World Academy of Science, Engineering and Technology
International Journal of Environmental and Ecological Engineering
Vol:3, No:2, 2009

Implementing an Intuitive Reasoner with a Large Weather Database

Yung-Chien Sun and O. Grant Clark

***Abstract*—**In this paper, the implementation of a rule-based intuitive reasoner is presented. The implementation included two parts: the rule induction module and the intuitive reasoner. A large weather database was acquired as the data source. Twelve weather variables from those data were chosen as the "target variables" whose values were predicted by the intuitive reasoner. A "complex" situation was simulated by making only subsets of the data available to the rule induction module. As a result, the rules induced were based on incomplete information with variable levels of certainty. The certainty level was modeled by a metric called "Strength of Belief", which was assigned to each rule or datum as ancillary information about the confidence in its accuracy. Two techniques were employed to induce rules from the data subsets: decision tree and multi-polynomial regression, respectively for the discrete and the continuous type of target variables. The intuitive reasoner was tested for its ability to use the induced rules to predict the classes of the discrete target variables and the values of the continuous target variables. The intuitive reasoner implemented two types of reasoning: fast and broad where, by analogy to human thought, the former corresponds to fast decision making and the latter to deeper contemplation. . For reference, a weather data analysis approach which had been applied on similar tasks was adopted to analyze the complete database and create predictive models for the same 12 target variables. The values predicted by the intuitive reasoner and the reference approach were compared with actual data. The intuitive reasoner reached near-100% accuracy for two continuous target variables. For the discrete target variables, the intuitive reasoner predicted at least 70% as accurately as the reference reasoner. Since the intuitive reasoner operated on rules derived from only about 10% of the total data, it demonstrated the potential advantages in dealing with sparse data sets as compared with conventional methods.

***Keywords*—**Artificial intelligence, intuition, knowledge acquisition, limited certainty.

I. Introduction

IN the literature of psychology and computational cognition, human intuition has been described as an inherent mental capability for fast, effortless recognition and as being able to accommodate incomplete information and poorly-defined goals [1], [2]. In this study, we look at intuition as the human capacity to make decisions under novel, complex situations in which information and goals are presented such a way. The objectives of this study were to characterize some of the aspects of intuitive thought, particularly those concerning problem solving with knowledge which is incomplete and of variable quality, and to model these aspects in a computer system. Sun and Clark [3] proposed a rule-based conceptual model for an intuitive reasoner (e.g. a classification algorithm) where the rules available to the reasoner could be fuzzy and produce multi-valued outputs associated with variable levels of certainty. Intuitive reasoning was modeled as a rule inference process that was carried out by the reasoner iteratively firing the rules and then consolidating similar outputs based on a predefined algorithm. When corroborating outputs reinforced each other and were consolidated, the associated certainty level of the consolidated datum was increased. Ideally, the conclusions produced at the end of the reasoning session, when no more rules can be fired, would be of high certainty. This conceptual model served as the foundation for the implementation presented in this paper.

This implementation consisted of two parts: the rule induction module and the intuitive reasoner, both were programmed in the computer language Matlab (R2007). A large weather database with 50 years of hourly measurements of 54 weather variables was acquired as the data source for rule induction. Because this was a pioneer implementation of the proposed conceptual model, the scope of the project was modest. A "complex" situation was simulated by making only a fraction of the data in the weather database available to the rule induction module. As a result, each induced rule involved only a small number of the variables from the weather database and was usually low in certainty. This mimicked, by analogy, the kind of scenario often confronted by human problem solvers, wherein they rarely have complete information about a complex situation. The reasoner was tested to predict the class or value of certain target weather variables with the induced rules.

Data records in the weather database contained measurements of weather variables observed at the same hour, so that the rules induced from these data also describe the relationships between variables at those specific times. These relationships, hence, are descriptive rather than predictive. However, the word 'predict' is used in this paper in a statistical sense, as in "the intuitive reasoner predicts that the relative humidity is 30%," meaning it produced an estimate of the measured value.

The certainty level associated with a rule or a datum (input or output) was indicated by a metric called "Strength of

Y.-C. Sun is with the Department of Bioresource Engineering, McGill University, Ste. Anne de Bellevue, QC H9X 3V9 Canada (e-mail: yungchien.sun@mail.mcgill.ca).

O. G. Clark is with the Department of Bioresource Engineering, McGill University, Ste. Anne de Bellevue, QC H9X 3V9 Canada (phone: 514-398-7784; fax: 514-398-8387; e-mail: grant.clark@mcgill.ca).

Belief" (*SB*), the value of which could vary between 0 and 1, corresponding to no certainty and complete certainty, respectively. Rules induced from the input data were called "basic rules" and each was associated with an *SB* to indicate its certainty level. Identical or similar basic rules were consolidated to create "super rules" that represented repetitive occurrences and usually had higher *SB* than basic rules. The algorithms for computing the *SB* of rules varied with respect to the rule induction techniques used and were discussed in detail in the rule induction section. The final rule base contained the super rules and those basic rules which were not represented by any of the former. A reasoning process where only the rules with higher *SB* values were referenced was faster than one involving all basic and super rules. These two types of inference were termed *fast* and *broad* reasoning, resembling, by analogy to human thought, "snap" decision-making and lengthier reflection without time pressure. They were both tested and the results were compared.

A "reference reasoner" was created by adopting a weather data analysis approach that had been employed to analyze similar weather data [4]. The reference reasoner consisted of predictive models generated from the weather data for the same target variables. Contrary to the situation for the intuitive reasoner, these models for the reference reasoner were generated using complete information from the full weather database. The predictive capability of these models was considered to be the "upper bound" of what could be expected from the intuitive reasoner. During evaluation, the intuitive reasoner and the reference reasoner were given the same set of input data for a given target variable and the actual measured values of the target variable were compared with the predictions made by the two reasoners. The intuitive reasoner had an advantage when dealing with incomplete data in that it matched the performance of the reference reasoner for two continuous target variables and reached at least 70% of the classification accuracy of the reference reasoner for all six discrete target variables, while working with rules induced from only about 10% of the total data.

II. The Weather Database

Sets of hourly data for 54 weather variables from the years 1953-2005 for Montreal were acquired from the weather archive of Environment Canada and assembled into a database From the 54 weather variables, three variable types were identified: continuous, ordinal, and categorical. Variables of the latter two types are also termed *discrete variables* in this paper. A *continuous weather variable* is a variable of which the values are real numbers, e.g., global solar radiation and temperature. Contrary to continuous variables, discrete weather variables take only discrete integers as their values. The *ordinal weather variables* represent various weather events, e.g., rain and snow. Bigger values correspond to greater intensity or level of the weather event. Zero corresponds to a nonevent. The order in the values of *categorical variables*, on the other hand, does not reflect any rank of the measured intensity or level, e.g., the cloud type at various cloud layers. The type of the target variables determined the type of rule induction algorithm to use. Regression was used to create rules for predicting continuous target variables, by modeling the relationships between a given dependent variable and the independent variables in a data subset. The decision tree algorithm was employed for discrete target variables, to classify the dependent variable based on the independent variables involved. In tree growing, different types of dependent variable postulated different ways of node splitting. More details are discussed in the description of rule induction (Section 4.1).

As with many kinds of archived data, some preprocessing of this weather database was required. Missing entries in the data set were imputed by the nearest neighbor method [5]. Theoretically, this imputation should not increase the amount of information in the data set. Some measured variables, such as cloud ceiling and cloud layer heights, were flagged in the original data with the value 888 to indicate, for example, an infinite height or a cloudless sky. To avoid potential errors caused by computing with infinite values, the values of these variables were converted to their multiplicative inverses except for the value of 888, which was replaced by zero.

Six ordinal variables and six continuous variables were chosen as the target variables (Table 1), for which the values were predicted. In this paper, the target variables are sometimes referred to by their unique three-digit codes, as defined in the Environment Canada archive. The selection of target variables was based on the completeness of the data in the original archive, with preference given to variables with fewer missing entries. In addition, if similar weather variables existed with similarly complete data, one representative variable was chosen as a target variable. For example, Rain (086) was chosen and Rain Showers (087) was not. Discrete target variables were chosen from ordinal weather variables with no more than four different discrete values. The reason was to simplify the associated prediction task for the intuitive reasoner.

III. Intuitive Reasoning in Novel, Complex Situations

Sun and Clark [3] reported that, when solving a novel or complex problem, a problem solving algorithm is usually presented with many kinds of data of variable quality and the knowledge (e.g. rule set) available with which to reason about the situation is usually also incomplete and of limited certainty. Contrary to the scenario in which few, high-certainty rules can be used to solve a task, the problem solver may have to draw instead on many low-certainty rules, given that no single rule or small set of rules can lead to conclusions which merit any confidence. In the approach adopted here, a conclusion is sought through iteratively consolidating any intermediate outputs from the rule set which might corroborate one another, thus gradually increasing the certainty level. This is the underlying mechanism of intuitive thought proposed in [3]. Based on this mechanism, an

intuitive reasoning algorithm was developed based on existing knowledge acquisition and reasoning techniques. First, a rule induction module was used to create a large number of rules with varying, usually low, levels of certainty, from a large, sparse database. Decision tree and multi-polynomial regression algorithms were employed in this step to induce

TABLE I SELECTED TARGET WEATHER VARIABLES, THEIR VARIABLE TYPE, THREE-DIGIT VARIABLE CODE, AND VARIABLE NAMES

Variable Type	Variable Code	Variable Name
Continuous	071	Ceiling (inversed)
	072	Visibility
	073	Sea level pressure
	074	Dew point temperature
	076	Wind speed
	080	Relative humidity
Ordinal	085	Thunderstorm
	086	Rain
	088	Drizzle
	091	Snow
	099	Fog
	101	Smoke

different types of rules. The intuitive reasoner then iteratively fired the resulting set of rules using the known values of variables as inputs. Each iteration was followed by a consolidation step in which corroborating intermediate data (e.g. rule outputs) were combined and the resulting values were assigned strengthened certainty levels. The iterative reasoning stopped when no more rules could be fired, and the outputs at that time were reported as the final conclusions.

The three principle characteristics of the knowledge involved in intuitive reasoning are variable certainty, fuzziness, and multi-valuedness. These characteristics are important aspects of the way the rule induction module and the intuitive reasoner manage information. The certainty level was represented by a metric called Strength of Belief (*SB*), which took a real number value in [0, 1], corresponding the range from no certainty to absolute certainty. Values of *SB* were associated with input information, rules and the outputs of any rules that were fired. *SB* increased if there was corroborating information, e.g., rules that described similar relationships among the same variables. Corroborating data or rules were consolidated, reinforcing each other and resulted in a single consolidated datum or rule with a higher *SB* than any of the contributing rules or data. The reinforced *SB* was calculated using the algebraic sum operator as in (1).

$$SB_{new} = Reinforce(SB_1, SB_2)$$
$$= SB_1 + SB_2 * (1 - SB_1), \quad (1)$$

where SB_1 and SB_2 correspond to the two original data or rules, and SB_{new} corresponds to the reinforced value of the consolidated datum or rule. The use of the algebraic sum operator in the *Reinforce* function ensures that the calculated value of SB_{new} never exceeds unity. Values of *SB* were assigned to basic rules and new values of *SB* were calculated for combined rules ("super rules") from the *SB* values associated with the basic rules that they replaced. Generally, the magnitude of *SB* assigned to a basic rule was proportional to the amount of its supporting data. The scheme for calculating the *SB* for basic rules was detailed in the next section. During reasoning, the reinforcement function was also called to compute *SB* during the consolidation of conclusions produced by the rules.

Fuzzy set theory was applied in this study to convert continuous weather variables into linguistic variables for the development of decision trees. The fuzzification process allowed to simulate human's use of linguistic terms in information processing. In addition, the resulting fuzzy variables had fewer distinct values than the original continuous variables, and might result in higher efficiency in subsequent rule induction. This effect is similar to data discretization in many data mining tasks [6].The fuzzification process was based on membership functions pre-defined on the domain of the continuous variable under consideration. Each real variable value was changed to the linguistic tag of the fuzzy subset to which it had the highest membership among all fuzzy subsets (Fig. 1).

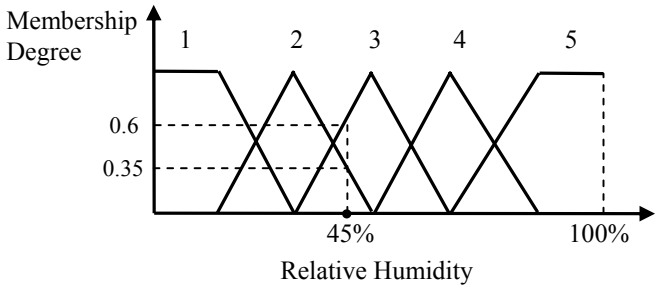


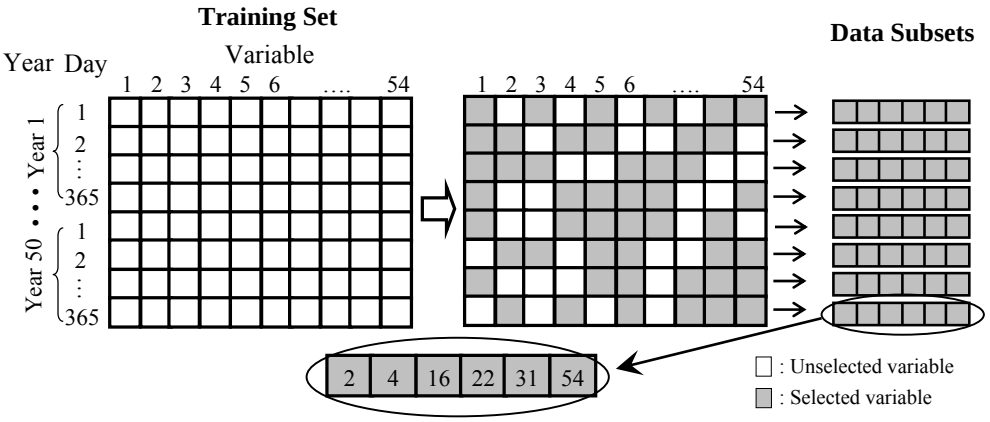
Fig. 1 An example of fuzzification for a real number valued variable, Relative Humidity. Five fuzzy subsets were defined and tagged by integers 1, 2, 3, 4, and 5, denoting "Very Low", "Low", "Medium", "High", and "Very High." The value 45% had the highest membership in the subset with fuzzy tag 3. It was hence converted to 3 with an associated membership degree, 0.6.

In this study, the tags for each fuzzy subset were represented by integers. The values of the fuzzified variables therefore appeared as integers, similar to the values of ordinal variables, except that each fuzzy variable value comprised not only an integer tag (indicating a fuzzy subset) but also an associated degree of membership (in that subset). For the sake of consistency and generality, the values of the ordinal and categorical variables, which also appear in the database as integers, were also interpreted in a fuzzy paradigm as fuzzy tags, but with membership values of 1. Each of these latter values represented a different class of the weather event to which the variable corresponded.

A "multi-valued" variable possesses more than one value simultaneously. Multi-valued variables might be presented to the intuitive reasoner as inputs or they might also result from the firing of rules that suggest distinct values for the same variable. This is one way to accommodate conflicting rules in the same rule set. Each value of a multi-valued variable has associated with it an *SB*, which is either assigned to that value prior to input or calculated from the *SB* of the rules and the inputs which produce the value.

IV. Rule Induction from Data Subsets

Records from 80% of the days in the 50-year database were randomly selected as the training set for rules generation and the rest were reserved as the test set. Rule induction proceeded as described earlier, whereby rules were generated from subsets of data sampled from the training set, i.e. only a small part of the training set was used to generate each rule. For each day, a fixed number of variables were randomly chosen and their hourly measurements for that day were extracted to form a data subset, from which a set of rules were induced (Fig. 2). This procedure was repeated for all the days represented in the training set. Six of the 54 variables were randomly chosen for each day and rules were induced from these data subsets using two commonly employed knowledge discovering tools: the multi-polynomial regression (MPR) and the classification and regression tree (CART) [7]. In this study, multi-polynomial regression models the relationships among independent variables and a continuous dependent variable using least-squared lines of specified orders to fit the data [8]. The resulting rules contained the coefficients of these lines and were used to predict continuous target variables. In this implementation, CART only operated on discrete variables, including those converted from continuous type variables, and created production rules that could be used to classify the dependent variables based on the values of a set of independent variables in the data subset. Both techniques, MPR and CART, were implemented using Matlab built-in functions. The rules induced from different data subsets were termed *basic rules* and were grouped into two basic rule sets, one for each rule type, from which identical or "similar" basic rules were replaced by representative rules, called *super rules*.



Decision Tree

Iteration	Dependent Variable	Independent Variable
1	2	4 16 12
2		4 16 31
3		4 16 54
⋮		⋮
10		22 31 54
11	4	2 16 22
12		2 16 31
13		2 16 54
⋮	⋮	⋮
51	54	2 4 16
⋮		⋮
60		16 22 31

Multi-Polynomial Regression

Iteration	Dependent Variable	Independent Variable
1	22	2 4
2		2 16
3		2 31
⋮		⋮
10		31 54
11	31	2 4
12		2 16
13		2 22
⋮	⋮	⋮
21	54	2 4
⋮		⋮
30		22 31

Fig. 2. Schematic representation of the extraction of data subset and iterative induction of rule. For each record in the training set, six variables were randomly chosen and extracted to form a data subset. Two techniques were use to induce rules from each data subset: the classification and regression tree (CART), which took three independent variables, and the multi-polynomial regression (MPR), which took up to two independent variables. Rule induction was carried out iteratively: in each iteration, a variable in the subset was designated as the dependent variable, a number (three for CART and one or two for MPR) of the remaining variables would be considered as the independent variables, and rule induction was performed by one of the two techniques. The example illustrates the process for a data subset which includes variables 2, 4, 16, 22, 31, and 54, where variables 22, 31, and 54 were continuous. As a result, 10 iterations were carried out for each dependent variable.

A number of characteristics of this rule induction process are noteworthy. First, all the data subsets that were extracted accounted for only part of the complete training set, and thus represented incomplete information about the task. Secondly, data subset extraction and rule induction were carried out on a "daily" basis: a number of rules were derived based from each record in the training set. This simulated a continuous learning scenario where new rules could be created and added to the

rule set whenever a small amount of new data became available. Thirdly, rule induction was independent of the task for which the rules were used later on. That is, during rule induction, the system had no "prior knowledge" of what the target variables it might be required to predict. Rules were therefore induced from all of the possible combinations of available in each data subset, resulting in a generic rule set that would be used to predict all target variables. Traditional approaches, by contrast, are usually task-specific. For instance, the reference model in this study required that, for each target variable, a group of "important" independent variables be identified prior to the creation of models for predicting that target variable. Lastly, with respect to the rule-based nature of this implementation, the initial data were not used further after the rule set was generated.

A. Basic Rule Induction

A first-order MPR was employed to establish linear relationships between a continuous dependent variable and the independent variables. In this study, the regression was done using either one or two independent variables. The process iterated through all combinations of variables in a given data subset, so that each time one of the continuous variables would be considered as the dependent variable and two of the remaining variables as the independent variables. In the example illustrated in Fig. 2, only variables 22, 31, 54 are continuous, and so only they would be considered as dependent variables during the process. The resulting rules contained the form of linear equations obtained from the least-square regressions. In addition, the number of data records involved in the creation of a rule was also included. For each iteration, as exemplified in Fig. 2, one linear relationship was found using both independent variables and two others were found, each using one of the two independent variables. An example is shown in (2), for a dependent variable (Y) and two independent variables, X_1 and X_2, resulting in the corresponding coefficients a_1, a_2, and a_0.

$$Y = a_1 X_1 - a_2 X_2 + a_0 \quad (2)$$

The classification part of CART was utilized to generate decision trees for discrete variables, from which production rules were then created. A rule is comprised of a set of conditions that are tested (premise), and an action that is performed when the premise is satisfied (consequent). A typical production rule in this study is of the form:

$$R_i\text{: IF } X_1 \# A_1 \text{ and } \ldots X_j \# A_j \text{ THEN } Y \text{ is } B, \quad j = 1, 2, \ldots, J, \quad (3)$$

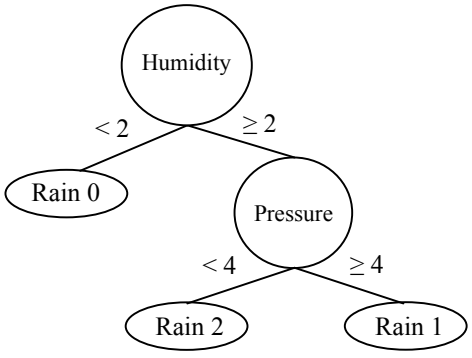
where the ith rule, R_i, has J conditions in its premise, $X_1, \ldots, X_j$ and Y are the independent variables and the dependent variable, $A_1, \ldots, A_j$ and B are fuzzy tags corresponding to fuzzy subsets defined on the domain of these variables, and # is one of the operators: <, ≥, and =. This rule assigns class B to Y when the J conditions are met.

CART is a data analysis algorithm that partitions the original data into groups through binary recursive splitting, so that, within each group, the class of the dependent variable can be determined by a set of independent variables. The resulting tree structure consists of a *root node*, which contains all the original data, and a number of *branches*, which consist of nodes and links. At the end of each branch is a *leaf node*. In each node, except the leaf nodes, data of that node are partitioned into two groups, each forming a child node (Fig. 3).

The splitting is aimed to sort the most data of one class of the dependent variable into the same child nodes. As the tree grows, the impurity of the dependent variable at the newly created nodes decreases. In a node, the splitting variable and its value used to split this node, the *split point*, are determined based on the Gini criterion: the combination of the splitting variable and the split point which leads to the biggest reduction in the Gini diversity index is chosen for splitting [7]. The Gini diversity index, $d(t)$, for a node t is calculated by (4):

$$d(t) = 1 - \sum_{i=1}^{k} p^2(i \mid t) \quad (4)$$

where k is the number of classes existing in that node and $p(i \mid t)$ corresponds to the relative frequency of class i in node t. The Gini diversity index of a node is biggest when all classes in the node are equally numerous and is minimal when the node contains only one class. To calculate the reduction in impurity for a split, the Gini diversity indexes of the parent node and the two child nodes are first calculated, then the mean of the indexes of the child nodes weighted by the number of records in each node is subtracted from the Gini diversity index of the parent node. The child nodes can themselves be split into more child nodes and this procedure continues until no further split can be made: all leaf nodes contain only one class or contained the same values of the independent variables. At the time, the leaf nodes represent the classification of the dependent variable.



R_1: IF Humidity < 2 THEN Rain 0
R_2: IF 2 ≤ Humidity and Pressure < 4 THEN Rain 2
R_3: IF 2 ≤ Humidity and 4 ≤ Pressure THEN Rain 1

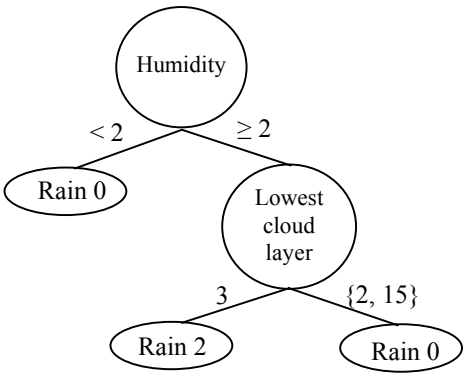
Fig. 3. An illustration of a decision tree and the resulting production rules. Circle represent non-leaf nodes of which the data are partitioned into two groups based on the variable specified in the circle and the conditions for the partition are denoted on the link to the corresponding child nodes. Each group forms a child node. At the end of each branch of the tree is a leaf node (oval) whereby the class of the dependent variable of that branch is determined, e.g., class "0" of Rain. Three leaf nodes are indicated here, which result in the three rules: R1, R2, and R3.

In this implementation, the CART classification procedure operated only on discrete data and continuous variables needed to be discretized (through fuzzification) in prior. The combinations of dependent and independent variables were similar as for the MPR procedure. For example, in Fig. 2, the values of variables 22, 31, 54 are converted into discrete fuzzy tags and each time one of the six variables 2, 4, 16, 22, 31, 54 was considered as the dependent variable and three of the rest the independent variables. After tree growing, each leaf node resulted in one production rule where the premise consisted of the test conditions corresponding to the tests along the path from the root node to this leaf node, and the consequent contained the class of the dependent variable as concluded in the leaf node (Fig. 3). When the data in a leaf node had diverse classes of the dependent variable, the process opted for the most numerous one as the representative class.

When an ordinal variable was used for splitting, the data in the parent node were sorted into one of the child nodes by comparing their value of the splitting variable to the split point, as shown in Fig. 3, where both Humidity and Pressure were ordinal variables. When a categorical variable was chosen to split the data, one child node would be assigned the data of which the value of the splitting variable equal to split point and the other node received the remaining data. For example, in Fig. 4, the categorical variable Lowest cloud layer type had three values in a node, 2, 3, and 15, and was chosen to split the data with the split point of 3. Note that this scheme might lead to the creation of rules with multiple values in it conditions, e.g., "IF 2 ≤ Humidity and Lowest cloud layer type = 2 OR 15 THEN Rain 0." This rule would be recorded in the rule base as two rules each with one value for the condition of Lowest cloud layer type, as R_2 and R_4 in Fig. 4. This assured that conditions in all production rules were of the same format and avoided complicated computation in the subsequent super rule induction and rule inference.

Rules which were induced from data subsets were "basic" rules. The basic rules were subject to a preliminary screening step during which unsatisfactory rules were discarded. A basic rule induced by MPR was retained only if it had an R-squared value of at least 0.5 and a p-value of less than 0.05. A tree-derived rule was retained only if its classification accuracy was at least 50%.

The *SB* associated with a basic rule was calculated based on the number of supporting data records represented by that rule. For instance, the *SB* assigned to an MPR-induced rule was calculated based on the number of data records used in the regression and the corresponding R-squared value. The *SB* assigned to a rule derived by CART was based on the number of data records in the class defined by the associated leaf node. For rule induced by CART, the *SB* was further adjusted by the averaged membership degree of fuzzy tags among all involved variables. For continuous variables, the values used in rule induction were converted from their real numbered values. Higher membership degrees meant the original real numbered values were better represented by the fuzzy tags. In other words, the relationship expressed in the rule in terms of fuzzy tags was better supported by these data. This adjustment term, therefore, could be considered as the average "weight" of the supporting data record. Data of discrete variables, given their membership degree was unity, always had a weight of one. Finally, the *SB* associated with each rule was multiplied by a scaling factor, so that *SB* values that were consolidated using the reinforcement operator would not easily reach the upper bound of 1. For this study, the scaling factor was set as 0.0001. During testing the reasoner, *SB* was used only as a relative measure, so the value of the scaling factor did not impact the reasoner's effectiveness.



R_1: IF Humidity < 2 THEN Rain 0
R_2: IF 2 ≤ Humidity and Lowest cloud layer type = 2 THEN Rain 2
R_3: IF 2 ≤ Humidity and Lowest cloud layer type = 3 THEN Rain 1
R_4: IF 2 ≤ Humidity and Lowest cloud layer type = 5 THEN Rain 0

Fig. 4. An illustration of node splitting based on a categorical independent variable, Lowest cloud layer type, and the production rule creation. Three values of this variable are present in this node: 2, 3, and 15. The value of 3 is found to be the split point and divides the data in that node into two groups: those with Lowest cloud layer type equal to 3 and the rest. Three leaf nodes result from this tree classification procedure and three rules are originally created. In the rule base, the rule "IF 2 ≤ Humidity and Lowest cloud layer type = 2 OR 15 THEN Rain 0" is recorded as two rules, R2 and R4, each with a single value in the condition for Lowest cloud layer type for the ease of computation.

Note that confliction in the rule base was not checked and resolved as in most other rule-based systems [9], where confliction was referred to the rules which had identical premise yet distinct conclusions for the same dependent variables. The reason for this was, because the induction in this study was based on incomplete data subsets, rules might appear conflicting yet actually represent no state of conflict. Consider the following two rules, R1 and R2, between which conflict would exist according to the above definition if only Humidity and Pressure have been included as the independent variables.

R1: IF 3 ≤ Humidity < 4 and Pressure = 6 and Temperature < 2 THEN Rain 0

R2: IF 3 ≤ Humidity < 4 and Pressure = 6 and 3 ≤ Temperature < 6 THEN Rain 1

This is possible if the data subset from which they were induced did not contain the data of Temperature. Conflicting

rules thus could be looked at as a trait of rules concluded from incomplete data sets, and their dependent variables would be considered as multi-valued variables.

B. Super Rule Induction

Identical or similar basic rules were consolidated into "Super rules", which represented repetitive scenarios recorded in the data archive. In this sense, super rules were of a higher abstraction than basic rules. Generating super rules and discarding the basic rules which they represented usually allowed the rule set to become more concise and the subsequent reasoning process that would make use of the rule set to be more efficient. Rules created from MPR and CART were of different formats and required different procedures for consolidation, or super rule creation.

1) MPR type rules

Basic rules obtained from MPR were consolidated into super rules using a fuzzy subtractive clustering algorithm. Basic rules for the same dependent and independent variables were first sorted into groups. For each group, clusters of similar basic rules were identified. The fuzzy subtractive clustering algorithm [10] is advantageous, on one hand, because it does not require the number of clusters to be specified in advance. On the other hand, a number of clustering parameters are required, e.g., cluster radius and penalty radius, on the basis of which the algorithm identifies as many clusters as can be postulate from the data and locates their centers. This algorithm was originally designed to operate on data points. However, in this study, it was used to identify similar basic rules, which were lines (Fig. 5). One modification was hence necessary: the mean squared distance between lines was substituted for the Euclidean distance between points. The parameter values were chosen based on preliminary trials conducted on a subset of the basic rules. The chosen parameter values resulted in the lowest averaged inter-cluster mean squared error among all trials.

Unlike a traditional clustering algorithm, in which an element is assigned binary membership to a cluster, this algorithm is fuzzy in that it allows a basic rule to be associated to multiple clusters with various degrees of membership. In this study, rules associated with a cluster by a membership degree greater than 0.5 were considered to belong to that cluster. If a rule belonged to more than one cluster, the one with the biggest membership was chosen as its cluster. The rule (line) which represented the center of a cluster then was considered as the consolidated ("super") rule that replaced all of the basic rules included in that cluster. Fig. 5 exemplifies this process for the dependent variable Station Pressure (kPa) and the independent variable Third Cloud Layer Type, represented by numerical codes ranging from 0 to 28. The original eleven basic rules in the group are drawn in Fig. 5a. Two clusters were found (Fig. 5b). The process selected an almost neutral relationship between the dependent and independent variables for the first cluster and a positive relationship for the second. Basic rules with high membership in one cluster had low membership in another (Fig. 5c). This indicates the two clusters were well separated. Since all basic rules in this group had membership of more than 0.5 in one of

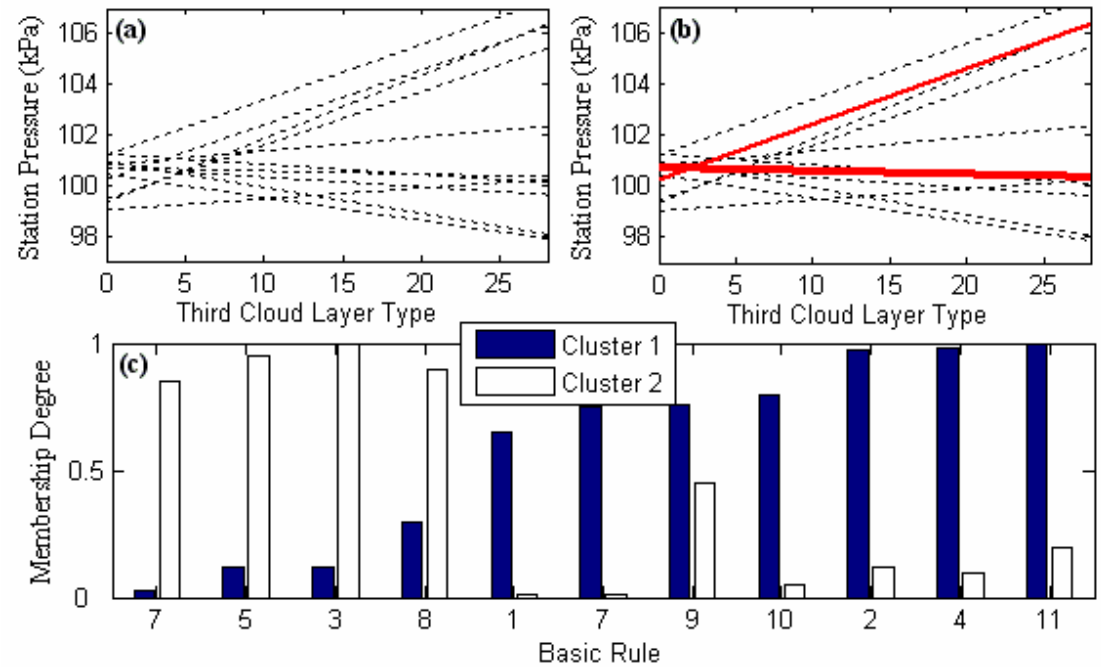


Fig. 5. Example of the consolidation of basic rules induced by MPR. The 11 dashed lines in (a) represent the 11 basic rules in a group. Two clusters were identified by a fuzzy subtractive clustering algorithm, the centers highlighted in (b), bold solid for the first cluster and plain solid for the second cluster. Fig. (c) illustrates the degrees of membership of the 11 rules in the two clusters. Rules are ordered in ascending order of their membership degree in the cluster 1. Rules 7, 5, 3, and 8 belong to cluster two and the rest belong to cluster one.

the clusters, they would all be removed from the rule set and would be represented by two consolidated ("super") rules.

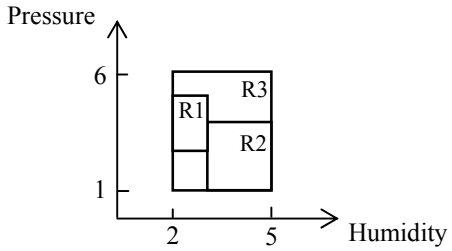
The basic rules belonging to a cluster were considered to reinforce the super rule representing that cluster. Therefore, the *SB* of a super rule was calculated as the reinforced *SB* of all of the associated basic rules, as if all lines corresponding to these member basic rules overlapping at the center, adjusted by how divergent this cluster was, as shown in (5).

$$SB_s = Reinforce(SB_{bn}, n = [1, N]) \times \overline{\mu} \quad (5)$$

where SB_s and SB_b stand for the *SB* of the super rule and the basic rules, respectively; N stands for the number of basic rules included in that cluster; $\overline{\mu}$ is the adjusting factor obtained by averaging the membership degrees of the N basic rules in that cluster.

2) CART type rules

The basic rules induced by the CART algorithm are production rules, where the premise consists of a number of test conditions and the dependent variable is assigned the class specified in the consequent if the test conditions are satisfied. In this application, groups of similar production rules were consolidated into "super rules". The consolidation procedure varied according to the degree of similarity among the basic rules in a given group. The first step was to group identical production rules and replace each group with a single, representative rules. Next, those rules that were not duplicated exactly were grouped together if their premises include some of the same independent variables with similar ranges of values. Also, a basic rule could be subsumed by another if the two rules had the same conclusion and the latter was more general [9]. In other words, a more specific rule could be subsumed if its condition were tighter or more numerous than those of the rule by which it was subsumed (Fig. 6). Any test data which satisfied the premised of the subsumed rule there satisfied those of the more general rule as well.



R1: IF 3 ≤ Humidity < 4 and 4 ≤ Pressure < 6 THEN Rain 1

R2: IF 4 ≤ Humidity < 6 and 2 ≤ Pressure < 5 THEN Rain 1

R3: IF 3 ≤ Humidity < 6 and 2 ≤ Pressure < 7 THEN Rain 1

Fig. 6. Examples of super rule creation from subsumed basic rules. Three basic rules, R1, R2, and R3, generate the same conclusion based on the dependent variable, Rain, but their premises differ with respect to Humidity and Pressure. These conditions are represented graphically as rectangles. R1 and R2 are more specific and are therefore subsumed by R3, which becomes the super rule of this group and R1 and R2 are removed from the rule set.

Groups of basic rules could also be consolidated by generalization, if they shared similar premises but yielded slightly different outcomes. Consolidating such a group of rules into a single super rule might increase the occurrence of misclassification because some of the rules being replaced could result in different conclusions than would the super rule. In addition, the premises of one rule might only partially overlap with those of others in the group, instead of being completed coincident, as in the case of subsumed rules. To determine whether or not a super rule should be generated from a group of basic rules, the ranges of the premise variables were extended to contain those of the contributing rules. If the majority of the contained rules had the same outcome, then a super rule was created to replace them (e.g., R1 in Fig. 7), as discussed in the following subsections.

A rule induction approach through generalization, RISE [11], was adopted and a generalization based procedure for creating super rules from similar basic rules was established. The objective of this procedure was to generate as many super rules while keep misclassification rate to minimum. Note that since subsumed rules were a special case of similar rules, they could also be identified and handled using this procedure. There was no need for extra procedures for subsumed rules. To start, basic rules with the same dependent variable were grouped together, and, each time, the procedure took one group as the input. The steps of the generalization procedure for a group are illustrated in Table 2. At first, the basic rule group *RB* was duplicated to become the initial set of super rules that were to be generalized, *RG*. During the procedure, *RB* would stay unchanged.

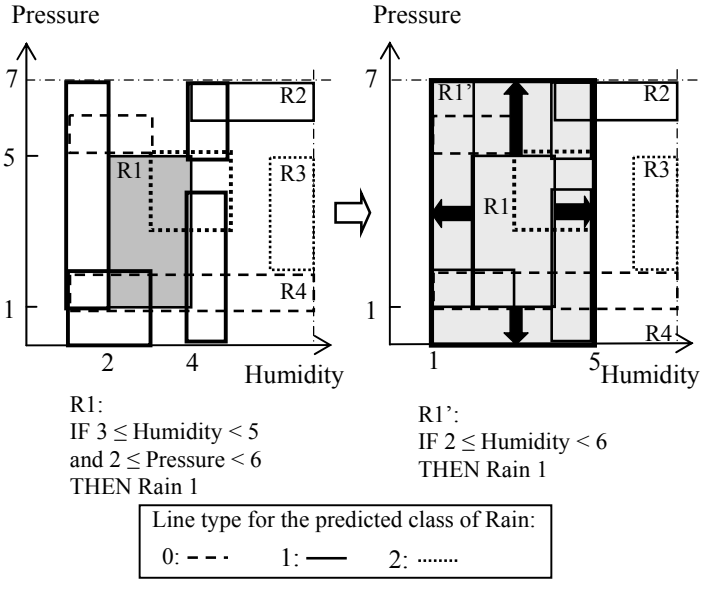


Fig. 7. Examples of rule consolidation through generalization. The dependent variable is Rain and the independent variables are Humidity and Pressure. The original basic rules are drawn in the left diagram with respect to the range of the conditions in their premise; different line types indicate their predicted classes of Rain. R1 was generalized to become R1', in the right diagram, that contained six basic rules, of which four had the same predicted class as R1'. R1' was thus a potential super rule. Note the range for Pressure was so

extended that it encompassed all seven classes of Pressure. That is, this condition would be always tested true and was removed from R1'. The procedure did not generalize R1' to further include R2 although it also predicted class 1. This was to avoid R1' from also including R3 and R4 with different predicted classes that would worsen the overall classification accuracy.

A super rule is called to "cover" a basic rule if the former has a more general premise than that of the latter; a super rule is called to "win" a basic rule if it is the nearest rule to that basic rule. When a basic rule was won by a given super rule, it was represented by that super rule. The definition of the distance between two rules is provided later. Next, each basic rule in *RB* was assigned to be won by its nearest super rule in *RG*. This created an initial state of representation. This initial state might exhibit a low overall classification accuracy, or high misclassification. However, as the procedure underwent, the classification accuracy would be gradually increased. The overall classification accuracy for a super rule set, *RR*, which represented *RB* was noted as *Acc*(*RR*) and was defined as the percentage of correctly classified basic rules.

A basic rule would not be won by its corresponding super rule. That is, in seeking a basic rule's nearest super rule, the procedure considered all rules except that basic rule's corresponding super rule in *RG*.

TABLE II THE STEPS OF THE GENERALIZATION PROCEDURE FOR CREATING SUPER RULES OF A GIVEN DEPENDENT VARIABLE

Input: *RB*, basic rules of a certain dependent variable
Output: *RG*, super rules of a certain dependent variable
Generalization based super rule creation
  Let *RG* be *RB*, set all rules in *RG* as active
  Assign each rule in *RB* to be won by its nearest rule in *RG* and calculate *Acc*(*RG*)
  Repeat
    For each active rule R_g in *RG*
* Find its nearest basic rule R_b with the same predicted class and not already covered by it
    IF no such basic rule is found, go to **
    Generalize R_g to R_g' to cover R_b
    IF generalization fails, go to **
    Update *RG* to *RG*' with R_g replaced by R_g'
    If R_s' does not win any new basic rules or *Acc*(*RG*') < *Acc*(*RG*), go to **
    If R_g' is identical to an existing rule in *RG*, go to **
    Let *RG* be *RG*', go to * with the next active R_g
** set this R_g inactive, go to * with the next active R_g
  End For
  Stop when no rules in *RG* are still active or when no further generalization can improve or sustain the overall classification accuracy
Let *RS* be the set of rules from *RG* which win no less than *Min_Rb_Win* basic rules
Calculate *SB* for super rules

TABLE III EXAMPLES OF THE COMMON AND UNCOMMON VARIABLES FOR DIFFERENT COMBINATIONS OF INDEPENDENT VARIABLES INVOLVED IN R_x AND R_y IN COMPUTING $\Delta(R_x, R_y)$. VARIABLES ARE REPRESENTED BY NUMBERS FOR SIMPLICITY

Example	1	2	3	4	5	6	7
Independent variables in R_x	13	13	13	13, 20	13, 20	13, 20	13, 20
Independent variables in R_y	13	1	1, 13	13	31	31, 44	13, 20
Common variables	13	N/A	13	13	N/A	N/A	13, 20
Uncommon variables	N/A	13	N/A	20	13, 20	13, 20	N/A

Identifying the nearest super rule involved computing the distance between two rules. The way the distance was computed was where the current procedure departed from RISE. For RISE, the input is a set of data points and distance is a measure of similarity between a data point and a rule. That is, a vector of values to a set of conditional intervals, e.g., between a point (Humidity 6, Pressure 3, Rain 1) and a rule "IF 4 ≤ Humidity < 6 and 1 < Pressure < 3 THEN Rain 1." For the super rule creation procedure in this study, on the other hand, the input was already a set of rules and the distance being computed was between two sets of intervals. In addition, data points and rules in RISE have the same variables. However, here rules were induced from different data subsets and usually shared few common independent values. Therefore, a new distance metric was established.

Let $R = (a_1, a_2, \ldots, a_A; c_R)$ be a rule, where a_i is the condition of the ith independent variable and c_R the predicted class. When there is no condition on the ith independent variable, a_i = True. For ordinal variables, a_i is $r_{i,lower} \le t_i \le r_{i,upper}$; for categorical variables, a_i is $t_i = r_i$, where t_i represents the value of the test case, $r_{i,lower}$ and $r_{i,upper}$ represent the lower and upper limits of the condition, and r_i the sole class specified in the condition. The distance between tow rules, R_x and R_y, is denoted as $\Delta(R_x, R_y)$ and is defined as the extent to which R_x needs to be generalized in order to cover R_y. It can be computed using (6):

$$\Delta(R_x, R_y) = \sum_{i=1}^{A} (\delta_{i,lower}^2 + \delta_{i,upper}^2) \quad (6)$$

where $\delta_{i,lower}$ and $\delta_{i,upper}$ are the lower and upper component distances for the ith independent variable involved in R_x.

The way to calculate δ_i for an independent variable was determined by whether that variable was a common variable or not. A *common variable* defined in calculating $\Delta(R_x, R_y)$ was a variable which appeared in both rules' premises, e.g., variable 13 in examples 1, 3, 4, and variables 13 and 20 in example 7 in Table 3. If it was an ordinary variable, the component distances were calculated as the amount the associated interval limits have to be extended to cover R_y, normalized by the class range of that variable:

$$\delta_{i,lower} = \begin{cases} 0 & if \quad r_{x,i,lower} \le r_{y,i,lower} \\ \dfrac{r_{x,i,lower} - r_{y,i,lower}}{c_{i,\max} - c_{i,\min}} & if \quad r_{x,i,lower} > r_{y,i,lower} \end{cases}$$

$$\delta_{i,upper} = \begin{cases} 0 & if \quad r_{x,i,upper} \ge r_{y,i,upper} \\ \dfrac{r_{y,i,upper} - r_{x,i,upper}}{c_{i,\max} - c_{i,\min}} & if \quad r_{x,i,upper} < r_{y,i,upper} \end{cases} \quad (7)$$

where $c_{i,max}$ and $c_{i,min}$ represent the maximum and the minimum classes for the ith variable. For common categorical variables, the component distance was obtained as:

$$\delta_{i,lower} = \delta_{i,upper} = \begin{cases} 0 & if \quad r_{x,i} = r_{y,i} \\ 1 & otherwise \end{cases} \quad (8)$$

Alternatively, an independent variable was an *uncommon variable* if it appeared in R_x but not in R_y, e.g., variable 20 in example 4 in Table 3. Because now R_x was more specific than R_y unless the conditions of uncommon variables were removed, the component distance was calculated as the amount of extension of the conditional interval that led it to cover all class range, i.e., a_i = True. For ordinary variables, (9) was followed:

$$\delta_{i,lower} = \frac{r_{x,i,lower} - c_{i,\min}}{c_{i,\max} - c_{i,\min}}$$

$$\delta_{i,upper} = \frac{c_{i,\max} - r_{x,i,upper}}{c_{i,\max} - c_{i,\min}} \quad (9)$$

For categorical variables, both $\delta_{i,lower}$ and $\delta_{i,upper}$ were assigned to be 1.

The order of the arguments given to $\Delta(R_x, R_y)$ is thus important. For example, $\Delta(R_1, R_2)$ and $\Delta(R_2, R_1)$ might have different outputs if the extents of generalization for R_1 to cover R_2 differs from that conversely.

Following a similar idea, to generalize a rule, R_g, in order to cover a basic rule, R_b, consisted of extending the limits of the conditions in R_g or removing certain conditions to make the resulting rule, R_g ', as general as or more general than R_b. A condition is called to be *removed* if its limits are so extended that the whole range of possible classes is covered. That is, the condition would be tested true no matter what the test value is. A generalization step could not remove all existing conditions of R_g. Detection of removing the last condition of R_g during generalization caused the step to be cancelled. In addition, since it suggests that rule would not be able to win its nearest basic rule through any further generalization, the rule was denoted as inactive and would remain unchanged until the end of the procedure. Otherwise, generalization on R_g was accepted as long as any new basic rules were won by R_g' and the global classification accuracy was either improved or sustained. The procedure iterated through all active rules in *RG* repetitively until none remained active or no further generalization could improve or sustain the overall classification accuracy.

When the iteration stops, super rules were identified from *RG* based on the number of basic rules they had won. An R_g had to win no less than *Min_Rb_Win* basic rules to become a super rule. Currently, *Min_Rb_Win* was set to 3. A super rule represented its corresponding basic rule and the basic rules it won through generalization, which formed the basis for calculating the SB for that super rule.

The *SB* of a super rule, R_s, contained three components: 1. The reinforced *SB* computed as if all *N* basic rules won by it had had identical premise, 2. The classification accuracy of this super rule when representing the *N* basic rules, $Acc(R_s)$, and 3. The complement of a generalization factor, F_g, as in (10):

$$SB_s = Reinforce(SB_{bn}, n = 1–N) * Acc(R_s) * (1\text{-}F_g) \quad (10)$$

The generalization factor measured the averaged dissimilarity between the premise of this super rule and those of the basic rules it won. It was assumed that the greater this factor, the less the involved basic rules contributed toward reinforcing the degree of certainty for the super rule. F_g was calculated as the square root of the summed distances between R_s and each of the *N* R_b, and normalized by the maximum possible number of independent variables, A_{max}, and *N*, as in (11):

$$Fg = \sum_{n=1}^{N} \sqrt{\Delta(R_{bn}, R_s)} \; * \frac{1}{A_{\max} \cdot N} \quad (11)$$

As discussed in the section of basic rule induction with CART, A_{max} was set to 3 in this implementation.

C. The Rule Base

The final rule set comprised super rules and the stand-alone basic rules not represented by any of the former. For example, a MPR type basic rule might exhibit too low a membership to be accepted by any cluster; a CART type basic rule could contain such a premise that no super rules could cover it while sustaining the overall classification accuracy. By and large, the super rules and basic rules represented, respectively, the frequent and exceptional association or causalities among weather variables that had been recorded in the data archive. They formed the base of knowledge for the intuitive reasoner to reason about related problems. A typical rule base contained 32,640 CART type rules, of which 6438 super rules, and 12937 MPR type rules, of which 590 were super rules. On average, the *SB* of a CART type super rule appeared 5 times larger than a CART type basic rule; for MPR, this number rose to about 15.

V. The Intuitive Reasoner

The reasoning process followed a forward-chaining rule inference scheme. A higher level of certainty in reasoning conclusions were sought by consolidating intermediate rule outputs through multiple runs of rule firing until no more rules could be fired. The results at that time became the final results. Each iteration consisted of two steps: rule firing and consolidation (Fig. 8). If the target variable was a continuous variable, rules induced by MPR were used for reasoning; otherwise production rules obtained by CART were used. In rule firing, un-fired rule were examined and fired if the following criteria were satisfied. For the MPR type rules, the independent variables were known and the computed dependent variable fell in a pre-defined range of reasonable values for that variable. Currently, the range of reasonable values for a variable was defined by the maximum and minimum values present in the data archive for that variable. For production rules, the premise was met. The *SB* of the rule output was computed by taking the minimum of the *SB* from the rule and from the input variables. For reasoning with the MPR type rules, all values involved were assumed to be crisp and thus no fuzziness was involved. For reasoning with production rules, the input values were tested against the conditions of the premise. A condition was satisfied if the test value has a non-zero membership in any of the eligible fuzzy tags within the range. For example, the condition "4 ≤ Pressure < 6" was interpreted as "Pressure = 4 or 5." The degree of membership for the output was derived following the Mamdani type fuzzy inference [12], i.e., the "max-min" composition with "min" as the implication function. However, the "de-fuzzification" step usually carried out in fuzzy controlling systems [13] to convert fuzzy tags to crisp numbers was not necessary here because the fuzzy tags were taken directly as the output. An example of rule inference which resulted in the class of Rain from Humidity and Pressure is illustrated in Fig. 9.

In the consolidation step intermediate rule outputs were integrated after rule firing. For reasoning with rules obtained by MPR, distinct output values of a variable were consolidated to produce one representative value of that variable. The representative value was calculated by taking the average of the outputs of the top *n* biggest *SB*, weighted by their *SB*. The consolidated *SB* was calculated from the *n*

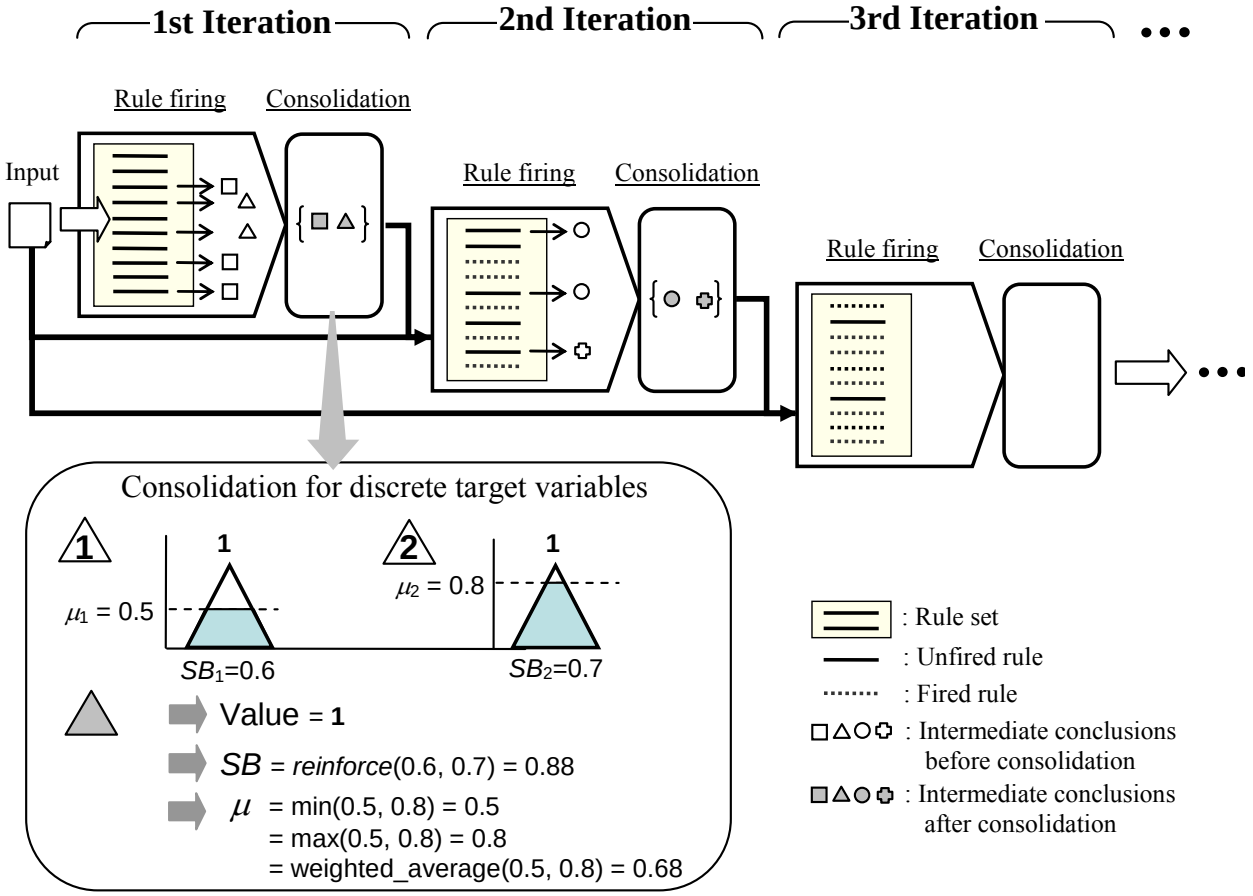


Fig. 8. A forward-chained rule-based intuitive reasoning scheme with production rules induced from CART. After rule firing, recurring outputs of the same values of a specific variable are consolidated so there is only one instance for each value of that variable. The consolidated values become inputs for the next iteration. For example, in the first iteration two rules might fire which result in fuzzy variables, the first giving a degree of membership of 0.5 (μ_1) in the class "1" and an associated SB_1 of 0.6, and the second giving a degree of membership of 0.8 (μ_2) in the fuzzy set "Medium" with SB_2 of 0.7. After consolidation, these two values will be combined into a single one, with an associated *SB* of 0.88 and a degree of membership (0.5, 0.8, or 0.68) in "Medium". The iterative process continues until no rules can be fired.

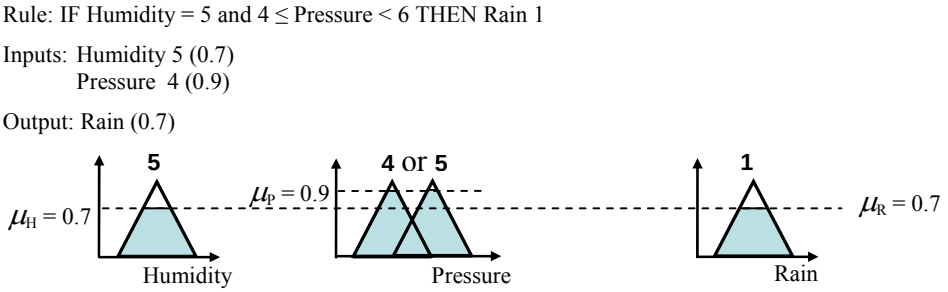


Fig. 9. An example of rule firing with Mamdani type fuzzy inference. The independent variables are Humidity and Pressure and the dependent variable is Rain. Inputs for Humidity and Pressure (membership in parentheses) are presented to the reasoner and the resulting degree of membership of the class of Rain is obtained as the minimum of the membership degrees of the inputs. Note that the condition for pressure is interpreted as "Pressure = 4 or 5." The output of this rule includes a fuzzy tag for the class of Rain, 1, its degree of membership, 0.7, and its *SB* (not shown here).

outputs using the *reinforce* function. For example, for a given variable, if *n* was set to two and the first two outputs with the highest *SB* were 10.5 with an *SB* of 0.8 and 12.3 with an *SB* of 0.6, the consolidated value became 11.3 and the reinforced *SB* became 0.92. This parameter of how many output values were included in calculating the consolidated value was a system parameter and could be set to different values to test its effect on reasoning accuracy.

For reasoning with production rules, rule firing might result in multiple outputs for one linguistic tag of a variable, each associated with a *SB* and a degree of fuzzy membership. These distinct *SB* and membership degrees were replaced by only one representative *SB* and one representative membership for that value. The consolidated (representative) *SB* for each value of a given variable was computed through reinforcement from the *SB* of the contributing outputs. The distinct degrees of membership were consolidated through fuzzy information aggregation. The aggregation operators chosen here were the conventional min and max, and, to allow a degree between the two extremes, a mean operator [14]. The mean operator was taken as the average weighted by the *SB*. Consider the example shown in Fig. 8, where two rules predicted class Medium for a particular dependent variable, yet with different *SB*, 0.6 and 0.7, and fuzzy membership degrees, 0.5 and 0.8. The consolidated *SB* became 0.88 and the three aggregation operators resulted in consolidated membership degrees of 0.5, 0.8, and 0.68 respectively.

The consolidated conclusions from one iteration, along with the originally given variables, became the inputs for the rule firing of the next iteration. The process continued until no rules were fired in an iteration. The intuition reasoner then presented the final results, which contained all of the values produced for each variable and their corresponding *SB* and, if the variables are fuzzy, an associated degree of membership.

VI. The Reference Reasoner

The reference reasoner in this study served as a reference for evaluating how well the target variables could be modeled by other weather variables observed on the same hour. If adequate relationships cannot be established, the design of this implementation scheme will be of little value. The approach proposed by Burrows [4] was adopted for building the reference case because it was similar to the intuitive reasoner in the following ways: 1. It had also been applied to a large weather archive to develop predictive models for certain target variables, 2. The classifier in this approach was also base on CART, and 3. The predictions for continuous target variables and discrete target variables were also made in separate routines. The results of this technique were considered as a suitable benchmark for evaluating the proposed intuitive reasoner. The data analysis resulted in one model for each of the 12 selected target variables. The major steps for creating the predictive models for the reference reasoner are listed in Table 4, where the particular methods used at each step were noted in parentheses

The procedures for discrete and continuous target variables shared the first part, where decision trees were generated using the CART algorithm. Like in rule induction for the intuitive reasoner, the values of continuous variables were converted into integers, each representing a class of the weather condition with which the weather variables were associated. For discrete target variables, the CAIM discretization algorithm [15] was adopted to define classes by determining intervals over the data range. It was a supervised procedure and postulated that, for each independent variable, the minimum numbers of classes were created and the interdependencies between that independent variable and the target variable were maintained. For continuous target variables, a simple equal-width discretization was employed. Next, irrelevant independent variables were identified and removed. Pair-wise dependence tests was conducted between the target variable and each independent variable, and an independent variable was deemed as irrelevant and was excluded from the data set if the null hypothesis — the two variables were mutually independent — could not be rejected with an alpha level of 0.01. If the test involved a categorical variable, the chi-square test was used; otherwise, the Kendall's Tau test was used. The remaining variables were taken for decision tree induction. The resulting decision tree became the predictive model for a discrete target variable. For a continuous target variable, the procedure continued to build a fuzzy inference system (FIS) as the predictive model.

"Important" independent variables were identified from those used in the decision tree and were included in developing the FIS. The fuzzy subtractive clustering algorithm was then employed to create clusters, each resulting in one equation for calculating the target variable. The optimal parameters of these equations were estimated through a least-squares method [16].

TABLE IV MAJOR STEPS OF THE WEATHER DATA ANALYSIS FOR THE REFERENCE REASONER

Procedure for discrete target variables
Repeat for each target variable
Discretize continuous independent variables (CAIM)
Conduct dependency tests between the target variable and each independent variable (Kendall's Tau test, Chi-square test)
Remove independent variables with no or low dependence with the target variable
Generate the classification tree for this target variable (CART)
Procedure for continuous target variables
Discretize continuous independent variables (equal-width discretization)
Repeat for each target variable
Conduct dependency tests between the target variable and each independent variable (Kendall's Tau test, Chi-square test)
Remove independent variables with no or low dependence with the target variable
Generate the pruned classification tree for this target variable (CART)
Identify important independent variables from the resulting decision tree
Create a fuzzy inference system for the target variable (fuzzy subtractive clustering, least squares estimation method)

VII. TESTING PLAN AND RESULTS

A five-fold simulation scheme was employed: the weather data set was randomly partitioned into five equal-sized subsets, each subset being the testing data set for one fold and the remaining four subsets used for rule induction, and such activities were repeated five times. For each of the 12 target variables, 100 test cases were created from records randomly chosen from the testing data set. A test case consisted of the values of a set of input variables from which the reasoners predicted the value of the target variables. For the reference reasoner, the relevant input variables for predicting a given target variable were identified during model development. For the intuitive reasoner, however, this information was not produced during rule induction. In this implementation, the intuitive reasoner was provided with the same set of input variables as used in the reference reasoner for each target variable. A heuristic for identifying important input variables for the intuitive reasoner is under development and will be presented in a follow-up article. Reasoning about continuous and discrete target variables postulated different rule sets or models and was carried out in separate reasoning sessions. Also, the fast and broad reasoning of the intuitive reasoner were carried out individually.

A. Class Imbalance Problem in Predicting Discrete Target Variables

The discrete weather variables, e.g., Rain, commonly exhibit an uneven class distribution. For example, for variable Rain, the class which represents nonevent accounts for 95% of all the data, while the other three classes which corresponds to various intensity of rain account of only 5%. Classifiers created from an imbalanced data set tend to over-fit the majority classes while ignoring the rare classes. This is called the class imbalance or class skew problem [17]. A test run of the intuitive reasoner revealed that it usually failed to predict the rare classes due to the presence of imbalanced variables in the training data set. To remedy this problem, a heuristic in which rules' *SB* was adjusted to offset the impact from uneven class distribution was developed. This approach attempted a similar effect of one common solution for this problem: re-sampling [18].

Re-sampling is aimed to create a data set with designed instead of natural class distribution for classifier development. Two sampling schemes are available: down sampling, which involves discarding certain majority class data, and up sampling, which involves duplicating some minority class data, until in the resulting data set the classes are represented by a pre-defined, desired ratio [18]. This procedure, however, was not readily suitable for this study because, for the intuitive reasoner, the raw data were assumed inaccessible after rules were induced from them, rending impossible a re-sampling operation. As a result, a solution based on the resulting rules was in need.

An infrequent class, e.g., "3," for the rare event of "heavy rain" of variable Rain, registered fewer records in the data set and, in turn, usually appeared in fewer rules than the majority classes (Fig. 10). As a result, during reasoning, the SB for the majority classes that were reinforced from the overwhelming amount of rules often outweighed that of the minority classes and made the majority classes seem to be the most certain prediction, even when actually they were not. That the amount of the supporting data from which each rule was induced, termed "rule support" hereafter, was recorded in the rule set and was utilized here: the proposed approach updated rules' SB based on their support to counterbalance the difference in rule abundance for different classes. The effect of re-sampling thus might be achieved without physically changing the size of supporting data.

Each time, the heuristic ran for the rules of one discrete weather variable. It started with calculating the summed rule support for each class and then identified the median of these summations. Next, for each class, an adjustment factor, which

equaled to the ratio of the median to the summation of rule support for that class, was multiplied to the value of SB of individual rules for that class to obtain the new values of rule SB. For the infrequent classes, the adjustment factors were greater than unity and the adjustment would increase the associated rules' SB value, resembling an up-sampling operation. For the majority classes, on the other hand, the adjustment factors were below unity and the adjustment worked toward the effect of a down-sampling operation. Ideally, the result of this procedure was a rule set whereby variable classes were more balanced than in the original rule set in terms of the value of SB of the rules.

To examine the effectiveness of this heuristic, for discrete target variables, the intuitive reasoner was tested with both the original rule set and the new rule set with adjusted rule SB. Furthermore, when choosing records from the testing data set to create the test cases, the sampling process was supervised to ensure a balanced discrete target variable in the test cases. Otherwise, the test cases tended to be dominated by the majority classes and the reasoners' ability to predict rare cases would not be tested.

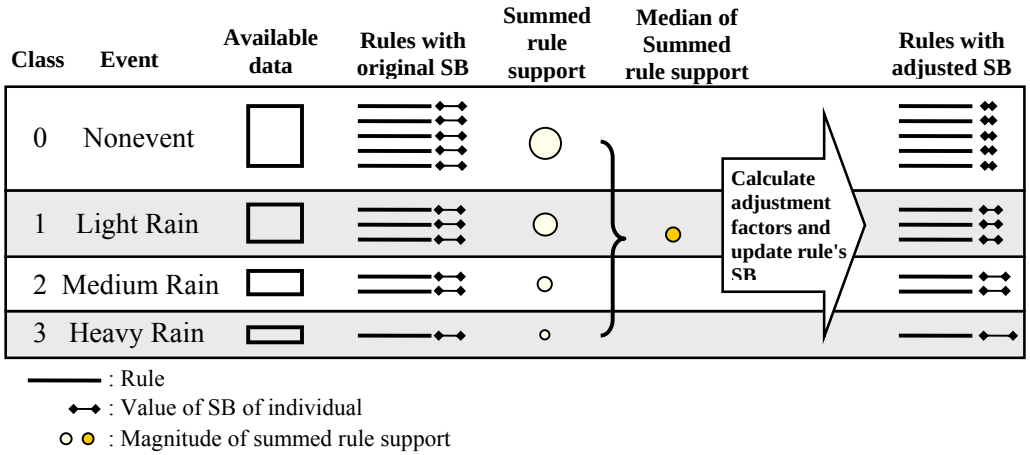


Fig. 10. An illustration of the class imbalance scenario for variable Rain and the heuristic of rule SB adjustment. The four classes, 0, 1, 2, and 3, correspond to four events of Rain recorded in the weather archive. Uneven class distribution exists in the data set and is carried out to the induced rule set, where the majority classes are usually predicted by more rules than the minority classes. For the sake of simplicity, all rules in this illustration are illustrated to originally have the same value of SB. Each time the heuristic operates for one dependent variable. For each class, the value of SB of individual rules is multiplied by a factor that is calculated as the ratio of the summed rule support for that class to the median found from the by-class summations of the rule support for all classes.

B. Simulation Results for Continuous Target Variables

The error rate and the correlation between the predicted and the actual values were used for evaluating the reasoners for predicting continuous target variables. The error rate was obtained, for a specific target variable, by dividing the averaged error between the actual and the predicted data from the results of the 100 test cases by the data range of reasonable values as defined in Section 5. It indicated the magnitude of difference between the predicted and the actual data. The correlation, on the other hand, was a measure of the linear dependency between the two groups of data.

A series of preliminary test runs were conducted to determine a better parameter n in combining intermediate outputs during consolidation, as discussed in Section 5. Three values of n, 1, 2, and 3, were used in three distinct runs. The value 1 appeared to provide the best prediction accuracy and thus the parameter was set to 1 in the real tests. This means the reasoner always took the sole intermediate output with the maximum SB as the consolidated value. As a result, the broad reasoning tended to be dominated by the outputs of the super rules and its conclusions should converge to that of the fast reasoning. Therefore, for the sake of simplicity, only the results of the fast reasoning are presented here for the intuitive reasoner.

The mean and standard deviation of the error rate and the correlation over the five runs are presented in Table 5. The reference reasoner reached error rates below 5% and correlations close to unity for variables 071 (Ceiling, inversed), 073 (Sea Level Pressure), 074 (Dew Point Temperature), and 080 (Relative humidity). The intuitive reasoner predicted almost as accurately as the reference reasoner for variables 073 and 074.

Figure 11 illustrates a detailed comparison between the predicted values and the actual values for variables 071 (Ceiling, inversed), 072 (Visibility), and 074 (Dew Point Temperature) for each test case. For variable 071, the predictions made by the intuitive reasoner were consistently below unity and it failed test cases with big values. For variable 072, the intuitive reasoner was accurate for a number of test cases. When it missed, it tended to overestimate the values. The intuitive reasoner was a very good predictive tool for variable 074 with usually miniature errors for most of the test cases presented to it.

TABLE V THE MEAN AND STANDARD DEVIATION OF THE ERROR RATES AND THE CORRELATIONS BETWEEN THE PREDICTED VALUES AND THE REAL VALUES FOR THE CONTINUOUS TARGET VARIABLES, CALCULATED FROM THE RESULTS OF THE FIVE TEST RUNS

	Intuitive Reasoner				**Reference Reasoner**			
Target Variable	Mean of Error Rate (%)	STD of Error Rate (%)	Mean of Correlation	STD of Correlation	Mean of Error Rate (%)	STD of Error Rate (%)	Mean of Correlation	STD of Correlation
071 Ceiling (inversed)	7.02	2.01	0.32	0.25	3.52	2.01	0.93	0.05
072 Visibility	11.06	1.44	0.57	0.21	7.72	1.11	0.74	0.03
073 Sea Level Pressure	0.04	0.00	1.00	0.45	0.03	0.00	1.00	0.00
074 Dew Point Temperature	2.71	0.58	0.98	0.43	0.18	0.02	1.00	0.00
076 Wind Speed	23.26	3.75	0.04	0.13	10.51	1.12	0.57	0.08
080 Relative Humidity	18.56	0.72	0.30	0.14	1.65	0.17	0.99	0.00

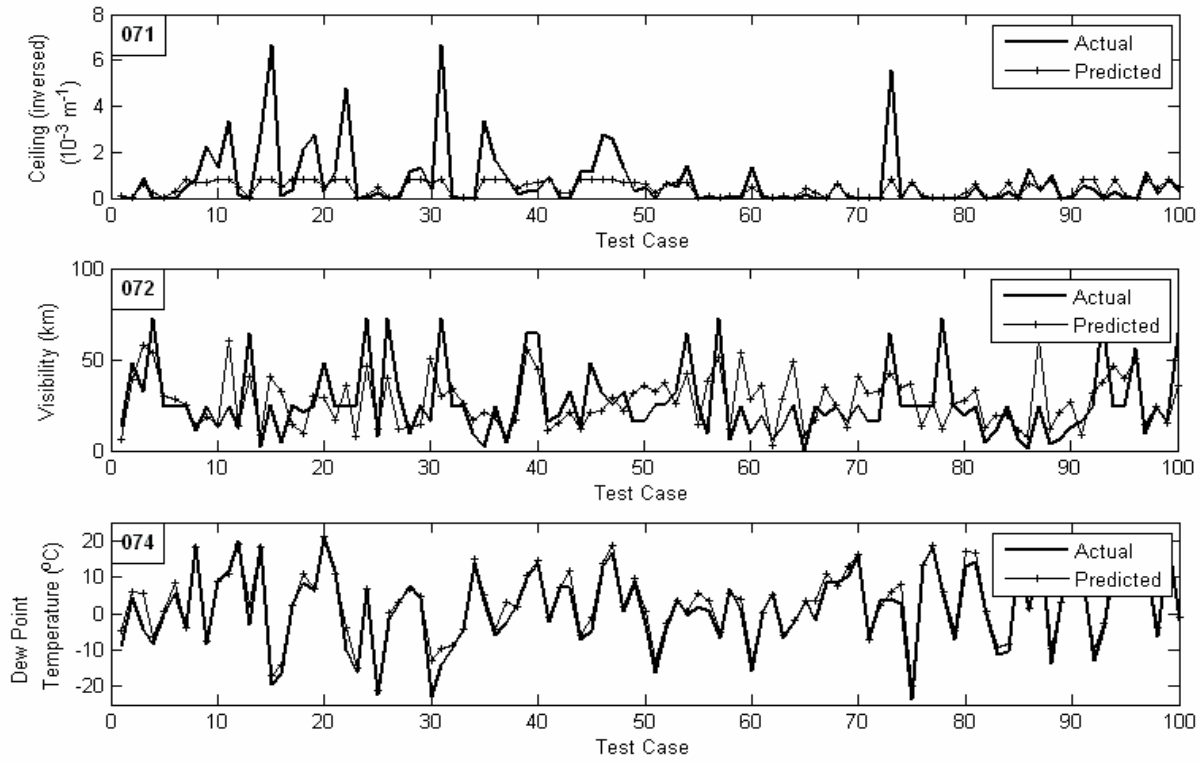


Fig. 11. Detailed by-case predicted values vs. actual value comparison for target variables 071 (Ceiling, inversed), 072 (Visibility), and 074 (Dew Point Temperature).

C. Simulation Results for Discrete Target Variables

The classes of discrete target variable were predicted from the input variables given in the test cases. The reference reasoner reported the class which was statistically the "best" estimation of the target variable in terms of the least prediction error [7]. Alternatively, the output of the intuitive reasoner included more information: a list of classes each associated with an *SB* and a fuzzy membership degree. An example of one such output for target variable 015, Thunderstorm, is shown in Table 6. The content in Table 6 might read "the intuitive reasoner is certain, with an SB of 0.82, about a state of thunderstorm that is 0.92 compatible to the nonevent, with an SB of 0.63 about a state that is 0.85 compatible to a class-2 thunderstorm, and with an SB of 0.32 about a state that is 0.87 compatible to a class-3 thunderstorm." Note that the low-SB classes are also reported so that a final decision might be made through conflict resolution. For example, the decision maker might decide to take the actions corresponding to a class-3 thunderstorm, thought the certainty level about its occurrence is not the highest, because the cost of misclassifying a class-3 thunderstorm is high.

The classification accuracy was used to evaluate the reasoning results, which was calculated as the percentage of the correctly predicted test cases. For intuitive reasoner, the class with the highest *SB* was taken as the answer in calculating the classification accuracy so a comparison of prediction performance with the reference reasoner could be made.

TABLE VI A TYPICAL OUTPUT FROM THE INTUITIVE REASONER FOR DISCRETE TARGET VARIABLE 015, THUNDERSTORM

Target Variable	Class	SB	Fuzzy Membership Degree
015	0	0.82	0.92
015	2	0.63	0.85
015	3	0.32	0.87

For each fold, five tests with different reasoners or settings were conducted with the same test cases: the reference reasoner, fast and broad reasoning of the intuitive reasoner based on the rules with SB adjusted for the class imbalance problem, and fast and broad reasoning of the intuitive reasoner with the original rule set. The results of these tests are compared in Fig. 12 in terms of their averaged classification accuracy over the five folds. The reference reasoner again demonstrated higher accuracy than any of the four intuitive reasoner settings, except for variable 088 for which the fast type intuitive reasoner demonstrated the best prediction accuracy. For the intuitive reasoner, the rules with adjusted SB appeared to improve the averaged classification accuracy by from 6% to 30%. The broad type outperformed the fast type reasoning for three target variables, 086, 091, 101. They delivered similar results for target variables 085 and 099. When the results were broken down to the class level, it appeared that fast reasoning produced lower accuracy when it failed more of the relatively rare classes than the broad reasoning. An example is given in Table 7, where the true positive rate and the false positive rate [19] for each class of variables 088 and 091 resulted from the two types of intuitive reasoning as well as the reference reasoner are compared. The statistics are drawn from the result of one fold and the intuitive reasoning was carried out based on rules with

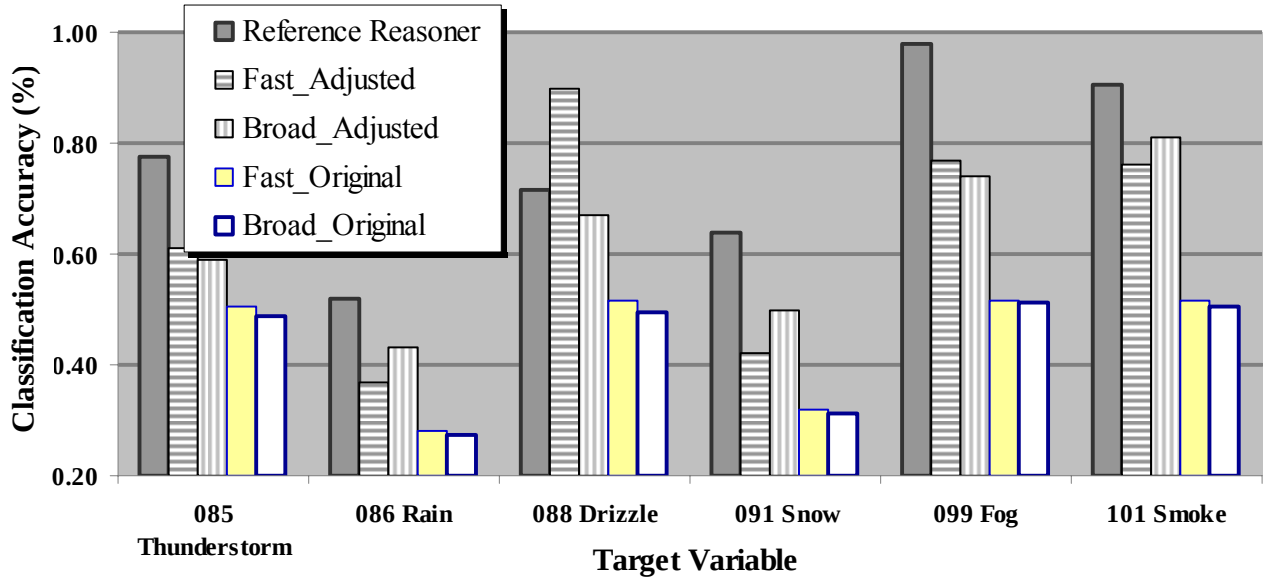


Fig. 12 Comparison of the classification accuracy for the discrete target variables among the reference reasoner and four different settings of the intuitive reasoner: fast and broad reasoning based on rules with adjusted SB (Fast_Adjusted and Broad_Adjusted) and fast and broad reasoning based on rules with the original SB (Fast_Origianl and Broad_Original). The numbers shown are averages from the results of the five folds.

TABLE VII TRUE POSITIVE RATE (TPR) AND FALSE POSITIVE RATE (FPR) COMPARISON BY TARGET VARIABLE AND CLASS AMONG TWO TYPES OF INTUITIVE REASONING AND THE REFERENCE REASONER. NUMBERS ARE OBTAINED FROM THE RESULTS OF ONE FOLD AND THE INTUITIVE REASONING WAS BASED ON RULES WITH ADJUSTED SB

Target Variable	Class	Intuitive Reasoner Fast			Intuitive Reasoner Broad			Reference reasoner		
		TPR	FPR	Accuracy	TPR	FPR	Accuracy	TPR	FPR	Accuracy
088	0	0.96	0.16	0.90	1.00	0.66	0.67	1.00	0.52	0.74
088	1	0.84	0.04		0.34	0.00		0.48	0.00	
091	0	0.76	0.03	0.42	0.64	0.03	0.50	1.00	0.09	0.59
091	1	0.92	0.75		0.52	0.11		0.72	0.36	
091	2	0.00	0.00		0.84	0.53		0.44	0.09	
091	3	0.00	0.00		0.00	0.00		0.20	0.00	

adjusted SB. For target variable 088, the fast type intuitive reasoner was able to correctly predict more test cases of class 2 than the other two reasoners and reach an overall higher accuracy. However, for target variable 091, the fast reasoning misclassified all infrequent classes, 2 and 3 and resulted in an accuracy rate below 0.5.

VIII. Discussion

It appears that the relationships among same-hour weather variable measurements could be properly modeled by the employed weather data analysis scheme (the reference reasoner), at least for the majority of the selected target variables. This marked the cornerstone for the following tests and comparison because otherwise these activities would be meaningless. As for those target variables whose values were not predicted with high accuracy by the reference reasoner, data of more weather variables might be needed or the dynamics of inter- or intra-variables, e.g., temporal autocorrelation or multivariate correlation, need to be studied, either case is beyond the implementation scope of this project and will not be further addressed.

The reference reasoner outperformed the intuitive reasoner in all categories except for one target variable. However, the intuitive reasoner matched the performance of the reference reasoner for two continuous target variables and reached at least 70% of the classification accuracy of the reference reasoner for all six discrete target variables. This result, given that the intuitive reasoner operated on rules induced from only about 10% of the data used by the reference reasoner, is reasonable.

For the classification task, the fast reasoning performed better than or almost equally well as the broad reasoning. This result manifests that by consolidating many identical or similar basic rules to create fewer, highly abstract super rules, the system might gain efficiency without the cost of performance when operating on these super rules. Due to the small body of learning data, the information of certain infrequent classes of some variables was not available and thus those classes were not represented in the resulting rule sets. For example, for variable 091, class 2 and 3 were not represented in the super rule set, neither was class 3 in the basic rule set. As a result, the intuitive reasoner failed to predict them during testing, particularly the fast type reasoning, which operated on the super rules that represented only frequent events in the data. Compared to the fast type reasoning, the broad type reasoning was at times, more successful in predicting rare events. For these infrequent classes which were not represented even in the basic rule set, more data were needed. This might be done through supervised data collection and rule induction to ensure the system has the necessary information about infrequent events.

IX. Conclusion and Future Work

The overall objective of this project was to investigate approaches to construct a computer system that can induce rules from low-quality data and solve problems with these rules by mimicking some aspects of human intuition. In this paper, the implementation of a basic intuitive reasoner is presented. The rule induction scheme was able to induce rules from discrete chunks of data existing in a sparse database. The rule-based intuitive reasoner was capable of solving questions with a set of rules that generated multiple outputs and were fuzzy and limited in certainty. This suggests that by consolidating the outputs of numerous low-quality rules, an "intuitive" reasoner can effectively perform computational tasks, such as classification, on the basis of incomplete information of variable quality. Computationally, the proposed approach is advantageous when the available data set is very sparse and heterogeneous, whereby different kinds of data are available in different records. Compared to conventional knowledge exploration methods such as artificial neural network, which usually require complete data sets [20], this approach appears more robust and saves effort and time that might be otherwise spent on database imputation.

The current intuitive reasoner has no knowledge in distinguishing relevant input variables from trivial ones. Since the original data are not further used after rules are induced from them, traditional variable selection schemes that depend on the data might not be readily useful to this end. A new approach that can identify important input variables based on the rules instead of the data is under study and its implementation will better complete the system. In addition, for discrete target variables, the *SB* adjustment factors for all variables were derived uniformly from the median of the by-class summed values of SB. Obviously this approach resulted in different extents of effectiveness for various target variables. A more sophisticated adjustment algorithm that calculates different adjustment factors for individual variables based on the corresponding rules and their SB values might further improve the classification accuracy. Also, it would be interesting to implement more knowledge discovering techniques, e.g., higher order MPR, different tree splitting schemes, time series analysis, or association rule exploration, and examine how or if the reasoning engine would gain in performance.

References

[1] R. M. Hogarth, *Educating Intuition*. Chicago, IL: The University of Chicago Press, 2001.
[2] M. Gladwell, *Blink: The Power of Thinking without Thinking*. New York, NY: Little, Brown and Company, 2005.
[3] Y.-C. Sun and O. G. Clark, "A computational model of an intuitive reasoner for ecosystem control," unpublished.
[4] W. R. Burrows, "Combining classification and regression trees and the neuro-fuzzy inference system for environmental data modeling," in *Proc. 18th International Conf. of the North American Fuzzy Information Processing Society*, New York, 1999.
[5] G. Latini and G. Passerini, *Handling Missing Data: Applications to Environmental Analysis*. Billerica, MA: Computational Mechanics Inc., 2004
[6] H. Liu, F. Hussain, C. L. Tan, and M. Dash. (2002). "Discretization: an enabling technique," *Data Mining and Knowledge Discovery*, vol. 26, pp 393–423, 2002.

[7] L. Breiman, J. Friedman, C. J. Stone, and R. A. Olshen, *Classification and Regression Trees*. Belmont, CA: Wadsworth International Group, 1984.
[8] G. W. Snedecor and W. G. Cochran, *Statistical Methods*. Ames, IA: The Iowa State University Press, 1967.
[9] J. P. Ignizio, *Introduction to Expert System*. New York, NY: McGraw-Hill, Inc., 1991
[10] K. Demirli, S. X. Cheng, and P. Muthukumaran, "Subtractive clustering based modeling of job sequencing with parametric search," *Fuzzy Sets Systems*, vol. 137, pp 235–270, 2003.
[11] D. Pedro, "Unifying instance-based and rule-based induction," *Machine Learning*, vol. 24, pp 141–168, 1996.
[12] E. H. Mamdani and S. Assilian, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Human-Computer Studies*, vol. 51, pp 135–147, 1999.
[13] W. Pedrycz, *Fuzzy control and fuzzy systems* (2nd, extended ed.). Taunton, UK: Research Studies Press Ltd., 1993.
[14] A. Loskiewicz-Buczak and R. E. Uhrig, "Information fusion by fuzzy set operation and genetic algorithms," *Simulation*, vol. 65, pp 52–66, 1995.
[15] L. A. Kurgan and K. J. Cios, "CAIM discretization algorithm," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, pp 145–153, 2004.
[16] S. L. Chiu, "A cluster estimation method with extension to fuzzy model identification," In *Proc. of the 3rd IEEE World Congress on Computational Intelligence*, vol. 2, pp 1240–1245, 1994.
[17] S. Visa and A. Ralescu, "Issues in mining imbalanced data sets - a review paper," in *Proc. of the 16th Midwest Artificial Intelligence and Cognitive Science Conf.*, Dayton, 2005.
[18] A. Estabrooks, T. Jo, and N. Japkowicz,. "A multiple resampling method for learning from imbalanced data sets," *Computational Intelligence*, vol. 20, pp 18–36, 2004.
[19] M. Stäger, P. Lukowicz, and G. Tröster, "Dealing with class skew in context recognition," in *Proc of the 26th IEEE International Conf. on Distributed Computing Systems Workshops*, Lisboa, Portugal, 2006.
[20] C. M. Ennett, M. Frize, and C. R. Walker, "Influence of missing values on artificial neural network performance," *Medinfo*. V. L. Patel, R. Rogers and R. Haux. London, UK: IOS Press, 2001.