SELF LEARNING REAL TIME EXPERT SYSTEM

Latha B. Kaimal[1], Abhir Raj Metkar[2], and Rakesh G[3]

[1,2,3] Control & Instrumentation Group, C-DAC,Thiruvanathapuram

ABSTRACT

In a Power plant with a Distributed Control System (DCS), process parameters are continuously stored in databases at discrete intervals. The data contained in these databases may not appear to contain valuable relational information but practically such a relation exists. The large number of process parameter values are changing with time in a Power Plant. These parameters are part of rules framed by domain experts for the expert system. With the changes in parameters there is a quite high possibility to form new rules using the dynamics of the process itself. We present an efficient algorithm that generates all significant rules based on the real data. The association based algorithms were compared and the best suited algorithm for this process application was selected. The application for the Learning system is studied in a Power Plant domain. The SCADA interface was developed to acquire online plant data.

KEYWORDS

Machine learning, Data Mining, Root cause analysis, Inference Engine, Tertius algorithm

1. INTRODUCTION

Machine Learning Technique is a scientific discipline that is concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data, such as from sensor data or databases. A major focus of machine learning research is to automatically learn to recognize complex patterns and make intelligent decisions based on data; the difficulty lies in the fact that the set of all possible behaviors given all possible inputs is too large to be covered by the set of observed examples (training data). Hence the learner must generalize from the given examples, so as to be able to produce a useful output in new cases[1][2].

Knowledge Based System (KBS) has an important role to play, particularly in fault diagnosis of process plants, which involve lot of challenges starting from commonly occurring malfunctions to rarely occurring emergency situations.

Machine Learning is discovering new knowledge from large databases (Data Mining). The field of machine learning is concerned with methods that can automatically learn from experience. The experience is usually given in the form of learning examples from which machine learning methods can automatically learn a model. Sometimes it is very important that the learned model can be read by human, and one of the most understandable types of models are rule sets[3][4].

Expert systems are Knowledge-Based Systems(KBS) in which The Knowledge Base(KB) holds information and logical rules for performing inference between facts. The KBS approach is promising as it captures efficient problem-solving of experts, guides the human operator in rapid fault detection, explains the line of reasoning to the human operator, and supports modification and refinement of the process knowledge as experience is gained. Real time Expert System Shell is a software development environment containing the basic components of expert systems. The Shell Toolkit can be used to develop Intelligent System for Diagnosis and Root Cause Analysis and operator guidance in process plants etc. The Expert System when enhanced with Learning Engine, updates the KB with new learned rules & thus improves the efficiency of the system in root cause analysis of faults.

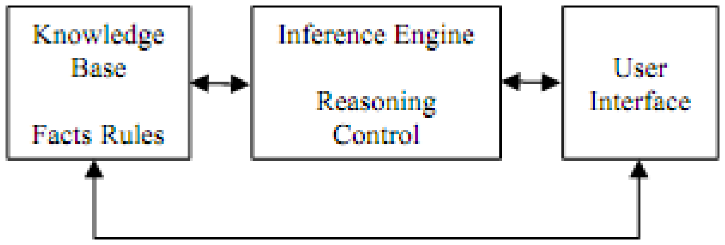


Figure. 1 Expert System Components

2. Knowledge Representation

Knowledge Base (KB): The system uses a Hybrid Knowledge base, the main components of which are Meta Rules, Rules & Frames. The Knowledge Base can be built with Domain specific information using the web based, user friendly Knowledge Base Editor GUI. A web based Knowledge Base Editor (KBE) provides facility to input different rules of an application by the domain expert using user-friendly GUI. Rules and other application specific attributes will be represented using Frames and Production rules.

Knowledge base consists of domain specific information that is fed into the system through GUI. This information is used to interpret and diagnose real-time data received from controllers and sensors. The structures used to store knowledge consist of:

Meta-rule: Structure - If Complaint Then Hypothesis

Meta-Rules are the supervisory rules to categorize Sub-rules in broad areas or nodes. Meta rules decide what domain rules should be applied next or in what order they should be applied. Rules: Structure - If Condition Then Action

Rules are basic components of the ‘Knowledge base’. The condition and action parts of the rules are defined as parameters. The condition and action parts can have more than one parameter separated by logical operators. The parameters are specific data variables, which are configurable by the user.

Frames: Frames are general data structure holding the static information about any equipment or unit.

3. Inference Engine

The Inference Engine is the heart of the expert system. Since the expert system is intended for diagnosis of faults, a Backward Chaining inference strategy is preferred. This inference engine will use knowledge stored in Production Rules and Frames for arriving at a conclusion or a goal while performing a fault diagnosis using backward reasoning method

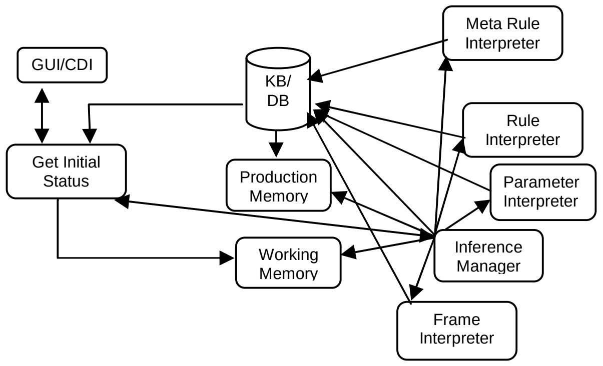


Figure. 2 Internal Block Diagram of Inference Engine

Working Principle of Inference Engine: The critical alarms are made as Meta-rules and sub-causes are depicted as rules in the knowledgebase. When the critical alarm occurs in the system, the relevant meta-rule will be fired; consequently the rules having the condition part true will get fired in sequence till the final root cause of the failure is reached. The inference engine algorithms is as follows:

Backward reasoning: Backward chaining involves, working back from possible conclusions of the system to the evidence, using the rules and frames. Thus backward chaining behaves in a goal-driven manner. One needs to know which possible conclusions of the system one wishes to test for. In frame based systems rules play an auxiliary role. Frames represent here a major source of knowledge both ‘methods’ and ‘demons’ are used to add actions to the frames. Goal can be established either in a method or in a demon.

Inference Manager is used for processing of all rules based on the user input or SCADA input. Meta Rule Interpreter search through the Meta Rule set for finding out the complaints part (antecedent) which matches the user defined complaint/Alarm. Inference Manager gets the hypothesis list (in the consequent part of Meta Rule) and Load it into the Working memory. Then Inference Manager will process hypothesis based on the certainty factor one by one (Hypothesis with the highest CF processed first).Then Inference Manager searches through the Rule set for finding out the action part which matches the hypothesis (THEN) part of the Meta rule. Rule Interpreter gets the condition part of Rule and evaluate each parameter in it. For this Parameter Interpreter checks the parameter type. Parameter can be database type or (within the working memory) or frame type or GUI or Computed. Evaluated parameters are added to working memory. This process is repeated till all rules are evaluated. Fired rules and parameters evaluated are logged into the database for generating explanation by the Explanation module

4. Learning Engine Developement For Enhancing Expert System

The Learning Engine is an extension of the Expert systems where new rules have to be learned by the shell for any domain. For this we need to do data preprocessing and data mining of the historic Data. The new rules generated have to be validated by the expert and then can be updated to the knowledge base. In our work we used Association based Learning methods for generating new rules from data[5].

To enhance the inferencing capability of Real time Expert System Shell new rules have to be found from the historic data. The parameter name and values from the historic data are used as input for Learning Engine. For learning, the association between parameters at different time has to be found. The major Machine Learning Techniques are as below:

Classification – Supervised data mining. Forms unseen classification.

Numerical prediction – Subgroup of Classifier. Formed outcome is a numeric quantity and not a discrete class or classifier.

Clustering – Unsupervised learning.

Association – Any association of one or more than one attribute at a time.

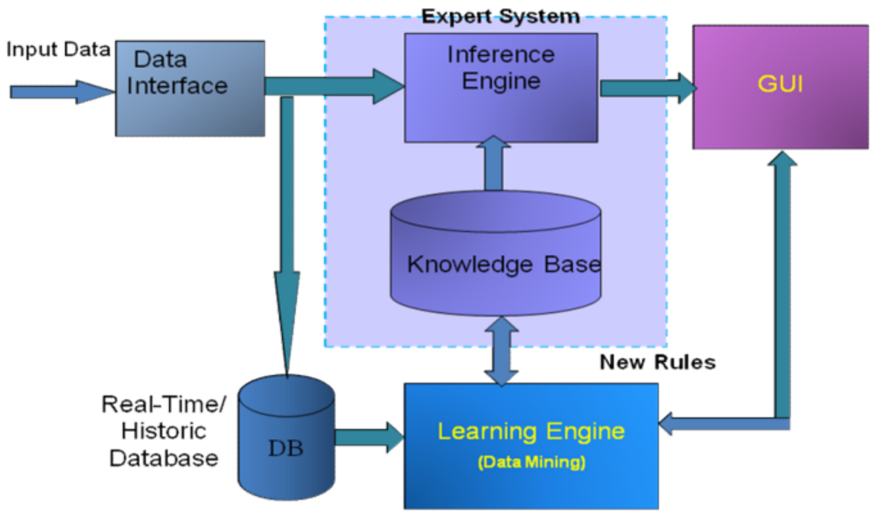


Figure 3: Architecture Diagram of Expert System with Learning Engine

The output of Machine Learning algorithms can be of the form of Decision tables, Decision trees, Association rules, Clustering Diagrams etc. Association rule learning (Dependency modeling) - Searches for relationships between variables. Thus we have studied and compared the Association algorithms.

4.1 Association rule learning

Association Mining is finding frequent patterns, associations, correlations or causal structures among sets of items or objects in transaction databases and other information repositories. When given a set of transactions, this will find rules that will predict the occurrence of an item based on the occurrences of other items in the transaction.

Association rules are usually required to satisfy a user-specified minimum support and a user-specified minimum confidence at the same time. Association rule generation is usually split up into two separate steps:

1. First, minimum support is applied to find all frequent item sets in a database.

2. These frequent item sets and the minimum confidence constraint are used to form rules.

In Association rule learning following algorithms were compared & results summarized:

1. Apriori
2. Predictive Apriori
3. FPGrowth
4. Tertius

4.1.1 Apriori

In computer science and data mining, Apriori is a classic algorithm for learning association rules. Apriori is designed to operate on databases containing transactions (for example, collections of items bought by customers, or details of a website frequentation).

Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as candidate generation), and groups of candidates are tested against the data. The algorithm Apriori, suffers from a number of inefficiencies or trade-offs, which have spawned other algorithms. Candidate generation generates large numbers of subsets (the algorithm attempts to load up the candidate set with as many as possible before each scan). Bottom-up subset exploration (essentially a breadth-first traversal of the subset lattice) finds any maximal subset S only after all 2 | S | − 1 of its proper subsets.

4.1.2 Predictive Apriori

It searches with an increasing support threshold for the best 'n' rules concerning a support-based corrected confidence value. When evaluating association rules, rules that differ in both support and confidence have to be compared; a larger support has to be traded against a higher confidence.

The constraint of Predictive Apriori is that it also creates transaction database before rule making and the time required for rule making was much high compared to other algorithms.

4.1.3 FP Growth

FP Growth allows frequent itemset discovery without candidate itemset generation. It has two step approach:

Step 1: Build a compact data structure called the FP-tree

Built using 2 passes over the data-set.

Step 2: Extracts frequent itemsets directly from the FP-tree

Traversal through FP-Tree

FP-Tree is an efficient algorithm for finding frequent patterns in transaction databases

- A compact tree structure is used
- Mining based on the tree structure is significantly more efficient than Apriori transaction databases

The constraint of FP Tree is that it supports only binary attributes and also creates transaction database before rule making.

4.1.4 Tertius

Tertius uses a top-down search algorithm when creating the association rules. The search first starts with an empty rule. Next, Tertius iteratively refines the rule by adding new-attribute values. Tertius continues to refine the rule as long as such refinements increase the confidence value. Finally, Tertius adds the rule and restarts the search to create new rules. Tertius ends when no additional rules can be created with sufficient confidence values. Afterwards, it returns the set of rules along with their confidence values.

Tertius does not create itemset and transaction databases. It starts directly from creating rules thus the algorithm implementation of Tertius requires less memory compared to the other association algorithms.

4.2 The Learning Module Implementation: Working Principle

Knowledge discovery in databases is the process of identifying valid, novel, potentially useful, and ultimately understandable patterns or models in data. Data mining is a step in the knowledge discovery process consisting of particular data mining algorithms that, under some acceptable computational efficiency limitations, find patterns or models in data. This Learning Engine was developed as an enhancement module to the Real Time Expert System for Root Cause Analysis of alarms in Plants. The development is done using Java and Flex languages and uses MySql database. In power plants application we have to take into consideration the historic data, online data for updating the knowledge base with the new knowledge acquired. Figure 4 shows the general architecture of the Learning Engine Module.

The Learning Engine Module mainly consists of the following sub modules:

User Interface
Preprocessor
Algorithm implementer
Post processor
Rule Validator

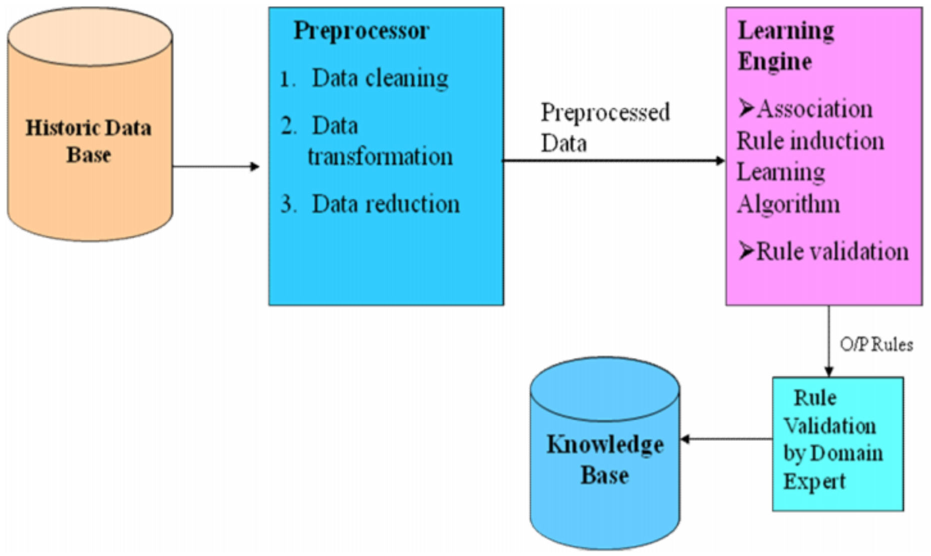


Figure 4: Block Architecture Diagram of Learning Engine

The administrator/user can initiate the Learning process from the GUI. The data has to be selected from the history by configuring the start time/date to end time/date for Learning Engine. Figure 5 shows the start/end date and parameter selection configuaration.

4.2.1 Preprocessor

This data in the historian data base has to be preprocessed and the suitable output structure has to be created before feeding to the Learning algoritm model to get the output.

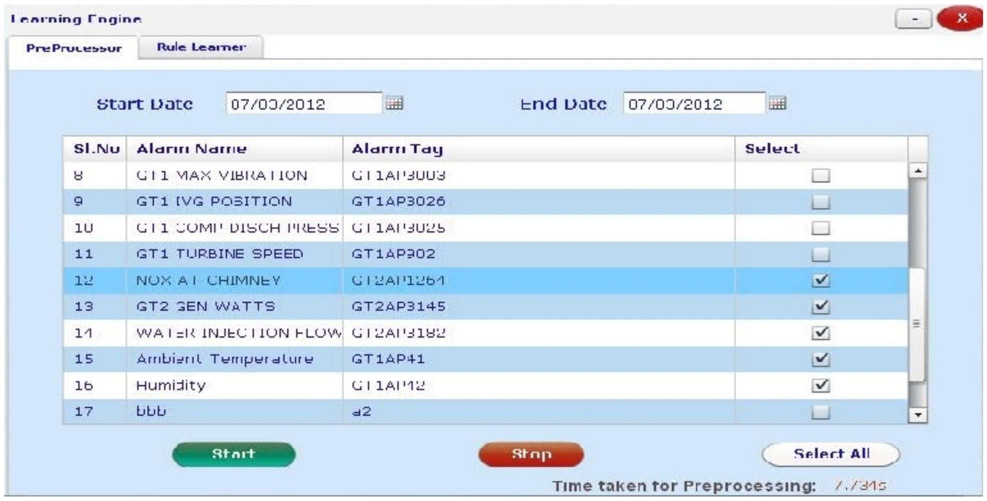
The history data preprocessing involves the following methods.

1. Data cleaning - The historic data contains the field ‘Quality’ with data 1(good) and 0 (bad). Filter out or discard instances with quality attribute value 0.

2. Data transformation- Data Transformation involves conversion of the data value (Parameter Value) using alarm code mapping to corresponding nominal values

3. Data reduction –The unwanted columns in the historic data are to be removed.

The historic data after preprocessing has to be fed to the Learning algorithms.



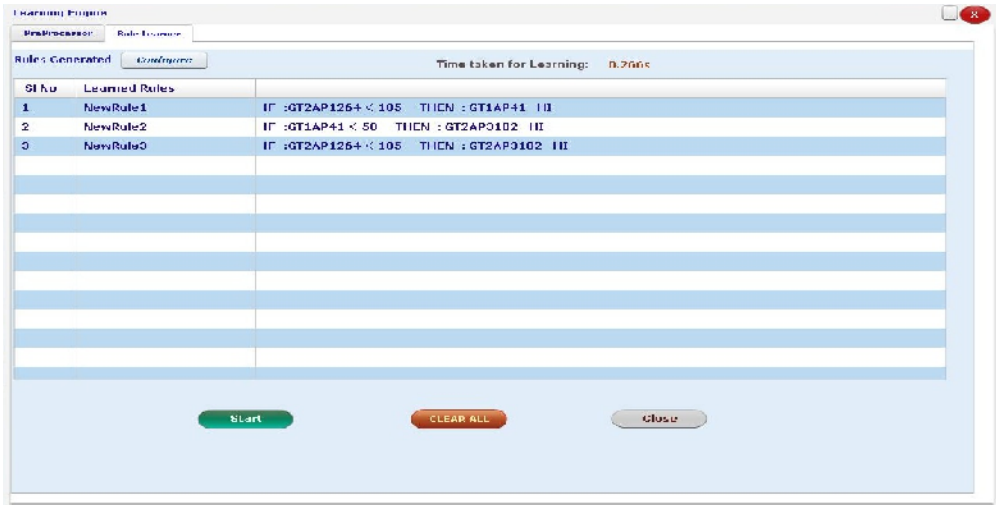
Start Date 07/03/2012 End Date 07/03/2012

Sl.No	Alarm Name	Alarm Tag	Select
8	GT1 MAX VIBRATION	GT1AP3003	☐
9	GT1 IVG POSITION	GT1AP3026	☐
10	GT1 COMP DISCH PRESS	GT1AP3025	☐
11	GT1 TURBINE SPEED	GT1AP302	☐
12	NOX AT CHIMNEY	GT2AP1264	✓
13	GT2 GEN WATTS	GT2AP3145	✓
14	WATER INJECTION FLOW	GT2AP3182	✓
15	Ambient Temperature	GT1AP41	✓
16	Humidity	GT1AP42	✓
17	bbb	a2	☐

Time taken for Preprocessing: 7.7345

4.2.2 Algorithm Implementer

The pre-processed learning data table in database is the input to Algorithm Implementer module. This module is using Tertius algorithm as found most suitable among the association algorithms for this domain of application. Tertius does a complete top-down A* search over the space of possible rules. If you have ‘A’ attributes with on the average ‘V’ values and search for rules with up to ‘n’ literals, the number of possible rules is of the order (AV)n. The output is the set of rules from the Rule generator block. The parameters which are configurable for Tertius algorithm are Confirmation Threshold and Frequency Threshold whose default values are set as 0.3 and 0.1 respectively.



Rules Generated

Time taken for Learning: 0.266s

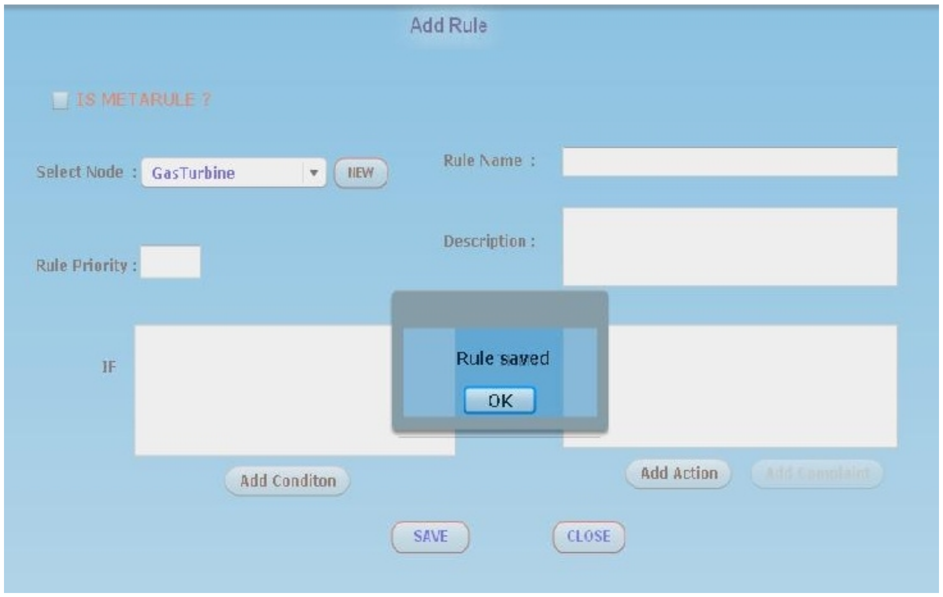
Sl No	Learned Rules	
1	NewRule1	IF :GT2AP1264 < 105 THEN : GT1AP41 HI
2	NewRule2	IF :GT1AP41 < 50 THEN : GT2AP3102 HI
3	NewRule3	IF :GT2AP1264 < 105 THEN : GT2AP3102 HI

4.2.3 Post processor Module

The input to this module is the set of rules generated from the rule generator(Algorithm Implementer) module. The rules generated have to be post processed so that it is in the same format as the Knowledge Base(KB) Rules. The rules with normal conditions (in both head and body) have to be discarded as only rules for alarm conditions are present in Knowledge Base. The duplicate rules have to be removed as part of post processing i.e with literals reverse in head and body. The rules shall be in the if-then format as in the KB. Post processor module converts the rules generated to if-then format.

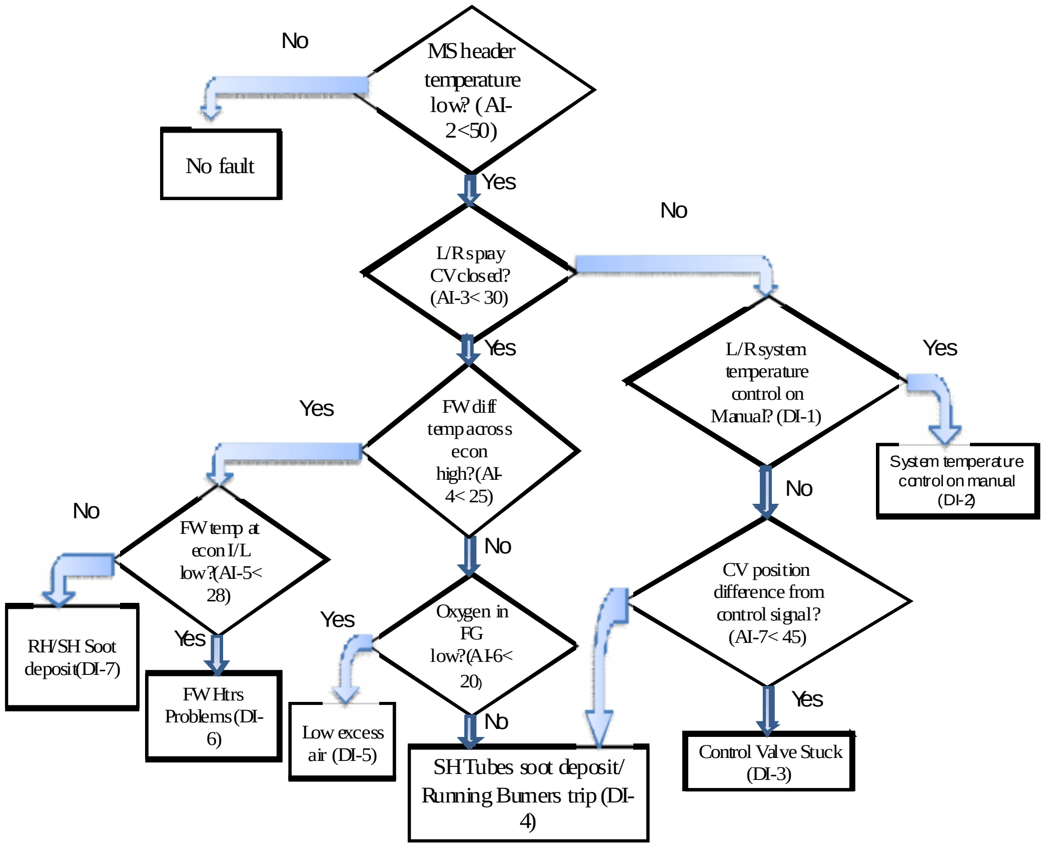
4.2.4 Rule Validator

The input to the Rule Validator module is the Postprocessor module output rules. The Rule Validator compares the input rules with the existing Knowledge Base Rules. If the rule already exists in Knowledge Base, it is discarded. The new rules are output to the GUI. After confirmation from domain expert, the Knowledge Base. is updated with the new rules.



5. APPLICATION

The data used for the testing is generated from a Plant Test Setup using Analog and Digital Input cards connected to suitable sources. The interface between Plant Distributed Control System (DCS) and Expert System is via OPC DA. The alarm “MS Header Temp LO” is considered for testing. This is an actual alarm studied from NTPC Plant.


MS header temperature low? (AI-2<50)
No
No fault
Yes
L/R spray CV closed? (AI-3< 30)
No
Yes
L/R system temperature control on Manual? (DI-1)
Yes
System temperature control on manual (DI-2)
No
FW diff temp across econ high?(AI-4< 25)
Yes
FW temp at econ I/L low?(AI-5< 28)
No
RH/SH Soot deposit(DI-7)
Yes
FW Htrs Problems (DI-6)
No
Oxygen in FG low?(AI-6< 20)
Yes
Low excess air (DI-5)
No
SH Tubes soot deposit/ Running Burners trip (DI-4)
CV position difference from control signal? (AI-7< 45)
Yes
Control Valve Stuck (DI-3)

Historic data is read from database and analog values are converted to nominal type for inputting to association algorithms. Here 13 attributes (6 analog and 7 digital) and 70 instances are used for testing. The test data is given as shown below.

attribute 1- AI-3 {LO,NORMAL} attribute 2- AI-2 {LO,NORMAL}
attribute 3- DI-7 {TRUE,FALSE}attribute 4- DI-6 {TRUE,FALSE}
attribute 5- DI-5 {TRUE,FALSE}attribute 6- DI-4 {TRUE,FALSE}
attribute 7- DI-3 {TRUE,FALSE}attribute 8- DI-2 {TRUE,FALSE}
attribute 9- DI-1 {TRUE,FALSE}attribute 10-AI-7 {LO,NORMAL}
attribute 11-AI-6 {LO,NORMAL}attribute 12-AI-5 {LO,NORMAL}
attribute 13-AI-4 {HI,NORMAL}

Existing Knowledge Base Rules

If AI-3<30(LO) OR DI-1=True OR AI-7<45(LO) OR DI-4=True Then AI-2<50(LO)
If AI-4>25(HI) OR AI-6<20(LO) OR DI-4=TRUE Then AI-3<30(LO)
If AI-5 < 28(LO) OR DI-7 = True Then AI-4>25(HI)
If DI-6 = True Then AI-5<28(LO)
If DI-5 = True Then AI-6<20(LO)
If DI-2 = True Then DI-1=True
If DI-3 = True Then AI-7<45(LO)

Algorithm Run Outputs

1. Apriori

Configuration
No of rules=100000 , Minimum support: 0.25 (17 instances)
confidence: 0.1, Instances: 70, Attributes: 13
AI-3, AI-2, DI-7, DI-6, DI-5, DI-4, DI-3, DI-2, DI-1, AI-7, AI-6, AI-5, AI-4
Best rules found:
Seven best rules are found using Apriori algorithm. One sample rule is shown below
AI-3=LO 40 ==> AI-2=LO 40   conf:(1)

2. Predictive Apriori

Configuration
No of rules=100000, Instances: 70, Attributes: 13
AI-3, AI-2, DI-7, DI-6, DI-5, DI-4, DI-3, DI-2, DI-1, AI-7, AI-6, AI-5, AI-4
Best rules found:
Twenty three best rules are found using Predictive Apriori algorithm. One sample rule is shown below
AI-4=HI 20 ==> AI-3=LO AI-2=LO 20   acc:(0.99454)

3. FPGrowth

Configuration, No of rules=100000, Minimum support: 0.25 (17 instances)
Minimum confidence: 0.1, Instances: 70, Attributes: 13
AI-3, AI-2, DI-7, DI-6, DI-5, DI-4, DI-3, DI-2, DI-1, AI-7, AI-6, AI-5, AI-4
FPGrowth found 47458 rules
Rules without NORMAL and FALSE attribute values are only displayed and no such rules are found
FPGrowth supports only binary attribute values.

4. Tertius

Instances: 70, Attributes: 13
AI-3, AI-2, DI-7, DI-6, DI-5, DI-4, DI-3, DI-2, DI-1, AI-7, AI-6, AI-5, AI-4
Thirty two best rules are found using Tertius algorithm. Some sample rules is shown below.
AI-4 = HI ==> DI-7 = TRUE or DI-6 = TRUE
AI-5 = LO ==> AI-4 = HI
AI-4 = HI ==> DI-7 = TRUE
AI-4 = HI ==> DI-6 = TRUE
AI-4 = HI ==> AI-5 = LO
Highlighted rules are existing KB rules.
Number of hypotheses considered: 73507
Number of hypotheses explored: 38490

6. Comparison Of Association Algorithms

Comparison of Machine learning Association algorithms was required to find the pros and cons of each algorithm. The Table 1 shows that out of these four algorithms Apriori and Tertius give satisfactory results with respect to this application. The Apriori Association Algorithm requires more time and memory if the number of attributes are high (more than 20 attributes). The FPGrowth Association Algorithm supports only binary attribute values. The Predictive

Association Algorithm takes more time and will not work with more attributes. Tertius Association Algorithm has been tested up to 250 attributes and got acceptable results. So it is concluded that Tertius Association Algorithm is more suitable for our application as it can support quite high number of attributes with highest number of valid results.

Table 1: Association Algorithms output comparison

	Apriori	FPGrowth	Predictive Apriori	Tertius
No of attributes	13	13	13	13
No of instances	70	70	70	70
Time	Less than 1 min	Less than 1 min	More than 5 hrs	Less than 1 min
Properties	Min CF =0.1	Min CF =0.1		Min CF =0.1
	Min support=0.25	Min support=0.25		Min support=0.25
	Max support=1	Max support=1		Max support=1
	Num rules=10^5	Num rules=10^5	Num rules=10^5	
No of rules generated	7	0	23	32
Existing KB rules	4	0	0	9

7. Conclusions

The Learning Engine enhances the Expert System by forming new rules dynamically. The new rules formed are validated by the domain expert and then the rules are updated to the knowledgebase. The association based learning algorithms are compared and it was found that the Tertius Algorithm is most suitable for this process application as it generates maximum number of valid rules in shortest possible time. Hence this algorithm has been used for implementing the Learning Engine.

Acknowledgements

Our special thanks to NTPC’s Rajiv Gandhi Combined Cycle Power Project (RGCCPP) at Kayamkulam, Kerala for offering their domain expertise in developing the Expert System and the Learning Engine for Operator Guidance and for allowing us to implement this system in the plant.

References

[1] Peter Flach, Valentina Maraldi, Fabrizio Riguzzi. Algorithms for Efficiently and Effectively Using Background Knowledge in Tertius.
[2] Rakesh Agrawal, Tomasz Imielinski, Arun Swami. 1993 Mining Association Rules between Sets of Items in Large Databases. ACM
[3] Tom M. Mitchell. July 2006 The Discipline of Machine Learning
[4] Peter a. Flach, Nicolas Lachiche. Kluwer Academic Publishers, Boston. Manufactured in The Netherlands.
[5] Confirmation-guided discovery of first-order rules with Tertius.

[6] Randall W. Pack, Paul M. Lazar, Renata V. Schmidt, Catherine D. Gaddy. Expert systems for plant operations training and assistance.

[7] T. S. Dahl. March 1999. Background Knowledge in the Tertius First Order Knowledge Discovery Tool. University of Bristol.

Authors

Abhir Raj Metkar obtained his bachelor's degree in Instrumentation Engineering in 1999 and Master's degree in Process Control Engineering in 2001. Currently he is working as a Senior Engineer in Control and Instrumentation Group of CDAC Trivandrum

Latha B Kaimal Graduated in Electronics and Communication Engineering from the College of Engineering, Trivandrum in 1977. Joined the Digital Electronics R&D Group of the State owned KELTRON group of companies in 1978 as Development Engineer. Was a key member of the Computer and Communication Group in Electronics Research and Development Centre of India (ER&DCI). Joined the Control & Instrumentation Group(CIG) in 1994 and involved in the implementation of Distribution Automation Project for Thiruvananthapuram city.Presently Associate Director in CIG in CDAC(T) and working as the Project leader of “Development of Real Time Expert System Shell” project

Rakesh G obtained his bachelor's degree in Computer Engineering in 2005.Currently he is working as a Senior Engineer in control and instrumentation group of CDAC Trivandrum Involved in the design and development of "Real Time Expert System Shell” project under the Automation Systems Technologies Programme (ASTeC - “Learning” domain).