J. Ind. Soc. Agril. Statist.
62(1), 2008 : 75-84

Drawing Conclusions from Forest Cover Type Data — The Hybridized Rough Set Model

Rajni Jain and Sonajhania Minz[1]
National Centre for Agricultural Economics and Policy Research, New Delhi

(Received: July 2007)

SUMMARY

Classification is an important research topic in the field of data mining and knowledge discovery. There have been many data classification methods including decision tree methods, statistical methods, neural networks, rough sets, etc. A reduct in the rough set theory refers to a set of dominant attributes in the dataset. A dataset may have zero, one or multiple reducts. A classification problem utilizing information contained in a single reduct is well examined in rough set literature. However, it means ignoring the available knowledge from the multiple reducts. An approximate core is proposed as an important tool to deal with the datasets which are having multiple reducts. In this paper, Forest cover type, a large benchmarking dataset having multiple reducts is used for experiments. The performance parameters - accuracy, complexity, number of rules and number of attributes in the resulting classifiers are compared among various algorithms employed. The results using approximate core are comparable with the other published results for this dataset.

Key words: Rough sets, Decision tree, RDT model, Approximate core, Forest cover type.

1. INTRODUCTION

Classification is an important research topic in the field of data mining and knowledge discovery. It finds the common properties among a set of objects in a dataset and classifies them into different pre-identified classes. There have been many data classification methods including Decision Tree (DT) methods, statistical methods, neural networks, rough sets etc. (Han and Kamber 2001, Pawlak 1991, Murthy 1998, Witten and Frank 2000).

In machine learning studies, a decision tree classification method, developed by Quinlan (1993) has been influential. A typical decision tree learning system is ID-3 (Iterative Dichotmiser) which adopts a top-down irrevocable strategy that searches only part of the search space. C4.5 algorithm is an extension to ID-3 and extends the domain of classification from categorical attributes to numerical ones (Quinlan 1993). Shan *et al*. (1996) proposed a novel approach, which uses rough sets to ensure the completeness of the classification and the reliability of the probability estimate prior to rule induction. Minz and Jain (2003) proposed Rough set based Decision Tree (RDT) model which uses reducts to refine decision trees. Jain and Minz (2003) proposed approximate core as a tool to deal with a dataset having large number of correlated attributes. It facilitates less memory requirements for the subsequent steps of learning and classification. This paper further explores approximate core for handling a very large benchmarking dataset called Forest cover type (Murphy).

The Forest cover type data from the UCI repository is one of the large datasets containing 581012 examples. The 54 attributes of this dataset actually represent 12 features. The classification task is to predict the Forest cover type (7 classes) given only cartographic data. The actual forest cover type for a given observation (30 × 30 meter cell) was determined from US Forest Service (USFS) Region 2 Resource Information System (RIS) data. Independent variables were derived from data originally obtained from US Geological Survey (USGS) and USFS data. Data is in raw form (not scaled) and

[1] *Jawaharlal Nehru University, New Delhi-110067*

contains binary (0 or 1) columns of data for qualitative independent variables (wilderness areas and soil types). This dataset is a representative instance of a domain of problems common in data mining and thus provide a useful benchmark for comparing classifiers. There are no missing values in this dataset. The characteristics that make it of particular interest are - a large number of cases; a relatively large number of attributes and large number of target class values; an unequal class distribution (36.46, 48.76, 6.15, 0.47, 1.63, 2.99, 3.53) for the response classes; and both continuous and categorical variables.

This paper proceeds as follows. Section 2 deals with the essential preliminaries required to understand the paper. Section 3 presents the experimental details. Results are discussed in Section 4, followed by conclusions in Section 5.

2. PRELIMINARIES

Rough Set theory was introduced in early 1980s by Z. Pawlak and since has come into focus as alternative to the more widely used methods of machine learning and statistical data analysis (Pawlak 1991, Ziarko 1999, Witten *et al*. (2000)).

2.1 RDT Model

Rough Set theory (RS) offers a simplified search for dominant attributes in datasets, called reducts. Reducts are the optimal number of attributes which preserve the indiscernibility between the objects in the dataset. Reducts are further used by Minz and Jain (2003) in the Rough set-based Decision Tree (RDT) model for classification. RDT model combines merits of both RS and DT induction algorithm. Thus, it aims to improve efficiency, simplicity and generalization capabilities of both the base algorithms (Minz and Jain 2005).

Algorithm RDT

1. Input the training data set T1.
2. Discretize the numeric or continuous attributes and label the modified data set as T2.
3. Obtain the minimal decision relative reduct of T2, say R.
4. Reduce T2 based on reduct R and label reduced data set as T3.
5. Apply ID3 algorithm on T3 to induce decision tree.
6. If needed, convert decision tree to rules by traversing all the possible paths from root to each leaf node.

In step 3 of the RDT algorithm for computation of a minimal decision relative reduct Johnson's algorithm (1974) can be directly used (Pawlak 1991, Ohrn 1999). Alternatively one can generate population of reducts using Genetic Algorithm (GA) followed by random selection of a reduct from the population (Wroblewski 1995). Genetic algorithm gives a good approximation to a problem of multiple reduct computation with very little effort. This provides flexibility to the data miner for choosing the desired set of attributes in the induction of the decision tree. For this, the reducts may be ranked in terms of the cost of obtaining the values of the required attributes. The reduct having least cost are preferable for further steps (Ziarko 1999).

2.2 RDT Variants

Basic RDT model as discussed above is suitable for small datasets. RDT variants are first proposed by Minz and Jain (2005) for improving the suitability of RDT model for large datasets. RDT variants are produced by modifying at least one step of the RDT algorithm. For example, if smallest reduct is selected from the population of reducts the corresponding algorithm is referred as RDTGAsmallest. Similarly, replacing ID3 algorithm in step 5 by J4.8 (Witten and Frank 2000) algorithm - Java implementation of C4.5 algorithm produces a variant called RJU or RJP depending on whether decision tree is unpruned or pruned. Other RDT variants can be produced using approximate core discussed below.

2.3 Approximate Core

For datasets having large number of reducts, the core instead of reduct is useful (Pawlak 1991). Such a variation is motivated by the desire to analyze and identify data patterns which represent strong statistical trends rather than only strict ones identified by a reduct (Jain and Minz 2003). The classical definition of Core = $\bigcap_{i=1}^{i=n} R_i$ where R_i is the i^{th} reduct in the population of reduct of the information system. However, it is observed that for some domains, some of the relevant attributes may be missed by using this classical definition of core. In order to address this issue, *approximate core*, denoted by Core_α, instead of core may be used for relevant features of the information system. Let $\hat{R} = \{R_1, R_2, ..., R_n\}$ and $|\hat{R}| = n$. Let a set of attributes $\hat{A} = \bigcup_{i=1}^{n} R_i$ and $|\hat{A}| = m$. Consider a Boolean matrix C of the dimension m × n, where matrix element C_{ij} is a Boolean variable to indicate the presence of the jth attribute from $\hat{A}$ in the ith reduct. In other words

$$C_{ij} = \begin{Bmatrix} 0 & \text{if} & a_j \notin R_i \\ 1 & \text{if} & a_j \in R_i \end{Bmatrix} \quad j = 1 \ldots n,\ i = 1 \ldots m \tag{1}$$

Using matrix of (1), occurrences of all the attributes in $\hat{R}$ can be summarized by a characterization vector $\vec{R}$ as given below in (2). The purpose of the characterization vector is to indicate the importance or relevance of each attribute in $\hat{A}$ with respect to $\hat{R}$.

$$\vec{R} = (w_1 a_1 \ldots w_k a_k \ldots w_m a_m) \tag{2}$$

where $w_k = \frac{1}{n} \sum_{i=1}^{n} c_{ik}$ and $1 \le k \le m$

In equation (2), $0 < w_k \le 1$ is called the relative frequency of the attribute a_k with respect to $|\hat{R}|$. Let α denote the measure of the approximation of the core then using (2), approximate core with the value α' can be expressed as

$$\text{Core}_{\alpha=\alpha'} = \{a_k \mid a_k \in \hat{A}\} \tag{3}$$

where $\alpha' = \min_{k=1}^{l}(w_k)$ iff $|\text{Core}_{\alpha=\alpha'}| = l$

Approximate core with approximation level of α', denoted by $\text{Core}_{\alpha=\alpha'}$, consists of all attributes which have at least a relative frequency of value α'. If α' is one then approximate core is same as the core. The parameter α determines the degree of the accuracy of the approximate core relative to the classical core. Use of approximate core allows identifying relevant attributes that would be missed if only the classical definition of core were used for the purpose. An algorithm, Compute CoreAlpha, is given below for the computation of approximate core for a given threshold of performance, followed by description of some of the identifiers used in the algorithm.

Algorithm ComputeCoreAlpha

Input SortAttrStack, SortAttrRelFreqStack, ValidationData, TrainingData, SmallestReductLength, Core, PerformanceThreshold

Output CoreAlpha, Alpha

Method

```
1. CoreAlpha = Core; i = 0; Alpha = 1
2. Do steps 3-7 while
   |CoreAlpha| < Smallest Reduct Length
3. InduceRDTclassifier(CoreAlpha, TrainingData)
4. CS = ComputePerformance(DTclassifier,
   ValidationData)
5. If CS>PerformanceThreshold then ArrayCS [i]
   = CS; ArrayCoreAlpha[i] = CoreAlpha;
   ArrayAlpha[i] = alpha; i = i + 1
6. a=POP(SortAttrStack); Add a to CoreAlpha
7. Alpha=POP(SortAttrRelFreqStack)
8. Compute index corresponding to maximum
   value in ArrayCS.
9. Return ArrayCoreAlpha[index],
   ArrayAlpha[index]
```

The relative frequencies w_i are sorted in the decreasing order and stored in the SortAttrRelFreqStack. SortAttrStack stores attribute elements from $\hat{A}$ corresponding the elements of the stack SortAttrRelFreqStack so that the attribute corresponding the maximum w_i is pointed to by the top and the following ones corresponding the decreasing values of w_i in the order of increasing value of w_i from bottom to top of the stack. PerformanceThreshold is the input value of performance in order to ensure the acceptable performance of the final classifier from the approximate core. InduceRDTClassifier follows RDT algorithm by replacing the reduct by the CoreAlpha using the training data. Procedure ComputePerformance returns a common performance score after measuring performance parameters of induced DT on validation data. Procedure

Table 1. Learning schemes and their descriptions used for Forest cover type dataset

Algorithm	Desc4ription
RS	Classical Rough set approach with full discernibility decision relative reduct
CJU	Continuous data, J4.8 algorithm, Unpruned - Java implementation of Quinlan's C4.5 algorithm
CJP	Continuous data, J4.8 Algorithm, Pruned - Java implementation of Quinaln's C4.5 with pruning
RJU	Discretized, Filtered using Reducts, J4.8 unpruned
RJP	Discretized, Filtered using Reducts, J4.8 pruned
RDTGA-smallest	Discretized, Filtered using smallest reduct selected from the population of reducts from GA, ID3
RJUGA-smallest	Discretized, Filtered using smallest reduct selected from the population of reducts from GA, J4.8 unpruned
RJPGA-smallest	Discretized, Filtered using smallest reduct selected from the population of reducts from GA, J4.8 pruned
RJUcore$_{\propto : c}$	Discretized, filtered using approx. core with $\alpha \geq c$, J4.8 unpruned
RJPcore$_{\propto : c}$	Discretized, filtered using approx. core with $\alpha \geq c$, J4.8 pruned

ComputePerformance may be defined by the user as per preferences among the performance parameters (Minz and Jain 2005). Use of approximate core in RDT algorithm generates RDT variants namely RDTcore$_\alpha$, RJUcore$_\alpha$ and RJPcore$_\alpha$.

3. EXPERIMENTAL DETAILS

Some of the previously reported experiments with Forest cover type dataset have used first 11340 records for training data, the next 3780 records for validation data and last 565892 records for testing data (Blackard and Dean 1999). The data were so arranged that the class frequencies are equal in training and validation data (thus significantly reducing the number of rare classes in the testing data). Balancing the data by sampling such that classes are equally represented in the training data, can make the final accuracy measure unrepresentative of the true model, therefore balancing of the class values was not done for experiments in this study. In (Bagnall and Cawley 2003) the experiments were carried out by modifying the classification task from predicting all 7 cover types to only majority cover type identification, thus reducing the problem to binary class identification. RDT and its variants (Table 1) can be directly applied to classification problems having multiple class values; hence no such modifications are required for experiments with RDT.

For experiments in this study, the dataset was randomly split into two parts (one for sampling training data and the other for sampling test data) using SPSS software. Five random samples for training data and five random samples for test data are drawn using stratified random sampling from the corresponding part such that distribution of the class values is nearly the same. Each training sample (total 5) is randomly associated to one of the test sample out of 5. This is referred as train-test pair or a sample for further reference in the chapter. The size and the distribution of the class values for each of the train-test pair is kept same as in the original data (Table 2).

The data has numeric attributes which requires discretization for employing rough sets. Discretization of continuous attribute values is done by using the Fayyad and Irani (1993) method. RS, CJU (C4.5 without pruning), CJP (C4.5 with pruning), RJU and RJP algorithms are explored for classification. On applying GA based algorithm to the training sample of this dataset, a population of reducts (RED) is obtained. To select the most suitable attributes from multiple reducts, RJUGA-smallest, RJPGA-smallest, RJUcore$_\alpha$ and RJPcore$_\alpha$ are also employed for learning from this dataset (Table 1).

Tuning of parameters of algorithms is identified as an important step in the classification experiments. For example, the parameter M (the minimum number of leaves allowed in a leaf of the DT) is required to be tuned so as to avoid any output variations because of its variability. In this dataset as considerable variations in accuracy are observed with the change in values of the parameter M, hence parameter M was tuned. Due to the large number of reducts from this dataset, parameter

Table 2. Size of samples and the distribution of their class values for cover type dataset

Sample pairs	Number of records		Class Value	Class value distribution			
				Training		Test	
	Training	Test		Records	%	Records	%
1	29024	29051	1	10573	36.43	10526	36.23
			2	14164	48.80	14259	49.08
			3	1773	6.11	1750	6.02
			4	125	0.43	123	0.42
			5	487	1.68	468	1.61
			6	878	3.03	864	2.97
			7	1024	3.53	1061	3.65
2	28760	28961	1	10412	36.20	10565	36.48
			2	14073	48.93	14127	48.78
			3	1737	6.04	1760	6.08
			4	146	0.51	151	0.52
			5	486	1.69	501	1.73
			6	857	2.98	827	2.86
			7	1049	3.65	1030	3.56
3	28789	29206	1	10444	36.28	10703	36.65
			2	14129	49.08	14262	48.83
			3	1766	6.13	1725	5.91
			4	123	0.43	124	0.42
			5	442	1.54	472	1.62
			6	851	2.96	910	3.12
			7	1034	3.59	1010	3.46
4	29207	28928	1	10643	36.44	10509	36.33
			2	14219	48.68	14065	48.62
			3	1811	6.20	1779	6.15
			4	144	0.49	138	0.48
			5	466	1.60	518	1.79
			6	858	2.94	857	2.96
			7	1066	3.65	1062	3.67
5	29056	28934	1	10753	37.01	10669	36.87
			2	14081	48.46	14022	48.46
			3	1744	6.00	1776	6.14
			4	126	0.43	134	0.46
			5	468	1.61	469	1.62
			6	869	2.99	821	2.84
			7	1015	3.49	1043	3.60

tuning for each reduct will be very time consuming. Thus, instead of taking each reduct in turn for tuning of parameters in classification algorithms, a tool, called Maximum Possible Combined Reduct (MPCR), representing attributes from all reducts is expected to be useful and convenient for implementation. For the experiments, MPCR is computed from training data of Sample-1. However, not much variation is expected even by choosing MPCR from any other sample because any attribute that belongs to at least one of the reducts in the population of reducts from GA (Table 3) also belongs to MPCR. Note that MPCR is referred as $\hat{A}$ in Section 2. For example, MPCR for the Sample-1 is {1-14,16, 18-20, 23-28, 30, 31, 33-38, 41, 43-47, 49, 52-54}

Table 3. Frequencies of attributes in population of reducts from 5 randomly selected samples of training data for approximate core identification

Training set 1		Training set 2		Training set 3		Training set 4		Training set 5	
#1	w1	#2	w2	#3	w3	#4	w4	#5	w5
1	1.00	1	1.00	1	1.00	1	1.00	1	1.00
2	1.00	2	1.00	2	1.00	2	1.00	2	1.00
3	1.00	3	1.00	3	1.00	3	1.00	3	1.00
4	1.00	4	1.00	4	1.00	4	1.00	4	1.00
5	1.00	5	1.00	5	1.00	5	1.00	5	1.00
6	1.00	6	1.00	6	1.00	6	1.00	6	1.00
7	1.00	7	1.00	7	1.00	7	1.00	7	1.00
8	1.00	8	1.00	8	1.00	8	1.00	8	1.00
9	1.00	9	1.00	9	1.00	9	1.00	9	1.00
10	1.00	10	1.00	10	1.00	10	1.00	10	1.00
37	0.99	52	0.99	26	1.00	24	1.00	46	1.00
45	0.96	25	0.97	43	0.97	43	0.99	47	0.99
44	0.96	26	0.97	20	0.97	36	0.98	25	0.98
52	0.96	43	0.97	30	0.96	52	0.98	37	0.98
34	0.95	45	0.97	36	0.96	47	0.97	34	0.97
36	0.96	46	0.97	37	0.96	26	0.97	24	0.96
46	0.96	36	0.96	53	0.95	34	0.97	43	0.96
47	0.94	47	0.96	24	0.93	25	0.95	26	0.93
30	0.93	37	0.95	45	0.93	37	0.95	13	0.90
43	0.93	53	0.95	16	0.90	38	0.91	36	0.89
26	0.91	28	0.92	34	0.90	45	0.91	33	0.87
16	0.89	44	0.92	46	0.90	46	0.91	38	0.84
24	0.89	24	0.91	38	0.89	16	0.88	18	0.82
20	0.87	16	0.90	27	0.87	33	0.85	20	0.82
33	0.86	19	0.90	44	0.87	30	0.72	44	0.82
27	0.83	31	0.90	33	0.84	14	0.71	19	0.82
13	0.80	34	0.89	52	0.82	27	0.66	52	0.81
25	0.79	38	0.88	13	0.80	11	0.65	53	0.81
38	0.78	13	0.86	49	0.80	35	0.65	45	0.80
28	0.77	33	0.81	47	0.75	44	0.65	30	0.79
23	0.76	54	0.81	28	0.70	13	0.61	40	0.75
11	0.75	27	0.80	31	0.70	12	0.60	31	0.71
12	0.71	11	0.79	19	0.65	31	0.54	11	0.70
31	0.67	12	0.75	25	0.63	28	0.53	15	0.68
53	0.62	18	0.74	11	0.61	40	0.47	12	0.60
41	0.58	20	0.62	12	0.59	53	0.47	27	0.55
14	0.56	48	0.60	40	0.55	19	0.42	54	0.54
35	0.54	30	0.57	18	0.40	41	0.42	35	0.53
54	0.51	49	0.50	17	0.30	18	0.22	49	0.51
18	0.40	35	0.48	54	0.28	20	0.22	28	0.51
19	0.39	15	0.43	14	0.09			14	0.41
49	0.31	17	0.41					17	0.32
		14	0.22					16	0.31

(Table 3). Due to the nature of MPCR which is union of all reducts, the identified value of M = 1 is expected to be suitable for further experiments with any single reduct or core because the cardinality of core or a single reduct would be lower than the MPCR (Table 4).

Table 4. Tuning of parameter M using MPCR

M	Accuracy (%)	
	RJU	RJP
1	81.37	79.85
2	80.70	79.25
5	77.95	77.37
10	76.53	76.02
25	74.35	73.99

Table 5. Sizes and number of reducts from 5 samples

Id	# Red	$\|R_{smallest}\|$	$\|R_{Largest}\|$	$\|Core_{\alpha = 1}\|$
1	160	30	37	10
2	145	32	38	10
3	115	28	35	11
4	116	27	34	11
5	148	30	36	11

Size of a reduct in the population of reducts obtained using the training data varies from 27 to 37 (Table 5). Experiments using RJU algorithm show that training as well as test accuracy performance is stable among the reducts of varying sizes. This observation supports the selection of smallest reduct from each population for experiments with RDT variants (Fig. 3).

Further in quest of simpler classifier, approximate core is employed. The experiments are performed by varying values of parameter α in approximate core to observe the corresponding trend in the variation of the performance measures. However, the most suitable approximate core as produced by the algorithm ComputeCoreAlpha is used for comparison with other learning algorithms (Fig. 1, Fig. 2).

Performance parameters accuracy (fraction of test instances correctly classified), complexity (number of selectors), number of rules and number of attributes in the classifier are used for comparison of algorithms. Cumulative Score (CS) is used for ranking of algorithms (Minz and Jain 2005).

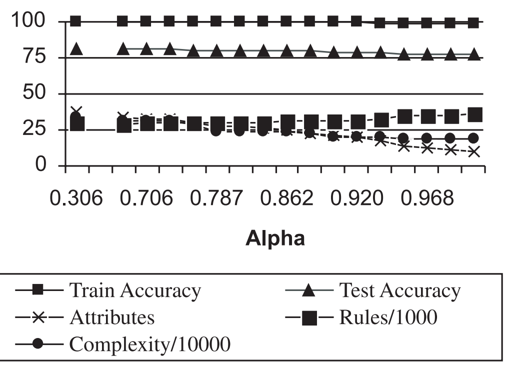

Fig. 1. Comparing performance values using RJUcore$_\alpha$ by varying α (alpha) of approx. core

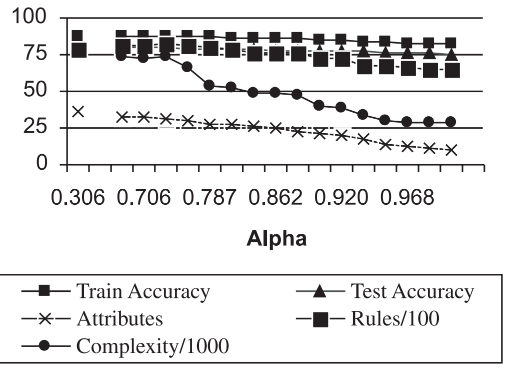

Fig. 2. Comparing performance values using RJPcore$_\alpha$ by varying α (alpha) of approx. core

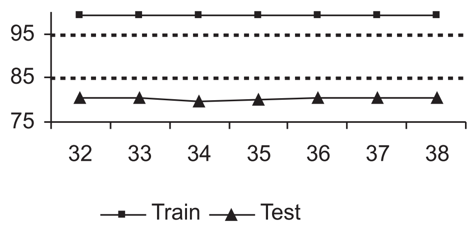

Fig. 3. Comparing accuracy values using RJU by using the reducts of varying cardinalities

4. RESULTS

The performance measures obtained from using RJUcore$_\alpha$ and RJPcore$_\alpha$ on Sample-1 is presented in Fig. 1 and Fig. 2. Results from Sample-2 to Sample-5 also showed similar trends. Table 6 compares the average performance measures for all the algorithms employed for this dataset. The observations as noted are discussed below.

During the initial experiments with Sample-1 for tuning the value of the parameter M, it is observed that by decreasing the value of M from 25 to 1, test accuracy of RJP improves from 73.99% to 79.85%. Under the same variations of M test accuracy of RJU improves from 74.35% to 81.37% (Table 4). This implies that for Forest cover type dataset the efforts towards generalised classifier either by pruning or by using the higher values of M results in loss of some information (refer Appendix 1 for details on decision tree induction algorithm). This loss of information is due to the generalization of specific example cases in the training data for higher values of M (for M = 25). Hence, this loss of information is minimized by choosing M = 1. In general, higher values of M are preferred to avoid overfitting of the decision tree for the training data unless it causes decrease in prediction accuracy for the test cases. However, proper value of M is chosen by predicting the error on the test cases. Decrease in accuracy with increasing values of M is a reflection of the information loss. Some times, depending on the preference for simple model, small information loss may be accepted at the cost of little decrease in accuracy.

By using reducts of varying cardinality in RJU experiments, it is observed that training as well as test accuracy is independent of the cardinality of the reduct (Fig. 3).

In the results of experiments involving RJUcore$_\alpha$ and RJPcore$_\alpha$, it is observed that by increasing alpha not only number of attributes in the classifier decreases but number of rules as well as complexity steadily decreases. However, this is at the cost of slight degradation of training as well as test accuracy (Fig. 1, Fig. 2). The decrease in accuracy is due to the loss in information that was imparted by some attributes present in an approximate core having lower alpha. The quantification for loss of information is not attempted in this work. However such quantification can be attempted by employing entropy measures for the information as defined by Shanon (Shanon 1948). However, this loss in information is acceptable because of the desire to have the reduced complexity for this very large dataset.

The attributes 1-10 are observed as the most relevant attributes because of their occurrence in all the reducts from each of the samples (Table 3). It is to be noted that 80% reduction of number of attributes in the induced classifier has resulted in 5% loss of accuracy (as observed for RJUcore$_\alpha$:1 and RJPcore$_\alpha$: 1 in Table 6). This strengthens the notion regarding capabilities of approximate core in inducing the simpler classifier.

Table 6. Comparison of CS for the dataset

Algorithm	Accuracy (%)	Attributes	# Rules (10^3)	#Selectors (10^4)	CS
RS	13.4	29	24	70	0.04
CJU	82.3	53	28	73	0.21
CJP	82.5	51	2	55	0.21
RDTGA	74.1	29	11	5	0.19
RJUGA	79.8	29	30	29	0.21
RJPGA	78.4	29	78	6	0.20
RJUcore$_{\alpha:1}$	77.6	10	35	18	0.22
RJPcore$_{\alpha:1}$	75.6	10	6	2	0.21

On using MPCR in place of approximate core, the performance measure accuracy is observed to improve along with undesirable increase in the complexity, the number of rules, and number of attributes as well. This is being shown by the points corresponding to the smallest value of alpha (Fig. 1, 2). This implies that inclusion of all attributes in the DT induction does not produce the desirable simpler classifier.

For RJPcore$_\alpha$, average rule coverage (number of training objects divided by number of rules) is computed to be in the interval 4-5 while it is less than one for RJUcore$_\alpha$. Hence, the rules from RJPcore$_\alpha$ seem to offer some generalization as compared to rules from RJUcore$_\alpha$ at the cost of small degradation in accuracy (Table 6).

Comparison of some selected RDT variants with classical approaches (RS,CJU and CJP) shows that RDT variants - RJUGAsmallest and RJPGAsmallest are comparable to benchmarking algorithms CJU and CJP in terms of test accuracy estimates (Table 6). CJU and CJP show the highest average test accuracy of 82.32% and 82.56% respectively by utilizing 51 and 53 attributes respectively. On the other hand RJUGA-smallest and RJPGAsmallest could provide an average test accuracy of 79.81% and 78.48% by utilizing 29 attributes only. The number of rules from CJP is lesser but the complexity is higher than that of RJPcore$_\alpha$: 1. This suggests that the length of the rules induced by CJP are greater than those induced by RJPcore$_\alpha$: 1 (Table 6).

It is observed that for Forest cover type dataset none of the algorithms could be ranked best in terms of all the performance parameters; hence Cumulative Score (CS) (Minz and Jain 2005) is applied to rank the algorithms. By assigning equal weights to the performance measures - accuracy, complexity, number of rules and number of attributes, CS is computed and presented in Table 6. As per the comparison of computed values of CS, hybridized RDT variant RJUcore$_\alpha$: 1 is ranked highest while classical RS approach is ranked lowest. Thus, induced ruleset using RS is a poor model for this dataset with the test accuracy less than 20%.

All RDT variants also compare well with the previously published benchmarking accuracy results (Table 7) for the Forest cover type dataset (Bagnall and Cawley 2003). Three grades of performance are recommended for this dataset: less than 70% is poor, 70-75% is adequate, and greater than 75% is good. Following this recommendation RDT variants RJUGAsmallest, RJPGAsmallest, RJUcore$_\alpha$: 1, RJPcore$_\alpha$: 1 are good models however RDTGAsmallest is adequate with an accuracy of 74.1%.

Table 7. Comparison of accuracy with previously reported results for Forest cover type dataset

Model	Accuracy (%)
Back Propagation	70.0
Linear Discriminant Analysis	58.0
SVM	71.0
SVM modified for unrepresentative class	73.4
C5	83.7
CHAID	72.7
CART	68.9
XCS	66.9
CJU	82.3
CJP	82.6
RDTGA-smallest	74.1
RJUGA-smallest	79.8
RJPGA-smallest	78.5
RJUcore$_\alpha$: 1	77.0
RJPcore$_\alpha$: 1	75.0

5. CONCLUSIONS

The experiments with the benchmarking large dataset having continuous attributes, suggest that classical RS alone is not suitable because of a very high complexity as well as a very low accuracy. Classical decision tree approach results in good accuracy but number of attribute requirements is high. RDT variants using approximate core generates simpler rules and removes irrelevant attributes at a stage prior to the tree induction to facilitate lesser memory requirements for the subsequent steps of learning and classification. This hybridized framework also provides mechanism to trade off between complexity and accuracy as per the requirements. Validity of approximate core on Forest cover type dataset suggests that the tool is useful for classification problems involving real time large datasets. In future, there is a need for the software integrating approximate core and other rough set tools with decision tree induction algorithms to encourage the application of approximate core for real time problems.

REFERENCES

Bagnall, A.J. and Cawley, G.C. (2003). Learning classifier systems for data mining: A comparison of XCS with other classifiers for the forest cover data set. *Proc. International Joint Conference on Artificial Neural Networks (IJCNN-2003),* **3**,1802-1807. Portland, Oregon, USA.

Blackard, J.A. and Dean, D.J. (1999). Comparative accuracies of artificial neural networks and discriminant analysis in predicting forest cover types from cartographic variables. *Compu. Electron. J. Agric*., **24**,131-151.

Fayyad, U.M. and Irani, K.B. (1993). Multi-interval discretization of continuous valued attributes for classification learning. *Proc. 13th Intl. Joint Conf. on AI*, Morgan Kaufmann, 1022-1027.

Han, J. and Kamber, M. (2001). *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 279-325.

Jain, R. and Minz, S. (2003). Classifying mushrooms in the hybridized rough sets framework. *Proc. 1st Indian International Conference on Artificial Intelligence (IICAI-03),* Hyderabad, 554-567.

Johnson, D.S. (1974). Approximation algorithms for combinatorial problems. *J. Compu. Sys. Sci*., **9**, 256-278.

Minz, S. and Jain, R. (2003). Rough set based decision tree model for classification. *Proc. of Fifth International Conference DaWaK 03,* Prague Czech Republic. LNCS 2737, Springer-Verlag, Berlin Heidelberg, 172-181.

Minz, S. and Jain, R. (2005). Refining decision tree classifiers using rough set tools. *Int. J. Hybrid Intell. Sys.,* **2(2)**, 133-148.

Murphy, P. M. UCI repository of machine learning databases. University of California, Irvine, available at http://www.ics.uci.edu/~mlearn/.

Murthy, S. K. (1998). Automatic construction of decision trees from data: A multidisciplinary survey. *Data Mining and Knowledge Discovery*, **2**, 345-389.

Ohrn, A. (1999). *Discernibility and Rough Sets in Medicine: Tools and Applications*, Ph.D. Thesis. Norwegian University of Science and Technology.

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

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

Shan, N., Ziarko, W., Hamilton, H.J. and Cercone (1996). Discovery classification knowledge in databases using rough sets. *Proc. Second International Conference on Knowledge Discovery & Data Mining*, Menlo Park, CA: AAAI Press.

Shanon, C. (1948). A mathematical theory of communication. *The Bell Systems Technical Journal*, **27**, 379-423, 623-656.

Witten, I.H. and Frank, E.(2000). *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann Publishers.

Wroblewski, J. (1995). Finding minimal reduct using genetic algorithms. Warsaw University of Technology, Institute of Computer Science Reports - 16/95.

Ziarko, W. (1999). Discovery through rough set theory. *Communications of ACM*, **42(11)**, 55-57.

APPENDIX 1: BUILDING SIMPLE DECISION TREES

Decision Tree Algorithm by Quinlan creates binary splits on interval inputs and multi-way splits on nominal inputs for a nominal target. The split is chosen that maximizes the gain ratio: Gain ratio = reduction in entropy / entropy of split. (Let P(b) denote the proportion of training cases a split assigns to branch b, b = 1 to B. The entropy of a split is defined as the entropy function applied to {P(b): b = 1 to B}.)

For nominal inputs, each category is first assigned to a unique branch, and then, in steps, two branches are merged, until only two branches exist. The split with the maximum gain ratio among splits examined becomes the candidate split for the input. This search method, of course, is heuristic and might not find the nominal split with the largest gain ratio. The search on an interval input will find the best split. Cases with missing values are excluded from the split search on that input and also from the numerator of the gain ratio. The entropy of the split is computed as if missing values constitute an additional branch. When a missing value prevents applying a splitting rule to a case, the case is replaced by B new ones, each being assigned to a different branch, and each is assigned a weight equal to the proportion of non-missing training cases assigned to that branch. The posterior probability of the original case equals the weighted sum of the probabilities of the generated observations.

Ross Quinlan advocates retrospective pruning instead of stopping rules. If enough data were available, the pruning process would use data withheld from training the tree to compare error rates of candidate sub tree. The software does not assume data may be withheld from training, so it implements “pessimistic” pruning. In each node, an upper confidence limit of the number of misclassified data is estimated assuming a binomial distribution around the observed number misclassified. The confidence limit serves as an estimate of the error rate on future data. The pruned tree minimizes the sum over leaves of upper confidences.