Evaluating the Quality of Attributes

Igor Kononenko

3rd May 2005

One of crucial tasks in machine learning is the evaluation of the quality of attributes. For that purpose a number of measures have been developed that estimate the usefulness of the attribute for predicting the target variable. We describe separately measures for classification (which are appropriate also for relational problems) and for regression. Most of the measures estimate the quality of one attribute independently of the context of other attributes. However, algorithm ReliefF and its regressional version RReliefF take into account also the context of other attributes and are therefore appropriate for problems with strong dependencies between attributes.

Attributional representation of instances is defined with:

- a set of attributes A = $\{A_i, i = 0...a\}$, where a is the number of attributes;
- each discrete attribute A_i has a set of m_i possible values $\mathcal{V}_i = \{V_1, ..., V_{m_i}\}$;
- each continuous attribute A_i has an interval of possible values $\mathcal{V}_i = [Min_i, Max_i]$;
- the dependent variable is given with the target attribute A_0: for classification problem it is discrete (class), for regressional problem it is continuous;
- one training instance is a vector of values of all attributes, including the target attribute: $t_l = \langle v^{(0,l)}, v^{(1,l)}, ..., v^{(a,l)} \rangle$ (for classification we label class with $c^{(l)} = v^{(0,l)}$);
- a set of n training instances is given with a set of vectors $\mathcal{T} = \{t_l, l = 1...n\}$

1 Measures for classification and relational problems

Measures for guiding the search in classification and relational problems, described in this section, are: information gain, Gain ratio, distance measure, minimum description length (MDL), *J*-measure, Gini-index and ReliefF.

1.1 Measures based on information content

Most measures used in ML are based on information content. The *amount of information* (measured in bits), necessary to determine the outcome X_j of an experiment, is defined as minus logarithm of its probability:

$$I(X_j) = -\log_2 P(X_j)$$

The average *amount of information*, necessary to determine the outcome among m disjoint possible outcomes X_j, $j = 1...m$, $\sum_j P(X_j) = 1$, is called *entropy* of an outcome:

$$H(X) = -\sum_j^m P(X_j)\log_2 P(X_j)$$

Entropy is an impurity measure

In the following the base of the logarithm is always 2 and will be omitted. Let us introduce the following notation:
n – the number of training instances,
$n_{k.}$ – the number of training instances from class c_k,
$n_{.j}$ – the number of training instances with j^{th} value of the given attribute A,
n_{kj} – the number of training instances from class c_k and with j^{th} value of the given attribute A.

The corresponding probabilities are approximated with relative frequencies (alternatively, a more sophisticated approximation can be used, such as Laplace's law of succession or the m-estimate) from the training set:
$p_{kj} = n_{kj}/n$,
$p_{k.} = n_{k.}/n$,
$p_{.j} = n_{.j}/n$,
$p_{k|j} = p_{kj}/p_{.j} = n_{kj}/n_{.j}$.

We define the following entropies:

H_C – class entropy:

$$H_C = -\sum_k p_{k.} \log p_{k.}$$

H_A – entropy of attribute A:

$$H_A = -\sum_j p_{.j} \log p_{.j}$$

H_{CA} – entropy of joint event class–attribute value:

$$H_{CA} = -\sum_k \sum_j p_{kj} \log p_{kj}$$

$H_{C|A}$ – conditional class entropy given the value of attribute A:

$$H_{C|A} = H_{CA} - H_A = -\sum_j p_{.j} \sum_k \frac{p_{kj}}{p_{.j}} \log \frac{p_{kj}}{p_{.j}}$$

$$H_{C|A} = -\sum_j p_{.j} \sum_k p_{k|j} \log p_{k|j}$$

Note that since H is nonnegative it follows that $H_{CA} \geq H_{C|A}$.

Information gain

Standard attribute quality measure is information gain. It is defined as the amount of information, obtained from the attribute, for determining the class:

$$Gain(A) = H_C - H_{C|A} \tag{1}$$

Information gain is also referred to as *mutual information* due to its symmetry:

$$H_C - H_{C|A} = H_C + H_A - H_{CA} = I(A;C) = H_A - H_{A|C} = I(C;A)$$

Information gain has the following properties:

$$Gain(A) \geq 0$$

$$max(Gain(A)) = H_C$$

$$Gain(A') \geq Gain(A)$$

where attribute A' is obtained from attribute A by replacing one of its values with two possible values. The latter property is undesired - information gain overestimates multivalued attributes. Therefore, in the following, we introduce two normalized versions of information gain.

Gain-ratio

The gain-ratio is defined as information gain, normalized with the attribute entropy:

$$GainR(A) = \frac{Gain(A)}{H_A} \tag{2}$$

The normalization eliminates the problem of overestimating the multivalued attributes, however, the gain-ratio overestimates attributes with small attribute entropy H_A. Therefore, when estimating the quality of attributes with the gain-ratio, the safe procedure is to take into account only attributes A_i with over average information gain:

$$Gain(A_i) \geq \frac{\sum_{j=1}^{a} Gain(A_j)}{a}$$

Distance measure

Distance measure, written in the form of proximity, is defined as information gain, normalized with joint entropy H_{CA}:

$$1 - D(C,A) = \frac{Gain(A)}{H_{CA}} \tag{3}$$

A measure D(C,A) is a distance, because it is:

1. nonnegative: $D(C,A) \geq 0$
2. equal zero only for identical events: $D(C,A) = 0 \Leftrightarrow C = A$
3. symmetric: $D(C,A) = D(A,C)$
4. it fulfills the triangular inequality: $D(C,A_1) + D(A_1,A_2) \geq D(C,A_2)$

Function (3) satisfactorily overcomes the problem of multivalued attributes and, contrary to the gain-ratio, is not problematic for small values of the denominator (H_{CA}).

MDL

Following the minimum description length principle (MDL) we can define the quality of attribute as its compressivity. Let us define the problem of data transmission through the communication channel. Both, the sender and the receiver

have the access to the domain definition - they know the number of values m of the given attribute A and the number of classes m_0. They also have the access to the value $v^{(l)}$ of attribute A for each training instance t_l. But only the sender knows the correct classes of instances. His task is to send the shortest possible message containing the classes of all instances to the receiver.

The sender has two options: The classes can be either explicitly coded, or the attribute A is used so that classes are separately coded for each value of the attribute. The former case requires that the message contains also the class probability distribution, so that receiver can decode the classification; in the latter case, however, each value of the attribute corresponds to different class distribution. Therefore, the latter case requires, that the message, besides classification, contains a class probability distribution for each attribute value.

The number of bits, required to code the classes of n instances with the given class probability distribution can be approximated with $H_C \times n$ plus the number of bits, required to code the class distribution. The number of different possible distributions of n instances over m_0 classes is equal to:

$$\binom{n+m_0-1}{m_0-1}$$

(if we split the interval $1...n$ into m_0 subintervals, some of them can be empty, is the same as splitting the interval $1...n+m_0$ into m_0 nonempty subintervals; there are $n+m_0-1$ possible boundaries from which m_0-1has to be selected).

Therefore, the number of bits, required to code the classes, is given with:

$$Prior\_MDL' = nH_C + \log\binom{n+m_0-1}{m_0-1}$$

The estimate of the number of bits for coding the classes, given the values of the attribute A, is the sum over all attribute values:

$$Post\_MDL'(A) = \sum_j n_{.j} H_{C|j} + \sum_j \log\binom{n_{.j}+m_0-1}{m_0-1}$$

$$Post\_MDL'(A) = nH_{C|A} + \sum_j \log\binom{n_{.j}+m_0-1}{m_0-1}$$

The quality $MDL'(A)$ of attribute A is defined as the compressivity of attribute, i.e. as the length difference of code without using attribute values and the code

with using them. The quality is normalized with the number of training instances:

$$MDL'(A) = \frac{Prior\_MDL' - Post\_MDL'(A)}{n} = Gain(A) +$$

$$+\frac{1}{n}\left(\log\binom{n+m_0-1}{m_0-1} - \sum_j \log\binom{n_{.j}+m_0-1}{m_0-1}\right) \quad (4)$$

The above quality measure uses coding, based on H_C. This coding is optimal only for arbitrary number of codewords (in our case one codeword corresponds to one training instance). If, however, the number of codewords (training instances) is known in advance (as is the case with the number of training instances), we can apply more optimal coding. The number of all possible classifications of n training examples is equal to:

$$\binom{n}{n_{1.},...,n_{m_0.}}$$

Therefore with optimal coding we obtain:

$$Prior\_MDL = \log\binom{n}{n_{1.},...,n_{m_0.}} + \log\binom{n+m_0-1}{m_0-1}$$

and

$$Post\_MDL(A) = \sum_j \log\binom{n_{.j}}{n_{1j},...,n_{m_0 j}} + \sum_j \log\binom{n_{.j}+m_0-1}{m_0-1}$$

Finally, the quality of the attribute is defined with:

$$MDL(A) = \frac{Prior\_MDL - Post\_MDL(A)}{n}$$

$$MDL(A) = \frac{1}{n}\left(\log\binom{n}{n_{1.},...,n_{m_0.}} - \sum_j \log\binom{n_{.j}}{n_{1j},...,n_{m_0 j}} +\right.$$

$$\left. +\log\binom{n+m_0-1}{m_0-1} - \sum_j \log\binom{n_{.j}+m_0-1}{m_0-1}\right) \quad (5)$$

This measure is most appropriate for estimating the quality of multivalued attributes. Its advantage is also the detection of useless attributes: if $MDL(A) < 0$ then the attribute A is non-compressive and therefore useless for theory construction (unless it is useful in combination with other attributes - see ReliefF algorithm below).

J-measure

Measures like information gain, gain-ratio, distance, and MDL are defined to evaluate the quality of an attribute as a whole. In practice we sometimes need to estimate the quality of a single attribute value. *J*-measure can be derived from information gain as follows:

$$\sum_j J_j = Gain(A) \tag{6}$$

The appropriate measure, stemming from the above equality, is

$$J_j = p_{.j} \sum_k p_{k|j} \log \frac{p_{k|j}}{p_{k.}} \tag{7}$$

J-measure can be used to evaluate the quality of single attribute value or the quality of a whole decision rule. Due to equation (6) it can be interpreted as the expected amount of information, obtained from condition j, for classification. In fact it evaluates the information content of the condition (indicated with index j), which can be either single attribute value (V_j) or a conjunction of several such conditions.

J-measure has the following two advantageous properties:

Nonnegativity: $J_j \geq 0$

Upper bound: $J_j \leq 0.53\ bit$

In addition, the following statistic approximately follows the χ^2 distribution:

$$2nJ_j \log_e 2 = 1.3863\ nJ_j \tag{8}$$

where n is the number of training instances. Therefore, J-measure can be used to verify the statistical significance of difference between prior ($p_{k.}$) and posterior ($p_{k|j}$) distribution. The test can be used to evaluate the significance of single attribute value or the significance of the whole decision rule.

1.2 Gini-index

Prior Gini-index is defined as follows:

$$Gini\_prior = \sum_k \sum_{l \neq k} p_{k.} p_{l.} = 1 - \sum_k p_{k.}^2$$

Prior Gini-index is an impurity measure.

The quality $Gini(A)$ of attribute A is defined with a difference between prior and expected posterior Gini-index:

$$Gini(A) = \sum_j p_{.j} \sum_k p_{k|j}^2 - \sum_k p_{k.}^2 \tag{9}$$

$Gini(A)$ has the following properties:

Nonnegativity: $Gini(A) \geq 0$

Maximum:

$$Gini(A) = Gini\_prior \Leftrightarrow \forall j : \exists! k : \; p_{k|j} = 1$$

Increasing the number of attribute values: $Gini(A)$ at most increases. This property is undesired - $Gini(A)$ overestimates multivalued attributes (just like Information gain).

1.3 ReliefF

All measures described so far evaluate the quality of an attribute independently of the context of other attributes. Equivalently, they assume independence of attributes with respect to class. We call them "myopic" measures because they cannot detect information content of the attribute that stems from dependencies between attributes. A small example learning task, provided in Table 1, illustrates the problem of attribute dependencies. We have three binary attributes, two possible classes, and eight training instances.

The correct target rule

$$C = (A_1 \neq A_2)$$

does not contain the third attribute. However, myopic measures (all measures, described so far) would estimate attribute A_3 as the most important and would consider the other two attributes irrelevant:

$$Gain(A_1) = GainR(A_1) = 1 - D(A_1, R) = Gini(A_1) = 0$$

and the same for attribute A_2. The reason for not detecting the importance of attributes A_1 and A_2 is their strong mutual dependance. If we evaluate each independently of the other, they both seem completely irrelevant for the given classification problem.

A_1	A_2	A_3	C
0	0	0	0
0	0	1	0
0	1	1	1
0	1	0	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	0	0

Table 1: Example learning task

The context of other attributes can be efficiently taken into account with algorithm ReliefF. Let us first describe a simpler variant, called RELIEF, which is designed for two-class problems. The basic idea of the algorithm, when analysing training instances, is to take into account not only the difference in attribute values and the difference in classes, but also the *distance* between the instances. Distance is calculated in attribute space, therefore similar instances are close to each other and dissimilar are far apart. By taking the similarity of instances into account, the context of all the attributes is implicitly considered.

The basic algorithm RELIEF (see Algorithm 1) for each instance from a random subset of m ($m \leq n$) training instances calculates a nearest instance from the same class (nearest hit H) and a nearest instance from the opposite class (nearest miss M). Then it updates the quality of (each) attribute with respect to weather the attribute differentiates two instances from the same class (undesired property of the attribute) and weather it differentiates two instances from the opposite class (desired property). By doing so, the quality estimate takes into account the local ability of the attribute to differentiate between classes. The locality implicitly takes into account the context of other attributes.

For (each) attribute A_{att} function *diff(att,Instance1,Instance2)* in Algorithm 1 returns the difference of attribute values of two instances:

$$diff(i,t_j,t_k) = \begin{cases} \frac{|v^{(i,j)}-v^{(i,k)}|}{Max_i-Min_i}, & A_i \textit{ is continuous} \\ 0, & v^{(i,j)} = v^{(i,k)} \wedge A_i \textit{ is discrete} \\ 1, & v^{(i,j)} \neq v^{(i,k)} \wedge A_i \textit{ is discrete} \end{cases} \tag{10}$$

Quality estimates W can be also negative, however, $W[A_i] \leq 0$ means that attribute A_i is irrelevant.

```
INPUT: n training instances I;
OUTPUT: for each attribute A_j a quality weight −1 ≤ W[j] ≤ 1;

function RELIEF(I: array[1..n] of instance): array[1..a] of real;
var
   inst, att, l : integer;
   W : array[1..a] of real; (* quality weights *)
   M, H : instance; (* nearest miss and nearest hit *)
begin
  for att := 1 to a do W[att] := 0.0;
  for l := 1 to m do
   begin
     randomly pick an instance I[inst];
     find its nearest hit H and nearest miss M;
     for att := 1 to a do
       W[att] := W[att] – diff(att,I[inst],H)/m + diff(att,I[inst],M)/m;
   end;
  return(W);
end;
```

Algorithm 1: Basic algorithm RELIEF; function diff is defined with eq. (10)

In fact RELIEF estimates the following difference of probabilities:

$$RELIEF(A_i) =$$

$$P(diff(i,\cdot,\cdot) = 1 | \textit{two nearest instances from the opposite classes}) -$$
$$-P(diff(i,\cdot,\cdot) = 1 | \textit{two nearest instances from the same class})$$
$$= P(diff(i,\cdot,\cdot) = 0 | \textit{two nearest instances from the same class}) -$$
$$-P(diff(i,\cdot,\cdot) = 0 | \textit{two nearest instances from the opposite classes})$$

If we omit the nearness condition we get a function, which is closely related to Gini-index. We call it (myopic) RELIEF':

$$RELIEF'(A_i) = constant \times \sum_j p_{\cdot j}^2 \times Gini'(A_i) \tag{11}$$

where $Gini'(A)$ is highly related with $Gini(A)$ from eq. (9):

$$Gini'(A) = \sum_j \left(\frac{p_{\cdot j}^2}{\sum_j p_{\cdot j}^2} \times \sum_k p_{k|j}^2 \right) - \sum_k p_{k\cdot}^2 \tag{12}$$

The only difference of $Gini'(A)$ to $Gini(A)$ is that instead of factor:

$$\frac{p_{\cdot j}^2}{\sum_j p_{\cdot j}^2}$$

in eq. (9) we have:

$$\frac{p_{\cdot j}}{\sum_j p_{\cdot j}} = p_{\cdot j}$$

However, the crucial difference between myopic RELIEF' in eq. (11) and $Gini(A)$ is in factor in front of $Gini'$ in eq. (11):

$$\sum_j p_{\cdot j}^2$$

This factor represents the prior probability that two randomly selected instances have the same value of the given attribute. The factor implicitly normalizes the RELIEF's quality estimates with respect to the number of attribute's values. While $Gini(A)$ overestimates multivalued attributes, RELIEF (and also myopic RELIEF') has no such undesired bias.

Basic RELIEF is able to evaluate the quality of continuous and discrete attributes, which are highly interdependent. For example, for very hard parity problems of arbitrary order, where the learning instances are described with additional number of irrelevant attributes, RELIEF is able to detect a subset of relevant attributes. The time complexity is $O(m \times n \times a)$, where a is the number of attributes, n the number of training instances and m is a user defined parameter. For calculation of the nearest hit and miss we need $O(n \times a)$ steps. Greater m implies more reliable evaluation of the attributes' qualities but also greater time complexity. If we set $m = n$ we get the most reliable quality estimates and the highest time complexity. This is often unacceptably slow, therefore, for large n, we set $m \ll n$, typically $m \in [30...200]$.

A more realistic variant of RELIEF is its extension, called ReliefF (see Algorithm 2). It is able to deal with incomplete and noisy data and can be used for evaluating the attribute quality in multi-class problems:

Missing attribute values: ReliefF can use also incomplete data. For that purpose we generalize function *diff* to calculate the probability that two instances have different value of the given attribute. We have two possibilities:

- one of instances (t_l) has unknown attribute value:

$$diff(A_i, t_l, t_k) = 1 - p_{v^{(i,k)}|c^{(l)}}$$

- both instances have unknown attribute value:

$$diff(A_i, t_l, t_k) = 1 - \sum_{j=1}^{m_i} \left(p_{V_j|c^{(l)}} \times p_{V_j|c^{(k)}} \right)$$

Noisy data: The most important part of algorithm RELIEF is searching for the nearest hit and miss. Noise (mistake) in class and/or attribute value significantly affects the selection of nearest hits and misses. In order to make this process more reliable in the presence of noise, ReliefF uses k nearest hits and k nearest misses and averages their contributions to attributes' quality estimates. k is a user defined parameter with typical values $k \in [5...10]$. This simple extension significantly improves the reliability of quality estimates.

Multi-class problems: Instead of k nearest hits and misses, ReliefF searches for k nearest instances from each class. The contributions of different classes are weighted with their prior probabilities. In Algorithm 2 the weighting

INPUT: n training instances I and number k of nearest instances from each class;
OUTPUT: for each attribute A_j a quality weight $-1 \leq W[j] \leq 1$;

```
function ReliefF(I: array[1..n] of instance; k : integer): array[1..a] of real;
var
   inst,att, kk, cl, c, cc, l : integer;
   W : array[1..a] of real;
   M : array[1..k,1..m0] of instance;
begin
  for att := 1 to a do W[att] := 0.0;
  for l := 1 to m do
  begin
   randomly pick an instance I[inst];
   for cl := 1 to m0 do
    begin
     find k nearest instances M[kk,cl] from class c_cl, kk=1..k;
     c := arg_cc c_cc = c^(inst); (* class of I[inst] *)
     for att := 1 to a do
      for kk := 1 to k do
       if cl = c (* nearest hit? *)
         then W[att] := W[att] – diff(att,I[inst],M[kk,cl])/(m*k)
         else W[att] := W[att] + p_cl./(1 – p_c.) * diff(att,I[inst],M[kk,cl])/(m*k)
    end; (* for c *)
   end; (* for l *)
  return(W);
end;
```

Algorithm 2: ReliefF

factor is $p_{cl.}/(1-p_{c.})$. The class of an instance is c, while cl is the class of its nearest miss. The factor is therefore proportional to the probability of class cl, normalized with the sum of probabilities of all classes, different from c.

2 Measures for regression

In regression the quality of attributes can be evaluated using the following measures:
- expected change of variance,
- regressional ReliefF, and
- minimum description length principle (MDL).

2.1 Change of variance

Instead of impurity measures, which are used in classification problems, in regression problems the *variance* of the continuous target (regression) variable is used. It is defined as the mean squared error:

$$s^2 = \frac{1}{n}\sum_{l=1}^{n}(c^{(l)} - \overline{c})^2$$

where $\overline{c}$ is the mean of the target variable over all n training examples:

$$\overline{c} = \frac{1}{n}\sum_{l=1}^{n} c^{(l)}$$

Variance is closely related to Gini-index, which is an impurity measure. If in the binary classification problem one class is transformed into value 0 and the other into value 1 of the regression variable (the discrete class variable is transformed into continuous one), we get the following equality:

$$Gini\_prior = 2s^2$$

For evaluating the quality of attribute A_i we use *the expected change of variance*:

$$ds^2(A_i) = \frac{1}{n}\sum_{l=1}^{n}(c^{(l)} - \overline{c})^2 - \sum_{j=1}^{m_i}\left(p_{.j}\frac{1}{n_{.j}}\sum_{l=1}^{n_{.j}}(c_j^{(l)} - \overline{c}_j)^2\right) \qquad (13)$$

The continuous value of the target variable for the l-th training instance with the j-th value of attribute A_i is denoted with $c_j^{(l)}$, while $\overline{c}_j$ denotes the mean of these values over all $n_{.j}$ instances with j-th value of attribute A_i:

$$\overline{c}_j = \frac{1}{n_{.j}} \sum_{l=1}^{n_{.j}} c_j^{(l)}$$

The expected change of variance behaves similarly to the expected change of impurity (i.e. measures Information gain and Gini-index in classification problems):

Nonnegativity: $ds^2(A_i) \geq 0$

Maximum: $max(ds^2(A_i)) = \frac{1}{n}\sum_{l=1}^{n}(c^{(l)} - \overline{c})^2$

Increasing the number of attribute values: If we change attribute A_i with m_i values into attribute A_i', so that value V_{m_i} is replaced with two possible values V'_{m_i} and V'_{m_i+1}, then it holds:

$$ds^2(A_i') \geq ds^2(A_i)$$

2.2 Regressional ReliefF

Like most of the attribute quality measures, defined for classification problems, the expected change of variance is also a myopic measure. When estimating the quality of an attribute it does not take into account the context of other attributes. In this subsection we develop a non-myopic measure for regression by appropriately adapting algorithm ReliefF.

In regression problems the target variable is continuous, therefore nearest hits and misses cannot be used in a strict sense as in algorithm ReliefF. RReliefF (Regressional ReliefF) uses a kind of "probability" that two instances belong to two "different" classes. This "probability" is modelled with distance between the values of the target variable of two training instances.

Myopic and non-myopic RELIEF calculates the following:

$$RELIEF(A_i) = \frac{P_{samecl|eq\_val} P_{eq\_val}}{P_{samecl}} - \frac{(1 - P_{samecl|eq\_val}) P_{eq\_val}}{1 - P_{samecl}} \qquad (14)$$

For estimating the quality in equation (14) we need the (posterior) probability $P_{samecl|eq\_val}$ that two (nearest) instances belong to the same class provided they

INPUT: n training instances I and number k of nearest instances;
OUTPUT: for each attribute A_j a quality weight $-1 \leq W[j] \leq 1$;

```
function RReliefF(I: array[1..n] of instance; k: integer): array[1..a] of real;
var
   N_dC, inst, att, kk, l : integer;
   W, N_dC&dA, N_dA : array[1..a] of real;
   M : array[1..k] of 1..n;
begin
  set all N_dC, N_dA[att], N_dC&dA[att], W[att] to 0;
  for l := 1 to m do
  begin
   randomly pick an instance I[inst];
   find indices M[kk] of k nearest instances, kk ∈ [1..k];
   for kk := 1 to k do begin
     (* index 0 in diff corresponds to target variable *)
    N_dC := N_dC + diff(0,I[M[kk]],I[inst])/k;
    for att := 1 to a do begin
       N_dA[att] := N_dA[att] + diff(att,I[M[kk]],I[inst])/k;
       N_dC&dA[att] := N_dC&dA[att] + diff(0,I[M[kk]],I[inst])×
          diff(att,I[M[kk]],I[inst])/k;
    end; (* for att *)
   end; (* for kk *)
  end; (* for l *)
(* for each attribute calculate the value of (15) *)
  for att := 1 to a do
    W[att] := N_dC&dA[att]/N_dC − (N_dA[att] − N_dC&dA[att])/(m − N_dC);
  return(W);
end;
```

Algorithm 3: RReliefF

have the same attribute value, and the prior probability P_{samecl} that two instances belong to the same class. We can transform the equation, so that it contains the probability that two instances belong to different classes provided they have different attribute values:

$$Relief(A_i) = \frac{P_{diffcl|diff}P_{diff}}{P_{diffcl}} - \frac{(1-P_{diffcl|diff})P_{diff}}{1-P_{diffcl}} \tag{15}$$

Here P_{diff} denotes the prior probability that two instances have different attribute values, and P_{diffcl} denotes the prior probability that two instances belong to different classes.

Algorithm RReliefF has to approximate the probabilities in equation (15). The details are provided in Algorithm 3. The algorithm calculates the "frequencies":

- N_{dC} – sum of "probabilities" that two nearest instances belong to different classes;
- $N_{dA}[att]$, – sum of "probabilities" that two nearest instances have different attribute values;
- $N_{dC\&dA}[att]$ – sum of "probabilities" that two nearest instances belong to different classes and have different attribute values.

Finally, from above "frequencies", it calculates the attribute qualities $W[att]$ using equation (15).

Time complexity of RReliefF is equal to that of basic RELIEF, i.e. $O(m \times n \times a)$. The most time consuming operation is searching for k nearest instances. We need to calculate n distances which can be done in $O(n \times a)$ steps. Building the heap requires $O(n)$ steps and k nearest instances can be extracted from heap in $O(k \log n)$ steps. In practice this is always less than $O(n \times a)$.

Both algorithms, ReliefF and RReliefF, calculate the quality of attributes according to equation (15), which represents a unified view on the attribute quality estimation – in classification and regression.

2.3 MDL in regression

In order to use the minimum description length principle (MDL) in regression we have to determine an appropriate coding of real numbers, which can be used for coding the values of target variable, the values of continuous attributes and the prediction errors. Coding has significant influence on the behavior of MDL

estimate. Because, in principle, the coding of an arbitrary real number requires an infinite code, we have to bound the precision with a fixed number of decimal places. In such a way the real numbers can be coded like natural numbers.

When coding the value of the target variable (or the value of a continuous attribute), all possible values have the code of equal length, defined with:

$$\log_2 \frac{Interval}{Precision} \tag{16}$$

It is obvious that the selection of precision influences the code length.

On the other hand, when coding the error, the desired code property is that smaller numbers (lower error) have shorter code length. The Rissanen code can be used for that purpose:

$$\begin{aligned} Rissanen(0) &= 1 \\ Rissanen(n) &= 1 + \log_2 n + \log_2(\log_2 n) + ... + log_2(2.865064...) \end{aligned}$$

The sum is limited to positive summands. The idea of the code is that for coding the number n we need $\log_2 n$ bits, for coding the length of the code, we need another $\log_2(\log_2 n)$ bits, etc.

When the appropriate coding is selected, the attribute quality can be estimated in the same manner as in classification (equations (4) and (5)). Instead of class distribution we code the mean value of target variable using equation (16) and for each training instance the deviation (distance) from the mean value using equation (17).