Comprehensiveness of tree based models: attribute dependencies and split selection

Marko Robnik Šikonja and **Igor Kononenko**
University of Ljubljana,
Faculty of Computer and Information Science,
Tržaška 25, 1001 Ljubljana, Slovenia,
Marko.Robnik@fri.uni-lj.si Igor.Kononenko@fri.uni-lj.si

Abstract

The attributes' interdependencies have strong effect on understandability of tree based models. If strong dependencies between the attributes are not recognized and these attributes are not used as splits near the root of the tree this causes node replications in lower levels of the tree, blurs the description of dependencies and also might cause drop of accuracy. If Relief family of algorithms which is capable of estimating the attributes' dependencies is used for split selectors we can partly overcome the problem. We describe ReliefF and RReliefF algorithms and their use in connection with tree based models. Some theoretical properties of Relief's estimate and a recent empirical study suggest that accuracy optimization near the fringe of the tree is not necessary with these algorithms.

1 Introduction

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

Most of the heuristic measures for estimating the quality of the attributes assume the independence of the attributes and indeed the experimental results (Kononenko et al., 1997) indicate that this myopia has no or only marginal effect on the majority of real world problems. However when we face a new problem it is unreasonable to try only these restricted measures unless we know in advance that there are no strong conditional dependencies between the attributes. Relief algorithms (Relief, ReliefF and RReliefF) do not make this assumption and are non-myopic in this sense. They are aware of the contextual information, efficient and can correctly estimate the quality of attributes in problems with strong dependencies between the attributes as well. While Relief algorithms have commonly been viewed as a feature selection methods that are applied in a preprocessing step before the model is learned (Kira and Rendell, 1992) and are one of the most successful preprocessing algorithms to date (Wettschereck et al., 1997; 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 et al., 1997), to select splits and guide the constructive induction in learning of the regression trees (Robnik Šikonja and Kononenko, 1997), as attribute weighting method in instance based learning (Wettschereck et al., 1997), and also in inductive logic programming (Pompe and Kononenko, 1995).

Decision and regression trees are popular description languages for representing knowledge in machine learning. While constructing a tree, a learning algorithm at each interior node selects a splitting rule (a feature) which divides the problem space into two separate subspaces. To select the 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 splitting rules seems to be of principal importance.

The focus of this paper is on the effect the attributes' interdependencies have on the understandability of decision and regression trees. Impurity based measures try to select as pure splits as possible i.e., separate instances with different (class) values into different subtrees. Strong dependencies between the attributes are not properly detected by these measures, therefore dependent attributes are not selected as splits near the root of the tree which would guarantee compact representation of dependencies. Instead they are selected in subsequent levels of the tree, mostly near the fringe (if at all) which causes node replication, blurs the description of dependencies and also might cause drop of accuracy.

One possible solution to this is to use multivariate splits (Brodley and Utgoff, 1995) but this can be computationally expensive. Another solution to this problem is to use non-myopic attribute estimator as split selection heuristics. This does not exclude the use of multivariate split on contrary the use of non-myopic attribute estimator to guide the constructive induction has turned out to be a successful technique to reduce its computational requirements (Robnik Šikonja, 1997). ReliefF was used for that purpose within a decision trees learning system (Kononenko et al., 1997) and achieved significantly better results in terms of accuracy and complexity of decision trees in domains with highly dependent features compared to impurity based measures. It achieved the same performance in domains with independent features. RReliefF was used as split selector in regression tree learning system CORE (Robnik Šikonja, 1997) and also achieved better results in terms of error and complexity of the tree compared to the mean squared error (Robnik Šikonja and Kononenko, 1997). The tree structure chosen by Relief algorithms is closer to human partition of the problem and improves on the understandability and extracted knowledge (Kononenko et al., 1997; Dalaka et al., 1999).

Usually if we want to improve the accuracy of tree based model we have to take into consideration that test nodes closer to the leaves of the tree should be chosen to maximize accuracy on the training set. This effect was investigated by (Brodley, 1995; Lubinsky, 1995) and showed empirically that this can improve classification accuracy, however the authors were working with myopic split selection measures and could not detect and address the effect the attribute dependencies have on the structure of the tree. Recently (Robnik Šikonja and Kononenko, 1999), it was shown that Relief algorithms already implicitly contain the switch to accuracy optimization and do not need any other mechanism in tree based models to perform it.

In this paper we are describing ReliefF and RReliefF and some of their properties related to split selection in tree based models. In the next Section we describe the problem of attribute dependencies, explain how Relief algorithms detect these dependencies and establish relation between Relief's estimations and myopic measures. Section 3 describes two algorithms to be used in practice: ReliefF in classification and RReliefF in regression problems. Section 4 describes some issues concerning the use of these two algorithms in tree based models especially their property to perform implicit switch to accuracy optimization in lower levels of the tree. The last Section summarizes the paper.

2 Attribute dependencies

When we are talking about attribute dependencies in this paper we have in mind the conditional dependencies (conditioned upon class values). Let us illustrate by three examples defined in Boolean space.

1. Let class value be defined as XOR (parity) of two attributes $C = X_1 \oplus X_2$. Knowing the values of the class and one of the attribute we have all the information about the other attribute, while the knowledge of the class value alone does not contain any information about the attributes' values. In this case we can say that X_1 and X_2 are (maximally) conditionally dependent.

2. Let class value be defined as conjunction of two attributes $C = X_1 \wedge X_2$. The knowledge about the class and one of the attributes is sometimes enough to determine the value of the other attribute (e.g., if $C = 0$ and $X_1 = 1$, the value of X_2 must be 0) but not always (e.g., if $C = 0$ and $X_1 = 0$, the value of X_2 is still undetermined), while the class value alone contains even less information (e.g., if $C = 0$ we cannot determine the values of X_1 and X_2). We can say that X_1 and X_2 are somehow conditionally dependent.

3. Let class be an arbitrary non-constant Boolean function, both values of the attributes are equiprobable ($P(X_1 = 1) = P(X_2 = 1) = 0.5$) and attributes are defined as $P(X_1 = 1|C = 1) = p$, $P(X_1 = 1|C = 0) = 1 - p$ and $P(X_2 = 1|C =$

$1) = q$, $P(X_2 = 1|C = 0) = 1 - q$. Each attribute contains some knowledge about the class value, but they contain this knowledge independently from each other. The knowledge of the class value conveys certain information about values of each of attribute, but this information does not increase even if we know the value of one of the attributes. So in this case we can say that X_1 and X_2 are conditionally independent. Another example of conditional independence are random attributes.

Relief algorithms estimate the quality of attributes in problems with strong dependencies between the attributes. How do they do that? The basic Relief algorithm (Kira and Rendell, 1992) is presented on Figure 1 and shortly described below.

The key idea is to estimate the quality of attributes according to how well their values distinguish between the instances that are near to each other. For that purpose, given a randomly selected instance R (line 3), Relief searches for its two nearest neighbors: one from the same class, called *nearest hit* H, and the other from a different class, called *nearest miss* M (line 4). It updates the quality estimation $W[A]$ for all the attributes A depending on their values for R, M, and H (lines 5 and 6). If the values of attribute A at instances R and H are different then attribute A separates two instances of the same class which is not desirable so we add negative update to the quality estimation $W[A]$. If the values of attribute A at instances R and M are different then attribute A separates two instances with different class values which is desirable so we add positive update to the quality estimation $W[A]$. The process is repeated for m times, where m is a user-defined parameter. Function $diff(Attribute, Instance1, Instance2)$ calculates the difference between the values of Attribute for two instances. For discrete attributes it was originally defined as:

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

and for continuous attributes as:

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

The function $diff$ is used also for calculating the distance between instances to find the nearest neighbors. The total distance is the sum of distances over all attributes.

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

$$W[A] = P(\text{diff. value of A}|\text{nearest inst. from diff. class}) - P(\text{diff. value of A}|\text{nearest inst. from same class}) \quad (3)$$

The estimations of Relief algorithms are strongly related to impurity functions (Kononenko, 1994) and shown below. When the number of nearest neighbors increases i.e., when we eliminate the requirement that the selected instance is the nearest, the Equation (3) becomes

$$\begin{aligned} W'[A] &= P(\text{different value of A}|\text{different class}) \\ &- P(\text{different value of A}|\text{same class}) \end{aligned} \quad (4)$$

If we rewrite

$$P_{eqval} = P(\text{equal value of A})$$

$$P_{samecl} = P(\text{same class})$$

$$P_{samecl|eqval} = P(\text{same class}|\text{equal value of A})$$

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:

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

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

Using the above equalities we obtain:

$$W'[A] = \frac{P_{eqval} \times Ginigain'(A)}{P_{samecl}(1 - P_{samecl})} \quad (5)$$

where

$$\begin{aligned} Ginigain'(A) &= \sum_V \left(\frac{P(V)^2}{\sum_V P(V)^2} \times \sum_C P(C|V)^2 \right) \\ &- \sum_C P(C)^2 \end{aligned} \quad (6)$$

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;
4. find nearest hit H and nearest miss M;
5. **for** A := 1 **to** #all_attributes **do**
6. $W[A] := W[A] - diff(\text{A,R,H})/m + diff(A,R,M)/m$;
7. **end**;

Figure 1: Pseudo code of the basic Relief algorithm

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 (5) (which we call myopic ReliefF), shows strong correlation of Relief's weights with the Gini-index gain. The probability $\sum_V P(V)^2$ that two instances have the same value of attribute A in Eq. (5) 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 et al., 1987)). Equation (5) 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 Equation (5) 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 Equation (5) is used as (myopic) attribute estimator what was empirically confirmed in (Kononenko, 1995).

The above derivation eliminated the "nearest instance" condition from the probabilities. If we put it back we can interpret Relief's estimates as the average over local estimates in smaller parts of the instance space. This enables Relief to take into account the context of other attributes, i.e. the conditional dependencies between attributes given the class value which can be detected in the context of locality. From the global point of view, these dependencies are hidden due to the effect of averaging over all training instances, and exactly this makes impurity functions myopic. Impurity functions use correlation between the attribute and the class disregarding the context of other attributes. This is the same as using the global point of view and disregarding the local peculiarities. We illustrate this in Figure 2 which shows dependency of ReliefF's estimate to number of nearest neighbors taken into account. The estimates are for a parity domain with two informative and 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 increasing number of nearest neighbors and how it eventually becomes indistinguishable from random attribute (Note that it is indistinguishable for all myopic estimators).

3 Used in practice: ReliefF and RReliefF

The original Relief can deal with discrete and continuous attributes. However, it can not deal with incomplete data and is limited to two-class problems. Its extension which solves these and some other problems is called ReliefF (Kononenko, 1994). One of the important and interesting differences is that ReliefF uses several nearest neighbors in the approximation of probabilities. The adaptation of ReliefF for regressional problems (con-

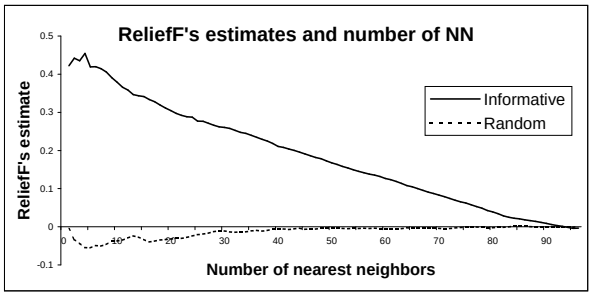

Figure 2: ReliefF's estimates of informative attribute are deteriorating with increasing number of nearest neighbors in parity domain.

tinuous class) is called RReliefF (Robnik Šikonja and Kononenko, 1997).

3.1 ReliefF

The ReliefF (Relief-F) algorithm (Kononenko, 1994) (see Figure 3) 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 (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 the attributes A depending on their values for R, hits H_j 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 $[0, 1]$ range and also symmetric we have to ensure that misses' probability weights sum to 1. As the class of the hits is missing in the summation we have to divide each probability weight with factor of $1 - P(class(R))$ (which represents the sum of probabilities for the misses' classes). The process is repeated for m times.

The selection of k hits and misses is the difference to Relief which ensures greater robustness of the algorithm concerning noise. The k is user-defined parameter which controls the locality of the estimates. For most purposes it can be safely set to 10.

To deal with incomplete data we change the $diff$ function. The missing values of attributes are treated probabilisticaly. We calculate the probability that two given instances have different values for the given attribute conditioned over class value. If one instance (e.g. I1) has unknown value:

$$diff(A, I1, I2) = 1 - P(value(A, I2)|class(I1)) \quad (7)$$

If both instances have unknown value:

$$diff(A, I1, I2) = 1 - \sum_{V}^{\#values(A)} \left(P(V|class(I1)) \times P(V|class(I2))\right) \quad (8)$$

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

3.2 RReliefF

Before the description of algorithm RReliefF (Regressional ReliefF) (Robnik Šikonja and Kononenko, 1997) we will first look at some theory which explains what Relief algorithm actually computes.

Taking into account Equation(3) we can see that the positive updates of the weights (line 6 in Figure 1 and line 9 in Figure 3) are actually forming estimate of probability that the attribute discriminates between the instances of different class, while the negative updates (line 6 in Figure 1 and line 8 in Figure 3) are forming the probability that the attribute separates the instances with the same class.

In regression problems the predicted value $\tau(\cdot)$ is continuous, therefore the (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 modeled with the relative distance between the predicted (class) values of the two instances.

Still, to estimate W[A] in Equation (3), the information about the sign of each contributed term is missing (where do hits end and misses start). In the following derivation the Equation (3) is reformulated, so that it can be directly evaluated using the probability that predicted values of two instances are different. If we rewrite

$$P_{diffA} = P(\text{different value of A}|\text{nearest instances}) \quad (9)$$

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;
4. find k nearest hits H_j;
5. **for** each class $C \neq class(R)$ **do**
6. from class C find k nearest misses $M_j(C)$;
7. **for** A := 1 **to** #all_attributes **do**
8. $W[A] := W[A] - \sum_{j=1}^{k} diff(A,R,H_j)/(m \cdot k) +$
9. $\sum_{C \neq class(R)} [\frac{P(C)}{1-P(class(R))} \sum_{j=1}^{k} diff(A,R,M_j(C))]/(m \cdot k)$;
10. **end**;

Figure 3: Pseudo code of ReliefF algorithm

$$P_{diffC} = P(\text{different prediction}|\text{nearest instances}) \quad (10)$$

$$P_{diffC|diffA} = P(\text{diff. prediction}|\text{diff. value of A and nearest inst.}) \quad (11)$$

we obtain from (3) using Bayes rule:

$$W[A] = \frac{P_{diffC|diffA}P_{diffA}}{P_{diffC}} - \frac{(1-P_{diffC|diffA})P_{diffA}}{1-P_{diffC}} \quad (12)$$

Therefore, we can estimate $W[A]$ by approximating terms defined by Equations 9, 10 and 11. This can be done by the algorithm on Figure 4.

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 (class) value $\tau(\cdot)$ (line 6), different attribute (line 8), and different prediction & different attribute (line 9 and 10) are collected in N_{dC}, $N_{dA}[A]$, and $N_{dC\&dA}[A]$, respectively. The final estimation of each attribute $W[A]$ (Equation (12)) is computed in lines 14 and 15.

Term $d(i,j)$ in Figure 4 (lines 6, 8 and 10) is used to take 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 decreased the influence of instance I_j with the distance from the given instance R_i:

$$d(i,j) = \frac{d_1(i,j)}{\sum_{l=1}^{k} d_1(i,l)} \quad \text{and} \quad (13)$$

$$d_1(i,j) = e^{-\left(\frac{rank(R_i,I_j)}{\sigma}\right)^2} \quad (14)$$

where $rank(R_i, I_j)$ is the rank of instance I_j in a sequence of instances ordered by the distance from R_i and σ is a user defined parameter controlling the distance. Since we want to stick to probability interpretation of results we normalize the contribution of each of the k nearest instances by dividing it with the sum of all k contributions. The reason for using ranks instead of the actual distances is that the 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. Default values used in our system are $k = 70$ and $\sigma = 20$.

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

3.3 Computational complexity

Although ReliefF (Figure 3) and RReliefF (Figure 4) look more complicated their asymptotical complexity is the same as that of original Relief i.e., $O(m \cdot N \cdot A)$, where N is the number of training instances, A the

Algorithm RReliefF
Input: for each training instance a vector of attribute values **x** and the predicted value $\tau(\mathbf{x})$
Output: the 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. $N_{dC} := N_{dC} + diff(\tau(\cdot), R_i, I_j) \cdot d(i,j)$;
7. **for** A := 1 **to** #all_attributes **do begin**
8. $N_{dA}[A] := N_{dA}[A] + diff(A, R_i, I_j) \cdot d(i,j)$;
9. $N_{dC\&dA}[A] := N_{dC\&dA}[A] + diff(\tau(\cdot), R_i, I_j) \cdot$
10. $diff(A, R_i, I_j) \cdot d(i,j)$;
11. **end**;
12. **end**;
13. **end**;
14. **for** A := 1 **to** #all_attributes **do**
15. $W[A] := N_{dC\&dA}[A]/N_{dC}$ - $(N_{dA}[A] - N_{dC\&dA}[A])/(m - N_{dC})$;

Figure 4: Pseudo code of RReliefF algorithm

number of attributes and m is the sample size. The most complex operation within the main **for** loop is the selection of **k** nearest instances. For it we have to compute the distances from all the instances to R, which can be done in $O(N \times A)$ 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 $O(N \times A)$.

If we use k-d tree to implement the search for nearest instances we can reduce the complexity of all three algorithms to $O(A \cdot N \cdot logN)$ (Robnik Šikonja, 1998) which is actually in the same order of complexity as multikey sort algorithms. The use of k-d trees within Relief algorithms becomes inefficient with increasing number of attributes (more than e.g., 10)(Robnik Šikonja, 1998).

4 Split selection and accuracy optimization in tree based models

In the process of building a (decision, regression) tree using ReliefF or RReliefF as the attribute selection heuristics we use all important dependent attributes in the upper part of the tree and come to a point where there are no more dependencies, i.e., the local view becomes the global view. This is the point when there are no more hidden dependencies and we should start optimizing the tree for accuracy. This point can be detected by measuring the correlation between the global estimate (Equation (5)) and the average over local estimates (Relief's weights, Equation (3)) for various attributes (Robnik Šikonja and Kononenko, 1999).

The (linear) correlation coefficient $\rho_{x,y}$ between two vectors of observations x and y is defined as

$$\rho_{x,y} = \frac{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu_x)(y_i - \mu_y)}{\sigma_x \sigma_y} \qquad (15)$$

where $-1 \leq \rho_{x,y} \leq 1$. This is a well known statistical measure which measures the degree of linear relationship between the x_i and y_i values. In our case x_i is the ReliefF's estimate of i-th attribute and y_i is the estimate of the i-th attribute by another estimator.

Accuracy as attribute estimator has some major drawbacks (Brodley, 1995) in terms of the structure of the tree but once we have extracted the dependencies it can be very useful if we want to optimize the tree for accuracy. If we classify with the majority class the ac-

curacy as the attribute estimator it is defined as

$$Acc = \sum_{V} P(V) \max_{C} P(C|V) \qquad (16)$$

If we want to compute the accuracy for a binary split of a multivalued attribute we group the attribute's values into two groups (left and right) and use the group probabilities in Equation 16.

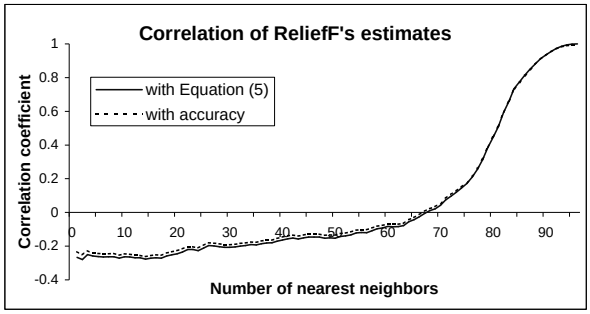

Figure 5: Correlation between ReliefF and Myopic Relief and ReliefF and accuracy in parity domain.

Figure 5 shows the correlation coefficient between the ReliefF's estimate and Equation (5)(solid line) and ReliefF's estimate and accuracy (dotted line) varying over the number of nearest neighbors taken into account for ReliefF's estimates. As before the estimates are for a parity domain with two informative and 10 random attributes and 200 examples. We see that the correlation coefficients are very similar and as it is indicative in other domains as well this leads to further simplification of our heuristic for selecting split criteria: by monitoring the correlation coefficient between ReliefF's estimate and accuracy we determine when there are no more dependencies in a local subspace (tree node). When the correlation coefficient is high enough (e.g., ≥ 0.8) we switch to accuracy optimization and select Equation (16) as the split selector measure in the subtree of a current node.

Empirical evaluation of the above described switching criteria on decision and regression trees (Robnik Šikonja and Kononenko, 1999) is summarized below. Some recent studies e.g., (Dietterich, 1998), have pointed out that the assumption of independence of training sets is violated in k-fold cross-validated paired Student's t-test, however (Dietterich, 1998) observes that this test is very powerful and has only somewhat elevated probability of Type I error. As most of the datasets we are using contain relatively small number of instances and 10-fold cross-validation is economical with the data we have selected 10-fold cross-validated paired Student's t-test for computing the significance of the differences. The significance of the differences were computed at 0.05 level. As we are comparing over several datasets we are also using Bonferroni adjustment (see e.g., (Salzberg, 1997)).

4.1 Decision trees

We have developed a learning system which builds binary decision trees by splitting training examples based on the values of attributes. The attribute in each node is selected according to the estimates of the attributes' quality. These estimates are computed on the subset of the examples that reach current node.

We run our system with three different estimators: ReliefF alone, accuracy alone, and combined ReliefF and accuracy (depending on the threshold for the correlation coefficient, which was set to 0.8). All other parameters for growing and pruning the tree were the same and set to their default values. We used two stopping criteria, namely the minimal number of cases in the leaf (=5) or the minimal purity of a leaf (97% of cases are from the same class) and postpruned the tree with the m-estimate ($m = 2$).

There are 7 datasets where ReliefF is significantly more accurate than accuracy (KRK1, KRK2, Parity-3, Primary, Rheumatic, Soya, Tic-tac-toe). The combined approach was significantly better than accuracy alone in 6 datasets (KRK2, Parity-3, Primary, Rheumatic, Soya, Tic-tac-toe). There were no significant differences in accuracy between ReliefF and combined approach. Accuracy was never significantly better than the other two methods. The significance of the differences are summarized in Table 1.

The combined approach produced the trees which are on average slightly larger than those produced solely by ReliefF and slightly smaller than those produced solely by the accuracy.

Visual inspection of the trees confirmed that mostly the splitting criteria was switched when there were no more dependencies between the attributes. In average 37% of splits at the bottom of the trees were selected by the accuracy.

We also experimented with the correlation threshold and it turned out that the switching method is quite robust against this parameter. By setting this parameter in the interval from 0.7 to 0.95 we get very similar results. Above 0.95 there are very few nodes selected by accu-

Table 1: Number of significant differences in favor of the estimators in the first column over the estimators in the first row.

estimator	ReliefF	accuracy	combination
ReliefF	-	7	0
accuracy	0	-	0
combination	0	6	-

racy and below 0.7 there are majority of nodes selected by the accuracy.

What can we conclude from the above experiments? Although in average 37% of splits at the bottom of the tree were selected by the accuracy this switch did not not help ReliefF in improving the score in accuracy. Comparing that with significant improvement which was observed when switching from gain ratio to accuracy (Brodley, 1995; Lubinsky, 1995) it seems that ReliefF does not need such switch as its self-normalization factors (see Equation (5)) and locality of estimates already take that into account.

To confirm this hypothesis we built this switching criterion into our regression tree learning system CORE (Robnik Šikonja, 1997) and test it with two different models in the leaves of the tree (mean prediction value and linear models). This extended definition of the accuracy puts Relief's abilities to a much harder test than in decision trees where the model in the leaf is a majority class.

4.2 Regression trees

Similarly to decision trees we have built the correlation based switching criterion into our regression tree learning system CORE[1] (Robnik Šikonja, 1997) which uses regressional version of ReliefF called RReliefF (Robnik Šikonja and Kononenko, 1997) or Mean Squared Error (MSE) as the attribute estimator. In the leaves of the tree we used two different predictors to forecast the function value: mean prediction value of the training instances in the leaf or linear model which fits the instances in the leaf computed with singular value decomposition method (Press et al., 1988). MSE estimate of the attribute A is minimum over all possible splits of attribute A of the weighted mean squared error of the predictor ϕ:

$$MSE(\phi, A) = \min_{split\ of\ A} p_L \cdot R_{t_L}(\phi) + p_R \cdot R_{t_R}(\phi), \tag{17}$$

where t_L and t_R are the subsets of cases that go left and right, respectively, by the split based on A, and p_L and p_R are the proportions of cases that go left and right. $R_t(\phi)$ is the mean squared error of the predictor ϕ compared to real prediction values c_i of instances x_i in the subset t :

$$R_t(\phi) = \frac{1}{N_t} \sum_{i=1}^{N_t} (c_i - \phi(x_i))^2. \tag{18}$$

$\phi(x_i)$ is the value predicted by ϕ.

MSE is used as attribute estimator and as the success criterion (counterpart of accuracy in decision trees). Previous study (Robnik Šikonja and Kononenko, 1997) already compared RReliefF and MSE on regression trees and it turned out that on the problems which contain dependent attributes RReliefF produced smaller trees with significantly lower error, while on problems without strong dependencies RReliefF and MSE produce comparable trees.

We run our system with three different estimators: ReliefF alone, accuracy alone, and combined ReliefF and accuracy (with the correlation coefficient set to 0.8). All other parameters for growing and pruning the tree were the same and set to their default values. We used the minimal number of instances in the leaf (=5) as the stopping criterion and postpruned the tree with the m-estimate ($m = 2$).

We ran our system on the artificial data sets and on data sets with continuous prediction value from UCI (Murphy and Aha, 1995). Altogether there were 15 datasets.

For each experiment we collected the results as the average of 10 fold cross-validation. We compared the relative mean squared error:

$$RE_t(\phi) = \frac{R_t(\phi)}{R_t(\mu)} \tag{19}$$

ϕ is the evaluated predictor (regression tree) and μ is a predictor which always returns the mean of the training instances. Sensible predictors have $RE(\phi) < 1$. The significance of the differences were computed by a two-sided paired Student's t-test at a 0.05 level using the Bonferroni adjustment and are presented in Table 2.

[1]Both learning systems which contain a number of estimators, constructive induction and various models in the leaves of the tree are freely available for non-commercial purposes upon e-mail request.

With linear models in the leaves there are 5 datasets where RReliefF has significantly lower error than MSE (Abalone, Fraction-3, Parity-2, Parity-3, and Wisconsin). The combined approach is significantly better than MSE in 4 datasets (Abalone, Fraction-3, Parity-2, and Parity-3). There are no significant differences in accuracy between RReliefF and combined approach. MSE never produces significantly lower error than the other two methods. With mean prediction value in the leaves we obtained similar results. Both RReliefF and the combined approach have significantly lower error than MSE in 4 datasets (Fraction-2, Fraction-3, Modulo8-2, and Parity-3)

Table 2: Number of significant differences in RE in favor of the estimators in the first column over the estimators in the first row.

estimator	RReliefF	MSE	combination
	linear models in leaves		
RReliefF	-	5	0
MSE	0	-	0
combination	0	4	-
	mean prediction value in leaves		
RReliefF	-	4	0
MSE	0	-	0
combination	0	4	-

When the switching criterion was used on average 25% and 36% of the splits at the bottom of the trees were selected by the MSE with linear models and mean prediction value, respectively.

We consider the above results as a confirmation of our hypothesis that (R)ReliefF does not need such a switch as it is already self-normalizing.

5 Conclusions

When we face a new (possibly difficult) problem we do not know much about it is unreasonable to assume that attributes are conditionally independent and use only any of restricted myopic measures to estimate their quality. Relief algorithms are efficient non-myopic measures successfully used in number of machine learning task: feature selection (Kira and Rendell, 1992), to select splits in the building phase of decision tree learning (Kononenko et al., 1997), to select splits and guide the constructive induction in learning of the regression trees (Robnik Šikonja and Kononenko, 1997), as attribute weighting method in instance based learning (Wettschereck et al., 1997), and also in inductive logic programming (Pompe and Kononenko, 1995).

In tree based models ReliefF and RReliefF recognize dependencies between the attributes and use them near the root of the tree which guarantees their compact representation. A nice theoretical property of Relief estimates allows ReliefF and RReliefF to avoid explicit switch to accuracy (error) optimization when there are no more attribute dependencies but this switch is performed implicitly within estimations.

References

Breiman, L., Friedman, J. H., Olshen, R. A., and Stone, C. J. (1984). *Classification and regression trees*. Wadsworth Inc., Belmont, California.

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)*. Morgan Kaufmann.

Brodley, C. E. and Utgoff, P. E. (1995). Multivariate decision trees. *Machine Learning*, 19(1):45–77.

Cestnik, B., Kononenko, I., and Bratko, I. (1987). Assistant 86: A knowledge-elicitation tool for sophisticated users. In Bratko, I. and Lavrač, N., editors, *Progress in Machine Learning, Proceedings of EWSL 87*, Wilmslow. Sigma Press.

Dalaka, A., Kompare, B., Robnik-Šikonja, M., and Sgardelis, S. (1999). Modeling the effects of environmental conditions on apparent photosynthesis of stipa bromoides by machine learning tools. (to appear).

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

Dietterich, T. G. (1998). Approximate statistical tests for comparing supervised classification learning algorithms. *Neural Computation*, 10(7):1895–1924.

Kira, K. and Rendell, L. A. (1992). A practical approach to feature selection. In D.Sleeman

and P.Edwards, editors, *Machine Learning: Proceedings of International Conference (ICML'92)*, pages 249–256. Morgan Kaufmann.

Kononenko, I. (1994). Estimating attributes: analysis and extensions of Relief. In De Raedt, L. and Bergadano, F., editors, *Machine Learning: ECML-94*, pages 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)*, pages 1034–1040. Morgan Kaufmann.

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

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)*. Morgan Kaufmann.

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

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

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

Press, W. H., Teukolsky, S. A., Vetterling, W. T., and Flannery, B. P. (1988). *Numerical recipes in C*. Cambridge University Press.

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

Robnik Šikonja, M. (1997). CORE - a system that predicts continuous variables. In *Proceedings of Electrotehnical and Computer Science Conference*, pages B145–148, Portorož, Slovenia.

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

Robnik Šikonja, M. and Kononenko, I. (1997). An adaptation of Relief for attribute estimation in regression. In Fisher, D., editor, *Machine Learning: Proceedings of the Fourteenth International Conference (ICML'97)*, pages 296–304. Morgan Kaufmann Publishers.

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

Robnik Šikonja, M. (1997). The developement of the heuristics for guiding the learning of the regression trees. University of Ljubljana, Faculty of Computer and Information Science, MSc thesis. (in Slovene).

Salzberg, S. L. (1997). On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Mining and Knowledge Discovery*, 1:317–328.

Wettschereck, D., Aha, D. W., and 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.