Int. J. Appl. Math. Comput. Sci., 2001, Vol.11, No.3, 675–690

ROUGH MODELING—A BOTTOM-UP APPROACH TO MODEL CONSTRUCTION

Terje LØKEN*, Jan KOMOROWSKI**

Traditional data mining methods based on rough set theory focus on extracting models which are good at classifying unseen objects. If one wants to uncover new knowledge from the data, the model must have a high descriptive quality—it must describe the data set in a clear and concise manner, without sacrificing classification performance. Rough modeling, introduced by Kowalczyk (1998), is an approach which aims at providing models with good predictive *and* descriptive qualities, in addition to being computationally simple enough to handle large data sets. As rough models are flexible in nature and simple to generate, it is possible to generate a large number of models and search through them for the best model. Initial experiments confirm that the drop in performance of rough models compared to models induced using traditional rough set methods is slight at worst, and the gain in descriptive quality is very large.

Keywords: knowledge discovery, rough sets, rough modeling, descriptive models

1. Introduction

Models in the form of propositional decision rules, extracted from data using methods such as rough set theory (Pawlak, 1991), are used for two main purposes: *prediction* and *description*. Prediction is concerned with predicting future or unknown values of some attributes using available data, while description means to identify important patterns in the data, and present them to the user in an understandable way.

When performing knowledge discovery from databases (KDD), we are interested in finding as good a model as possible from a set of data. However, what constitutes a good model may vary, depending on the goals of the particular KDD session. If the goal is to build a model able to classify unseen objects as accurately as possible, the predictive quality is all-important. If the goal of the KDD process is description, we need to formalize what it means that a model displays good descriptive qualities. This is a difficult task, but the *size* of the model is of fundamental importance. A model consisting of thousands of rules, or utilizing several hundred attributes, is incomprehensible, while a small model, containing in the neighborhood twenty rules or fewer is easily read and comprehended.

* Research and Development, FAST Search & Transfer ASA, Støperigata 2, P.O. Box 1677 Vika, N-0120 Oslo, Norway, e-mail: Terje.Loken@fast.no

** Knowledge Systems Group, Department of Information and Computer Science, Norwegian University of Science and Technology, N-7491 Trondheim, Norway, e-mail: janko@idi.ntnu.no

1.1. Compact Models

Very often we would like to find a model which has good descriptive quality and still retains good predictive quality. Our starting assumption is that a good descriptive model will be a *compact* model, meaning that it will contain a limited number of rules, and each rule will use a manageable number of attributes. There are two fundamentally different approaches to finding models displaying these characteristics. The first approach starts out by generating as good a predictive model as possible, ignoring the descriptive quality. It is then believed that within this model, there are some patterns which are of fundamental importance, and others which are redundant or apply only to a small number of objects in the data. Using various *filtering* strategies, one tries to reduce the size of the model while retaining classifier performance. This process is called *rule pruning*, and usually involves using some sort of *rule quality measure*, which gives an estimate on the importance of each individual rule. A large number of quality measures exist, and an overview of these, as well as an investigation of the performance of heavily filtered rule sets, can be found in (Ågotnes *et al.*, 1999a).

Another philosophy, and the one adopted here, is that one should try to locate the important patterns directly. There are several reasons for doing this. If we use traditional methods to find a good predictive rule-based model, the rule discovery process is often computationally very intensive. As an example, most rough set-based (Pawlak, 1991) methods involve the search for an optimal set of *reducts*, a problem which is NP-hard (Skowron and Rauszer, 1991). Approximation algorithms exist, but often even these approximations have a substantial complexity. In addition, algorithms for filtering the resulting model may be complex, and the resulting total computational demand may be very large. There is also the possibility that the best predictive model with no restrictions on size may contain only very specific rules, and filtering will therefore not uncover rules of a more general nature.

An alternative and very different approach is represented by *discovery oriented* approaches such as the Explore algorithm (Mienko *et al.*, 1996), which attempt to extract common patterns and regularities, focusing solely on descriptive qualities and not on the predictive quality.

2. Rough Models

Rough Data Modeling is a method introduced by Kowalczyk (1998), who attempts to address two common problems found in traditional data mining methods: the computation cost of model generation, and the inability to tailor the method to the specific needs of each data mining session. Kowalczyk argues, as do many others, that knowledge discovery should be looked upon as an iterative process. With the large number of alternatives found at each stage in the knowledge discovery process (feature selection, discretization, data mining, etc.), it is impossible to find general guidelines which will always produce the best model. In the words of Klösgen (1996):

> "... a KDD process cannot be specified in advance and automated completely, because it depends on dynamic, result dependent goals and intuitions of the analyst and emerges iteratively."

Thus viewed, a KDD process is a search for a model satisfying a set of criteria which is *unique for that particular session*. In order to maximize the probability of finding the best model, we would like to generate as large a number of models as possible. However, the computational costs of many commonly used algorithms prohibit the generation of numerous models, and in addition, most algorithms are specifically designed to maximize (or minimize) a certain measure, or a predefined combination of measures (accuracy, specificity, misclassification cost, etc.).

Searching through a large group of models is similar to the idea formalized in the *wrapper approach* for feature subset selection (Kohavi and John, 1997), where one uses the induction algorithm as a black box to search for the feature (attribute) subset which generates the best classifier. In this scheme, a model is created and evaluated at each point in the search space, using the selected induction algorithm. While the algorithm is geared towards feature subset selection, it will work just as well for model creation, since the optimal feature subset is the feature subset which results in the best model. This approach works well for finding accurate models, but as expected, it is very resource-consuming (Kohavi and John, 1997).

Recognizing this, rough data modeling (Kowalczyk, 1998) simplifies the model generation process and makes it feasible to search through a large number of models. In addition, it allows the user to tailor the data mining process, by specifying in detail how to evaluate each model. As in traditional rough set theory, the starting point is a decision system $\mathcal{A} = (U, A, \{d\})$, where U is the universe of objects, A is the set of condition attributes, and d is the *decision attribute*, with the set of possible decisions denoted by V_d.

Given a decision system $\mathcal{A} = (U, A, \{d\})$, a rough data model (RDM) of $\mathcal{A}$ is a triple

$$\mathcal{M} = \langle B, \bar{d}_B, \preceq \rangle, \tag{1}$$

where B is an attribute subset of A which, using the indiscernibility relation $\mathrm{IND}(B)$, induces a set of *equivalence classes* E^B, where all elements in each class E_i^B have the same values for the attributes in B. Here $\bar{d}_B : E^B \to V_d$ is a *class decision function* for the equivalence classes E^B, which assigns all the objects in each class the same decision value, producing a deterministic decision system $\mathcal{A}'$.

The linear ordering $\preceq$ of the classes E^B is achieved using a combination of different class characteristics, such as the size of the class, the number of elements with a particular decision value, the number of correctly classified elements, etc. Note that $\preceq$ is an equivalence[1] relation, meaning that the set of classes E^B is *totally ordered*. Both the class decision function and the ordering on the classes are decided on by the user. Further details can be found in (Kowalczyk, 1998).

Each equivalence class in a particular rough data model is uniquely identified by the values of the attributes in B and is thus equivalent to a single rule of the form

$$\big(a_1(x) = v_1\big) \wedge \cdots \wedge \big(a_n(x) = v_n\big) \to \big(d(x) = \bar{d}_B([x]_B)\big), \tag{2}$$

[1] A reflexive, antisymmetric and transitive relation.

where $B = \{a_1, \ldots, a_n\}$ and $\{v_1, \ldots, v_n\}$ are the characteristic values of the equivalence class of x, $[x]_B$. Note that at no point in the process are reducts or any similar concepts calculated, the selected subset of attributes directly forms the equivalent of a reduct, and all the attributes in B are used to form the model, regardless of any redundancies.

Example 1. *(Rough Data Model)* A simple decision system is shown in Table 1, with simple numerical values for each attribute. A rough data model for this decision system

Table 1. Our example decision system, and a sample rough data model created using the attributes X and Y.

	X ($\mathbf{a_1}$)	**Y** ($\mathbf{a_2}$)	**Z** ($\mathbf{a_3}$)	**D** (**d**)
x_1	1	1	1	0
x_2	1	1	1	0
x_3	2	1	2	1
x_4	1	3	1	1
x_5	3	2	1	1
x_6	2	3	2	0
x_7	1	1	2	1
x_8	4	3	2	1
x_9	4	3	2	1

Rough Data Model
$X = 1 \wedge Y = 1 \rightarrow D = 0$
$X = 4 \wedge Y = 3 \rightarrow D = 1$

can be generated by setting $B = \{X, Y\}$. This results in a set of equivalence classes $E_1^B = \{x_1, x_2, x_7\}, E_2^B = \{x_3\}, E_3^B = \{x_4\}, E_4^B = \{x_5\}, E_5^B = \{x_6\}, E_6^B = \{x_8, x_9\}$. If we rank the classes according to size, we get the following ranking: $E_2^B \preceq E_3^B \preceq E_4^B \preceq E_5^B \preceq E_6^B \preceq E_1^B$. The model in Table 1 is created by including only classes with more than one member, and setting $\bar{d}_B$ to return the dominating decision in each class. As can be seen, the rough data model does not cover all the possible combinations of attribute values, and we therefore need a *fallback* decision value[2]. ♦

2.1. Rough Modeling as a Search Problem

The procedure of rough data modeling, that is, the search for a rough data model that maximizes a certain performance measure, is an optimization problem which can be summarized by the following steps:

1. Formulate a numerical measure for assessing the quality of each model. This measure will be a reflection of the goal of the data mining session. The function used to calculate the measure, $q_{\mathcal{M}} : \mathcal{M} \rightarrow \mathbb{R}$, will serve as the quality function for the optimization problem.

[2]A default decision value assigned to all objects which match none of the patterns in the model.

2. Determine the parameters of the search:
   (a) Set the search space of the search S, a starting point s_0, and a successor function $\sigma : S \to S$.
   (b) Determine the nature of the goal test. Since this is an optimization problem, no path-cost function is needed.
   (c) Decide on a search strategy.
3. Find a decision function $\bar{d}$, which maps all the objects in each equivalence class to a decision value $d \in V_d$.
4. Define a ranking relation to be used to rank the equivalence classes, $\preceq$.
5. Explore the search space, by generating a rough model at each node, and evaluate it using the $q_{\mathcal{M}}$ function. Use the successor function to move between one node and the next.

The search space S for a rough model search consists of a set of states S, where each state $s \in S$ corresponds to a model $\mathcal{M}_s$. When evaluating the quality of a state, $q(s) = q_{\mathcal{M}}(\mathcal{M}_s)$. As the characteristics used to rank each class are generated using only the objects in the particular class (Kowalczyk, 1998), and it is possible to find the equivalence classes E^B by a single run through the data set, the complexity of rough data modeling is linear with respect to the number of objects in the data set. The size of the search space is decided on by the user-specified upper and lower bounds on how many attributes to include in the model, and the worst-case expected complexity is thus

$$T(n) = \sum_{m=\text{min\_attribs}}^{\text{max\_attribs}} \frac{k!}{m!(k-m)!} O(n). \tag{3}$$

Since the search will find a model which optimizes the objective function, formulating an objective function amounts to formulating the goal of the data mining session. Depending on the objective function, a subset of the classes in a model may be selected, or the whole set of classess may be used. Some examples of different objective functions and model pruning schemes are found in (Kowalczyk, 1998).

Originally, rough data modeling employed only a simple exhaustive search through all possible models, and a natural extension is to investigate the use of different search strategies in order to search more effectively (for an overview, see, e.g. (Russell and Norvig, 1995)). However, the nature of the problem makes some strategies better suited than others. Using algorithms which rely on heuristics to reduce the size of the explored search space is difficult, for several reasons. First of all, it is impossible to know how good the best model is until it is found, and although the perfect model (no misclassifications) could be used as an approximation, this works only when searching for predictive models, and will likely result in overfitting unless additional steps are taken. At a given node in the search space, it is difficult to decide how to increase the quality of the model (which attributes to include), and controlling

the search is thus difficult. As illustrated in the study of attribute selection criteria by Imam (1996), the attributes reported as relevant by the investigated criteria did not turn out to be relevant in the learning system (a decision tree induced by the C4.5 algorithm). Since the search space is non-continuous (a collection of points corresponding to different attribute subsets), iterative improvement strategies such as hill-climbing and simulated annealing are ill-suited to the task.

Whilst ill-suited to conventional heuristics and iterative improvement algorithms, our experience has shown that the problem of searching for a rough model is well-suited to the mechanisms of a simple genetic algorithm. Genetic algorithms (Goldberg, 1989) solve optimization problems not by searching for a single optimal solution, but by gradually improving from a large number of starting points, with random elements, to ensure that the evolution does not get trapped in a dead end. This makes the search more robust and less likely to get stuck in a local maximum when better solutions exist. In our case, each individual corresponds to a specific rough model, and each gene indicates the inclusion/non-inclusion of an attribute. Thus, each individual has $|A|$ genes, and the value of gene i indicates whether attribute number i is included in the model or not. The *fitness function* is used to measure the quality of a given individual. In our case, the quality of an individual corresponds to the quality of the model which the individual represents. This can be done using any number of quality functions (accuracy, AUC, model size, etc.), or weighted combinations of several quality measures.

In a basic genetic algorithm, such as the one used in our experiments, the initial population is randomly generated, and each subsequent generation is the result of applying three basic genetic operations: *reproduction*, *crossover* and *mutation*. The algorithm halts once the goal test is fulfilled. The criteria for this may vary, but they usually involve tracking changes in a selected portion of the population (we used average fitness of the 10 best individuals), and halting once there are no changes over a set number of generations (10–15 generations).

As a genetic algorithm is indeterministic, it is impossible to give meaningful estimates for the running time of the algorithm. It all depends on the nature of the search problem, the formulation of the goal test, and the basic genetic parameters (size of population, frequency of mutation, etc.). Our observed results are presented in Section 4.1.

3. Other Rough Models

In earlier research (Ågotnes *et al.*, 1999b), we have investigated various extensions to the rough data modeling method introduced by Kowalczyk and presented above. In order to increase flexibility, two of the restrictions of rough data modeling are relaxed: the nature of the object decision function $\bar{d}_B$ and the partitioning of the universe into equivalence classes.

What we will refer to as a rough model consists of an attribute subset $B \subseteq A$ and a set of object classes E^B (not necessarily equivalence classes) which cover the universe U, with no restrictions on how these classes are found. In addition, we

replace the class decision function $\bar{d}_B : E^B \to V_d$ with an *object decision function*, $\hat{d}_B : U \to V_d$, which operates on each object rather than on an entire object class. This allows us the added flexibility of being able to assign different decision values to objects which belong to the same decision class, if we so desire.

For a decision system $\mathcal{A} = (U, A, \{d\})$, a rough model is a quadruple

$$\mathcal{M} = \langle B, \hat{d}_B, E^B, \preceq \rangle, \tag{4}$$

where $B \subseteq A$ is a set of attributes, $\hat{d}_B$ is an object decision function, E^B is a set of object classes which cover the universe U, $\preceq$ is a linear ordering on the classes in E^B. The rule generation process will depend on the nature of the classes E^B, and the relation used to divide the universe of objects. Some examples will be given below.

3.1. Partitioning the Universe

Within the rough modeling framework, it is possible to define many kinds of models, simply by varying the way in which we generate the object classes E^B, and the nature of the function $\hat{d}_B$. Rough data models are a kind of rough models where $E^B = U/\mathrm{IND}(B)$, using classical discernibility (inequality of attribute values), and $\hat{d}_B$ assigns the same value to all objects in each object class.

Equivalence classes are usually found using the notion of indiscernibility. The notion of indiscernibility is tied to the *discerns*/3 predicate, which states which objects are discernible from each other. The classical definition of this predicate uses equality, an attribute a_i discerns between two objects x_1 and x_2 if $a_i(x_1) \neq a_i(x_2)$. This creates a set of equivalence classes E^B where the objects in each class have identical values for each attribute in B. There are, however, different ways of defining the *discerns*/3 predicate, and we can use *similarity* instead; two objects x_1 and x_2 are indiscernible from each other using attribute a_i if $a_i(x_1)$ is similar to $a_i(x_2)$. The semantics behind similarity may vary across different domains and attribute types, but for a numerical attribute a_i similarity usually means that $|a_i(x_1) - a_i(x_2)| < r_i$. Since the similarity relation is not transitive, it is not an equivalence relation, and the resulting classes will overlap. The classes *cover* the universe U, but do not *partition* it as the classical indiscernibility relation does. Each class E_i^B will yield a single rule of the form

$$(x \in E^B) \to d(x) = \hat{d}_B(x_i), \tag{5}$$

where x_i is the object representing similarity region E_i^B. For example, $\hat{d}_B$ could be set to return the dominating decision on class E_i^B.

Another possible variation are *rough dominance models*, where the dominance relation, introduced by (Greco *et al.*, 1998), is used to order the value domains for each attribute, including the decision attribute. While those authors use the relation to perform multi-criteria decision analysis, we will simply use it as a different way to partition our universe. The dominance relation allows the creation of a partial order on the universe of objects, meaning that the objects can be organized in a *lattice*. If an object is above another object in the lattice, it is said to *dominate* this object. The

ordinal properties of the attribute value domains make it possible to generate rules of the form $a_1(x) > v_1 \wedge \ldots \wedge a_n(x) > v_n \rightarrow d(x) = v_d$, which are both more compact and more general than normal decision rules.

Generating a rough dominance model is done as follows: Select an attribute subset $B \subseteq A$ and use the dominance relation to partition the universe. The dominance relation will create a lattice which is smaller than the lattice created using all the attributes in A, but it is still possible to trace the region containing the objects with a specific decision value.

When generating rules, we can do it directly from the rough dominance model, in the same manner as one generates decision rules using the dominance relation, as explained by Greco *et al.* (1998). There may be inconsistencies of the kind shown in the following example, and these may be ignored (producing an inconsistent model). Or, we may try to create a new, deterministic, decision system. This means finding a set of classess within the lattice that do not overlap, and assigning them a decision. How this is done is up to the user and the goals of the data mining session, and several different decision functions, may be used in analogy with rough data modeling. If we let Δ_i^j denote the interval for attribute j to be within, in order to belong to class i, then each rule looks as follows:

$$a_1(x) \in \Delta_i^1 \wedge \ldots \wedge a_n(x) \in \Delta_i^n \rightarrow d(x) = \bar{d}_B(C_i). \tag{6}$$

Example 2. In Fig. 1, two rough dominance models of the decision system presented in Example 1 are shown. As can be seen, the dominance relation has been defined in such a way that for $a_i \in A$, $a_i(x_1)$ dominates $a_i(x_2)$ if $a_i(x_1) < a_i(x_2)$.

As can also be seen, the lattice also says something about the ordering of all the possible variations of attribute values, not only the known examples (represented by black dots). If V_d is also ordered, this enables us to spot inconsistencies such as the fact that $d(x_6) = 0$, even when x_6 is dominated by several objects with $d = 1$ (x_3, x_4).

If we want to create a consistent decision system, we may let $\hat{d}_B(x_6) = 1$, and let the remaining decision values stay untouched. This means that our rough dominance model is

$$X(x) \in \{1\} \wedge Y(x) \in \{1, 2\} \rightarrow d(x) = 0, \tag{7}$$

$$X(x) \in \{2, \ldots, 4\} \wedge Y(x) \in \{1, \ldots, 3\} \rightarrow d(x) = 1. \tag{8}$$

It is evident that only attribute X is necessary to distinguish between the two object classes introduced. ♦

This simple example shows that when compared with simple equality-based discernibility, rough dominance models offer two main benefits: They are more compact due to the increased expressive power of the syntax. They are also more general—the dominance relation generates rules which also apply to objects not found in the learning set, because of the partial ordering of the universe of objects. In addition, the same ordering also allows the detection of inconsistencies in the learning set.

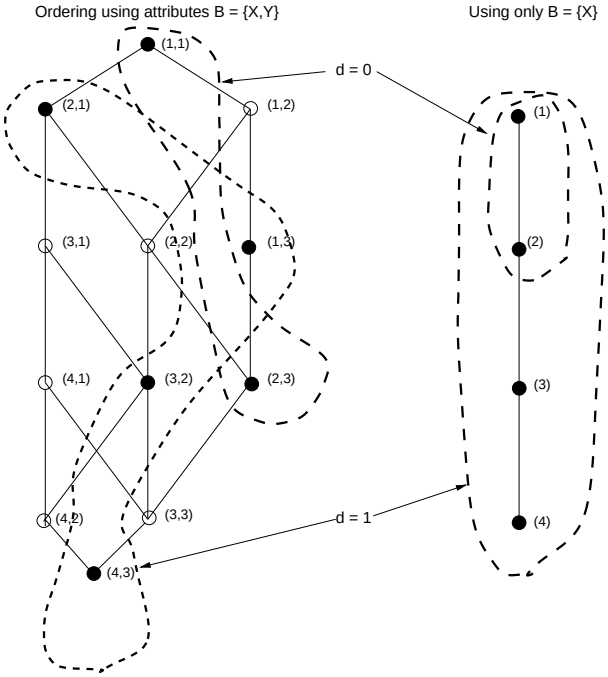

Fig. 1. Example rough dominance model. The black dots are objects which exist in the decision table; the white dots are objects for which no decision is known.

3.2. Bottom-Up Model Construction

When generating decision rules using the concept of a reduct, the starting point is the entire decision table. Redundant information (in the form of attributes) is removed from the table, and once no more information can be discarded, the reduct(s) remain, and rules are generated from these. In a rough modeling approach, one starts at the bottom. The question here is: "How much information do I need in order to reasonably represent the knowledge stored in this table?" What is meant by the phrase "reasonably represent" may vary, but a good classifier score on unseen data may be a good indication. Information (in the form of attributes) is added to the model, and the best possible model using that information is found. As the model reaches an acceptable quality, or the level of information approaches a predefined limit, a model which represents the knowledge in the table to a reasonable degree has been found.

As Fig. 2 illustrates, rough modeling represents a bottom-up approach to knowledge discovery versus the top-down approach used in reduct computation. While the rough model search is specifically limited to the lower part of the search tree (models with few attributes), the search for reducts spans the entire tree. While some reducts may indeed be within the rough model search area, practical experiments have shown that they often contain more attributes than the rough model search space. Experi-

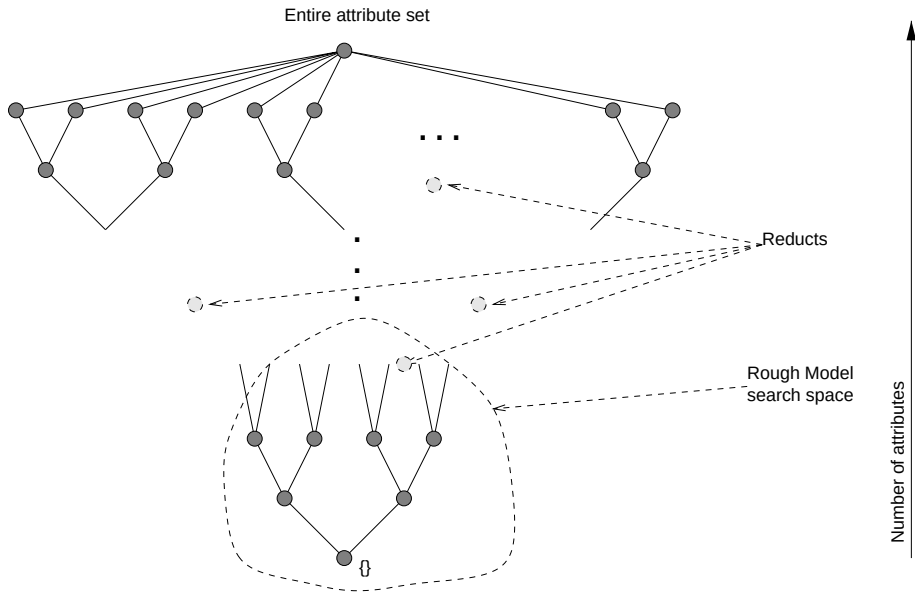

Fig. 2. Rough modeling vs. reduct search. The search space for rough models is explicitly limited to the models with few attributes.

ence has shown that models with large classess generalize better, and it is therefore recommended that a relatively small subset of attributes (2–5 is an often-mentioned figure) is used to generate the rough model.

This means that the rough modeling approach will not result in a perfectly fitting model. However, models which represent their data very well will often be *overfitted*, representing relationships found only in that particular data set, often due to chance. Such models do not perform well when classifying new cases, as their specific and detailed patterns are not found in the new data. The results reported among others by Holte (1993), Kohavi and Frasca (1994), and Mollestad and Komorowski (1998) suggest that simple rules formed using few attributes tend to perform better when classifying unseen objects.

4. Evaluating Rough Modeling

Several variations of the rough modeling approach have been implemented in the ROSETTA system (hrn *et al.*, 1998), which is a system for analyzing data using rough set methods. It was developed as a joint effort of the Institute of Mathematics at the University of Warsaw and the Knowledge Systems Group at the Norwegian University of Science and Technology (NTNU). A public version is available from the ROSETTA home page, (hrn, 2001). Several data sets were analyzed using rough modeling, as well as traditional rough set methods for comparison. All results shown below were computed using a genetic algorithm for rough model search.

When evaluating the results of the experiments, we have used *Receiver Operating Characteristics*, ROC, as the quality measure instead of classification accuracy, as is commonly used. For an in-depth explanation of why accuracy should be discarded as a measure of classifier performance, see (Provost *et al.*, 1998). In order to obtain a simple numerical measure of a classifier's performance, we have used the area under the ROC curve (AUC).

4.1. Experiment Results

Rough models were generated from the Pima Indian diabetes data set (Blake *et al.*, 1998) and the acute appendicitis data set (Hallan *et al.*, 1997b). The diabetes data, being part of the UCI machine learning repository, have been used in a large number of published studies. Kohavi and Sommerfield (1995) analyzed the data using a variety of algorithms, and reported accuracies of between 0.68 and 0.76. No published AUC values for the data are known. The acute appendicitis data set has been analyzed using logistic regression (Hallan *et al.*, 1997a; 1997b), obtaining AUC values of 0.920. Using dynamic reduct computation (Bazan *et al.*, 1994, Carlin *et al.*, 1998) analyzed the data set, and reported a mean AUC value of 0.923 (and standard deviation of 0.023). Details, as well as a detailed description of the data set, including the discretization used for the numerical attributes, can be found in (Carlin *et al.*, 1998).

In order to investigate the effect of excluding the smallest equivalence classes from the rough data models, the size threshold for inclusion into the rough model was varied, and a rough model generated from 67% of the objects in the data set. The resulting model was used to classify the test set, consisting of 33% of the objects, and the AUC value was calculated using the trapezoidal method of approximating integrals. The results on the acute appendicitis and diabetes data are shown in Table 2.

In addition to a decision function $\hat{d}_B$ which sets the decision value for each object to the dominating decision value for the equivalence class of that object, a decision function which copies the original decision value for each object was implemented. This produces a model which may contain indeterministic rules. The performance of the models generated when keeping the original decision values is also shown in Table 2. The same splits as for the dominating decision results were used.

No precise observations of the running time were recorded; we did, however, observe that the search for rough models is faster than the rough set methods used here (the difference varies for different parameters and different sessions). The genetic algorithm consistently returned models of equal quality to the models returned from an exhaustive search, in usually 20–30% of the time. The search time was further improved by introducing a cut-off on the size of the classes E_i included in the model, which had the added benefit of increasing model performance.

4.2. Analysis

The results on the appendicitis data set indicate that the performance of the rough data models is somewhat poorer than that of the optimal model generated using traditional rough set methods. In order to investigate this further, the Hanley-McNeil

Table 2. Results from experiments carried out on the appendicitis and diabetes data. The RS model is the best reduct-based model reported by Carlin *et al.* (1998).

	Size	**AUC (SD)**			
Model / Data set	**limit**	**Split 1**	**Split 2**	**Split 3**	**Mean**
RS model	N/A	0.923 (0.028)	0.891 (0.031)	0.908 (0.031)	**0.907 (0.031)**
RDM Dominating Appendicitis	0	0.804 (0.047)	0.768 (0.050)	0.776 (0.050)	**0.783 (0.049)**
	5	0.856 (0.040)	0.807 (0.046)	0.842 (0.042)	**0.835 (0.043)**
	10	0.850 (0.041)	0.834 (0.043)	0.765 (0.051)	**0.816 (0.045)**
	15	0.710 (0.056)	0.795 (0.048)	0.807 (0.046)	**0.771 (0.050)**
RDM Original Appendicitis	0	0.829 (0.044)	0.764 (0.051)	0.790 (0.048)	**0.794 (0.048)**
	5	0.891 (0.034)	0.887 (0.035)	0.914 (0.030)	**0.897 (0.033)**
	10	0.914 (0.030)	0.792 (0.048)	0.866 (0.038)	**0.857 (0.039)**
	15	0.813 (0.046)	0.806 (0.046)	0.799 (0.048)	**0.806 (0.047)**
RDM Dominating Diabetes	0	0.678 (0.038)	0.708 (0.035)	0.691 (0.035)	**0.692 (0.036)**
	5	0.666 (0.038)	0.728 (0.034)	0.689 (0.036)	**0.694 (0.036)**
	10	0.702 (0.037)	0.784 (0.032)	0.712 (0.034)	**0.732 (0.035)**
	15	0.690 (0.037)	0.733 (0.034)	0.721 (0.034)	**0.715 (0.035)**
RDM Original Diabetes	0	0.726 (0.036)	0.741 (0.034)	0.778 (0.034)	**0.748 (0.034)**
	5	0.754 (0.035)	0.774 (0.032)	0.664 (0.036)	**0.730 (0.034)**
	10	0.768 (0.034)	0.762 (0.033)	0.760 (0.033)	**0.763 (0.033)**
	15	0.771 (0.034)	0.779 (0.032)	0.750 (0.033)	**0.767 (0.033)**

test for comparing correlated AUC values (Hanley and McNeil, 1982) was used. This was only done for the appendicitis data, as no known benchmark existed for the diabetes data. The p-values from the Hanley-McNeil test are shown in Table 3.

On the whole, the results from the appendicitis data indicate that the difference in performance is not significant if the correct threshold for class size is selected. In order to briefly examine the increase in descriptiveness, Table 4 lists the size of the different rough models for the appendicitis data set.

While it is certainly not impossible to systematically analyze and investigate a model containing more than 800 rules, it is undoubtedly a daunting task. On the other hand, a model containing 10–15 rules, where each rule uses 3–4 attributes (since the search is explicitly limited to models of this size), is easily inspected. In addition, by accepting the slight performance drop from using dominating decision functions, all the rules in the rough model will be deterministic (have a consequent consisting of a single indicated decision value). On all other data sets examined, the sizes of the models found by a rough model search were comparable to the results reported for the appendicitis set (between 5 and 20 rules, if a small cut-off on class size is used).

Table 3. Statistical analysis of the results on the appendicitis data. The AUC values for the models using the dominating and original decision function, listed in Table 2, were compared with the AUC value of the best rule set induced using rough set methods.

		Size limit			
	Decision function	**0**	**5**	**10**	**15**
Split 1	dominating	0.0180	0.0972	0.0775	0.0001
	original	0.0414	0.3584	0.7668	0.0065
Split 2	dominating	0.0154	0.0899	0.1570	0.0528
	original	0.0156	0.9167	0.0375	0.0483
Split 3	dominating	0.0100	0.0906	0.0112	0.0122
	original	0.0181	0.8677	0.2498	0.0122

Table 4. The number of rules in each RDM with varying size limit. The number of rules in the best RS rule set for the particular split is included for comparison.

	# of rules in the model		
Size limit	**Split 1**	**Split 2**	**Split 3**
RS model	893	872	851
0	107	70	77
5	17	18	17
10	9	10	8
15	5	5	5

5. Conclusions

The results obtained so far are not a foundation solid enough to make strong claims about the performance of rough models versus models mined using traditional rough set methods on. There is some indication that the performance of rough models falls slightly short of the performance of larger rough set induced models, but the evidence is not conclusive. However, there is overwhelming evidence supporting the conjecture that rough data models of performance comparable to traditional RS models are far more descriptive. No universally agreed upon measure of descriptiveness exists, but a decrease in the size from thousands of rules to between five and twenty, the use of few attributes, and the option of determinism add up to a very large improvement in descriptive capability.

It is interesting to note that while we found that rough models performed no better than the RS-based models, Kowalczyk (1998) reported that rough data models performed better than models generated using other approaches, including rough set

methods. There are a lot of possible reasons: Inherent differences in the data sets examined, his use of accuracy versus our use of ROC, as well as a lack of established well-documented benchmarks, all contribute to making comparisons between learning algorithms and uncertain science.

While the ideas of rough dominance models look promising in their expressive power, it is difficult to find domains which are well-suited to these kinds of models, and comparing the results obtained to other results is also difficult, since using the dominance relation will imply making several additional assumptions about the domain and data. It should be observed that any rule set obtained from a rough dominance model is easily transformed into an ordinary rough data model, with a single rule for each combination of attribute values. This means that rough dominance models mainly offer a more compact and thus more readable model, as well as the ability to generate rules for value combinations not found in the learning data, and the ability to detect inconsistencies in the learning data.

It is somewhat unfair to compare unfiltered models to rough models created from reducts. Even very simple filtering strategies will reduce the number of rules significantly without sacrificing performance. Ågotnes *et al.* (1999b) reported results from heavily filtered rule sets which are comparable to the results achieved using rough modeling, with rule sets containing 6–10 rules. While the results are similar, the time used to first approximate reducts and then employ one or more filtering strategies, is far greater than the time spent on searching for rough models.

The simplicity of rough model generation means that rough modeling is well-suited to large databases, and as an initial analysis approach. Models which are of a high descriptive and predictive quality may be generated quickly. The insight gained from inspecting the rough models may then be used to guide a more complex KDD process in order to achieve the desired results.

Acknowledgements

Aleksander hrn was very helpful with questions concerning ROSETTA and the implementation of rough modeling. Jan Komorowski's work was in part supported by a grant from the Norwegian Research Council.

References

Ågotnes T., Komorowski J. and hrn A. (1999a): *Finding small high performance subsets of induced rule sets.* — Proc. 7-th Europ. Congress *Intelligent Techniques and Soft Computing, EUFIT'99*, Aachen, Germany, p.174.

Ågotnes T., Komorowski J. and Lken T. (1999b): *Taming large rule models in rough set approaches*, In: Principles of Data Mining and Knowledge Discovery (J.M. Zytkow and J. Rauch, Eds.). — Berlin: Springer-Verlag.

Bazan J.G., Skowron A. and Synak P. (1994): *Dynamic reducts as a tool for extracting laws from decision tables*, In: Methodologies for Intelligent Systems (Z.W. Raś and M. Zemankova, Eds.). — New York: Springer, pp.346–355.

Blake C., Keogh E. and Merz C.J. (1998): *UCI repository of machine learning databases.* — Available at: `http://www.ics.uci.edu/~mlearn/MLRepository.html`.

Carlin U., Komorowski J. and hrn A. (1998): *Rough set analysis of patients with suspected acute appendicitis.* — Proc. 7-th Conf. *Information Processing and Management of Uncertainty in Knowledge-Based Systems, IPMU'98*, Paris, France, pp.1528–1533.

Goldberg D.E. (1989): *Genetic Algorithms in Search, Optimization, and Machine Learning.* — New York: Addison-Wesley.

Greco S., Matarazzo B. and Słowiński R. (1998): *New developments in the rough set approach to multi-attribute decision analysis.* — Bull. Int. Rough Set Soc., Vol.2, No.2/3, pp.57–87.

Hallan S., sberg A. and Edna T.-H. (1997a): *Additional value of biochemical tests in suspected acute appendicitis.* — Europ. J. Surgery, Vol.163, No.7, pp.533–538.

Hallan S., sberg A. and Edna T.-H. (1997b): *Estimating the probability of acute appendicitis using clinical criteria of a structured record sheet: The physician against the computer.* — Europ. J. Surgery, Vol.163, No.6, pp.427–432.

Hanley J.A. and McNeil B.J. (1982): *The meaning and use of the area under a receiver operating characteristic (roc) curve.* — Radiology, Vol.143, pp.29–36.

Holte R.C. (1993): *Very simple classification rules perform well on most commonly used datasets.* — Mach. Learn., Vol.11, pp.63–91.

hrn A. (2001): *The* ROSETTA *home page, 2001.* — `http://rosetta.sourceforge.net/` and `http://www.idi.ntnu.no/~aleks/rosetta/`.

hrn A., Komorowski J., Skowron A. and Synak P. (1998): *The design and implementation of a knowledge discovery toolkit based on rough sets—the* ROSETTA *system*, In: Rough Sets in Knowledge Discovery 1: Methodology and Applications. — Heidelberg: Physica-Verlag, Ch.19, pp.376–399.

Imam I.F. (1996): *An empirical study on the incompetence of attribute selection criteria*, In: Foundations of Intelligent Systems (Z.W. Raś and M. Michalewicz, Eds.). — Berlin: Springer, pp.458–467.

Klösgen W. (1996): *Knowledge discovery in databases and data mining*, In: Foundations of Intelligent Systems (Z.W. Raś and M. Michalewicz, Eds.). — Berlin: Springer, pp.623–632.

Kohavi R. and Frasca B. (1994): *Useful feature subsets and rough set reducts.* — Proc. 3-rd Int. Workshop *Rough Sets and Soft Computing (RSSC '94)*, San Jose, USA.

Kohavi R. and John G.H. (1997): *Wrappers for feature subset selection.* — Artif. Intell. J., Vol.97, No.1–2, pp.273–324.

Kohavi R. and Sommerfield D. (1995): *Feature subset selection using the wrapper method: Overfitting and dynamic search space topology.* — Proc. 1-st Int. Conf. *Knowledge Discovery and Data Mining, KDD'95.*

Kowalczyk W. (1998): *Rough data modelling: A new technique for analyzing data*, In: Rough Sets and Knowledge Discovery 1: Methodology and Applications. — Heidelberg: Physica-Verlag, Ch.20, pp.400–421.

Mienko R., Stefanowski J. and Vanderpooten D. (1996): *Discovery-oriented induction of decision rules*. — Tech. Rep., Cahier du Lamsade No. 141, Universite de Paris Dauphine.

Mollestad T. and Komorowski J. (1998): *A rough set framework for propositional default rules data mining*, In: Rough-Fuzzy Hybridization: A New Trend in Decision Making. — Berlin: Springer-Verlag.

Pawlak Z. (1991): *Rough Sets—Theoretical Aspects of Reasoning about Data*. — Dordrecht: Kluwer.

Provost F., Fawcett T. and Kohavi R. (1998): *The case against accuracy estimation for comparing induction algorithms*. — Proc. 15-th Int. Conf. *Machine Learning (ICML'98)*.

Russell S. and Norvig P. (1995): *Artificial Intelligence—A Modern Approach*. — Prentice-Hall.

Skowron A. and Rauszer C. (1991): *The discernibility matrices and functions in information systems*, In: Intelligent Decision Support Systems—Handbook of Applications and Advances in Rough Set Theory (R. Słowiński, Ed.). — Dordrecht: Kluwer, pp.331–362.