Development of a Knowledge-Driven Constructive Induction Mechanism

Lo, S.; Famili, Fazel

Development of a Knowledge-Driven Constructive Induction Mechanism *

Lo, S., and Famili, A.
August 1997

* published in the Proceedings of the Second International Symposium on Intelligent Data Analysis (IDA-97). London, England. August 4-6, 1997. NRC-40208.

Presented at the IDA-97 Symposium, London, UK, Aug. 4-6, 1997

Development of a Knowledge-Driven Constructive Induction Mechanism

Suzanne Lo(1) and A. Famili

Institute for Information Technology, National Research Council Canada, Ottawa, ON, K1A 0R6, Canada, famili@ai.iit.nrc.ca

(1) Current address: Dept. of Computer Science, University of British Columbia, Vancouver, BC, V6T 1Z1, Canada

Abstract

This paper discusses the advantages of knowledge-driven constructive induction (KDCI). Development, testing, and evaluation of a KDCI (or in short CI for constructive induction) system are explained in detail. The objectives of developing this system were to demonstrate the usefulness of the approach and to provide knowledge-driven constructive induction support in our data analysis research. Technical details, particularly the process of building new attributes and changing the representation space, are discussed. Other issues concerning the design and implementation of the CI mechanism are presented. The evaluation process and comparison measures used for evaluation of the system are briefly explained. Experimental results, using 4 data sets from a real-world application, are given.

Keywords: induction, constructive induction, feature generation

1.0 Introduction

Today's world is overwhelmingly dominated by generation and acquisition of large amounts of data in all businesses and industries. Over the last ten years, there have been many advances in data integration, data warehousing, data analysis, and generation of useful knowledge. Examples are several data analysis tools that have become commercially available [9]. Almost all these tools assume that the data representation space is accurate and complete. The data representation space is normally spanned over descriptors such as attributes, variables, terms, relations, or transformations that are used to describe an object. For example, most inductive tools use all initial attributes (numeric and non-numeric) as provided in the data sets, assuming that useful concepts can be generated from these attributes. However, researchers in real-world data analysis applications have experienced that more meaningful concepts can be generated from the data and more accurate knowledge can be discovered if the data is properly transformed to provide a new representation space. This is through constructive induction.

Constructive induction is the process of manually or automatically transforming the data through creating new features from the existing ones and possibly removing the less relevant ones before or during the data analysis. The goal of constructive induction is therefore to create a new representation of the domain so that the overall prediction or explanation accuracy of induction based data analysis tools are improved. The prediction accuracy can be estimated through analysing and comparing two randomly selected subsets of a large data set when the representation space is modified on one and not on the other. The explanation accuracy is normally evaluated at the end of the data analysis process by a domain expert.

Generating new features may involve different constructive induction strategies. Three strategies introduced in the literature [16] are: (i) **hypothesis-driven** in which changes to the data representation space are based on the analysis of hypotheses generated in each data analysis iteration and the discovery of patterns, (ii) **data-driven** in which data characteristics, such as interrelationships between parameters, are used to generate new data representation space, and (iii) **knowledge-driven** in which expert provided domain knowledge is applied to modify, construct and/or verify new data representation space. When two or more of the above strategies are combined, it is called **multistrategy constructive induction** approach.

Two of the most common applications where a **knowledge-driven constructive induction** approach would be useful are:

- Data analysis applications in which initial parameters have to be corrected through the use of domain knowledge and new features have to be created for an accurate and meaningful data analysis. Examples are industrial operations where environmental conditions affect the operation and performance of a system.
- Data analysis applications in which exploratory research is needed to incorporate qualitative models of a process into the data analysis process and create new features from the existing ones. Examples are applications in which dimensionless terms (π terms) can be introduced to replace all or some of the initial attributes. Each dimensionless term represents two or more of the original attributes. The idea is to transform any number of dimensionally invariant variables (m) represented by n dimensions into m-n dimensionless terms.

The research reported in this paper is focused on **knowledge-driven constructive induction**. Our goal is to introduce a knowledge-driven construction induction mechanism that we have developed as part of our research in data analysis and demonstrate, using real-world data, how creation of new features could improve the overall performance of this system. In Section 2, we review some of the related work. Section 3 includes description of the problem that motivated this work. In Section 4 we provide an overview of our approach where we introduce technical details of the constructive induction mechanism that we have built and show how new features are generated and used. Section 5 includes testing and evaluation methodology and in Section 6 we provide the results. We conclude the paper in Section 7 and discuss our future research.

2.0 Related Work

The general form of knowledge-driven constructive induction has been advocated by many researchers [10, 15]. Most of the related works were motivated by the fact that inductive learning algorithms were sensitive to the original representation space. This section briefly reviews some aspects of knowledge-driven constructive induction research performed by other researchers.

Pagallo [12] developed FRINGE that iteratively constructs new attributes using structural information of the decision tree. Matheus and Rendell [8] and Rendell [14] also proposed a number of frameworks for feature construction based on four main aspects of detection, selection, generalization, and evaluation. In these studies, the process of building new features is at the time of building the decision tree where simple domain knowledge is used for constructive biases. AM (Automated Mathematician), developed by Lenat [7], changes the data representation space by employing pre-defined heuristics for: (i) defining new concepts represented as frames, (ii) creating new slots and their values, and (iii) adapting concept frames developed in one domain to another domain.

Callan and Utgoff [1] used the domain knowledge, in the form of first order predicate calculus, to describe the problem solving operators. This information was then used to generate a better vocabulary by increasing the resolution and by decomposing the description of the goal states into a set of functions that were used to map search states to feature values. On two domains that this method was tested, it was shown that the features it generated, were more effective for inductive learning than the original features. A similar concept has been incorporated into LAIR [3] which is a constructive induction system that acquires conjunctive concepts by applying a domain theory to introduce new features into the evolving concept description. The system works through an iterative process in which it weakens the inductive bias with each iteration of the learning loop.

Most systems that incorporate knowledge into constructive induction use almost complete domain knowledge. Use of fragmentary knowledge in constructive induction, due to varying degrees of completeness, may still result in generation of useful features. Donoho and Randell [3] showed that production of new features, using fragmentary knowledge increases inductive accuracy and it also results in determining knowledge reliability. Use of domain knowledge during the process of building the decision tree was also investigated by Nunez [11] who developed an algorithm to generate more logical and understandable decision trees than are normally generated by learning algorithms like ID3. His algorithm executes various types of generalization and at the same time reduces the classification costs by means of background knowledge.

Constructive induction could also be treated as a preprocess to data analysis. In GALA, developed by Hu [6], a small number of new attributes are generated from the existing nominal or real-valued attributes. GALA was designed for use in preprocessing data sets for inductive algorithms and was tested with C4.5 [13]. Reiger [15] also views constructive induction as a data preprocessing step and proposes a framework in the robotics domain where the numeric data is transformed to logic-based data using domain knowledge.

3.0 The Problem

Some of the best applications of inductive techniques have been in areas where domain knowledge has been used to generate a new representation space. In some cases, complex domain knowledge, such as causal qualitative models, are used to generate new attributes. The main questions are: (i) where the required knowledge comes from, (ii) how can we appropriately use this knowledge to modify the representation space, (iii) are the new attributes meaningful and (iv) how can we evaluate the results when new attributes are used. This section includes a brief discussion of the above points.

3.1 Types of Knowledge

The knowledge for creating a new representation space could vary from simple concepts of domain knowledge to complex causal models. The best representation space can be created when complete domain theories are used for constructive induction [2]. Two of the most common types of domain knowledge are:

- Process or System Knowledge that represents various aspects of domain theory. This type of knowledge comes from domain experts, empirical results, system design information, and system documentation.
- Data Characteristics Knowledge that is acquired during data preprocessing. Various approaches are taken. Examples are: data visualization, principal component analysis, dimensional analysis and data fusion. The main goal is by understanding the nature of the data (combined with domain theory), one can generate new attributes.

3.2 Validating New Features

The most difficult task in knowledge-driven constructive induction is validating features when a new representation space is introduced since the new representation space has a profound effect on the quality of generated results. One approach is to investigate the relationship between attributes in the new representation space when new features are included. This can be done using design of experiment techniques to evaluate the performance of a system or process represented by the new representation space. The other approach is to evaluate the performance of the new representation space in an induction process and on iterations in which sets of new attributes are added in groups of related features. This approach requires involvement of domain experts [2] who would evaluate the results of induction each time the representation space is modified.

4.0 Overview of the Approach

Most learning systems have been concerned with learning concepts from examples from a fixed set of attributes. In other words, the attributes relevant to describing the examples are normally provided before the learning process starts, and the resulting concepts are expressed in terms of these attributes. However, inconsistency and incompleteness of data, as well as the need for incorporating the domain knowledge, have shown the importance of learning systems to have the ability to construct new attributes. Our goal in constructive induction is to convert domain knowledge into a set of numeric attributes for which an evaluation function can be generated via existing methods. We have developed a constructive induction facility that is added to our data analysis tool, IMAFO [Intelligent MAnufacturing FOreman]. IMAFO is a data mining tool that is used for failure analysis and process optimization. It helps engineers to discover the reasons for unsuccessful productions or operations [4]. IMAFO has two engines for data analysis. Engine 1 is a variation of Quinlan's ID3 algorithm; Engine 2 uses Quinlan's C4.5 algorithm [4,13].

4.1 Design of the CI Mechanism

The CI module was designed as one of the modules of IMAFO. The criteria for designing the CI module was: once new attributes are created, they are added to the measurement space for any application set-ups and data analysis. Creation of new attributes would be based on using domain knowledge and all new attributes will have the capabilities of the existing ones:

- **Subspace Definition**: New attributes can be used to define new classes (problem definitions).
- **Dimension Control**: New attributes can be enabled or disabled for certain classes during data analysis.
- **Correlation Check**: Principal components can be identified using new attributes.
- **Graphics**: New attributes can be used for data visualization.

Figure 1 shows the interface to the CI facility. The New Variable Names list (top scroll list) contains the names of all the new attributes. The first set of buttons (Insert, Rename and Delete) are used to define new attribute names. The Available Numeric Variables list (bottom scroll list) contains all numeric (integer or real) attributes that can be used in the definition of a new attribute. The Relations selection buttons display the selected relation which can be used to define a new attribute. The New Variable Definition field shows the definition of the new attribute that is highlighted in the New Variable Names list. The Build button is used to build a new attribute definition.

4.2 Technical Details

The relationships between the objects in the interface are illustrated in Figure 2. A new attribute consists of a name and its definition. A definition is a mathematical expression that is built from domain knowledge. When a new attribute name and its definition are entered, they are added to the New Variable Names list. The most important step in defining a new attribute is the Build process. This process ensures that the expression is syntactically and semantically valid.

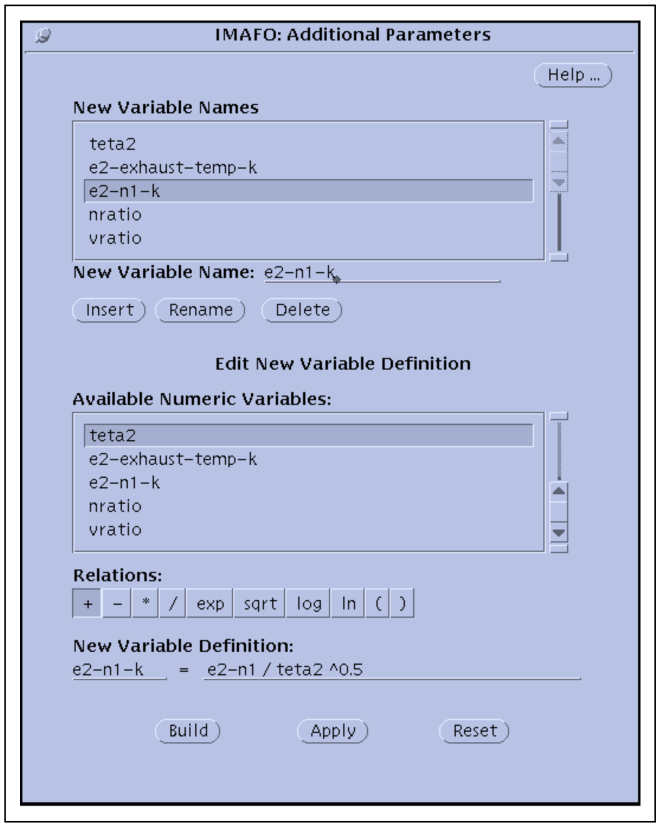
The operations in the Relations section for constructing new attributes are based on the generality of the individual relation and completeness of the overall relation set. These relations are relatively common operators that can be used frequently in changing the representation space. It should be noted that the current set of relations in our system can be extended to include operators for strings and logical expressions. They can result in new attributes of type string or boolean.

4.3 Changing the Representation Space

The process of changing the representation space involves three operations: *expansion, correction and contraction*. Expansion aggrevates the learning process as the representation space is expanded. For example, if two of the original attributes were angular velocity of a rotor blade (ω) and its radius (r), and its linear velocity (v) was also useful for learning a concept, then the linear velocity can be added to the representation space:

$$v = r \times \omega$$

The original representation space can also be corrected for inconsistencies in the values of the attributes to provide a better representation space for the learning process. This can be accomplished by constructing new attributes with more consistent values from existing attributes and eliminating the original attributes. For example, if one of the original attributes were the angular velocity of the rotor bade of an engine (ω), since it is influenced by several operation conditions, it would be corrected by θ:



New Variable Names: teta2, e2-exhaust-temp-k, e2-n1-k, nratio, vratio

New Variable Name: e2-n1-k

Available Numeric Variables: teta2, e2-exhaust-temp-k, e2-n1-k, nratio, vratio

New Variable Definition: e2-n1-k = e2-n1 / teta2 ^0.5

Fig. 1. Interface for the Constructive Induction Facility

$$\omega_c = \omega / \theta$$

where θ is calculated from domain knowledge. ω_c represents the corrected value of ω.

Contraction decreases the complexity of the learning process as the representation space is restricted. This is performed through abstraction and implies that the learning process can be sped up due to narrowing the search space and decrease in generality. Use of dimensionless terms is a good example of contraction. For example, Reynolds Number (RN) of a moving object is space, a dimensionless term, is derived from air speed (U), air mass density (ρ), length of the moving object (l), and coefficient of air viscosity (μ), as:

$$RN = \frac{\rho \times U \times l}{\mu}$$

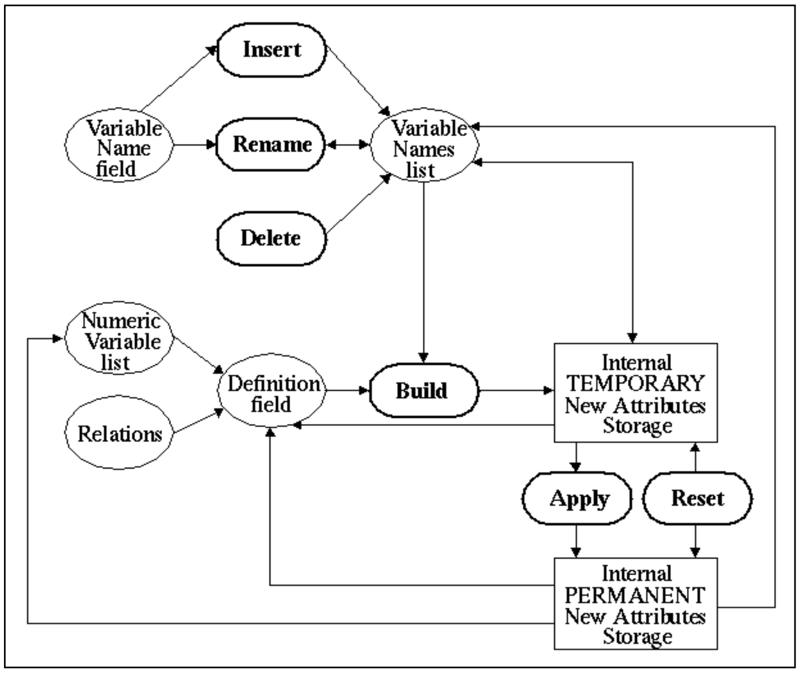


Fig 2. Interaction between Objects in CI Interface

The most important effect of changing the representation space is that the overall complexity of the resulting concept description may decrease when domain knowledge is applied. An example is when irrelevant attributes are eliminated from the representation space. Table 1 shows how the existing relations (shown in figure 1) can be used for expansion, contraction, and correction of the original attributes to generate a new representation space.

TABLE 1. Summary of representation space changes

Operator	Arguments	Notation	Interpretations	E	C	K
+	Attributes x, y	x + y	Sum of x and y	*	*	*
-	Attributes x, y	x - y	Difference between x and y	*	*	*
*	Attributes x, y	x * y	Product of x and y	*	*	*
/	Attributes x, y	x / y	Quotient of x and y	*	*	*
exp	Attributes x, y	x ^ y	x raised to the power of y	*	*	*
sqrt	Attributes x	sqrt x	Square root of x	*		*
log	Attributes x	log x	Logarithm of x in base 10	*		*
ln	Attributes x	ln x	Natural logarithm of x	*		*
(	-	-	Beginning of sub-expression			
)	-	-	End of sub-expression			

E: Expansion, C: Contraction, and K: Correction

5.0 Testing and Evaluation Methodology

The overall goal of testing was to focus on the logics of the algorithm that generates the new representation space. Evaluation was performed to investigate improvements in **predictive accuracy** of the concepts (rules) generated as a result of our data analysis. Predictive accuracy is a criterion that evaluates the performance, consistency, and other behaviours of learning systems, as a measure of success. Having high predictive accuracy means that the learning system learned the correct concept. The correct concept can be represented in different forms or different organizations of a single representation, or different combination of attributes.

The experimentation process consisted of running IMAFO, with and without the new representation space, to evaluate the performance based on improvements on the following comparison measures:

- **Coverage**: The percentage of records with problems that the rule explains.

$$\textit{coverage} = \frac{\textit{\# of records with the rule and the problem}}{\textit{\# of records with the problem}}$$

- **Error rate**: The percentage of records without problems that the rule explains.

$$\textit{error rate} = \frac{\textit{\# of records with only the rule}}{\textit{\# of records with the rule and the problem}}$$

- **False-negatives rate**: The percentage of records with problems that the rule does not explain.

$$\textit{false-negatives rate} = \frac{\textit{\# of records with the problem but without the rule}}{\textit{\# of all records}}$$

- **Rule complexity**: The number of variables acting together in a rule to characterize a problem.
- **Accuracy**: The percentage of records with problems that the rule explains and without problems that the rule does not explain.

$$\textit{accuracy} = \frac{\textit{\# of records with problem \& rule + \# of records without problem \& rule}}{\textit{\# of all records}}$$

For experimentation, 46 data sets, representing data collected from an aerospace domain, grouped into four groups, were analyzed. The steps for starting the data analysis were as follows:

1. Filtered data files for corrupt data.
2. Performed data visualization for trend monitoring and class threshold selections.
3. From graphs prepared in step 2, selected proper thresholds for problem (class) definitions.
4. Removed irrelevant attributes from the measurement space.
5. Analyzed all corrected data sets using both engines 1 and 2. For consistency, data analysis was repeated three times.

6. Using the domain knowledge, changed the representation space.
7. Modified all problem (class) definitions to make use of the new attributes, if necessary.
8. Analyzed all data sets using both engines 1 and 2. For consistency, both engines were run three times on each data set.

The following steps were taken for summarizing the results.

1. For each run, the top rule was selected for comparison. The top rule is the rule in which the comparison measures collectively behave better than all the other rules generated (see explanation below).
2. Determined the coverage, error rate, and complexity from the output of all three runs. From the contingency tables of the runs, determined the false-negatives rate and accuracy.
3. Calculated the average of each comparison measures listed in step 2 for the three runs.
4. Repeated steps 1, 2 and 3 for the results, using the new representation space.
5. For each data set, the values of comparison measures without new attributes were subtracted from the values obtained when including new attributes. Thus a positive difference indicates an increase in the comparison measure by using the CI facility, and a negative difference indicates a decrease. Note that an increase does not imply improvement in quality and a decrease does not imply deterioration.
6. The values of differences obtained in step 5 were further averaged for each data group.

For engine 1, the top rule was selected for evaluation because from our experience in the past we have noticed that the first rule had by far the best performance. The first rule usually gives the highest coverage and the lowest error rate. As an example, when we compared two rules generated for the same problem, rule 1 performed better than rule 2 in coverage (64.5% vs. 9.7%), false-negatives rate (7.4% vs. 18.9%), complexity (2 levels vs. 3 levels) and accuracy (92.6% vs. 81.1%). The performance of both engines were the same in error rate. For engine 2, it follows from engine 1 that the top rule is also selected for evaluation for consistency in the overall evaluation process.

6.0 Results

The data came from an aerospace domain that contained several parameters from the operation of the aircraft main engines and the auxiliary power unit. Results of analyzing four data groups are presented below. Data group 1 consisted of 34 data sets of auxiliary power unit described by 7 attributes; they are divided into individual data sets by aircraft identification number. Data group 2, also represented auxiliary power unit data, consisted of four data sets for four aircraft from data group 1, but described by 55 attributes. Data group 3 consisted of four sets of main engines cruise data for four aircraft. This group consisted of 52 attributes. Data group 4 consisted of four sets of main engines cruise performance data for the four aircraft in data group 3. This group consisted of 53 attributes.

Table 2 summarizes the results of data analysis obtained with the two data analysis engines in IMAFO. This table reports the averages of the differences in the comparison measures obtained for the four data groups. A positive value indicates an overall increase in the comparison measure between the original and the new representation space, for the data set(s) in the data group; a negative value indicates an overall decrease.

For engine 1, the results in Table 2 shows improvements in performance measures with the new representation space. Each comparison measure improved in two to three data groups. In other words, for two data groups, coverage increased (data groups 2 and 4) and error rate decreased (data groups 2 and 4). In addition, false-negatives rate decreased for three data groups (2, 3 and 4) and complexity decreased for three data groups (1, 2 and 3). Finally, accuracy increased in data groups 2, 3 and 4.

For engine 2, the results shows similar improvements. While coverage increased for three data groups (2, 3 and 4), error rate decreased for the same data groups. On the other hand, false-negatives rate decreased for two data groups (2 and 4). As for complexity, it decreased for three data groups (1, 2 and 3). Accuracy increased for three data groups (2, 3 and 4).

TABLE 2. Averages of Results of Experimentation

Comparison Measures		Cov(%)	ER(%)	FN(%)	RC	A(%)
Engine 1	Data Group 1	-3.76	1.44	1.80	-0.21	-1.61
	Data Group 2	5.86	-0.94	-1.79	-0.19	2.01
	Data Group 3	-1.04	5.68	-1.14	-0.26	1.04
	Data Group 4	2.39	-2.84	-0.09	0.08	0.09
Engine 2	Data Group 1	-2.92	3.17	0.29	-0.17	-0.49
	Data Group 2	0.49	-0.50	-0.60	-0.29	0.53
	Data Group 3	0.48	-0.48	0.10	-0.44	0.07
	Data Group 4	2.93	-2.93	-0.13	0.13	0.11

In the above table, Cov=Coverage, ER=Error Rate, FN=False Negatives, RC=Rule Complexity, and A=Accuracy.

When using engine 1 of IMAFO, changing the representation space showed slightly greater magnitude of improvement than using engine 2. The improvement in coverage using engine 1 ranged from 2.39% to 5.86% while using engine 2, the improvement ranged from 0.48% to 2.93%. The decrease in error rate for engine 1 ranged from 0.94% to 2.84% while for engine 2, the decrease ranged from 0.48% to 2.93%. The change in false-negatives rate using engine 1 ranged from 0.09% to 1.79% while using engine 2, the change ranged from 0.13% to 0.60%. The average ranges of improvement in complexity were 0.19 to 0.35 rules and 0.17 to 0.44 rules using engines 1 and

2, respectively. The increase in accuracy for engine 1 ranged from 0.09% to 2.01% while for engine 2, the increase ranged from 0.07% to 0.53%.

In general, the new representation space resulted in improvements in the data analysis. On average, while it does not always improve analysis results in the comparison measures, no comparison measure has shown constant deterioration with the data groups used in the experimentation.

7.0 Conclusions

The main goal of a constructive induction approach is to understand what knowledge from the application domain can improve the representation space. One is also interested to investigate the effects of incorporating domain knowledge on the constructed feature space and any improvements in accuracy and predictability of the learned knowledge. Our experiments have shown that knowledge-driven constructive induction helps to incorporate domain knowledge into the measurement space and create a new representation space. However, the procedure for incorporating domain knowledge and the process of evaluating the results may require involvement of domain experts. Although we did not investigate how using a new representation space may speed up the knowledge discovery we found that constructive induction provides means to generate more reliable knowledge.

We have designed and implemented a constructive induction facility into our existing data analysis system. This data analysis tool has been successfully applied in several domains, including semiconductor manufacturing. With the addition of the constructive induction facility, we hope to obtain improvements in predictive accuracy in data analysis. In our experimentation plan, we tested and evaluated the performance of our constructive induction using four groups of data, containing 46 data sets. A significant contribution of this system is that the rules generated can be easily interpreted by the domain expert or can be incorporated into an expert system. The comparison between the existing induction system and the one with constructive induction facility was based on change of the representation space that resulted in improvements in coverage, error rate, false-negatives rate, rule complexity and accuracy.

According to the results based on the five comparison measures, the performance of our data analysis tool for learning new concepts improved with the addition of the constructive induction facility. The user is now able to select the type of relation (operator) to be used to create new attributes and generate a new representation space. This gives engineers and data analysts a way for generating data descriptions that are most suitable based on the available domain knowledge.

We conclude that with knowledge-driven constructive induction and creation of a new representation space we can obtain more accurate and more reliable results in data analysis. There are a number of issues yet to be further investigated: (i) how can generation of new attributes be optimized before the data is analyzed, (ii) is it possible to automate the process of creating new attributes, by taking into account the data characteristics and domain knowledge, and (iii) expansion of our work to include string and logical expressions.

Acknowledgements:

The authors would like to thank Riad Hartani and Sylvain Létourneau for providing useful comments on an earlier version of this paper. The preliminary design of our constructive induction mechanism was done by Serge Oliveira.

References

[1] J.P. Callan and P.E. Utgoff, Constructive Induction on Domain Information, *Proceedings of AAAI-91,* AAAI Press, (1991), Vol. 2, 614-619.

[2] S. Donoho and L. Rendell, Constructive Induction Using Fragmentary Knowledge, *Proceedings of International Conference on Machine Learning*, Bari, Italy, (1996), 113-121.

[3] R. Elio and L. Watanabe, An Incremental Deductive Strategy for Controlling Constructive Induction in Learning from Examples, *Machine Learning*, Kluwer Academic Publishers, **7** (1), (1991), 7-44.

[4] A. Famili, and P. Turney, Intelligently Helping Human Planner in Industrial Process Planning, *AIEDAM*, **5**(2), (1991), 109-124.

[5] T.E. Fawcett, Knowledge-Based Feature Discovery for Evaluation Functions, *Computational Intelligence*, **12**(1), (1996), 42-64.

[6] Y-J. Hu, Constructive Induction: A Preprocessor, *Proceedings of Canadian AI Conference*, Toronto, Canada, Springer-Verlag, (1996), 249-256.

[7] D.B. Lenat, The Role of Heuristics in Learning by Discovery: Three Case Studies, in R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, eds., *Machine Learning: An Artificial Intelligence Approach* I, Morgan Kaufmann, Palo Alto, CA, (1983), 243-306.

[8] C.J. Matheus and L.A. Rendell, Constructive Induction on Decision Trees, *Proceedings of IJCAI-89,* Detroit, MI, AAAI Press, (1989), Vol.2, 645-650.

[9] J. Mena, Automatic Data Mining, PC AI, **10**(6), (1996), 16-20.

[10] R.S. Michalski, *et al*, The Multi-Purpose Incremental Learning System AQ15 and its Testing Application to Three Medical Domains, *Proceedings of AAAI-86*, Philadelphia, PA, AAAI Press, (1986), 1041-1045.

[11] M. Nunez, The Use of Background Knowledge in Decision Tree Induction, *Machine Learning*, Kluwer Academic Publishers, **6** (3), (1991), 231-250.

[12] G. Pagallo, Learning DNF by Decision Trees, *Proceedings of IJCAI-89*, Detroit, MI, AAAI Press, (1989), Vol.2, 639-644.

[13] J. R. Quinlan, *C4.5: Programs for Machine Learning,* Morgan Kaufmann Publishers, San Mateo, CA, (1993).

[14] L. Rendell, Learning Hard Concepts Through Constructive Induction: Framework and Rational, Report No. UIUCDCS-R-88-1426, Dept. of Computer Science, University of Illinois, Urbana, Il. (1988).

[15] A. Rieger, Data Preparation for Inductive Learning in Robotics, *IJCAI Workshop on Data Engineering for Inductive Learning,* (1995), 70-78.

[16] J. Wnek and R.S. Michalski, Hypothesis-Driven Constructive Induction in AQ17-HCI: A Method and Experiments, *Machine Learning*, **14** (2), (1994), 139-168.