# Post-processing of qualities of decision rules within a testing phase

Frantisek Franek and Ivan Bruha
Department Computing & Software
McMaster University
Hamilton, Ont., Canada L8S4L7

*Email*: {franya|bruha}@mcmaster.ca

## ABSTRACT

Knowledge bases are often generated by learning (data mining) algorithms. One possible form of such a knowledge base (decision scheme, model) is a set of decision rules. If a classifier uses an unordered set of rules (which happens frequently) a problem arises concerning what to do if the classification of an unseen object 'fires' rules of different classes.

One possible solution consists of calculating a numerical factor that would explicitly indicate a quality (predictive power) of each rule and giving a higher priority to the rule with a higher quality. In the literature, there can be found several formulas for calculations of *rule quality*. So far as we know, in existing models the rule qualities are calculated by the learning (data mining) algorithm itself and remain constant during the phase of classification.

This paper introduces a new strategy that allows to modify (refine) the rule qualities during the classification of unseen objects. The refinement is performed in a *feedback loop* that can increase or decrease the quality of the rule involved with the classification of each unseen object. Thus, the entire method of refinement of rule qualities may be viewed as a post-processing of a set of decision rules performed within the testing phase.

The results of experiments with both real-world and artificial data are discussed and analysed.

**Key words**
data mining, post-processing, decision rules, rule quality, refinement, feedback loop

## 1. SPECIFICATION OF THE PROBLEM

### 1.1 Knowledge Discovery

Knowledge discovery in databases (KDD), as its name reveals, deals with processing of usually very large databases in a profound and robust way in order to reveal unknown and unanticipated patterns within the data. Research in KDD is concerned with developing methods and techniques how to reveal knowledge hidden in the data. The paper [1] defines knowledge discovery as a nontrivial process of identifying valid, novel, and ultimately understandable knowledge in data.

Hence, the term KDD is commonly used for the overall process of determining useful knowledge from databases, i.e. extracting high-level knowledge from low-level data in the context of large databases. Knowledge discovery can be viewed as a multidisciplinary activity because it exploits several research disciplines of artificial intelligence (AI), such as machine learning, pattern recognition, expert systems, and knowledge acquisition, as well as mathematical disciplines, such as statistics, theory of information, and uncertainty processing.

The entire process of KDD can thus be characterized by these stages:
(a) selecting a problem area,
(b) collecting the data,
(c) preprocessing of the data,
(d) data mining, i.e. extracting pieces of knowledge,
(e) postprocessing of the knowledge derived.

In our paper, we focus on the last two stages of KDD: partly data mining, but mainly post-processing.

### 1.2 Induction of Decision Rules

A commonly used strategy in machine learning is *divide-and-conquer*. It is widely used by the family of TDIDT (Top-Down Induction of Decision Trees) learning algorithms that induce decision trees from a given set of examples. One if its first pioneers was the ID3 [12], but currently the most used and well-known member of the family is the C4.5 algorithm [13].

Another widely used strategy in machine learning relies on the *covering* paradigm. The AQx family of algorithms [11] was one of the first utilising this approach. The family the CNx of learning algorithms relies on covering paradigm as well. The second author of this paper developed the CN4 algorithm [5], a significant extension of the well-known CN2 [8]. CN4 is able to process unknown attribute values, numerical attributes, continuous classes, perform economy learning, etc.

In the project described here, we used the learning algorithm CN4 that induces a knowledge base (model, concept description) in a form of a set of decision rules; this set produced by CN4 may be *ordered* or *unordered*. If the decision set is ordered, then classification of an unseen object is quite straightforward: the classifier goes through the list of rules from its beginning to the end looking for the first rule that matches ('fires') the given object, which is then categorized into the class attached to the rule.

However, more frequent and natural form seems to be an unordered set of rules, often utilized in expert systems. The classification relying on an unordered set of decision rules exhibits a significant deficiency, that may not be immediately apparent. The three cases listed bellow illustrate the situation:

**1/** If the unseen object satisfies one or more rules of the same class, then the object is categorized to the unique class assigned to the rule(s).

**2/** If the unseen object is not covered by any rule, then either the classifier informs the user about its inability to decide, or the object is assigned by default to the majority class in the training set, or some similar techniques are invoked.

**3/** Difficulty arises if the object satisfies more rules assigned to different classes. Then some schemes have to be applied to assign the unseen object to the most appropriate class.

A possible resolution principle to clarify the conflict situation (case **3/**) associates each rule in the decision scheme of a classifier with numerical factors expressing certain attributes of the rule, such as a measure of belief in the rule, its power, predictability, reliability, likelihood, and so forth. A collection of these attributes is commonly called the *rule quality*.

When we decide on a formula for the rule quality, we have to provide a scheme for combining these qualities if many rules fire. The conflict case is then resolved using the (combined) quality of the rules of the same class that fired for the given object: the rule combination with the maximum value will determine the class of the unseen object. (Note that this resolution principle does not address the case when qualities of two rule combinations for different classes come out with the same maximum value.)

Formulas for assignment of rule quality have been studied and tested in several papers [3], [7], [14]. The paper [6] provides a survey of such formulas as well as three fundamental schemes for rule quality combination.

In the recently studied systems utilising the quality measure there is no feedback loop between the classifier and the corresponding learner (data mining algorithm). In this project we proposed and studied such a feedback loop whose main purpose is the refinement (modification) of rule qualities according to the ratio correct/false predictions made by the classifier. The feedback loop increases or decreases the qualities of the rules that were involved in the classification of the currently processed unseen object.

We categorize this method as a post-processing algorithm that modifies (refines) the qualities of the rules induced by a "traditional" learning (data mining) algorithm.

The paper is organized as follows. Section 2 introduces some formulas for rule qualities and schemes for their combination. Section 3 describes the actual feedback loop for refining the rule qualities. Section 4 then presents some experiments performed with both real-word databases as well as some artificial ones. An analysis of the feedback loop and some future work are presented in Section 5.

## 2. Rule Qualities and Quality Combination

Let a given task to be classified be represented by a set of training examples that belong to two classes, named $C$ and $\bar{C}$. Let $R$ be a decision rule of the class $C$, that is, of the form

```
R:  if Condition then class is C
```

From the statistical viewpoint, the above rule can be described by the 2x2 *contingency table* [4]:

| | class $C$ | not class $C$ | |
|---|---|---|---|
| rule $R$ covers | $rc$ | $r\bar{c}$ | $r$ |
| does not cover | $\bar{r}c$ | $\bar{r}\bar{c}$ | $\bar{r}$ |
| | $c$ | $\bar{c}$ | $K$ |

where $rc$ is the number of training examples that are covered by the rule $R$ and belong to the class $C$;
$r\bar{c}$ is the number of examples covered by the rule $R$ but not belonging to the class $C$, etc.;
$r = rc + r\bar{c}$ is the number of examples covered by $R$;
$c = rc + \bar{r}c$ is the number of training examples of the class $C$, etc.; and
$K = c + \bar{c} = r + \bar{r}$ is the number of all training examples.

Using the elements of the contingency table, we may define the *consistency* of the rule $R$ by

$$cons(R) = \frac{rc}{r} \tag{1}$$

and its *completeness* by

$$compl(R) = \frac{rc}{c} \qquad (2)$$

The paper [6] surveys both empirical and statistical formulas for rule quality. Rule quality can be viewed as a function of the contingency table:

$$quality(R) = f(rc, r\bar{c}, \bar{r}c, \bar{r}\bar{c})$$

or even just a function of consistency and completeness:

$$quality(R) = F(cons(R), compl(R))$$

[2] works with the simplest formula in a form of a weighted sum:

$$quality_{\text{Mich}}(R) = w_1\, cons(R) + w_2\, compl(R) \qquad (3)$$

where $w_1, w_2 \in (0, 1)$ are user-defined weights, usually $w_1 + w_2 = 1$ . The formula (3) is used quite often; however, its great disadvantage is the user has to specify the weights. Therefore some systems use a formula with a product of consistency and completeness, without weights:

$$quality_{\text{BT}}(R) = cons(R)\ .\ h(compl(R)) \qquad (4)$$

In particular, the authors of [7] came with the following form of the function $h$ in (4) that they use in their integration system Integ.3 :

$$h(x) = \exp(x\text{-}1)$$

As soon as a formula for rule quality is selected and a knowledge base (model) is induced, we may employ several different classification schemes. Each such scheme has to resolve the conflict case (case **3**/ above). If one or more rules from different classes are satisfied (fired) for a given input unseen object, then:

(i) the classification scheme has to combine the qualities of the fired rules belonging to the same class, thus yielding the so-called *combined quality*;

(ii) the unseen object is categorized to the class with the maximum combined quality.

More formally, let $R_{r,j}$ be the rules of the class $C_r$ , r=1...,R, that are satisfied (fired) for the input object $\mathbf{x}$ and $q_{r,j}$ the corresponding qualities of these rules; the sequence j=1,2,... differs for each class. Then, the classification scheme is a function *ClassScheme* that returns the combined quality *comb_quality* for each class $C_r$ in the form

$$comb\_quality_{r}(\mathbf{x}) = ClassScheme(q_{r,1}, q_{r,2}, \ldots)$$

r=1,...,R. The input object $\mathbf{x}$ is then classified to the class r* for which

$$comb\_quality_{r^*}(\mathbf{x}) = \max_{r}\ comb\_quality_{r}(\mathbf{x})$$

Some classification schemes are introduced and analyzed e.g. in [6], [10].

## 3. Refinement of Qualities in a Feedback Loop

The classification systems in the literature employing the rule qualities have not, to our knowledge, utilized any information from the result of classification of unseen objects for updating (modifying, refining) rule qualities. We have developed a new strategy that allows to modify (refine) the rule qualities during the classification of unseen objects. This is achieved by a *feedback loop* that can increase or decrease the qualities of the rules that were involved in the classification of the unseen object.

The power and predictability of a decision set (knowledge base, model) is usually measured by the classification accuracy, i.e. the number of correctly classified testing objects over the total size of the testing set. More sophisticated formulas may be used, e.g. based on the so-called loss-matrix.

In a traditional scenario the testing of a classifier is a passive process without any influence on the decision set induced by the learning (data mining) algorithm. Nevertheless, it seems to be quite natural to enhance the rule qualities according to the results of classification. Thanks to the feedback loop designed in our project, the rule qualities may be refined (modified) according to the correct/false predictions made by the classifier. This process thus represents a simple version of meta-learning that is performed above the existing knowledge base that was induced by a learner. Figure 1 illustrates the entire idea of the feedback loop.

The top-level flow chart can be described as follows:

```
for each testing object do
  1. (penalty)
        if the classifier made a wrong decision about the
        class of the testing object
        then decrease the quality of all fired rules attached
        to by the wrong class
  2. (reward)
        if the classifier made a correct decision about the
        class of the  testing object
        then increase the quality of all fired rules that are
        attached to by the correct class
enddo
```

The entire feedback loop represents a quite simple incremental learning procedure above the set of rule qualities. The testing data are presented to the classifier only once. Since the rule qualities are represented by an array of numerical values, we opted for a very simple learning algorithm for modifying numerical vectors. The formula is given below (derived from the *delta* method [9]):

$$q := q \pm c * \|Q\|$$

where $c$ is the delta-constant, selected by the user of the algorithm,

$\|Q\|$ is the normalized sum of all the rule qualities in the given knowledge base (model),

+ (−) is used if the rule classified the object correctly (incorrectly),
:= represents the assignment statement.

## 4. EXPERIMENTS

To test out the performance of the feedback loop mechanism, various datasets from Machine Learning Repository as well as a medical dataset from Bern University were used. We also ran experiments with two artificially created datasets. The following databases were used:

- ***Japanese Credit***, ***Iris***, ***Pima Indian Diabetes***, *and* ***Australian Credit***: These datasets were taken from the Machine Learning Repository.
- ***Thyreosis***: The data for diagnosing of thyroid gland disease were provided by the Institute of Nuclear Medicine of Inselspital, Bern, Switzerland. The database has been used at the Department of Advanced Mathematics, University of Bern, and also in the project CN4 [5]. The entire task in fact consists of four subsets (*Thyr21, Thyr5, Thyr7,* and *Thyr14*).
- ***ArtData 1*** *and* ***ArtData 2***: We utilize these artificially created datasets for comparative analysis.

Table 1 exhibits the characteristics of all the datasets.

We ran the following algorithms on the above ten datasets:

(i) the covering algorithm CN4 (the unordered mode, of course);
(ii) CN4-unordered with the feedback loop in the incremental mode.

Each of above ten databases has been randomly separated to three subsets:

- the learning subset used for inducing the set of decision rules (with their initial rule qualities), size 50%;
- the primary testing subset used for the feedback loop to modify the rule qualities, size 25%;
- the secondary testing subset used for the actual testing, i.e. calculating the classification accuracy, size 25%.

This scenario was executed 20 times for each combination. Consequently, Table 2 gives in each cell the average of the classification accuracy (of the secondary testing subsets) over the 20 runs.

## 5. ANALYSIS AND FUTURE WORK

There is a great effort to improve predictive power of models induced by learning (data-mining) algorithms. Such enhancements are obtained through various concepts of boosting, bagging, combining, stacking, etc. This paper introduces a new feature in learning (data mining) which can in a simple way increase the predictive power of the resulting model. Current data mining systems test the predictive power of a knowledge base (model, decision set) in a passive way, without influencing the induced knowledge base itself. Our approach introduces a feedback loop that can modify the qualities of the decision rules and thus improve the predictive power. The modification procedure uses almost standard reward/penalty scenario.

The analysis of the above experimental data reveals the following:

- The percentage of unknown attribute values yields a significant influence on the behaviour of the feedback loop and thus on better classification accuracy. One can observe that the datasets with larger percentage of unknown attribute values yield better performance of the feedback loop.
- The larger the size of a database, the better performance of the feedback loop.
- According to our analysis, the most important factor that influences the behaviour of the feedback loop is the so-called *redundancy rate* for a given knowledge base (set of decision rules) *KB* ; it is defined as the average numbers of rules in *KB* that fire for (are satisfied by) an object in the training set.
  The datasets of small sizes with small (or no) percentage of unknown attribute values have the redundancy rate from 1.0 to 1.10 .
  Therefore, this was the reason we employed artificial datasets, too; they were formed to exhibit higher redundancy rule: 1.5 to 2.5 .

Our future work in this topic will consequently carry out the following:

(i) Finding precisely how the feedback loop performance depends on the redundancy rate is a very important issue.
(ii) We concluded that a further study and the design of the data mining systems inducing redundant knowledge bases with a large redundancy rate should be done.
(iii) In the current system, the feedback loop is applied to modify the qualities of single rules. The same principle of a feedback loop could be applied to a set of several knowledge bases (models), each accompanied by the *model quality*. The updating (modifying) these model qualities can be done on the same principle as described in this paper.

## 6. REFERENCES

[1] R. Agrawal, G. Psaila: "Active Data Mining". *1st International Conference on Knowledge Discovery and Data Mining*, Menlo Park, Calif., 3-8 (1995).
[2] F. Bergadano et al.: "Measuring quality of concept

descriptions". *EWSL-88,* Glasgow (1988).

[3] F. Bergadano et al.: "Learning two-tiered descriptions of flexible concepts: The Poseidon system". *Machine Learning J.* 8, 5-43 (1992).

[4] Y.M.M. Bishop, S.E. Fienberg, P.W. Holland: *Discrete Multivariate Analysis: Theory and Practice.* MIT Press, Cambridge, MA (1991).

[5] I. Bruha, S. Kockova: "A support for decision making: Cost-sensitive learning system". *Artificial Intelligence in Medicine* 6, 67-82 (1994).

[6] I. Bruha: "Quality of Decision Rules: Definitions and Classification Schemes for Multiple Rules". In: G. Nakhaeizadeh, C.C. Taylor: *Machine Learning and Statistics: The Interface*, John Wiley, New York, 107-131 (1997).

[7] P.B. Brazdil, L. Torgo: "Knowledge acquisition via knowledge integration". In: *Current trends in knowledge acquisition*. IOS Press (1990).

[8] P. Clark, T. Niblett: "The CN2 induction algorithm". *Machine Learning J.* 3, 261-283 (1989).

[9] S.K. Fu: *Sequential methods in pattern recognition and machine learning*. New York-London, Academic Press (1968).

[10] I. Kononenko: "Combining decisions of multiple rules". In: B. du Boulay, V. Sgurev (eds.): *Artificial Intelligence V: Methodology, Systems, Applications*, Elsevier, Amsterdam (1992).

[11] R.S. Michalski: "Pattern recognition as rule-guided inductive inference". *IEEE Trans. Pattern Anal. Mach. Intell.* PAMI-2 (4), 349-361 (1980).

[12] J.R. Quinlan: "Induction of decision trees". *Machine Learning J.* 1, 81-106 (1986).

[13] J.R. Quinlan: "C4.5: Programs for Machine Learning". Morgan Kaufmann (1992).

[14] L. Torgo: "Controlled redundancy in incremental rule learning". *ECML-93*, Vienna, 185-195 (1993).

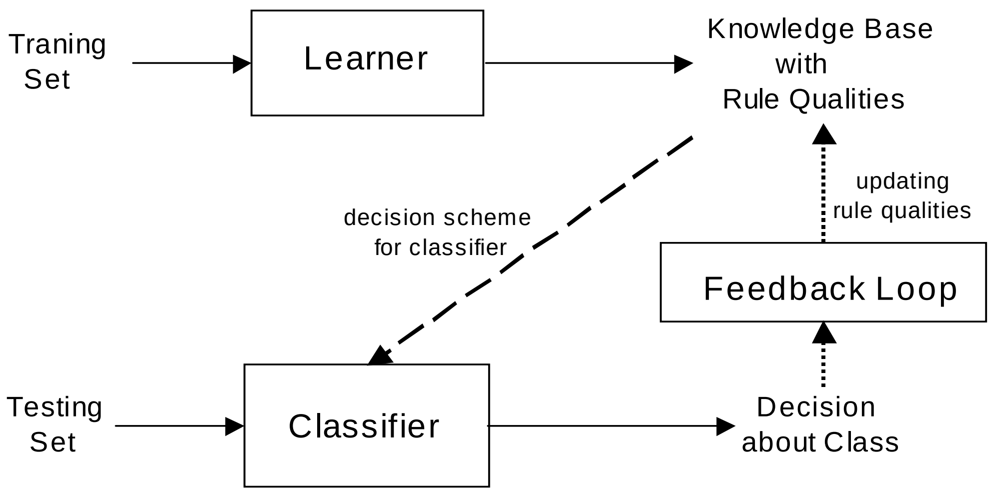

Figure 1. Feedback loop from Classifier to Knowledge Base

| dataset | # classes | frequency of majority class | # examples | # attributes | numerical attributes | average # of values per symbolic attribute | % unknown values |
|---|---|---|---|---|---|---|---|
| *Japanese Credit* | 2 | 68% | 125 | 10 | 5 | 3 | 0% |
| *Iris* | 3 | 33% | 147 | 4 | 4 | - | 0% |
| *Pima Indian Diabetes* | 2 | 65% | 768 | 8 | 8 | - | 0% |
| *Australian Credit* | 2 | 55% | 690 | 14 | 6 | 4.6 | 0% |
| *Thyr21* | 2 | 72% | 269 | 21 | 5 | 5.4 | 23% |
| *Thyr5* | 2 | 72% | 269 | 5 | 2 | 5.7 | 29% |
| *Thyr7* | 3 | 48% | 73 | 7 | 6 | 2.0 | 36% |
| *Thyr14* | 3 | 44% | 72 | 14 | 1 | 2.8 | 12% |
| *ArtData 1* | 4 | 40% | 600 | 20 | 5 | 4.5 | 30% |
| *ArtData 2* | 4 | 35% | 500 | 15 | 3 | 3.0 | 15% |

Table 1.  Characteristics of all ten datasets.

| database | CN4 | feedback loop |
|---|---|---|
| *Japanese Credit* | 96.8 | 97.0 |
| *Iris* | 97.0 | 97.5 |
| *Pima Indian Diabetes* | 95.8 | 96.5 |
| *Australian Credit* | 100 | 100 |
| *Thyr21* | 96.5 | 97.5 |
| *Thyr5* | 91.0 | 91.5 |
| *Thyr7* | 95.3 | 96.8 |
| *Thyr14* | 92.5 | 93.7 |
| *ArtData 1* | 81.0 | 84.1 |
| *ArtData 2* | 75.7 | 77.1 |

Table 2. Classification accuracy (in %) for all three algorithms.