_2		M_3	
Attribute	ReliefF	Gain r.	ReliefF	Gain r.	ReliefF	Gain r.
A_1	0.054	0.003	0.042	0.006	−0.013	0.004
A_2	0.056	0.004	0.034	0.006	0.324	0.201
A_3	−0.023	0.003	0.053	0.001	−0.016	0.003
A_4	−0.016	0.007	0.039	0.004	−0.005	0.008
A_5	0.208	0.160	0.029	0.007	0.266	0.183
A_6	−0.020	0.002	0.043	0.001	−0.016	0.003

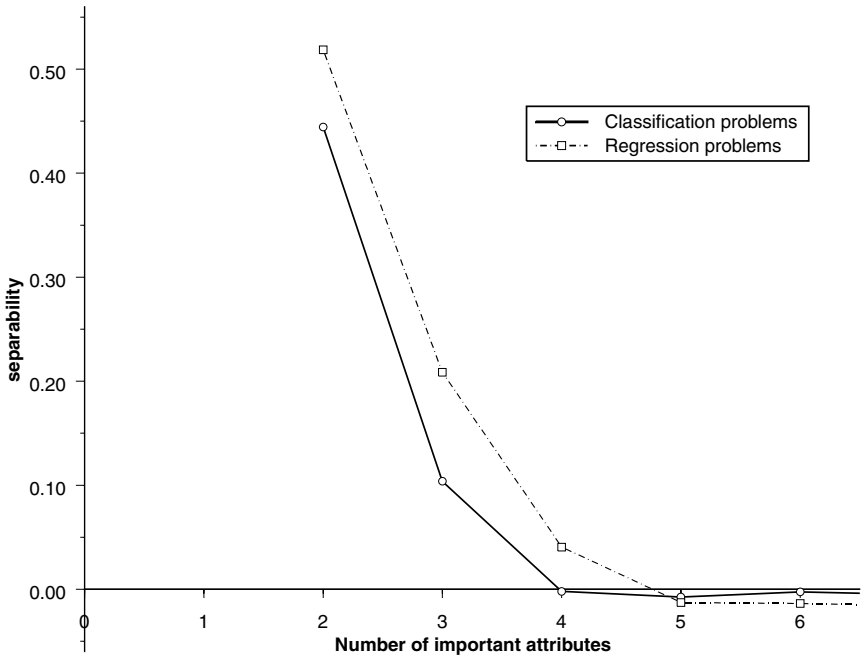

Figure 13. Separability for ReliefF and RReliefF on modulo 5 problems with changing the number of important attributes.

For the first problem we see that ReliefF separates the important attributes (A_1, A_2, and A_5) from unimportant ones while Gain ratio does not recognize A_1 and A_2 as important attributes in this task.

In the second problem where all the attributes are important ReliefF assigns them all positive weights. It favors attributes with less values as they convey more information. Gain ratio does the opposite: it favors attributes with more values.

In the third problem ReliefF and gain ratio behave similarly: they separate important attributes from unimportant ones and rank them equally (A_2, A_5, and A_4).

4.2.3. Linear and nonlinear problems. In typical regression problems linear dependencies are mixed with some nonlinear dependencies. We investigate problems of such type. The problems are described with numerical attributes with values from the [0, 1] interval and 1000 instances. Besides I important attributes there are also 10 random attributes in each problem.

We start with linear dependencies and create problems of the form:

$$\text{LinInc-I:} \quad \tau = \sum_{j=1}^{I} j \cdot A_j \tag{44}$$

The attributes with larger coefficient have stronger influence on the prediction value and should be estimated as more important.

Table 5 reports quality estimates of attributes for RReliefF and MSE. We see that for small differences between importance of attributes both RReliefF and MSE are successful in recognizing this and ranking them correctly. When the differences between the importance of attributes become larger (in LinInc-4 and LinInc-5) it is possible due to random fluctuations in the data one of the random attributes is estimated as better than the least important informative attribute (A_1). This happens in LinInc-4 for RReliefF and in LinInc-5 for MSE. The behavior of s and u are illustrated in figure 14.

Another form of linear dependencies we are going to investigate is

$$\text{LinEq-I:} \quad \tau = \sum_{j=1}^{I} A_j \tag{45}$$

Here all the attributes are given equal importance and we want to see how many important attributes can we afford. Figure 15 shows separability and usability curves for RReliefF and MSE.

Table 5. Estimations of the best random attribute (R_{best}) and all informative attributes in LinInc-I problems for RReliefF (RRF) and MSE. RReliefF assigns higher scores and MSE assigns lower scores to better attributes.

	LinInc-2		LinInc-3		LinInc-4		LinInc-5	
Attr.	RRF	MSE	RRF	MSE	RRF	MSE	RRF	MSE
R_{best}	−0.040	0.424	−0.023	1.098	−0.009	2.421	−0.010	4.361
A_1	0.230	0.345	0.028	1.021	−0.018	2.342	−0.007	4.373
A_2	0.461	0.163	0.154	0.894	0.029	2.093	0.014	4.181
A_3			0.286	0.572	0.110	1.833	0.039	3.777
A_4					0.180	1.508	0.054	3.380
A_5							0.139	2.837

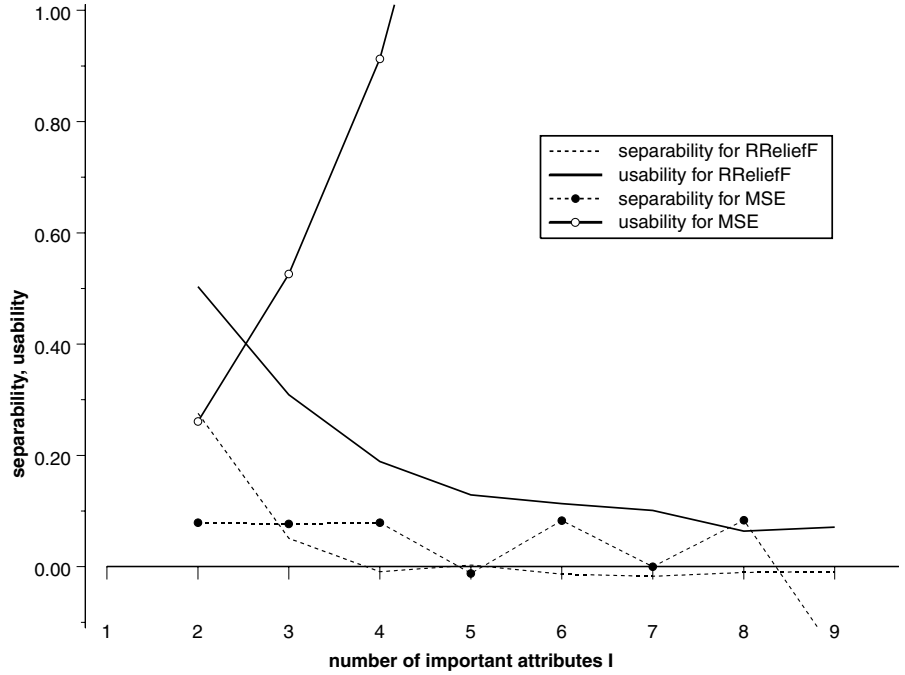

Figure 14. Separability and usability on LinInc concepts for 1000 examples.

We see that separability is decreasing for RReliefF and becomes negative with 10 important attributes. This is not surprising considering properties of Relief: each attribute gets its weight according to the portion of explained function values (see Section 2.7) so by increasing the number of important attributes their weights decrease and approache zero. The same is true for RReliefF's usability which, however, becomes negative much later. MSE estimates each attribute separately and is therefore not susceptible to this kind of defects, however, by increasing the number of important attributes the probability to assign one of them a low score increases and so s curve becomes negative. If we increase the number of examples to 4000, RReliefF's s curve becomes negative at 16 important attributes while the behavior of MSE does not change.

We end our analysis with non-linear dependencies where the prediction is defined as

$$\text{Cosinus-Hills:} \quad \tau = \cos 2\pi \left(A_1^2 + 3A_2^2\right). \tag{46}$$

In this problem there are 1000 examples and the attributes have values in the range $[-1, 1]$. Beside two important attributes there are 10 random attributes. Figure 16 visualizes this domain. We see that prediction values are symmetric along both important attributes with 3 times more hills along A_2.

The quality estimates of RReliefF and MSE are contained in Table 6. We see that RReliefF recognizes A_1 and A_2 as the important attributes and separates them from the random attributes, while MSE is not successful in this task.

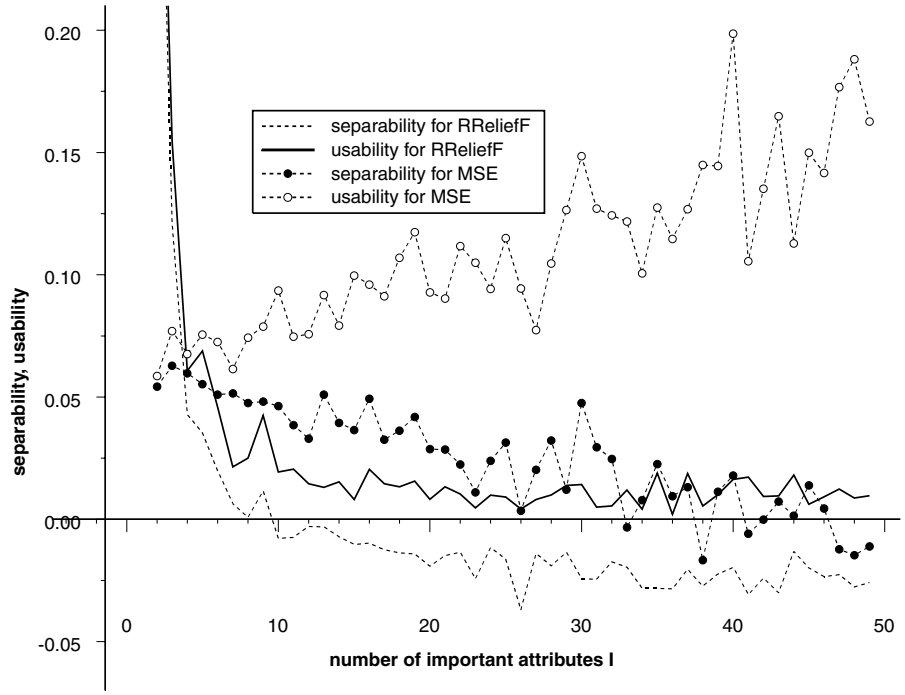

Figure 15. Separability and usability on LinEq concepts for 1000 examples.

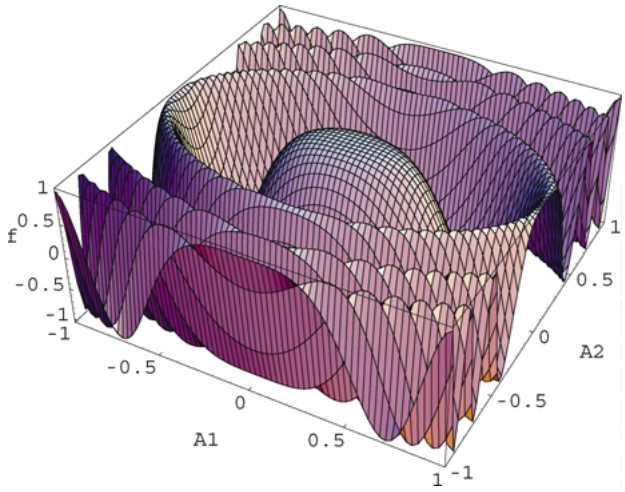

Figure 16. Visualization of Cosinus-Hills problem where: $f = \cos 2\pi(A_1^2 + 3A_2^2)$

Table 6. Estimations of two important attributes and the best of random attributes (R_{best}) in Cosinus-Hills problem for RReliefF (RRF) and MSE. RReliefF assigns higher scores and MSE assigns lower scores to better attributes.

Attr.	RRF	MSE
A_1	0.0578	0.5001
A_2	0.0398	0.5003
R_{best}	0.0194	0.4992

4.3. Number of examples

We investigate how the number of available examples influences quality estimates. For this purpose we generated a number of artificial domains with different number of examples, estimate the quality of the attributes and observe their s and u values. For each data size we repeat the experiment 10 times and test hypotheses that s and u are larger than zero. By setting the significance threshold to 0.01 we get the limiting number of examples needed to separate all important from unimportant attributes and at least one important from unimportant attributes. We are reporting these numbers of examples in the left hand side of Table 7. The datasets and their characteristics are described in Appendix (Table 9). Reported numbers are an indication of the robustness of the estimators.

We can observe that the number of required examples is increasing with complexity of problems within each group (e.g., Modulo-5, Modulo-10, LinInc, ...). Across the problem groups this increase is not correlated with the concept variation V (Eq. (40)) as complexity measure.

It is also interesting to compare Relief algorithms with other estimation measures (Gain ratio and MSE). Apart from the fact that some dependencies cannot be detected by these measures (sign '–' in Table 7), in other (easier) problems we see that Relief algorithms need approximately the same number of examples to detect the best important attribute and slightly more examples to detect the worst important attribute. This can be explained with the intrinsic properties of the estimators: while myopic attribute estimators estimate each attribute independently, Relief algorithms estimate them in the context of other attributes, i.e., all important attributes share the positive estimate and better attributes get more.

4.4. Adding noise by changing predicted value

We check robustness of Relief algorithms concerning by using the same setting as before with the number of examples from the first s column in the left-hand side of Table 7. We added noise to the data sets by changing certain percent of predicted values to a random value. We varied the noise from 0 to 100% in 5% steps.

The right-hand side of Table 7 gives the maximal percentage of corrupted prediction values where s and u were still positive with high probability (>0.99).

We can observe that Relief algorithms are quite concerning the noise in all the problems. When we increase the number of examples 10 times we are able to randomly

Table 7. Results of varying the number of examples and noisy prediction value.

	Number of examples				Noise			
	ReliefF		Gain r.		ReliefF		Gain r.	
Name	*s*	*u*	*s*	*u*	*s*	*u*	*s*	*u*
Bool-Simple	100	50	–	40	10	45	–	20
Modulo-2-2-c	70	60	–	–	30	50	–	–
Modulo-2-3-c	170	110	–	–	5	50	–	–
Modulo-2-4-c	550	400	–	–	20	50	–	–
Modulo-5-2-c	150	130	–	–	5	5	–	–
Modulo-5-3-c	950	800	–	–	15	35	–	–
Modulo-5-4-c	5000	4000	–	–	10	50	–	–
Modulo-10-2-c	400	400	–	–	20	20	–	–
Modulo-10-3-c	4000	3000	–	–	25	55	–	–
Modulo-10-4-c	25000	22000	–	–	10	40	–	–
MONK-1	100	30	–	20	30	70	–	55
MONK-3	250	10	250	20	25	75	5	75
	RReliefF		MSE		RReliefF		MSE	
Modulo-5-2-r	60	50	–	–	25	50	–	–
Modulo-5-3-r	400	350	–	–	20	45	–	–
Modulo-5-4-r	2000	1600	–	–	15	40	–	–
Modulo-10-2-r	90	80	–	–	25	40	–	–
Modulo-10-3-r	800	600	–	–	10	40	–	–
Modulo-10-4-r	7000	4000	–	–	5	40	–	–
Fraction-2	80	70	–	–	20	40	–	–
Fraction-3	650	400	–	–	15	35	–	–
Fraction-4	4000	3000	–	–	10	35	–	–
LinInc-2	60	10	70	20	15	50	10	50
LinInc-3	400	20	150	10	10	65	20	70
LinInc-4	2000	40	450	10	5	80	35	85
LinEq-2	50	40	20	20	30	60	25	45
LinEq-3	180	50	50	20	25	40	30	50
LinEq-4	350	70	70	30	10	40	30	45
Cosinus-Lin	300	40	–	200	15	50	–	40
Cosinus-Hills	550	300	4000	2000	5	20	–	–

set more than 80% of values in every problem. Comparison with myopic measures (Gain ratio and MSE) is included in two right-hand columns. We see that in problems where these measures are successful their noise tolerance is comparable with Relief algorithms.

Table 8. Results of varying the number of iterations and adding random attributes.

	Number of iterations				Random attr.	
	#Ex		10× #Ex		#Ex	
Name	*s*	*u*	*s*	*u*	*s*	*u*
Bool-Simple	83	1	21	1	12	150
Modulo-2-2-c	11	9	1	1	20	45
Modulo-2-3-c	144	15	2	1	14	25
Modulo-2-4-c	151	18	3	1	12	20
Modulo-5-2-c	51	35	1	1	17	30
Modulo-5-3-c	390	37	3	2	20	45
Modulo-5-4-c	3674	579	10	2	13	25
Modulo-10-2-c	31	26	1	1	110	160
Modulo-10-3-c	1210	263	3	1	50	85
Modulo-10-4-c	23760	3227	25	4	11	30
MONK-1	36	1	2	1	NA	NA
MONK-3	163	1	56	1	NA	NA
Modulo-5-2-r	15	12	2	1	19	20
Modulo-5-3-r	57	23	4	1	16	20
Modulo-5-4-r	975	182	17	3	11	19
Modulo-10-2-r	55	37	3	3	14	14
Modulo-10-3-r	316	164	16	9	13	19
Modulo-10-4-r	5513	929	126	17	13	25
Fraction-2	47	32	3	3	18	25
Fraction-3	379	51	29	11	14	20
Fraction-4	1198	78	138	13	13	18
LinInc-2	48	4	6	3	20	>10000
LinInc-3	109	4	52	3	16	>10000
LinInc-4	940	3	456	2	14	>10000
LinEq-2	17	9	3	2	45	140
LinEq-3	96	10	16	7	25	170
LinEq-4	215	13	45	9	16	300
Cosinus-Lin	188	10	16	7	20	3200
Cosinus-Hills	262	51	110	30	12	20

4.5. Number of iterations

We check how many iterations Relief algorithms need to compute by using the same setting as above. The left-hand side of Table 8 gives a minimal number of iterations needed in each problem so that s and u are positive with probability >0.99. The columns labelled '#Ex'

are results for the number of examples fixed to the number of examples from the first *s* column in Table 7, and two columns labelled '10× #Ex' contain results for 10 times that many examples.

There are two interesting observations. The first is that even with the minimal number of examples needed for distinguishing two sets of attributes we do not need to do that many iterations and the actual number of iterations is sometimes quite low. And secondly, if we have an abundance of examples available than we can afford to do only very little iterations as seen from the '10× #Ex' columns in Table 8.

4.6. Adding more random attributes

ReliefF and RReliefF are context sensitive and take all attributes into account when estimating their quality. They are therefore more sensitive to abundance of random attributes than myopic measures, which estimate each attribute separately. We test this sensitivity with similar settings as before (default parameters, the number of examples from the first *s* column in Table 7) and add various numbers of randomly generated attributes.

Right-hand side of Table 8 summarizes results (we consider MONK's data sets fixed and did not include them in this test—label NA). The two columns give the maximal number of random attributes that can be added before *s* and *u* are no more positive with probability >0.99. We see that while we can afford only a moderate number of random attributes to separate all important attributes from unimportant ones, this number is much higher for the best estimated important attribute. Again this confirms the intrinsic behavior of Relief algorithms: they estimate each attribute in the context of other attributes and better attributes get higher score. This can be clearly seen in Lin-Inc and Cosinus-Lin domains where the important attributes are not all equally important.

Since MSE and Gain ratio are less sensitive to this kind of noise (only to the extent that the probability of virtually 'good' random attribute increases with more random attributes) we did not include them into this experiment.

4.7. Number of important attributes

Although the answer to the question, how many important attributes in domain Relief algorithms can handle, is already contained in previous sections we will summarize the results here and give some explanations for this important practical issue.

If important attributes are not equally important then Relief algorithms ideally would share the credit among them in proportion of the explained concept (see Eq. (24)). In practice (with limited number of examples) less important attributes get less than this. The reason for this is that the differences in predicted value caused by less important attribute (see Eq. (12) are overshadowed by larger changes caused by more important attributes. The example of such behavior can be seen in figure 14 and Table 5 where attributes are not equally important. With four attributes in such conditions the separability was no longer positive for RReliefF. Usability on the other hand is robust and remains positive even with several hundred of important attributes.

If all attributes are equally important then we can afford somewhat larger number of attributes still to get positive separability as illustrated in figure 15. Note, however, that the exact number is strongly dependent on sufficient number of examples to cover the problem space adequately. Again the usability stays positive even for several hundreds of important attributes.

In short, Relief algorithms tend to underestimate less important attributes while recognition of more important attributes in not questionable even in difficult conditions.

4.8. Duplicate attributes

We demonstrate behaviors of Relief algorithms in the presence of duplicate attributes. As a baseline we take a simple Boolean XOR problem with 2 important and 10 random attributes with 1000 examples for which ReliefF returns the following estimates:

I_1	I_2	R_{best}
0.445	0.453	−0.032

Now we duplicate the first important attribute I_1 and get:

I_{1,C_1}	I_{1,C_2}	I_2	R_{best}
0.221	0.221	0.768	$-0.\overline{015}$

Both copies of I_1 now share the estimate and I_2 gets some additional credit. Before we explain these estimates we try with 3 copies of the first important attribute:

I_{1,C_1}	I_{1,C_2}	I_{1,C_3}	I_2	R_{best}
0.055	0.055	0.055	0.944	−0.006

and also with 10 copies:

I_{1,C_1}	I_{1,C_2}	...	$I_{1,C_{10}}$	I_2	R_{best}
0.000	0.000	...	0.000	1.000	0.006

The reason for this behavior is that the additional copies of the attribute change the problem space in which the nearest neighbors are searched. Recall that for each instance the algorithm searches nearest neighbors with the same prediction and nearest neighbors with different prediction and then updates the estimates according to the values of diff function (see Eq. (1)). The updates occur only when values of diff is non-zero i.e., when the attribute's values are different. The difference in c times multiplied attribute causes that the instances are now at least on the distance c which causes that this instance will slip out of the near neighborhood of the observed instance. As a consequence the odds of the multiplied attribute to get any update are diminishing and its quality estimate converges towards zero. The reverse is true for non-multiplied attributes: they get more frequently into the near neighborhood and are more frequently updated so the important attributes get higher score as deserved.

If all important attributes are multiplied (not very likely in practical problems) then this effect disappears because the distances are symmetrically stretched. Here is an example with 10 copies of both important attributes:

I_{1,C_1}	...	$I_{1,C_{10}}$	I_{2,C_1}	...	$I_{2,C_{10}}$	R_{best}
0.501	...	0.501	0.499	...	0.4999	−0.037

The conclusion is that Relief algorithms are sensitive to duplicated attributes because they change the problem space. We can also say that Relief algorithms give credit to attributes according to the amount of explained concept and if several attributes have to share the credit it is appropriately smaller for each of them. Of course, myopic estimators such as Gain ratio and MSE, are not sensitive to duplicate attributes.

5. Applications

In previous sections we browsed through theoretical and practical issues on the use of Relief algorithms. In this section we try to give a short overview of their applications. We begin with the feature subset selection and feature weighting which were the initial purposes of Relief algorithm and then continue with the use of ReliefF and RReliefF in tree based models, discretization of attributes, and in inductive logic programming.

5.1. Feature subset selection and feature weighting

Originally, the Relief algorithm was used for feature subset selection (Kira & Rendell, 1992a, 1992b) and it is considered one of the best algorithms for this purpose (Dietterich, 1997). Feature subset selection is a problem of choosing a small set of attributes that ideally is necessary and sufficient to describe the target concept.

To select the set of the most important attributes (Kira & Rendell, 1992a) introduced significance threshold θ. If weight of a given attribute is below θ it is considered unimportant and is excluded from the resulting set. Bounds for θ were proposed i.e., $0 < \theta \leq \frac{1}{\sqrt{\alpha m}}$, where α is the probability of accepting an irrelevant feature as relevant and m is the number of iterations used (see figure 1). The upper bound for θ is very loose and in practice much smaller values can be used.

Feature weighting is an assignment of a weight (importance) to each feature and can be viewed as a generalization of feature subset selection in the sense that it does not assign just binary weights (0-1, include-exclude) to each feature but rather an arbitrary real number can be assigned to it. If Relief algorithms are used in this fashion then we do not need a significance threshold but rather use their weights directly. ReliefF was tested as the feature weighting method in the lazy learning (Wettschereck, Aha, & Mohri, 1997) and was found to be very useful.

5.2. Building tree based models

In learning tree based models (decision or regression trees) in a top down manner we need an attribute estimation procedure in each node of the tree to determine the appropriate

split. Commonly used estimators for this task are impurity based e.g., Gini index (Breiman et al., 1984) or Gain ratio (Quinlan, 1993) in classification and mean squared error (Breiman et al., 1984) in regression. While these estimators are myopic and cannot detect conditional dependencies between attributes they also have inappropriate bias concerning multivalued attributes (Kononenko, 1995). ReliefF was successfully employed in classification (Kononenko & Šimec, 1995; Kononenko et al., 1997) and RReliefF in regression problems (Robnik Šikonja & Kononenko, 1996, 1997).[1] Relief algorithms perform as good as myopic measures if there are no conditional dependencies among the attributes and clearly surpass them if there are strong dependencies. We claim that when faced with an unknown data set it is unreasonable to assume that it contains no strong conditional dependencies and rely only on myopic attribute estimators. Furthermore, using impurity based estimator near the fringe of the decision tree leads to unoptimal splits concerning accuracy and switch to accuracy has been suggested as a remedy (Brodley, 1995, Lubinsky, 1995). It was shown (Robnik Šikonja & Kononenko, 1999) that ReliefF in decision trees as well as RReliefF in regression trees do not need such switch as they contain it implicitly.

We also employ Relief algorithms to guide the constructive induction process during growing of the trees. Only the most promising attributes are selected for construction and various operators were applied on them (conjunction, disjunction, summation, product). The results are good and in some domains the obtained constructs provided additional insight into the domain (Dalaka et al., 2000).

We also observe in our experiences with machine learning applications (medicine, ecology) that trees produced with Relief algorithms are more comprehensible for human experts. Splits selected by them seem to mimic human's partition of the problem which we explain with the interpretation of Relief's weights as the portion of the explained concept (see Section 2.7).

5.3. Discretization of attributes

Discretization divides the values of a numerical attribute into a number of intervals. Each interval can then be treated as one value of the new nominal attribute. The purpose of discretization, which is viewed as an important preprocessing step in machine learning is multiple: some algorithms cannot handle numerical attributes and need discretization, it reduces computational complexity and the splits may convey important information. While Relief algorithms are used for discretization in our tree learning system, the discretization capability was analyzed separately for ReliefF in Robnik (1995). The discretization on artificial datasets showed that conditionally dependent attributes might have important boundaries, which cannot be detected by myopic measures.

5.4. Use in ILP and with association rules

In inductive logic programming a variant of ReliefF algorithm was used to estimate the quality of literals, which are candidates for augmenting the current clause under construction when inducing the first order theories with a learning system (Pompe & Kononenko, 1995). To adjust to specifics of estimating literals diff function has to be changed appropriately.

The modified diff function is asymmetric. The learning system, which uses this version of ReliefF, obtains good results on many learning problems.

The use of ReliefF together with an association rules based classifier (Jovanoski & Lavrač, 1999) is connected also with feature subset selection. The adaptation of the algorithm to association rules changes diff function in a similar way as ReliefF in ILP.

6. Further work

Although there has been a lot of work done with Relief algorithms we think that there are plenty left to do. In this Section we will discuss some ideas concerning parallelization, use as anytime algorithm, applications in time series, in cost-sensitive problems, and in meta learning.

6.1. Parallelization of Relief algorithms

While Relief algorithms are computationally more complex than some other (myopic) attribute estimation measures they also have a possible advantage hat they can be naturally split into several independent tasks which is a prerequisite for successful parallelization of an algorithm. Each iteration of the algorithm is a natural candidate for a separate process, which would turn Relief into the fine-grained parallel algorithm.

6.2. Anytime algorithm and time series

Anytime algorithm is an algorithm, which has a result available at any time and with more time or more data it just improves the result. We think that Relief algorithms can be viewed as anytime algorithms, namely in each iteration they refine the result. As our experiments show (e.g., left-hand side of Table 8) even after only few little iterations Relief algorithms already produce sensible results. The problems we may encounter if we add more data during the processing, are in proper normalizations of updated weights.

Another idea worth considering is to observe changes in quality weights when we add more and more data. In this way it might be possible to detect some important phases in the time series e.g., a change of context.

6.3. Cost-sensitiveness

Typically, machine learning algorithms are not designed to take the non-uniform cost of misclassification into account but there are a number of techniques available which solve this problem. For ReliefF it has been proposed (Kukar et al., 1999) to change the weights of the attributes to reflect the cost sensitive prior probabilities. Another approach would be to change the diff function which could be directly mapped from the cost matrix. However, a normalization of estimates still remains questionable.

6.4. Meta learning

One of the approaches to meta learning is to obtain a number of features concerning the dataset and available learning algorithms and then try to figure out which algorithm is the most appropriate for the problem. Some of these features are obvious e.g., the number and type of attributes, the number of examples and classes, while others are more complex e.g., concept variation (Vilalta, 1999). We think that quality estimates of the attributes produced by Relief algorithms could be a very useful source of meta data. The estimate of the best attribute, difference between the best and the worst estimate, correlation between the Relief's estimate and estimates of myopic functions are examples of features, which could help to appropriately describe the dataset at hand.

7. Conclusions

We have investigated features, parameters and uses of Relief family of algorithms. Relief algorithms are general and successful attribute estimators and are especially good in detecting conditional dependencies. They provide a unified view on attribute estimation in regression and classification and their quality estimates also have a natural interpretation.

We explained how and why they work, presented some of their theoretical and practical properties and analyzed their parameters. To help at their successful application we showed what kind of dependencies they detect, how do they scale up to large number of examples and features, how to sample data for them, how robust are they regarding the noise and how irrelevant and duplicate attributes influence their output. In short, Relief algorithms are robust and noise tolerant. Their somewhat larger computational complexity can, in huge tasks, be alleviated by their nature: as anytime algorithms and intrinsic parallelism.

Appendix

The datasets

Some characteristics of the datasets, we used to demonstrate the features of Relief algorithms, are contained in Table 9. For each domain we present the name, reference to where it was defined, number of classes or label that it is regressional (r), number of important attributes (I), number of random attributes (R), type of the attributes (nominal or numerical) and concept variation V as a measure of concept difficulty (defined with Eq. (40) and measured on 1000 examples—except for MONK-1 and MONK-3, where we used all 432 examples).

Some datasets are not explicitly defined in the text and we provide definitions here.

Cosinus-Lin is a non-linear dependency with cosine stretched over two periods, multiplied by the linear combination of two attributes:

$$\text{Cosinus-Lin:} \quad f = \cos(4\pi A_1) \cdot (-2A_2 + 3A_3) \tag{47}$$

The attributes are numerical with values from 0 to 1.

Table 9. Short description of datasets.

Name	Definition	#class	I	R	Attr. type	V
Bool-Simple	Equation (38)	2	4	10	Nominal	0.283
Modulo-2-2-c	Equation (43)	2	2	10	Nominal	0.278
Modulo-2-3-c	Equation (43)	2	3	10	Nominal	0.379
Modulo-2-4-c	Equation (43)	2	4	10	Nominal	0.444
Modulo-5-2-c	Equation (43)	5	2	10	Nominal	0.651
Modulo-5-3-c	Equation (43)	5	3	10	Nominal	0.748
Modulo-5-4-c	Equation (43)	5	4	10	Nominal	0.790
Modulo-10-2-c	Equation (43)	10	2	10	Numerical	0.808
Modulo-10-3-c	Equation (43)	10	3	10	Numerical	0.891
Modulo-10-4-c	Equation (43)	10	4	10	Numerical	0.902
MONK-1	Section 4.2.2	2	3	3	Nominal	0.213
MONK-3	Section 4.2.2	2	3	3	Nominal	0.145
Modulo-5-2-r	Equation (43)	r	2	10	Nominal	0.658
Modulo-5-3-r	Equation (43)	r	3	10	Nominal	0.755
Modulo-5-4-r	Equation (43)	r	4	10	Nominal	0.803
Modulo-10-2-r	Equation (43)	r	2	10	Numerical	0.801
Modulo-10-3-r	Equation (43)	r	3	10	Numerical	0.885
Modulo-10-4-r	Equation (43)	r	4	10	Numerical	0.892
Fraction-2	Equation (48)	r	2	10	Numerical	0.777
Fraction-3	Equation (48)	r	3	10	Numerical	0.845
Fraction-4	Equation (48)	r	4	10	Numerical	0.869
LinInc-2	Equation (44)	r	2	10	Numerical	0.642
LinInc-3	Equation (44)	r	3	10	Numerical	0.707
LinInc-4	Equation (44)	r	4	10	Numerical	0.709
LinEq-2	Equation (45)	r	2	10	Numerical	0.660
LinEq-3	Equation (45)	r	3	10	Numerical	0.693
LinEq-4	Equation (45)	r	4	10	Numerical	0.726
Cosinus-Lin	Equation (47)	r	3	10	Numerical	0.588
Cosinus-Hills	Equation (46)	r	2	10	Numerical	0.848

Fraction-I is the same as the Modulo-∞-I domain with prediction and attributes normalized to [0, 1]. It can be described as floating point generalization of the parity concepts of order I with predicted values being the fractional part of the sum of I important attributes:

$$\text{Fraction-I:} \quad f = \sum_{j=1}^{I} A_j - \left\lfloor \sum_{j=1}^{I} A_j \right\rfloor \tag{48}$$

Note

1. A learning system that contains ReliefF and RReliefF is freely available upon e-mail request to the authors.

References

Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM, 15:9*, 509–517.

Breiman, L., Friedman, J. H., Olshen, R. A., & Stone, C. J. (1984). *Classification and Regression Trees*. Belmont, California: Wadsworth Inc.

Brodley, C. E. (1995). Automatic selection of split criterion during tree growing based on node location. In *Machine Learning: Proceedings of the Twelfth International Conference (ICML'95)* (pp. 73–80). Morgan Kaufmann.

Cestnik, B., Kononenko, I., & Bratko, I. (1987). ASSISTANT 86: A knowledge-elicitation tool for sophisticated users. In I. Bratko, & N. Lavrač (Eds.), *Progress in Machine Learning, Proceedings of European Working Session on Learning EWSL'87* (pp. 31–36). Wilmslow: Sigma Press.

Dalaka, A., Kompare, B., Robnik-Šikonja, M., & Sgardelis, S. (2000). Modeling the effects of environmental conditions on apparent photosynthesis of Stipa bromoides by machine learning tools. *Ecological Modelling, 129*, 245–257.

Deng, K., & Moore, A. W. (1995). Multiresolution instance-based learning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI'95)* (pp. 1233–1239). Morgan Kaufmann.

Dietterich, T. G. (1997). Machine learning research: Four current directions. *AI Magazine, 18:4*, 97–136.

Domingos, P. (1997). Context-sensitive feature selection for lazy learners. *Artificial Intelligence Review, 11*, 227–253.

Friedman, J. H., Bentley, J. L., & Finkel, R. A. (1975). An algorithm for finding best matches in logarithmic expected time. Technical Report STAN-CS-75-482, Stanford University.

Hong, S. J. (1994). Use of contextual information for feature ranking and discretization. Technical Report RC19664, IBM.

Hong, S. J. (1997). Use of contextual information for feature ranking and discretization. *IEEE Transactions on Knowledge and Data Engineering, 9:5*, 718–730.

Hunt, E. B., Martin, J., & Stone, P. J. (1966). *Experiments in Induction*. New York: Academic Press.

Jovanoski, V., & Lavrač, N. (1999). Feature subset selection in association rules learning systems. In M. Grobelnik, & D. Mladenič (Eds.), *Prooceedings of the Conference Analysis, Warehousing and Mining the Data (AWAMIDA'99)* (pp. 74–77).

Kira, K., & Rendell, L. A. (1992a). The feature selection problem: Traditional methods and new algorithm. In *Proceedings of AAAI'92*.

Kira, K., & Rendell, L. A. (1992b). A practical approach to feature selection. In D. Sleeman, & P. Edwards (Eds.), *Machine Learning: Proceedings of International Conference (ICML'92)* (pp. 249–256). Morgan Kaufmann.

Kononenko, I. (1994). Estimating attributes: Analysis and extensions of Relief. In L. De Raedt, & F. Bergadano (Eds.), *Machine Learning: ECML-94* (pp. 171–182). Springer Verlag.

Kononenko, I. (1995). On biases in estimating multi-valued attributes. In *Proceedings of the International Joint Conference on Aartificial Intelligence (IJCAI'95)* (pp. 1034–1040). Morgan Kaufmann.

Kononenko, I., & Šimec, E. (1995). Induction of decision trees using reliefF. In G. Della Riccia, R. Kruse, & R. Viertl (Eds.), *Mathematical and Statistical Methods in Artificial Intelligence, CISM Courses and Lectures No. 363*. Springer Verlag.

Kononenko, I., Šimec, E., & Robnik-Šikonja, M. (1997). Overcoming the myopia of inductive learning algorithms with RELIEFF. *Applied Intelligence, 7*, 39–55.

Kukar, M., Kononenko, I., Grošelj, C., Kralj, K., & Fettich, J. (1999). Analysing and improving the diagnosis of ischaemic heart disease with machine learning. *Artificial Intelligence in Medicine, 16*, 25–50.

Lubinsky, D. J. (1995). Increasing the performance and consistency of classification trees by using the accuracy criterion at the leaves. In *Machine Learning: Proceedings of the Twelfth International Conference (ICML'95)* (pp. 371–377). Morgan Kaufmann.

Mantaras, R. L. (1989). ID3 revisited: A distance based criterion for attribute selection. In *Proceedings of Int. Symp. Methodologies for Intelligent Systems*. Charlotte, North Carolina, USA.

Moore, A. W., Schneider, J., & Deng, K. (1997). Efficient locally weighted polynomial regression predictions. In D. H. Fisher (Ed.), *Machine Learning: Proceedings of the Fourteenth International Conference (ICML'97)* (pp. 236–244). Morgan Kaufmann.

Murphy, P. M., & Aha, D. W. (1995) UCI repository of machine learning databases. http://www.ics.uci.edu/mlearn/MLRepository.html.

Perèz, E., & Rendell, L. A. (1996). Learning despite concept variation by finding structure in attribute-based data. In *Machine Learning: Proceedings of the Thirteenth International Conference (ICML'96)* (pp. 391–399).

Pompe, U., & Kononenko, I. (1995). Linear space induction in first order logic with ReliefF. In G. Della Riccia, R. Kruse, & R. Viertl (Eds.), *Mathematical and Statistical Methods in Artificial Intelligence*. CISM Courses and Lectures No. 363. Springer Verlag.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1:1*, 81–106.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Rendell, L. A., & Seshu, R. (1990). Learning hard concepts through constructive induction: Framework and rationale. *Computational Intelligence, 6*, 247–270.

Ricci, F., & Avesani, P. (1995). Learning a local similarity metric for case-based reasoning. In *Proceedings of the International Conference on Case-Based Reasoning (ICCBR-95)*. Sesimbra, Portugal.

Robnik, M. (1995). Constructive induction in machine learning. *Electrotehnical Review, 62:1*, 43–49. (in Slovene).

Robnik Šikonja, M. (1998). Speeding up relief algorithm with k-d trees. In *Proceedings of Electrotehnical and Computer Science Conference (ERK'98)* (pp. B:137–140). Portorož, Slovenia.

Robnik Šikonja, M., & Kononenko, I. (1996). Context sensitive attribute estimation in regression. In M. Kubat, & G. Widmer (Eds.), *Proceedings of ICML'96 Workshop on Learning in Context Sensitive Domains* (pp. 43–52). Morgan Kaufmann.

Robnik Šikonja, M., & Kononenko, I. (1997). An adaptation of relief for attribute estimation in regression. In D. H. Fisher (Ed.), *Machine Learning: Proceedings of the Fourteenth International Conference (ICML'97)* (pp. 296–304). Morgan Kaufmann.

Robnik Šikonja, M., & Kononenko, I. (1999). Attribute dependencies, understandability and split selection in tree based models. In I. Bratko, & S. Džeroski (Eds.), *Machine Learning: Proceedings of the Sixteenth International Conference (ICML'99)* (pp. 344–353). Morgan Kaufmann.

Sefgewick, R. (1990). *Algorithms in C*. Addison-Wesley.

Smyth, P., & Goodman, R. M. (1990). Rule induction using information theory. In G. Piatetsky-Shapiro, & W. J. Frawley (Eds.), *Knowledge Discovery in Databases*. MIT Press.

Thrun, S. B., Bala, J. W., Bloedorn, E., Bratko, I., Cestnik, B., Cheng, J., De Jong, K., Džeroski, S., Fahlman, S. E., Fisher, D. H., Hamann, R., Kaufman, K. A., Keller, S. F., Kononenko, I., Kreuziger, J., Michalski, R. S., Mitchell, T., Pachowicz, P. W., Reich, Y., Vafaie, H., Van de Welde, W., Wenzel, W., Wnek, J., & Zhang, J. (1991). The MONK's problems—A performance comparison of different learning algorithms. Technical Report CS-CMU-91-197, Carnegie Mellon University.

Vilalta, R. (1999). Understanding accuracy performance through concept characterization and algorithm analysis. In *Proceedings of the ICML-99 Workshop on Recent Advances in Meta-Learning and Future Work* (pp. 3–9).

Wettschereck, D., Aha, D. W., & Mohri, T. (1997). A review and empirical evaluation of feature weighting methods for a class of lazy learning algorithms. *Artificial Intelligence Review, 11*, 273–314.

Received June 27, 2000
Revised May 24, 2001
Accepted May 31, 2001
Final manuscript February 7, 2002