A Study on Data Filtering Techniques for Event-Driven Failure Analysis

Yang-Ji Lee

Engineering and Systems Design Program

Graduate School of UNIST

A Study on Data Filtering Techniques for Event-Driven Failure Analysis

A thesis
submitted to the Graduate School of UNIST
in partial fulfillment of the
requirements for the degree of
Master of Science

Yang-Ji Lee

02. 20. 2013

Approved by

[illegible]

Major Advisor

Duck-Young Kim

A Study on Data Filtering Techniques for Event-Driven Failure Analysis

Yang-Ji Lee

This certifies that the thesis of Yang-Ji Lee is approved.

02. 20. 2013

signature
[illegible]

Thesis supervisor: Duck-Young Kim

signature
[illegible]

Gwan-Seob Shin

signature
[illegible]

Min-Seok Song

Abstract

High performance sensors and modern data logging technology with real-time telemetry facilitate system failure analysis in a very precise manner. Fault detection, isolation and identification in failure analysis are typical steps to analyze the root causes of failures. This systematic failure analysis provides not only useful clues to rectify the abnormal behaviors of a system, but also key information to redesign the current system for retrofit. The main barriers to effective failure analysis are: (i) the gathered sensor data logs, usually in the form of event logs containing massive datasets, are too large, and further (ii) noise and redundant information in the gathered sensor data that make precise analysis difficult. Therefore, the objective of this thesis is to develop an event-driven failure analysis method in order to take into account both functional interactions between subsystems and diverse user's behaviors. To do this, we first apply various data filtering techniques to data cleaning and reduction, and then convert the filtered data into a new format of event sequence information (called 'eventization'). Four eventization strategies: equal-width binning, entropy, domain knowledge expert, and probability distribution estimation, are examined for data filtering, in order to extract only important information from the raw sensor data while minimizing information loss. By numerical simulation, we identify the optimal values of eventization parameters. Finally, the event sequence information containing the time gap between event occurrences is decoded to investigate the correlation between specific event sequence patterns and various system failures. These extracted patterns are stored in a failure pattern library, and then this pattern library is used as the main reference source to predict failures in real-time during the failure prognosis phase. The efficiency of the developed procedure is examined with a terminal box data log of marine diesel engines.

Contents

List of Figures

List of Tables

I. Introduction

1.1 Background

As system complexity is increased, product quality is more emphasized in reliable operation of systems. Once system failures have occurred, they can cause unrecoverable losses, and the occurrence of a minor failure has resulted in disastrous effects. For example, it has been observed that failures have caused a 3~8% reduction of the oil production in the United States, leading to $20 billion losses in the country's economy each year (Wang et al. 2009). Recently, massive numbers of Toyota vehicles were recalled due to an accelerator pedal error, and this recall problem caused huge losses for the corporation. Product quality, economic operation and overall safety are essential factors in process design (Naik and Amol.S. 2010). Thus, a real-time monitoring system to observe abnormal behaviors is required, and systems should effectively react to unforeseen system failure.

A failure is an unpermitted deviation of at least one characteristic property of the system from the acceptable, usual, standard condition, and there exist many different types of failures, e.g. design, manufacturing, assembling, normal operation, wrong operation, maintenance, hardware, software, and operator's error (Isermann 2006). Failure detection, isolation and identification are typical steps to analyze the root cause of failures in the failure analysis as displayed in Figure 1.1. In the detection phase, the system detects the undesired symptoms of one of the system's functional components, and then localizes the different types of failures in the isolation phase. After the isolation phase, the system estimates the size, type and time in the identification phase (Naik and Amol.S. 2010). This systematic failure analysis provides not only useful clues to rectify the abnormal behavior of a system, but also key information to redesign the current system for retrofit.

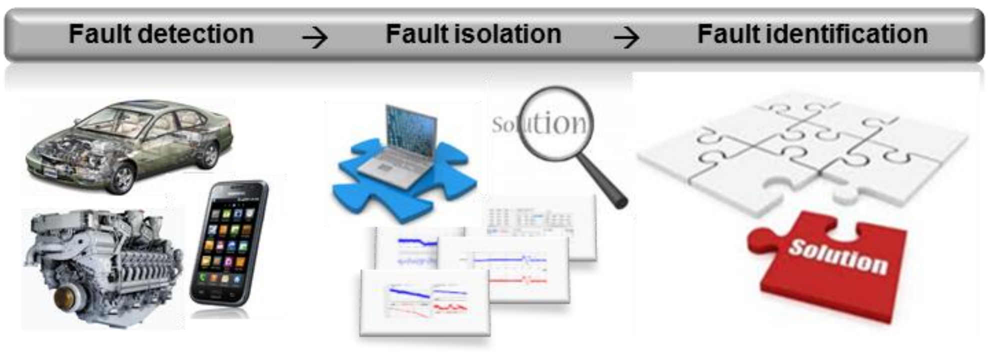


Figure 1.1: Procedure of failure analysis

Depending on whether the system model can be represented or not, analysis techniques can be grouped into MBFA (Model-Based Failure Analysis) and DDFA (Data-Driven Failure Analysis). MBFA can be used when a mathematical model is created by reading the input and output of the system. However, DDFA only uses data collected from systems. In addition, analysis techniques can be grouped into phases (detection, isolation, and identification), which have already been described. In this paper, analysis techniques are grouped in terms of each phase. Much research has been devoted to each phase, and detailed research is described in Chapter 2.

1.2 Motivation

MBFA can be used on-board real-time applications, and this analysis is widely adapted in the automotive industry. However, MBFA has some disadvantages in that this analysis needs extensive expertise and knowledge to derive dynamic relations of the system under nominal and failure conditions (Namburu et al. 2007). In vehicle or marine operating systems, it is very difficult to get an accurate model because the operating conditions often change depending on the driving conditions. In addition, the accurate input and output of the systems are essential factors (Luo 2006) to get system model. It is hard to get these accurate input and output factors, and then this analysis cannot consider the interaction of parameters. For these reasons, MBFA is sometimes difficult to adapt though it is accurate. Due to the increasing complexity of the mathematical model, DDFA is more flexible because it does not need to construct the detailed mathematical model. This DDFA can be used to detect failures that could not be detected by using MBFA.

DDFA uses the collected data to find causes of failures. These data are collected from sensors that are attached in the systems, and these data contain the significant information for failure analysis. However, these sensor data present obstacles to successful analysis because the data are too large in general, and furthermore, they usually contain noise and redundant data. Problems of gathering data can be divided into two factors: sensor and storing (Rabatel et al. 2009). In case of sensors, types, units and their normal state values are all different and data can be affected by environmental factors. The other problem occurs when data are transferred from sensors to the data acquisition system. At that time, some errors may occur and then readings can become incomplete or even missing (Rabatel et al. 2009). In addition, these data are difficult to analyze because they were collected over a long time.

For these reasons, data must be put into a comprehensive form and interpreted accurately to analyze the root causes of failure. High volumes of data must be changed into low-dimension space while containing the significant information. If the data size is reduced, it is more efficient to analyze the failure. For this reduction phase, some data reduction techniques should be applied, and then these reduced data have to be used for the failure analysis phase. When these filtered data are used for analysis, the interaction of parameters should also be considered, and the information about the causes can be provided more easily.

Once data are filtered, the accurate causes of failure should be provided. In general, failure analysis methods that are already developed only offer the occurrences of failures and it takes much time to construct a failure analysis model. The most important factor of failure analysis is to find the causes of failure, and then users have to provide a proper reaction to resolve the failures. At the result, the failure analysis should have to find the accurate causes of failure, and then a suitable solution be applied. Finally, the failures should be predicted in real-time.

1.3 Objective

The objective of this paper is to construct an even-driven failure analysis. Our event-driven failure analysis has five phases, which are data cleaning, reduction, failure pattern analysis, parameter interaction, and failure prognosis phase. The entire procedures of failure analysis are described, and the focus of this paper is the data reduction phase by applying different discretization techniques. The detailed objective of this paper is described as follows.

At first, the Kalman filter and statistical analyses are applied to raw data. Data can be converted to the transformed data, and the significant features can be extracted from raw data. After these analyses, eventization, which is in the focus of this thesis, is described. Event means the significant data extracted from the raw data. As described, DDFA deals with high dimensional data, which requires data reduction techniques. For this reduction phase, different discretization techniques, which are Equal-width binning, Entropy, Knowledge expert-based discretization, and Estimation of probability distribution-based discretization, are applied and compared to the performance. Through this eventization phase, data can be converted to event logs that are indicating the format of reduced data.

After extracting the event logs, FP-Tree algorithm is applied to find the causes of failure. Certain patterns can be extracted, and this information contains the interaction of sensor information. Furthermore, a temporal reasoning technique is applied to provide the time interval information. A suitable parameter for analyzing the failure is also estimated. From these patterns, the causes of failure can be known, and these results are stored. These stored results are used for the failure prognosis phase. From all of these failure analysis phases, failure can be predicted in real-time.

1.4 Outline of the thesis

This paper consists of six chapters. First Chapter introduces the general failure analysis and some problems of the failure analysis. In Chapter 2, literature surveys which are contained each phase for the failure analysis are described. The problems and procedures for DDFA which is suggested in this paper are defined in Chapter 3. Detailed procedures and methods are described in Chapter 4, and based on these procedures case study is applied in Chapter 5. At the result, the conclusion and future research is described in Chapter 6.

II. Literature Survey

Over the past years, many methods have been developed in failure analysis area. Methods for failure analysis can be broadly classified model-based approaches and data-driven approaches (Luo 2006). As mentioned before, model based approaches have some problems. DDFA (Data-Driven Failure Analysis) is efficient because they do not need to use accurate model and it can use a priori knowledge of system. Therefore, when system is complicated, DDFA is better than model based approaches because only the huge amount of historical data is used without accurate mathematical model. Failure analysis consists of three procedures as failure detection, failure isolation, and failure identification. Each phase, different methods are considered, and prognosis phase is added. In this Chapter, developed researches are described along with three different phases.

2.1 Failure detection

Failure detection is indicating the processes to find the failure. If something is going wrong in the monitored system, certain warning is occurred in failure detection phase. The detection performance is determined from bellowed factors (Gertler 1998).

- Sensitivity: it is the ability to detect failures of reasonably small size.
- Reaction speed: it is the ability to detect failures with reasonably small delay after their arrival.
- Robustness: it is the ability to operate in the presence of noise, disturbances and modeling errors with few false alarms.

Failure detection is binary decision, and this phase only detects the probability of occurrence. Usually the failure detection phase is used simultaneously with the failure isolation (Samy and Gu 2011) because failure type or size can be determined after detecting the failure. Therefore, failure detection and identification (FDI) is usually researched. In this failure detection phase, Limit checking, Parameter estimation, and Expert system are described.

2.1.1 Limit checking

Limit checking is easy to implement, and the most frequently used method in failure detection methods. This method finds that certain value exceeds specified threshold while monitoring the data trend. If certain signal exceeds specified threshold, this state would be regarded as a failure states (Gertler 1998).

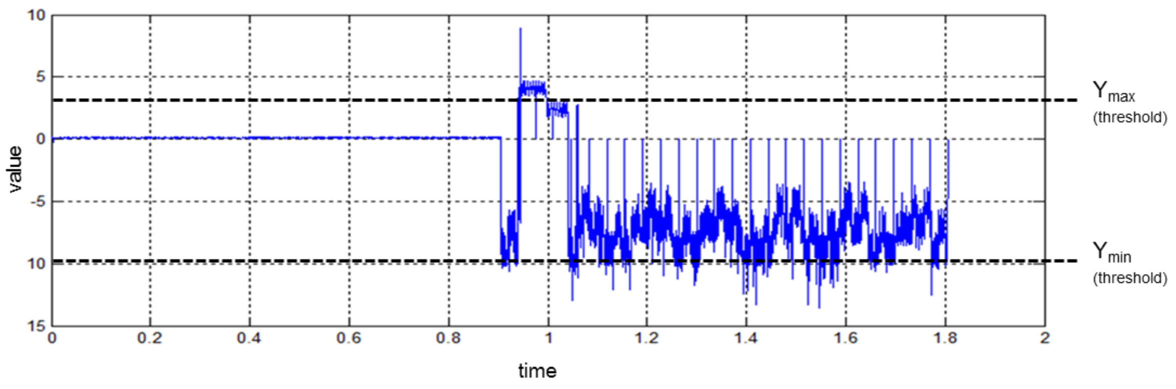


Figure 2.1: Example of limit checking method

In Figure 2.1, Y_{max} is a maximum value and Y_{min} is a minimum value of specified threshold. If certain values which are entering in real-time belong to specified range, these values are considered as normal state. Otherwise the value which is exceeded specified range, this signal is indicating the failures. Some researches applied the other values for limit checking. The standard deviation is set the limit and the maximum absolute value is used for limit checking (Fujimaki et al. 2005). Limit checking is a convenient algorithm. However, it is difficult to detect the failure when there is no big change. Failure analysis should detect these small change because small noisy or change can be also affected on the failure situation. Threshold also leads the problem because expert who has knowledge of system can only determine a suitable threshold for failure detection. Trend checking is the expended method of limit checking that uses differential of output and can monitor the trend of variables. The performance of trend checking method is better than limit checking method.

2.1.2 Residual generation

Residual is the difference between estimated values and measured value in real-time. The ideal difference is zero in stability systems however actual systems differ from the ideal one. This difference is occurred due to the failure, disturbance, noise and modeling error (Gertler 1991). Therefore, residual can be used to determine the failure state. Parity equations, Parameter estimation, and Expert systems are grouped as the failure detection methods using residual generation approach.

Parity equations

Parity equation generates the residuals by direct manipulation of the plant observables (Samy and Gu 2011). To get the residual, fixed model is needed which can be constructed by transfer function. Fixed mode can be created by using single input and output process. Performance will be increased if more measureable are available. Threshold is used for determine interval of residual. There is the existence of modeling errors which disturb outputs. For this reason, wider threshold should be used because this technique does not allow the small residual like noise (Isermann 2006). Failure detection and isolation (FDI) system used to generate input-output model using parity equation (Gertler and Singer 1990). Based on the constructed model, if measured residual is bigger than specified threshold, this state would be indicated the failure state. This technique is the simple method however it takes much time to construct the equation. Furthermore, it is difficult to detect the unfamiliar patterns which did not entered before.

Parameter estimation

Parameter estimation is usually used for failure detection and isolation of parametric failures. This technique assumed that the failure detection via parameter estimation relies in the principle that possible failures in the monitored system can be associated with specific parameters and states of the mathematical model of the system given in the form of an input and output relation (Escobet and Travé-Massuyès 2001). This model is displayed in the form or partial differential equations which generates process coefficients. It indicates a certain failure. The general procedure of parameter estimation is followed as below (Pouliezos et al. 1989).

- Step1: Establishment of the mathematical model of the system’s normal behavior.
- Step2: Determination of the relationship between the model parameters and the physical system parameters.
- Step3: Estimation of the model parameters from measurements of output and input by a suitable estimation procedure.
- Step4: Calculation of the physical system parameters, via the inverse relationship.
- Step5: Decision on whether a failure has occurred, based either on the changes of the non-measurable parameters or on the changes of the physical system parameters and tolerances limits. If the decision is made based on the non-measurable parameters the affected physical system parameters can be easily determined form step2. This may be achieved with the aid of a failure catalogue in which the relationship between process failures and changes in the coefficients of the change of the physical system parameter has been established. Decision can be made either by simply checking against the predetermined threshold levels, or by using more sophisticated methods from the field of statistical decision theory.

If one or several parameters are changed compared to normal parameters, system might be influenced by the failures. Parameter estimation is used not only failure detection but also failure isolation. At the result, failure types and size can be known using failure decision phase along with certain changes of parameters.

Expert systems

Expert system was first suggested by Hayes-Roth (Hayes-Roth et al. 1983) and then different ways about expert systems have been applied in a variety of domains. Expert system should be able to interpret the signals, and deliver the required control action. This techniques can be divided into rule-based expert system and model-based expert system (Angeli 2010).

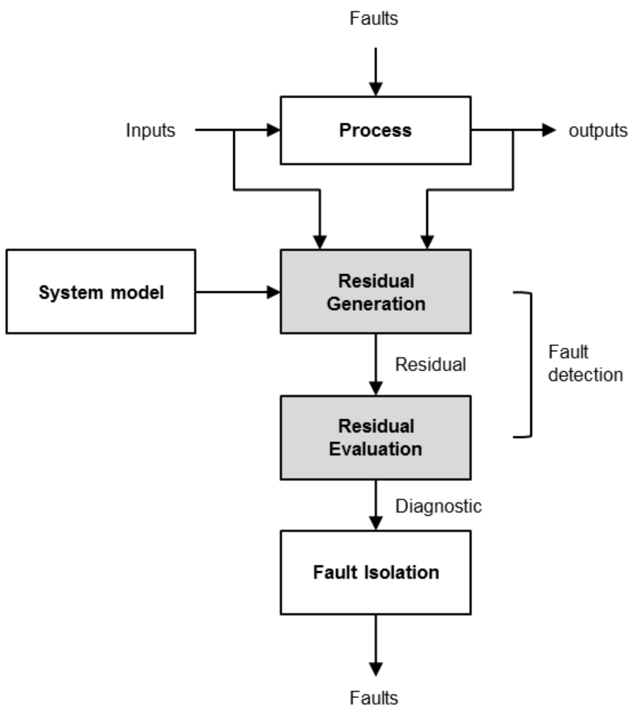


Figure 2.2: Structure of model-based expert system (Angeli 2010)

Rule-based expert systems need deep understanding of the physical system and should be constructed alternative solutions for certain symptoms prior to failure analysis procedures. Rules are formed “if (condition) then (action) and construction of alternative solutions can be possible through experiment repeatedly. Experts and knowledge engineers construct the inference engine using data, and then user can be used to analyze the failures. A decision tree is the main technique that defined the conclusions (specific failure type along with certain symptoms) along with logical paths. Rule-based expert systems do not need a mathematical model however they have to cover the all possible patterns. If unexpected failure is happened, there is no answer to solve the problem. For these reasons, this technique is very expensive and takes much time to build a system. In case of model-based approaches, these approaches should generate the residual signals. Model-based expert system covers disadvantages of rule-based expert system which cannot explain unexpected failure. Standard to compare the real-time input is the constructed mathematical mode. Therefore, if the difference between real-time data and mathematical models is occurred, it would be just regarded as the failure states. Figure 2.2 is represented procedures of model-based expert system. The disadvantage of this model-based expert system needs accurate input and output is needed to develop the mathematical model as mentioned before. There are many applications of expert systems for failure analysis. Integration of neural network and an expert system is proposed to diagnose the most commonly encountered failures (Becraft and Lee 1993). Wavelet sigmoid basis neural network and expert system based failure analysis have also be presented (Bingzhen Chen and Shen 1997).

2.1.3 Signal analysis

Failure detection using the signal analysis can be divided as stochastic signal analysis and trend analysis. Stochastic signal analysis uses the difference between the signal of normal state and the signal of abnormal state. It can also be considered as residual generation methods. The other method using the signal analysis is the trend analysis which uses the different types of signal. These two types of signal analysis methods are described.

Stochastic signals

Failure detection with signal models is usually used when there are some obvious changes in signal. It will calculate suitable features by assuming mathematical models for the measured signal (Miljkovic 2011). As shown in Figure 2.3, generated feature is compared to normal behavior and then it would be considered as analytical symptoms. Spectrum analysis is used for stochastic signal. Instead of original signal, convert the frequency domain using Fourier transform because massive data cannot extract failure relevant signal characteristic. Extracted failures relevant signal is restricted using band-pass filters (Stearns and Hush 1990). Certain values within interval using band-pass filter can be considered normal state range and the other values are considered as relevant signals. Bandwidth is usually determined by user.

Changes of vibration must be affected on the failures therefore vibration analysis is usually a well-established field (Wowk 1991). These researches assumed that certain vibration signal is associated with the certain components. If measured vibration signatures are as uncomplicated, detection would be easy matter. However, usually vibration signal contains high-level imbalance and misalignment components including random vibration associated with friction and other sources (McInerny and Dai 2003). For these reasons, vibration signal also should be converted frequency domain and extracted features. Using these features, these vibration signals can be compared with the normal state and the calculated the difference.

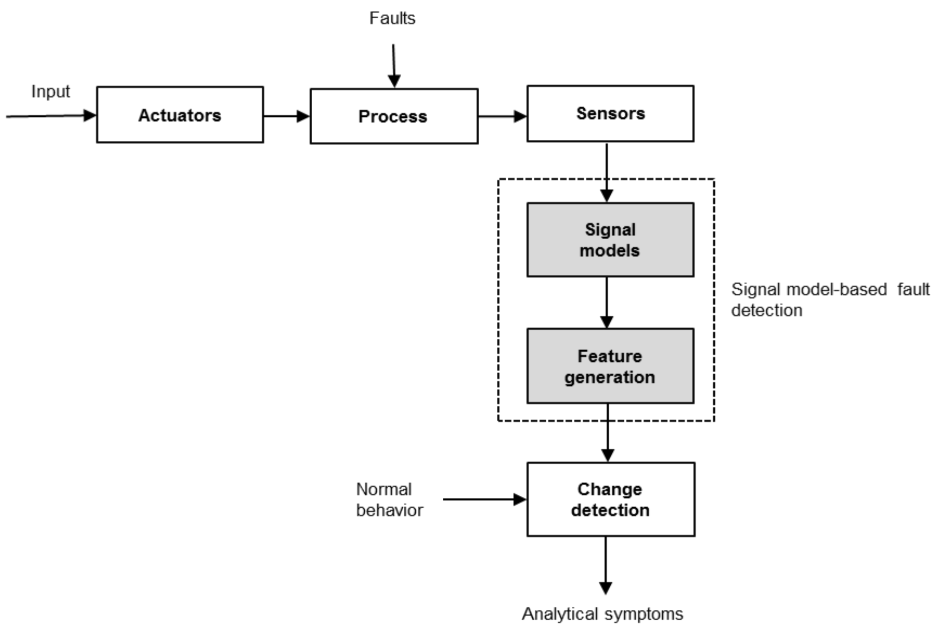


Figure 2.3: Process of failure detection with signal models (Miljkovic 2011)

Trend analysis

Trend analysis and prediction are important components of process monitoring and supervisory control (Venkatasubramanian et al. 2003). It needs some training data to store certain trend that correlated with failure state. It is possible to predict the failure state by mapping these certain trend and input data. Thus, if process trends could be detected early, the suitable solution along with failure trend would be processed as quickly.

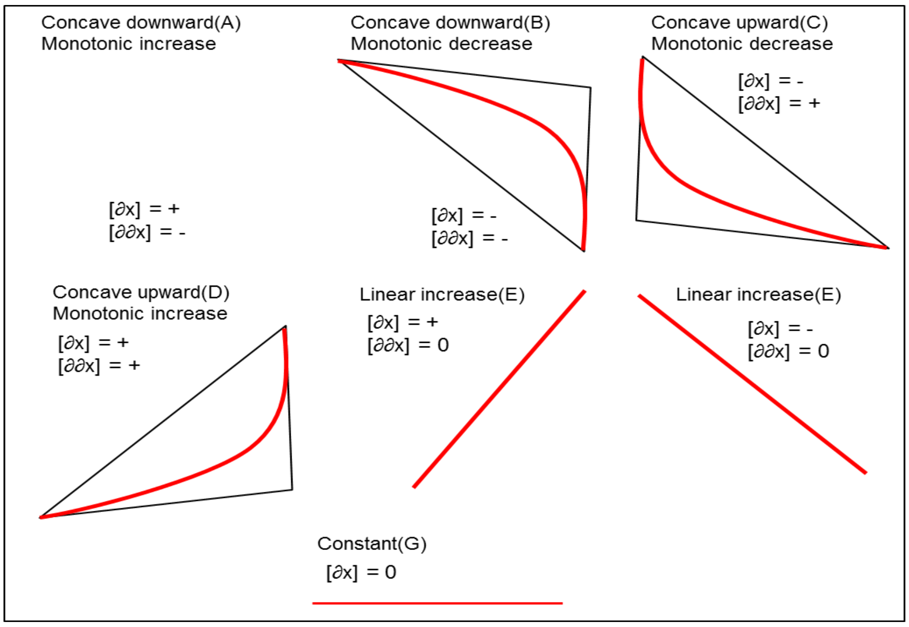


Figure 2.4: Various signal types in Triangular episodic presentation (Kivikunnas 1998)

One of the trend analysis methods is wavelet theory based trend adaptive trend analysis (Vedam and Venkatasubramanian 1997). Before the analysis, some signal samples are collected, and then the input data trend is compared with stored training data which are already classified along with failure classes. This technique is only possible to analyze accurately when training data contain all patterns of the certain failures cases. The other method triangular episodic presentation is developed based on temporal episodes and used to extracted trend (Cheung and Stephanopoulos 1990). Different types of episodes are defined as slope as Figure 2.4. Input data are classified based on the different shapes of episodes and compared with these types of slope. This triangular episodic method is similar to neural network that is used to identify the fundamental features of the trend observed. This method is helpful to construct an expandable shape library that stores shapes like decreasing concavely, decreasing convexly and so on (Konstantinov and Yoshida 1992). It means that input data is representative a symbolic form. Trend analysis has big problem that is to extract the trend from noise process data automatically.

2.2 Failure isolation

Failure detection is to determine whether failure is existed or not. After failure detection phase, failure isolation is applied to define the failure types and size based on symptoms. Failure isolation phase should determine the types, size, locations and time of detection correctly. These methods are classified with FMEA (Failure Mode and Effects Analysis), Failure trees, Classification and Data-driven pattern analysis. Data-driven pattern analysis is similar processes which are researched in this thesis. Therefore, phases of Data-driven pattern analysis are divided as the same phases which are researched in this research.

2.2.1 FMEA and Fault trees

FMEA and Fault trees are widely used in the failure analysis area. If certain failure is occurred, all possible ways would be searched using these two methods. Detailed methods are described in this Chapter.

FMEA (Fault Mode and Effects Analysis)

FMEA is a bottom-up approach and design analysis discipline that considers the effects of any failure in a design process. This technique also identifies the more serious problems as the certain areas where the design may need to be improved (Price and Taylor 2002). FMEA is mainly applied in industrial production, motor cars, mechanical and electronic components. The goal of FMEA is to predict the action of each cause and detect failure modes. Failure mode is an unstable state or the cause downstream operation to fail. Based on these failure modes, FMEA form can be constructed.

Initially, survey is needed about the functions of each component and effects to develop the FMEA form. The attributes of standardized form for FMEA analysis are described as below (de Queiroz Souza and Á lvares 2008).

- Function: action which the user desires that the item or system executes in a specified performance standard
- Component: identification of each component belonged to the system
- Component function: brief and accurate description of the task that the component must execute
- Functional failure: description of all the possible failures related to each component
- Failure mode: description of the form as the failure is observed by the operation team
- Failure cause: simple and concise description of the occurrence that they can origin to the considered type of failure
- Failure effect: consequence of the occurrence of the failure, perceived or not for the final user

Using these FMEA form, a product or process about the failures can be identified, and then estimated the risk. After that, the accurate actions can be applied to reduce the risk. FMEA have great importance for evaluation of potential failures in a system and it can possible classifying the failures for detecting certain reaction along with each failure mode.

Fault trees

Fault tree analysis (FTA) is a top-down approach for failure analysis, starting with a potential undesirable event (Rausand and Hoylanc 2004). It begins with undesired event at the top and determining all the specific events which are concerned with certain failures. Each analyzed event is connected to its causes by a gate (Ortmeier and Schellhorn 2007). Failure trees are a good graphical tool for displaying the binary relationships between failure and symptoms. By calculating the failure probabilities of each element, probability of failure can be predicted.

It is natural and intuitive to classify a pattern through a sequence of questions (Richard et al. 2001). As displayed in Figure 2.5, it is a flow chart like tree structure and internal node (circular) is a test on an attribute and branch represents an outcome of the test. Leaf (quadrangle,) is a class. Classification begins at the top node that indicates the root of all nodes. From the root node, outcome form the each node is evaluated by certain threshold (condition) of each branch. Based on the answer of certain condition node should follow the appropriate link repeatedly until the node is reached certain class. In failure analysis, each node indicates the symptoms along with the certain failures, and then the failure types can be determined along with symptoms. If some failure is occurred, failure type can be found along with symptoms condition, and the appropriate action will be applied. Failure tree is easy to understand and generate. However, this method is often complicated in case of large trees, and only expert who has a prior knowledge of system can construct the failure tree. Determining the suitable threshold which is to divide the state of symptoms is also difficult problem. This tree can be constructed by doing experiment repeatedly. For this reason, this method need huge amount of historical data. Example of FTA is displayed in Figure 2.6. Root of tree is displayed an undefined events of failure which are connected events. OR-gate indicates that the output event occurs if any of the input events occur and AND-gate indicates that the output event occurs only if all the input events occur at the same time. Following these gate conditions, the failure type can be defined. FTA main steps are as below (Rausand and Hoylanc 2004).

- Definition of the system, the top event and the boundary conditions
- Construction of the fault tree
- Identification of the minimal cut sets
- Qualitative analysis of the fault tree
- Quantitative analysis of the fault tree
- Reporting of results

Cut-set means that the condition to get the normal behaviors in the fault tree. There are many researches to find a suitable and minimal cut-set (Vatn 1992). FTA can easily interpret the relationship between failures and symptoms however only expert can only construct tree and there are other problems.

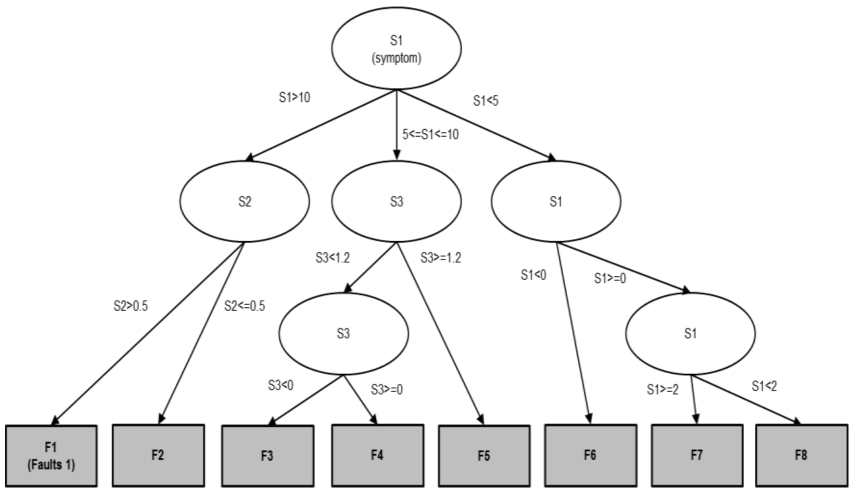


Figure 2.5: Structure of fault tree

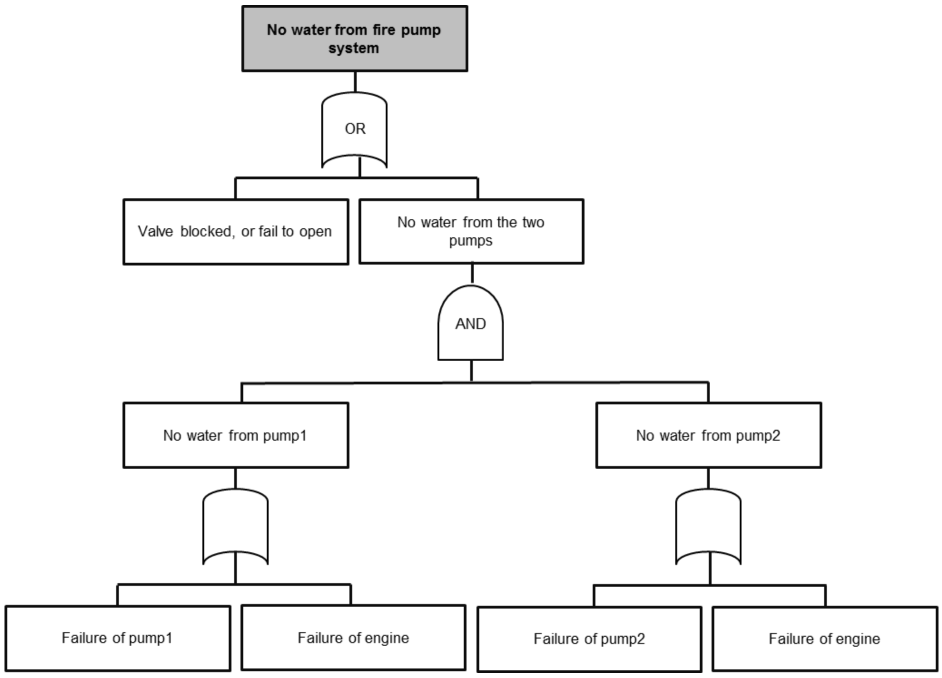


Figure 2.6: Example of fault tree (Rausand and Hoylanc 2004)

Classification

Generally, classification methods need two phases consisting of learning phase and classification phase (Han 2006). In the learning phase, classifier can be built to predict set of data classes using training data set, and then test data is classified along with predetermined classification rule in the classification phase. Symptoms are classified along with types of failure when classification is used for the failure analysis. Historical data are experimented repeatedly to determine the symptoms for the certain types of failure. After that the classification methods are used to represent the analysis functional mapping from the symptom space to the failure measure space. There are many methods in the classification methods. In this paper, Baysian, Geometric, and Neural network classification methods are considered.

The Baysian classifier is statistical approach which can predict the classes by calculating the probability. This classifier assumes that each instance must be contained one of certain classes. This method finds out the probability that unknown instance will belong to certain class, and then simply pick up the most probable class. This probability will be calculated as below (Leonhardt and Ayoubi 1997).

$$p(F_i|S) = \frac{p(S|F_i) \cdot p(F_i)}{p(S)}$$

Where:

- F: failures (F_0: no failure, F_1: failure)
- S: symptoms
- $p(F_i|S)$: probability of a certain failure once a specific symptom vector S has been observed
- $p(S|F_i)$: probability for the symptom vector S belongs to failure F_i
- $p(F_i)$: probability of the failure F_i to occur
- $p(S)$: probability for a specific symptom vector S to occur
- If given data sets have many attributes, probability is calculated as below when X is various attributes.

$$p(X|F_i) = \prod_{k=1}^{n} p(x_k|F_i) = p(x_1|F_i) \times p(x_2|F_i) \times \cdots \times p(x_n|F_i)$$

Accordance with these equations, the class whose probability is the maximum is selected. Bayes classification is fast because of the training data, and it is suitable for streaming data. However, this classification assumed that features are independence each other before the analysis. This assuming makes a problem when unfamiliar patterns are entering.

Geometric classifiers assumed that a certain symptoms vector should belong to the certain failure class. A failure class which has shortest distance is selected Figure 2.7. Distance between the symptoms and the failure types can be calculated by applying the nearest neighborhood distance. Geometric classifier is able to identify a failure type that has the minimum distance (Kim and Zuo 2007). There are many applications using geometric classifier (Liang et al. 2007). “Fatal (failure)” and “No fatal (no failure)” groups are determined along with certain symptoms and similarity is calculated. In summary, using training data, symptom patters are constructed along with certain failure, and then the similarity is calculated with training symptoms. This classification is simple to implement, and it can also handle the correlated features. However, it takes much time for large data for calculating and is very sensitive to irrelevant features. The commonly used method is neural network method in classification methods. The goal of neural network is to indicate a possibility of the failure occurrence. Given the appropriate network structure and weight vector, a neural network can isolate failures extremely quickly when such failures occur within the system for which it has been trained to detect (Naughton et al. 1996). Once prediction about the failure is applied, appropriate solution can be provided.

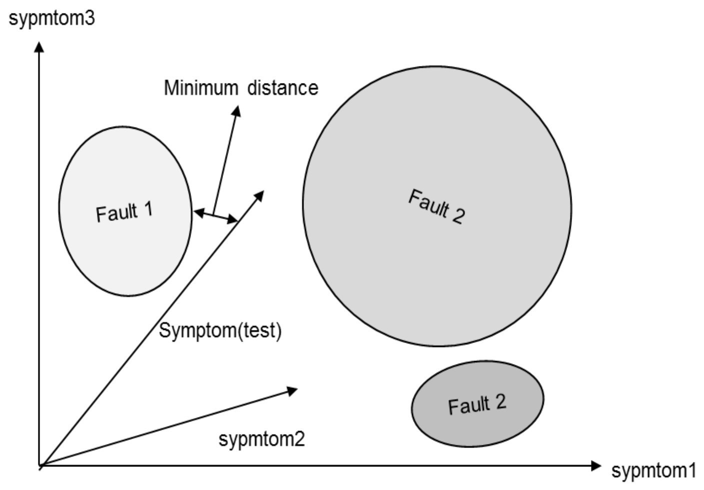


Figure 2.7: Geometric classification (Leonhardt and Ayoubi 1997)

Neural network system consists of three factors which are input layer, output layer and hidden layer. If input patterns are given, specific response is extracted through hidden layer. To extracted specific response along with input, the training phase is needed. During the training phase, general rules are created to connect the input and output by applying the weights. (Venkatasubramanian and Chan 1989). After training phase, network can generate responses for the input based on the training phase. Usually, the input is symptoms and the output is type of failure that system can predict the failure along with symptoms. The most popular method in neural network is back-propagation algorithm. Back-propagation algorithm calculates the difference between actual output and desired output, and then they change weight relative to error size. Through these phases, performance can be improved.

Neural network is mentioned before convention data-driven analysis. Multilayer perceptron (MPL) is the best known example (Rumelhart and McClelland 1986) that several layers are consisted. Training sets are repeatedly experimented to determine proper weights. After determine the proper weights, inputs (symptoms) are entered and they will connect with certain output (failures) from the calculated weights. In some applications, MLP can generate symptoms from the measurement signal directly (Bauer et al. 1996). A large problem of MLP is occurred when outside of the trained symptoms domain pattern is extracted. If the network is trained with many degrees of freedom to fit a complex decision boundary, these neurons will create completely unpredictable outputs in the extrapolation region, even close to the training area (Leonard and Kramer 1991). Besides of MLP methods, self-organizing networks and radius-basis function networks are widely used for failure analysis.

2.2.2 Data-driven Pattern Analysis

In this Chapter, historical data is used for failure analysis. Collected data need some techniques for accurate analysis, and then the causes of failures can be extracted. Therefore, data cleaning, reduction, and pattern extraction techniques are described in this Chapter.

Data cleaning

Applications of data-driven pattern analysis depend on the historical data gathered by physical sensor devices. One of the main problems in sensor data is unreliability caused by the sensors. The dirty or noisy data could be classified by two forms: unreliable readings and missed readings as below (Jeffery et al. 2006) .

- Missed readings: sensors often employ low cost, low power, hardware and wireless communication, which lead to frequently dropped messages.
- Unreliable readings: often, individual sensor readings are imprecise or unreliable.

How to remove these noises or minimize the effect from ‘dirty’ by noises is a key issue to detect events (Xue et al. 2006). One of the most popular methods to reduce noise and calculate the monitoring values is to use the moving average (Hellerstein et al. 2003) and the advanced moving average, which is a Weighted Moving Average-based Approach is also proposed (Zhuangt et al. 2007b). In addition, method which is to treat nasty sensors and outliers is explained (Subramaniam et al. 2006). Usually, signal processing is suggested as data cleaning method in terms of a model-driven (Jiang and Chen 2010).

As well as noise, one of the critical issues on data stream processing is how to treat the missing data. Processing methods for missing data in sensor data stream has been discussed in data processing community, and lots of papers have been published to deal this issue. Some methods delete all the missing data before analyzing them such as list-wise and pair-wise deletion (Wilkinson 1999). Otherwise, the missing value could be estimated by mean substitution, imputation by regression (Cool 2000), multiple imputation(Rubin 1987; Rubin 1996), hot-deck imputation (Little and Rubin 1987), cold deck imputation, maximum likelihood (Rubin 1987; Allison 2002), Bayesian analysis (Gelman et al. 2004). Recently, (Rodrigues and Gama 2006) describe a real-time system for online prediction in large sensor networks. Association rule mining is applied to estimate missing data in sensor networks (Rubin 1987). Kalman Filter with the dynamic linear model can also be used for estimating missing data in a sensor stream which have been used successfully for decade (Vijayakumar and Plale 2007a).

Another method for data cleaning is the data transformation. For safe operation of the machines, condition monitoring and failure analysis are critical part. In general, in order to monitor machine condition, signals are gathered by number of sensors in time series. The problem with handling time series data is its volume and the associated computational complexity. Therefore, the available information must be appropriately compressed via transformation of high-dimensional data into a low-dimensional feature space with minimal loss of class separately (Sarkar et al. 2011). To extract the available and important information among large sensor data, signal analysis have been developed which are simple and effective transform of original signal. For decade the fact that the description of a system in the frequency domain can reveal valuable insight into the system behavior and stability have proven (Wu and Liu 2009).

Data filtering

Gathered data from each sensor are huge and complex. In some cases, the information was containing errors or outliers (noisy) and they were not collected (missing data). Also, they were containing same value continually (redundant data) and containing discrepancies in codes or names (inconsistent data). These unrefined data make precise analysis difficult therefore some data-filtering methods are applied to improve the accuracy and efficiency. Data-filtering can be grouped into data reduction and discretization. Data reduction techniques reduce the dimensionality of a data set by creating new attributes that are a combination of the old attributes (Tan et al. 2006) and these techniques can be grouped dimensionality reduction and clustering techniques. Examples of dimensionality reduction are wavelet transform and attribute subset selection. Discretization is necessary to transform a continuous attribute into a categorical attributes. There are numerous discretization methods, these methods can be categorized in several dimensions (Liu et al. 2002). We will discuss the several of data-filtering methods.

Data reduction

Data reduction obtains a reduced representation of the data set that is much smaller in volume, yet produces the same analytical results (Han 2006). It may allow the data to be more easily visualized and the huge amount of time and computation memory is reduced because data reduction techniques can eliminate redundant features and reduce noise. These methods include Wavelet transformation and Attribute subset selection in dimensionality reduction and clustering.

Wavelet transformation: Signal analysis is one of the most important methods for failure analysis, whose aim is to transform the original signals simply (Peng and Chu 2004). Discrete Wave Transform (DWT) and Discrete Fourier Transform (DFT) are multi-resolution analysis for linear signal processing however DFT has loss compression. When applying wavelet transformation techniques to the original data, these data can be transformed to a numerically different vector of wavelet coefficients. These coefficients usually can be used as the failure features by using threshold function. Chen proposed dynamic transient signals by the discrete wavelet transform and selected the wavelet coefficients by the hard-thresholding method and discarding other smaller coefficients (Chen et al. 1999). Multi-scale wavelet analysis is used to represent the features characterizing the transients that can reduce the data and removes noise. Similarly, an improved soft-thresholding denoising method was used to extract the impulse component as failure features from vibration signals, and the time-frequency resolution can be adapted to different signals (Lin and Qu 2000). Lu and Hsu proposed that wavelet transform method used to detect the existence and location of structural damage (Lu and Hsu 2000). They defined that the maximum change of the wavelet coefficient was corresponding to the location of the damage because the changes in the wavelet coefficients of the vibration signals were very sensitive to minor localized damage.

The output signals usually contain peak point when the failures are occurred. Based on this assumption, principal of wavelet transform method is that the frequency components outside a certain range are set to zero, which may cause some useful failure information to be lost since some burst failures often appear as impulses in signals (Peng and Chu 2004). Therefore, wavelet transform method can be useful to solve the bottleneck of the massive data transmission and efficient to reduce the cost of data transmission while improving the performance.

Attribute subset selection: Another way to reduce the dimensionality is to use only a useful attributes of all attributes. Real data set contain many types of attributes because data are recorded from every sensor. These attributes contain some irrelevant or redundant attributes which can slow down the analysis process and result in poor quality. Therefore, some methods are needed to select the useful attributes. Actually, the most accurate method is to pick out some of the useful attributes from expert. However, sometimes it is difficult when especially the data behavior is not well known. Therefore, standard rules are needed to determine the useful attributes. Attribute subset selection reduce the data size by removing irrelevant or redundant attributes (Han 2006). This method is to try all possible subsets of attributes as input to the analysis algorithm, and then select the subset which produces the best result (Tan et al. 2006). Attribute subset selection consists of four steps. Subset generation is a first step to specify a candidate subset for evaluation. After generating subset, each candidate subset is evaluated and compared with previous best one according to a certain evaluation criterion such as dependency, distance, and information gain (Liu and Yu 2005).

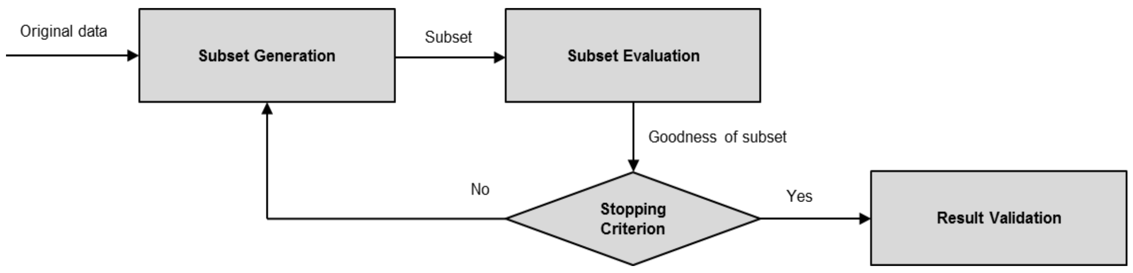


Figure 2.8: Process of Attribute subset selection(Liu and Yu 2005)

If the new subset is better than previous one, this new subset would be replaced with the previous one. At the last, process of subset generation and evaluation phases are repeated until specific stopping criterion is satisfied. These methods are usually heuristic search algorithm, and these help to extract the solutions as quickly. Heuristic search algorithms of attribute subset selection usually include stepwise-forward selection, stepwise backward elimination, combination of forward selection and backward elimination, and decision tree induction. The most methods of attribute subset selection are typically determined using statistical significance and information gain. These methods can reduce the number of attributes and help to extract the patterns more easily. However, it takes much time to test the significance of all of combinations and they need too much memory size to store all combinations.

Clustering: Clustering means that partition data set into clusters usually based on distance function. Data acquisition system only gets the specific data related with objective in failure analysis because it is difficult to store all of continuous data in time series data. Adaptive method is proposed for finding rules from time series (Das et al. 1998). They first form subsequence by sliding a window from the time series data, and then cluster these subsequences by using a suitable measure of similarity.

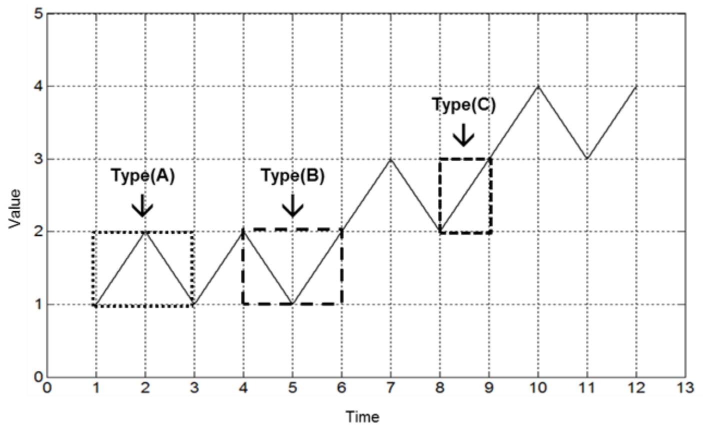


Figure 2.9: Clustering of time series data (Das et al. 1998)

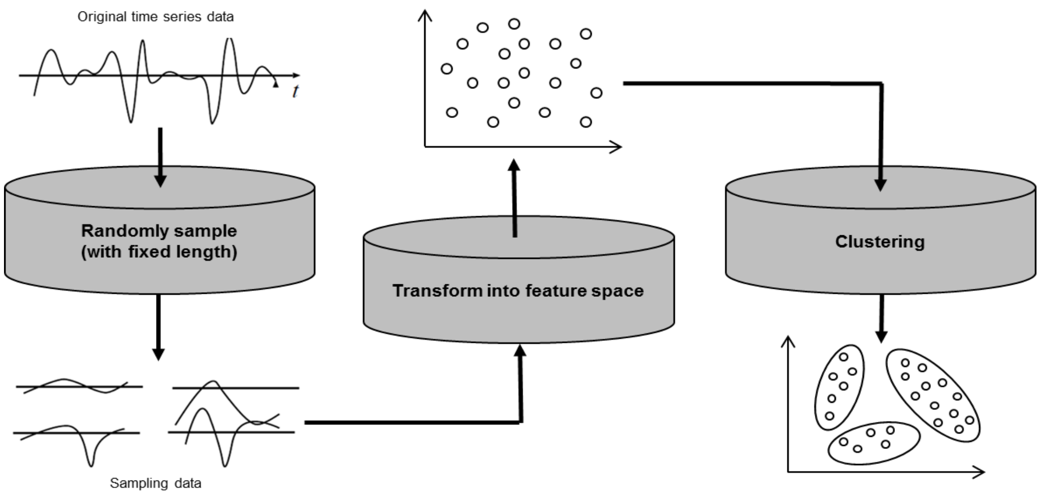

Original time series data
t
Randomly sample (with fixed length)
Transform into feature space
Clustering
Sampling data

Figure 2.10: Clustering for data reduction

In Figure 2.9, each symbol (‘a’, ‘b’, ‘c’) represent an original shape when window size is ‘3’. Using training data, different types of pattern are defined before getting time series data. After define the different types of patterns, distance is measured between entering data and defined pattern to measure similarity. Time series data is converted into symbol of similar shape (pattern) by looking for similar subsequence. For example, the original time series data is [1 2 1 2 1 2 3 2 3 4 3 4] and reduced data set is [a b a b c a b c a b]. In this application, the shape of the subsequence is used as the main factor in distance determination, and then time series data is obtained by taking the cluster identifiers corresponding to the subsequence.

To extract a set of specific patterns from each time series and detect anomaly, clustering and mining association rules are used in time series data (Yairi et al. 2001). For data reduction, clustering is used to extract a finite number of representative signal patterns from time series data. Procedure is displayed in Figure 2.10. The original time series data are sampled with fixed length, and then sampled data is converted into specified subsequences. Each subsequence is transformed into feature space by calculating a feature value including minimum or maximum value, etc. These subsequences are clustered by applying a k-means method because of measuring similarity among the subsequences. Each cluster is labeled as a pattern class which is corresponding to a certain event. These reduced data are used for association mining and certain relations can be extracted. However, this method has some problems because users should decide some parameters, and they require little a priori knowledge on the system. It is not easy to decide variety of parameters properly in advance. Therefore, automatic decision method is needed to increase accuracy.

Sometimes original data contain huge volume and mass, it is not suitable for data-driven failure analysis. Useful features in original data can be extracted by using data reduction techniques (wavelet transform, attribute subset selection, clustering). These data reduction techniques are usually proper to time series data however if time series data can be converted into vector space, data reduction techniques are useful in the data preprocessing. Data reduction techniques help to eliminate irrelevant feature and noisy, and data can be converted visual format to analyze more easily.

Discretization

The goal of discretization is to find a set of cut points to partition the range into a small number of intervals that have good class coherence, which is usually measured by an evaluation function (Kotsiantis and Kanellopoulos 2006). Many discretization algorithms have been proposed and tested to reduce the large data while improving accuracy. According to the data format, discretization algorithms features are divided as below and classification of discretization algorithms are given as below Table 2.1.

- Supervised or Unsupervised: It is divided depending on whether class information is considered or not. Supervised consider class information, but unsupervised do not consider class information. In unsupervised, just continuous data are divided into sub-ranges so that accuracy is lower than supervised.
- Global or Local: Global uses the entire instance space while local uses a localized region of the instance space.
- Direct or Incremental: Direct divides specified range of intervals, needing an user defined rang of interval k. Incremental divides range of intervals automatically through an improvement process, but users should define when to stop discretizing.
- Splitting or Merging: Split starts with empty of cut-points and adds new cut-point of all the continuous values as the discretization progresses. Merge starts with complete of cut-points and remove some of cut-points by merging intervals.
- Static or Dynamic: Static is processed before the classification task so that it is considered preprocessing phase. Dynamic would discretize continuous values when a classifier is being built so that it needs training data.

Usually, it is difficult to find a suitable discretization algorithm for certain data format, and different results can be extracted along with each discretization algorithm. However, data can be reduced and simplified through discretization techniques. In this Chapter, developed typical discretization algorithms are described.

Table 2.1: Classification of discretization algorithms (Liu et al. 2002)

Methods	Supervised/ Unsupervised	Global/ Local	Direct/ Incremental	Splitting/ Merging	Static/ Dynamic
Equal-width	Unsupervised	Global	Direct	Splitting	Static
Equal-frequency	Unsupervised	Global	Direct	Splitting	Static
1r	Supervised	Global	Direct	Splitting	Static
D2	Supervised	Local	Indirect	Splitting	Static
Entropy	Supervised	Local	Indirect	Splitting	Static
Mantaras	Supervised	Local	Indirect	Splitting	Static
Id3	Supervised	Local	Indirect	Splitting	Dynamic
Zeta	Supervised	Global	Direct	Splitting	Static
Accuracy	Supervised	Global	Direct	Splitting	Static
Chimerge	Supervised	Global	Indirect	Merging	Static
Chi2	Supervised	Global	Indirect	Merging	Static
Conmerge	Supervised	Global	Indirect	Merging	Static

Binning: It is the simplest algorithm using specified bins. Equal-width binning and Equal-frequency binning is considered as binning algorithm. Equal-width binning divides the continuous range of a feature into intervals that have an equal-width and each interval represents a bin (Liu et al. 2002). For example, the result is displayed in Figure 2.11 when attribute values are [0 4 12 16 16 18 24 26 28] and bin width is 3. Bin1 includes the range of 0-10 and the other bins include same format. Equal-width binning is the simples and fast algorithm however this binning technique can produce bad results when there are many occurrences of one range and very few of the other ranges.

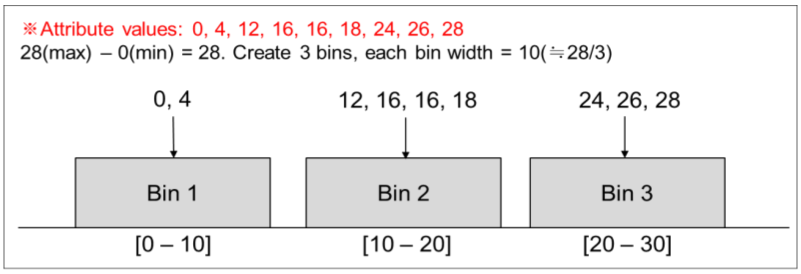

※Attribute values: 0, 4, 12, 16, 16, 18, 24, 26, 28
28(max) – 0(min) = 28. Create 3 bins, each bin width = 10(≒28/3)
0, 4
12, 16, 16, 18
24, 26, 28
Bin 1
Bin 2
Bin 3
[0 – 10]
[10 – 20]
[20 – 30]

Figure 2.11: Example of equal-width binning

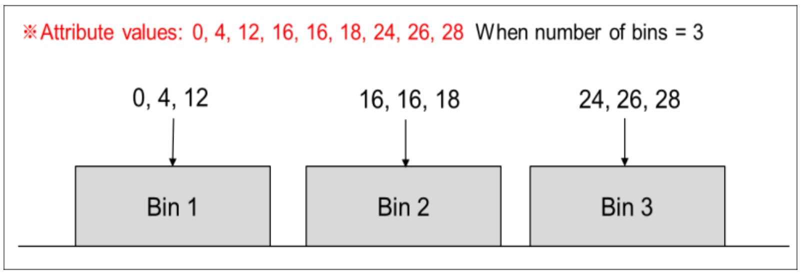

※Attribute values: 0, 4, 12, 16, 16, 18, 24, 26, 28 When number of bins = 3
0, 4, 12
16, 16, 18
24, 26, 28
Bin 1
Bin 2
Bin 3

Figure 2.12: Example of equal-frequency binning

Equal-frequency binning divides the equal number of continuous in each bin. For example, the result is displayed in Figure 2.12 when attribute values are [0 4 12 16 16 18 24 26 28] and bin width is 3. Each bin has same number of attribute values. Equal-width binning and equal-frequency binning do not use class information, and user just specified number of bins is needed. Binning algorithms are sensitive along with a given user specified number of bins, and it is difficult to select a suitable number of bins. Many researches have been proposed to choose number of bins automatically (Bay 2001; Boulle 2005; Jiang et al. 2009). Sometimes, classification information can be lost by binning as a result of combining values that are strongly associated with different classes into the same bin because binning(unsupervised discretization) algorithms do not utilize class information in setting partition boundaries (Kerber 1992).

1R (One rule discretization): Simple binning algorithms do not consider class information. 1R algorithm is supervised discretization using binning. 1R divides the range of continuous values into a number of separate intervals like equal-frequency binning. Each interval should contain a minimum of 6 instances and does not allow terminating an interval. If the next instance has the same class label as the majority class label seen until then in the interval (Kerber 1992). For instance, each bin could contain 6 values and 6 classes when data set is [11 14 15 18 19 20 21 22 23 25 30 31 33 35 36] and the class set is [R C C R C R C R C C R C R C R] and minimum values of each interval is 6 in Figure 2.13. Bin1 contains [11 14 15 18 19 20] and [R C C R C R] but number of class R and class C in class set of bin1 is same. Next value and class (value: 21, class: C) should include because 1R algorithm consider class attribute. Along with this rule, bin1 contains [11 14 15 18 19 20 21] and [R C C R C R C] so that bin1 is class C. After this phase, 6 values are contained in bin2 and this rule will be applied again. At the result, 14 values are converted into 2 standard classes set [C R] and there are 8 miss-classifications. 1R algorithm of all binning algorithms is the one way to improve the accuracy, and it can also overcome such problems that focused to put two instances with the same class in two different intervals (Kerber 1992). 1R algorithm is faster than other supervised algorithms however the accuracy is lower.

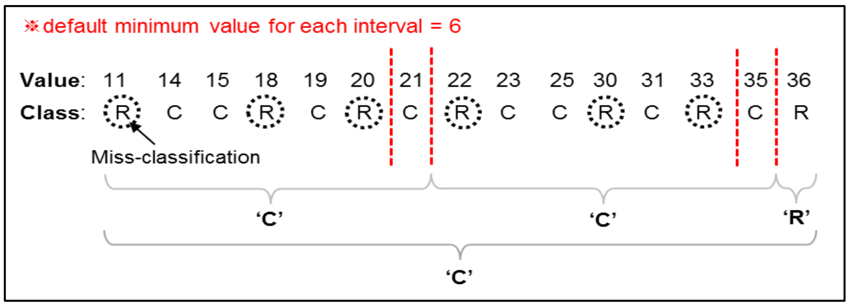

※default minimum value for each interval = 6
Value: 11 14 15 18 19 20 21 22 23 25 30 31 33 35 36
Class: R C C R C R C R C C R C R C R
Miss-classification
'C'
'C'
'R'
'C'

Figure 2.13: Example of 1R discretization

Entropy: Bin boundaries can be selected by calculating the entropy value which is denoting the uncertain degree of interval (Dougherty et al. 1995). This entropy value is calculated by considering the relation with objective attributes, and the smallest entropy of interval is selected as split point. Entropy value is calculated as below. 'x' is a value of X and p_x is a estimated probability of occurring.

$$Entropy(X) = -\sum_{x} p_x log p_x$$

To discrete the value, the splitting process can be repeated in the upper and lower parts of the range (Witten et al. 2011). If original data set is [64 65 68 69 70 71 72 72 75 80 81 83 85] and class set is [Y N Y Y Y N N Y Y Y N Y Y N] in Figure 2.14. The cut-point with the lowest entropy is chosen to split the range into two parts. It is continued until stopping criterion is satisfied. At first, all of entropy is calculated in all possible partition boundaries then boundary 84 is the lowest entropy value. After that this phase is repeated without values upper than 84 like Figure 2.14. Finally, the result of discretized data [64 65 | 68 69 70 | 71 72 72 | 75 75 | 80 | 81 83 | 85] is extracted then 14 values are reduced 6 discretized data set.

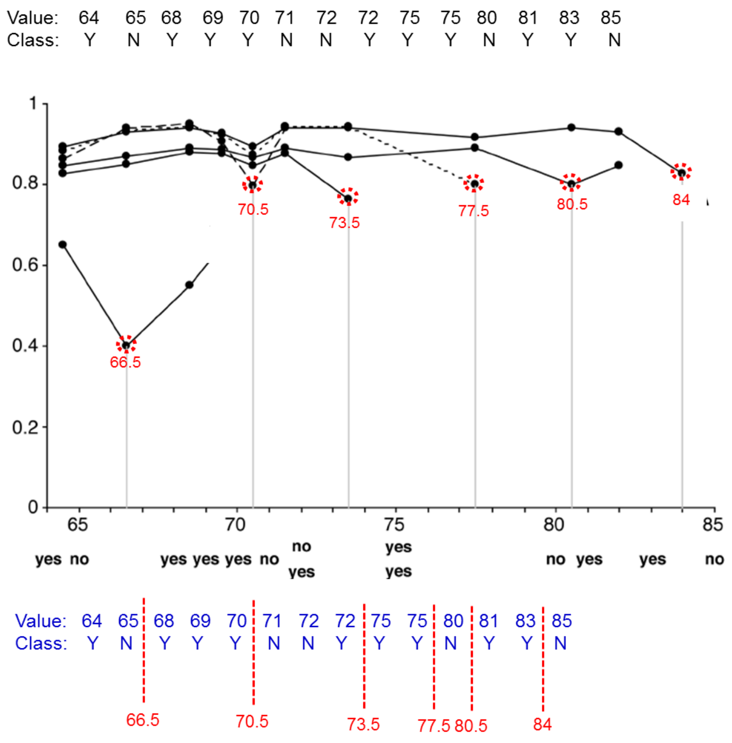


Figure 2.14: Entropy based discretization (Witten et al. 2011)

Entropy based discretization measure purity and specifies to which class a value belongs. Zeta maximization algorithm (Ho and Scott 1997) is similar to entropy based discretization which measure based on minimization of the error rate when each value of an independent variable must predict a different value of a dependent variable. These entropy and zeta algorithm can select the cut-point automatically but one problem is to determine stopping criterion.

ChiMerge: This technique tests the hypothesis that the two adjacent intervals are independent by measuring the expected frequency of the classes represented in each of intervals (Dougherty et al. 1995) . It is a top-down method and searches the suitable cut-point based on the maximizing the chi-square criterion. If two adjacent intervals are independent with the lowest chi-square value, two intervals would be merged. Otherwise, they should remain separate. Merging is repeated until chi-square value is smaller than already specified threshold. Chi-square value can be calculated as below.

$$\chi^2 = \sum_{i=1}^{2}\sum_{j=1}^{k}\frac{(A_{ij}-E_{ij})^2}{E_{ij}}$$

Where

- k = number of classes
- A_{ij} = number of samples in ith interval, jth class
- E_{ij} = expected frequency of A_{ij} (= (R_i x C_j) / N)
- R_i = number of samples in ith interval
- C_j = number of samples in jth class
- N = total number of samples on the two intervals
- If E_{ij}=0 then set E_{ij} to an small value for instance 0.1

The contingency table is needed for ChiMerge discretization. For instance, 12 samples are conducted ChiMerge discretization procedures in Table 2.2 with threshold 2.706. Chi-square value is 0.2 when splitting point is [7.5, 8.5]. This Chi-square value is less than threshold value therefore these two intervals ([value: 8, class: 1] and [class: 9, class: 1]) should be merged in Table 2 (b). However, chi-square value is 2.72 when splitting points are [0, 10] and [10, 42]. It is higher than threshold value therefore these, and then these two intervals ([value: 1 3 7 8 9, class: 1 2 1 1 1] and [value: 11 23 37 39, class: 2 2 1 2]) remain separate. At the result, final discretized intervals are [0, 10], [10, 42], [42, 60], and then 12 samples are converted into 3 intervals.

Table 2.2: Contingency table of the ChiMerge discretization

Sample	Value	Class
1	1	1
2	3	2
3	7	1
4	8	1
5	9	1
6	11	2
7	23	2
8	37	1
9	39	2
10	45	1
11	46	1
12	59	1

(a) Original data set

	Class1	Class2	Sums
Interval1	1(A_{11})	0(A_{12})	1(R_1)
Interval2	2(A_{21})	1(A_{22})	1(R_2)
Sums	2(C_1)	4(C_2),	2(N)

(b) The contingency table when splitting point is [7.5, 8.5] and [8.5, 10]: chi-square value is 0.2 (no significant different) = merge

	Class1	Class2	Sums
Interval 1	4(A_{11})	1(A_{12})	5(R_1)
Interval 2	1(A_{21})	3(A_{22})	4(R_2)
Sums	5(C_1)	4(C_2),	9(N)

(c)The contingency table when splitting points are [0, 10] and [10, 42]: chi-square value is 2.706 (significant different) = no merge

Threshold is determined by user however it also difficult to select suitable threshold such as the other discretization algorithms. Chi2 solves this problem by changing the statistical significance level and more adjacent intervals as long as the inconsistency criterion is satisfied (Liu and Setiono 1995). This technique can select the cut-point automatically and remove irrelevant features.

CAIM (Class-Attribute Interdependence Maximization): The goal of CAIM algorithm is to find a partition scheme that maximizes the interdependence and minimizes the information loss. To find the optimal discretization, class attribute interdependency is used. It measures the dependency between the class variable C and the discretization variable D for attribute F using a given quanta matrix which is displayed on the Table 2.3(Kurgan and Cios 2004).

$$CAIM(C,D|F) = \frac{\sum_{r=1}^{n} \frac{max_r^2}{M_{+r}}}{n}$$

Where:

- qir is the total number of continuous values belonging to the ith class that are within interval
- Mi+ is the total number of objects belonging to the ith class
- M+r is the total number of continuous values of attribute F that are within the interval (dr-1, dr] n is the number of intervals
- r is iterates through all intervals, i.e. r = 1, 2 ,..., n
- maxr is the maximum value among all qir values (maximum in the rth column of the quanta matrix), i = 1, 2, ..., S,

The larger value of CAIM indicates the higher interdependence between the class labels and the discrete intervals. The algorithm assumes that every discretized attribute needs at least the number of intervals that is equal to the number of classes (Cios et al. 1998). Candidate interval boundaries and initial discretization scheme are initialized, and then the new cut-point which is the locally highest value of the CAIM is added consecutively. CAIM algorithm does not need an input of the user specified parameters and it can generate the number of intervals automatically. It is fast and efficient supervised discretization algorithm however its execution time is comparable to the time required by the simplest unsupervised discretization algorithms (Cios et al. 1998).

CAIM have some more problems that generate a simple discretization scheme in which the number of intervals is very close to the number of target classes and considers only the class with the most samples and ignores all the other target classes (Tsai et al. 2008). For this reasons, CACC(Class-Attribute Contingency Coefficient) discretization algorithm (Tsai et al. 2008) is suggested by finding the best division point and recording a Global-CACC value.

Table 2.3: Quanta matrix of CAIM algorithm

Class	Interval					Class Total
	$[d_0, d_1]$	…	$(d_{r-1}, d_r]$	…	$(d_{n-1}, d_n]$	
C_1	q_{11}	…	q_{1r}	…	q_{1n}	M_{1+}
:	:	…	:	…	:	:
C_i	q_{i1}	…	q_{ir}	…	q_{in}	M_{i+}
:	:	…	:	…	:	:
C_S	q_{S1}	…	q_{Sr}	…	q_{Sn}	M_{S+}
Interval Total	M_{+1}	…	M_{+r}	…	M_{+n}	M

Table 2.4: Summary of data filtering methods

Filtering	Method	Algorithms	Authors/ Year	Description	Features
Data Reduction	Wavelet Transform	CWT (Continuous Wavelet Transform)	(Grossmann et al. 1989)	Compare the transformed wavelet and original signal to calculate the similarity	Good frequency resolution and low time resolution for low frequency resolution and good time resolution
		STFT (Short Time Fourier Transform)	(Koo and Kim 2000)	It performs a mapping of one-dimensional signal to a two dimensional function of time and frequency	Depending on window size, performance is good
		DWT (Discrete Wavelet Transform)	(Chen et al. 1999)	Signal is converted into a numerally different vector of wavelet coefficients. These coefficients are used as failure features.	It is useful to solve the bottleneck of the massive data.
		Soft-thresholding denoising algorithm	(Lin and Qu 2000)	It used the wavelet entropy as a rule to optimize parameters of the wavelet function and impulse components as failure are extracted.	Threshold is defined by user so it is difficult to select suitable threshold.
			(Lu and Hsu 2000)	They defined that the maximum change of the wavelet coefficient was corresponding to the location of the damage.	It is proper specified signal that can be changed sensitively by abnormal signal.
	Attribute subset selection	Stepwise forward selection	(Derksen and Keselman 1992)	It will be start from empty set and the best subset is added.	Sometimes this can be a difficult and time consuming task, especially when the behavior of the data is not well known.
		Stepwise backward elimination	(Derksen and Keselman 1992)	It will be start from full set and the worst attribute is removed.	
		Combination of forward selection and backward elimination	(Guyon and Elisseeff 2003)	At each step, the procedure selects the best attribute and removes the worst from among the remaining attributes.	It is more accuracy than forward selection and backward elimination. In addition it can slow down the mining process.
		Decision tree induction	(Siciliano and Conversano 2006)	A flow-chart is constructed and each internal node denotes a test on an attribute. At each node, the algorithm chooses the best attribute to partition the data into individual classes.	It takes much time to construct the decision tree and needs an exclusive knowledge of system.
	Clustering	Rule discovery in time series	(Das et al. 1998)	Subsequences are extracted by sliding window and cluster these subsequence by using a suitable measure of similarity.	It is useful to extracted features of signal. However, it is difficult to find the feature of signal for massive data.
		Sampling	(Yairi et al. 2001)	To extract a set of specific patterns from each time series and detect anomaly, clustering and mining association rules are used in time series data.	User should decide some parameters and it requires little a priori knowledge on the system. it is not easy to decide variety of parameters properly in advance.

Filtering	Method	Algorithms	Authors/ Year	Description	Features
Discretization	Unsupervised approaches	Equal-width binning	(Dougherty et al. 1995)	It divides the continuous range of a feature into intervals that have an equal-width and each interval represents a bin.	It is simple easy to implement but it may produce bad results when there are many occurrences of one range, and vey few of the other ranges.
		Equal-frequency binning	(Boulle 2005)	It divides the range into N interval, each containing approximately same number of samples.	Classification information will be lost because binning does not utilize class information.
	Supervised approaches	1R	(Liu et al. 2002)	It supervised discretization using binning. From partitions such that each group contains a large majority of a single classification.	It is faster than unsupervised discretization but there are some miss-classifications.
		Entropy	(Kotsiantis and Kanellopoulos 2006)	It uses class information entropy of candidate partitions to select bin boundaries. Minimal value among all candidates is selected as cut-point.	It takes much time to compute all of candidates and suitable stopping criterion is needed.
		ChiMerge	(Kerber 1992)	For two adjacent intervals, if chi-square test concludes that the class is independent of the intervals, the intervals should be merged. It is repeated until chi-square values of all adjacent pairs exceed a threshold.	Accuracy is higher than the other methods but threshold is determined by user. Chi2 algorithm is solved this problem.
		CAIM(Class-Attribute Interdependency Maximization)	(Kurgan and Cios 2004)	It uses greed top-down approach that finds local maximum values of CAIM. After calculate CAIM values of all candidates and it is compared with threshold.	It is fast and efficient and it can generate small number of intervals compared with entropy approach. Its execution time is comparable to the time required by the simplest unsupervised discretization algorithm.

Pattern extraction

Pattern extraction methods are used to extract frequent pattern among the massive data. Frequent pattern can be defined that certain combinations of its elements occur more than a predefined minimum support threshold in a given ordered data list or transaction set. Frequent pattern mining was first introduced for mining transaction databases (Agrawal et al. 1993). Many previous researches contributed to the efficient mining (Han et al. 2007). Depending on considering sequence, pattern mining methods are divided as association rules and sequence rules.

Association rule

Association rule relates two itemsets ‘A’ and ‘B’. Such a rule indicates that A is strongly dependent on B, which is denoted by ‘A→B’ (Ma and Hellerstein 2002). Initially used for market basket analysis to find how items purchased by customers are related. For example, huge amounts of customer purchase data are collected daily at grocery stores. Association rules discover interesting relationships hidden in collected data. when people buy tea, they are also likely to buy coffee (Brin et al. 1997). Support and confidence are used to measure the significant. Support is an important measure because a rule that has very low support may occur simply by chance (Tan et al. 2006). It is usually used to eliminate uninteresting rules. Confidence, on the other hand, measures the reliability of the inference made by rule. These two measurements are defined as below.

- Support: A and B occur together in at least s (support) % of the n baskets.
- Confidence: Of all the baskets containing A, at least c (confidence)% also contain B

Association rules usually generate all candidate itemsets of all data and determine which of the candidate itemsets are frequent. Apriori, AprioriTID, and FP-Tree algorithms are considered as association rules.

Apriori: It is proposed to solve the disadvantages of AIS (Agrawal et al. 1993) and SETM (Houtsma and Swami 1995) algorithm. AIS and SETM algorithm generate candidate itemset and counted on-the-fly as the database is scanned. After reading a transaction, it is determined which of the itemsets that were found to be large in the previous pass are contained in this transaction. New candidate itemsets are generated by extending these large itemsets with other items in the transaction (Agrawal and Srikant 1994).This algorithm takes much time and memory because it generates and counts every candidate itemset.

The candidate itemset which is determined as frequent itemsets in previous phase is generated and counted to reduce the number of scan. Apriori algorithm consists of two steps as below (Tan et al. 2005).

- Let k=1
- Generate frequent itemsets of length 1
- Repeat until no new frqeunt itemsets are identified
  - ✓ Generate length (k+1) candidate itemsets from length k frequent itemsets
  - ✓ Prune candidate itemsets containing subsets of length k that are infrequent
  - ✓ Count the support of each candidate by scanning the database
  - ✓ Eliminate candidates that are infrequent, leaving only those that are frequent

Table 2.5: Notation of association rules

k-itemset	An itemset having k items
L_k	Set of large k-itemsets (those with minimum support) Each member of this set has two field: i) itemset and ii)support count
C_k	Set of candidate k-itemsets (potentially large itemsets) Each member of this set has two field: i)itemset and ii)support count
$\overline{C}_k$	Set of candidate k-itemsets when the TIIDs of the generating transactions are kept associated with the candidates

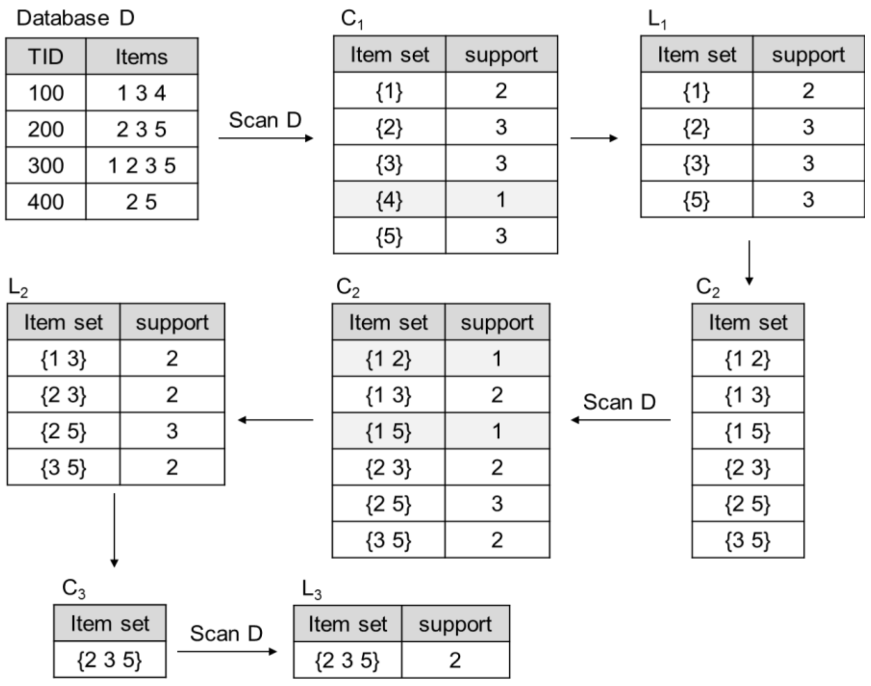
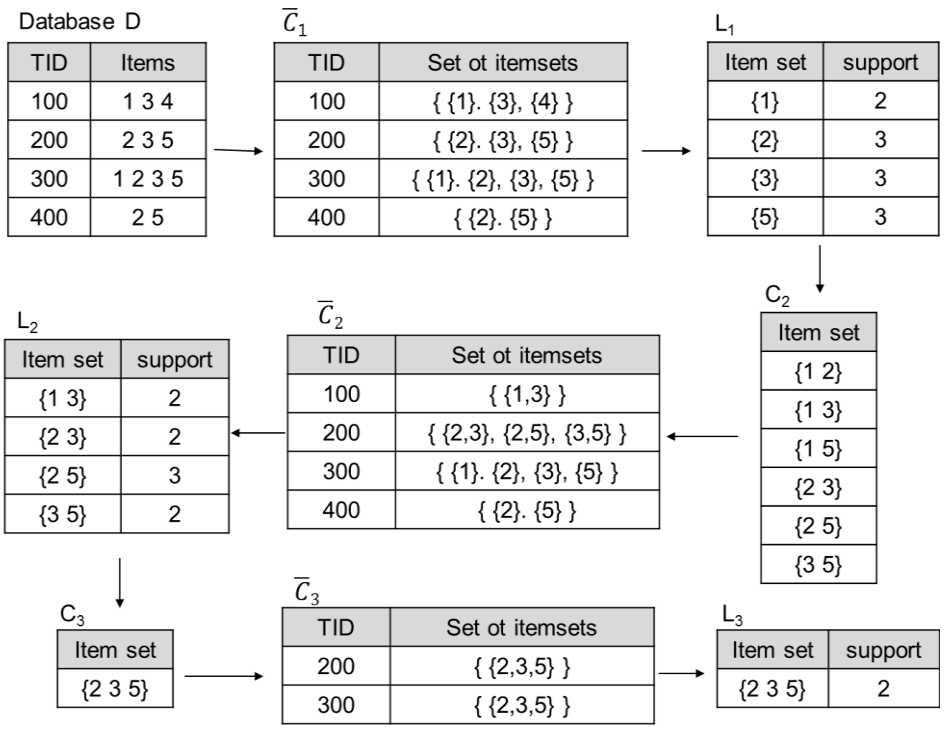


Figure 2.15: Apriori algorithm (Minimum support =2)

The example of Apriori algorithm is displayed in Figure 2.15 in accordance with Table 2.5. As shown in Figure 2.15, 1-length itemsets are generated and counted the support. If the support of certain item is less than minimum support, this item would be eliminated. Based on the remained 1-length itemsets, 2-length candidates are created, and then the support is counted again. These steps are repeated until there is no frequent itemset. Consequently, 3-length itemset is frequent itemset in this example. If an itemset is frequent, the all of its subsets must also be frequent in this Apriori algorithm. It uses large itemset properly and easy to parallelize and implement however, they construct huge candidate sets and multiple scans of database to count of the support. Though counting of the infrequent patterns are not necessary, after scanning the database. For these reasons, it takes much time and needs memory.

AprioriTID: It is used to eliminate the counting phase in Apriori algorithm. This algorithm does not use the transactions in the database for counting itemset support. Instead of using the counting itemsets $\bar{C}_k$ is used in Table 2.5. Each member of the set $\bar{C}_k$ is of the form <TID, X_k>, where each X_k is a potentially large k-itemset present in the transaction with identifier TID (Agrawal and Srikant 1994). The example of AprioriTID algorithm is displayed in Figure 2.16. Database is same as Apriori algorithm in Figure 2.15however the feature of AprioriTID algorithm is $\bar{C}_k$. As mentioned before, this algorithm does not need to scan for counting the support because possible itemsets in each TID are constructed before the scanning in $\bar{C}_k$.

If a transaction does not contain any candidate k-itemset, then $\bar{C}_k$ will not have an entry for this transaction (Agrawal and Srikant 1994). When k (length) is large, number of entries in $\bar{C}_k$ may be less than the number of transaction in database so speed is fast. However speed is slow and much space is needed when k is small. The number of entries in $\bar{C}_k$ may be large than the number of transaction in database. To solve these problem, AprioriHybrid algorithm is proposed (Agrawal and Srikant 1994; QIN and SONG 2004) by combining two algorithm (Apriori + AprioriTID). This algorithm begins with Apriori algorithm and AprioriTID algorithm is applied when $\bar{C}_k$ can fit in main memory.


Database D

TID	Items
100	1 3 4
200	2 3 5
300	1 2 3 5
400	2 5

$\bar{C}_1$

TID	Set ot itemsets
100	{ {1}. {3}, {4} }
200	{ {2}. {3}, {5} }
300	{ {1}. {2}, {3}, {5} }
400	{ {2}. {5} }

L_1

Item set	support
{1}	2
{2}	3
{3}	3
{5}	3

C_2

Item set
{1 2}
{1 3}
{1 5}
{2 3}
{2 5}
{3 5}

$\bar{C}_2$

TID	Set ot itemsets
100	{ {1,3} }
200	{ {2,3}, {2,5}, {3,5} }
300	{ {1}. {2}, {3}, {5} }
400	{ {2}. {5} }

L_2

Item set	support
{1 3}	2
{2 3}	2
{2 5}	3
{3 5}	2

C_3

Item set
{2 3 5}

$\bar{C}_3$

TID	Set ot itemsets
200	{ {2,3,5} }
300	{ {2,3,5} }

L_3

Item set	support
{2 3 5}	2

Figure 2.16: AprioriTID algorithm (minimum support = 2)

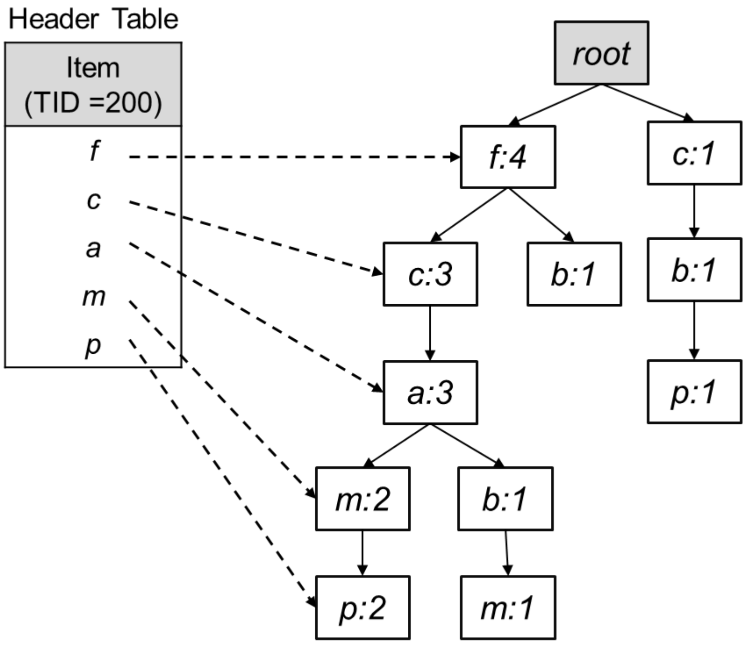
FP-Tree: Apriori algorithm is costly to handle a huge number of candidate sets. To avoid this problem, FP-Tree algorithm is constructed. Frequent pattern tree is novel and compact data structure which is an extended prefix-tree structure storing crucial, quantitative information about frequent patterns (Han et al. 2004). Only two scans are needed. One is to find the frequent items and the other to construct the FP-Tree (Mitsa 2009). When the numbers of transactions increase, FP-Tree has much scalability than the Apriori algorithm. Procedures of FP-Tree are as below.

- Scan database once, find frequent 1(length)-itemset
- Sort frequent items in frequency descending order as FList
- Scan database again, construct FP-Tree
- For each frequent item, construct its conditional pattern-base, and conditional FP-Tree
- Repeat the process on each newly created conditional FP-Tree
- Until the resulting FP-Tree is empty, or it contains only one path-single path will generate all the combinations of its sub-paths, each of which is a frequent pattern

Table 2.6 is displayed an example of FP-Tree. At first, original data set is sorted and F-List is constructed as [$f - c - a - b - m - p$]. Along with the F-List, the item whose frequency is less than threshold will be eliminated. Ordered frequent items which are remained from original items are displayed in Table 2.6 and FP-Tree is constructed based on these ordered frequent items. The FP-Tree top node is labeled as the root. Each subsequent node correspond to an item whose count is also noted in the label of the node in Figure 2.17 and the count represents the number of transactions that are present in the path reaching the node (Mitsa 2009). FP-Tree will merge transactions that share a common prefix item set. As shown Figure 2.17, items {f, c, a, b, m} are inserted into FP-Tree when TID is 200. Item ‘f’ can be shared with the other ‘f’ which was defined in TID 100, then item would be merged and the number of transaction is increased. If inserted node cannot share any other nodes, new branch would be made. These phases are repeated until TID 500 in example and then FP-Tree is constructed. Using FP-Tree, conditional pattern tree and conditional FP-Tree are constructed to find frequent patterns as displayed in Table 2.7. All the patterns containing frequent items that a certain node participates can be collected by starting at certain node’s node-link head and following its node-link (Han et al. 2004). For node ‘p’, FP-Tree in Figure 2.17 has two nodes ‘p’ and its immediate frequent is (p:3). Following these two node, it has two paths in the FP-Tree that are <f:4, c:3, a:3, m:2> and <c:1, b:1>. First path <f:4, c:3, a:3, m:2> means that ‘f’ appears four time and ‘c’ appears three times so on. However <f, c, a, m> can appear twice together with ‘p’ and <c, b> can appear once together with ‘p’. These two prefix paths of ‘p’, *{(fcam:2), (cb:1)}* is extracted as conditional pattern base. Construction of a FP-Tree on this conditional pattern base leas to only one branch (c:3) because ‘f’, ‘a’ and ‘m’ appear only two time even though ‘p’ appears three times in one branch. Similarity other nodes can extract conditional pattern base and conditional FP-Tree as displayed in Table 2.7. These are repeated until the result FP-Tree is empty, or it contains only one path-single path will generate all the combinations of its sub-paths, each of which is a frequent pattern.

Table 2.6: A transaction database in FP-Tree (minimum support=3) (Han et al. 2004)

TID	Items	(Ordered) Frequent items
100	{f, a, c, d, g, i, m, p}.	{f, c, a, m, p}
200	{a, b, c, f, l, m, o}	{f, c, a, b, m}
300	{b, f, h, j, o}	{f, b}
400	{b, c, k, s, p}	{c, b, p}
500	{a, f, c, e, l, p, m, n}	{f, c, a, m, p}



•

Figure 2.17: FP-Tree constructions

Table 2.7: Creating conditional pattern-base

Item	Conditional pattern-base	Conditional FP-Tree
p	{(fcam:2), (cb:1)}	{(c:3)|p
m	{(fca:2), (fcab:1)}	{f:3, c:3, a:3}|m
b	{(fca:1), (f:1), (c:1)}	{f:3, c:3, a:3}|m
a	{(fc:3)}	∅
c	{(f:3)}	{(f:3)}|c
f	∅	∅

The most important feature of FP-Tree is that it can avoid repeated scan of entire database. FP-Tree does not generate candidates and has compressed database format. For these reasons, it is usually faster than the Apriori algorithm. However FP-Tree will be complicated when transaction has a unique set of items. In this case, size is same as the original data since transaction does not have common data sets. To solve these problems, advanced FP-Tree algorithms are proposed (Grahne and Zhu 2005; Hong et al. 2008; Sui et al. 2008).

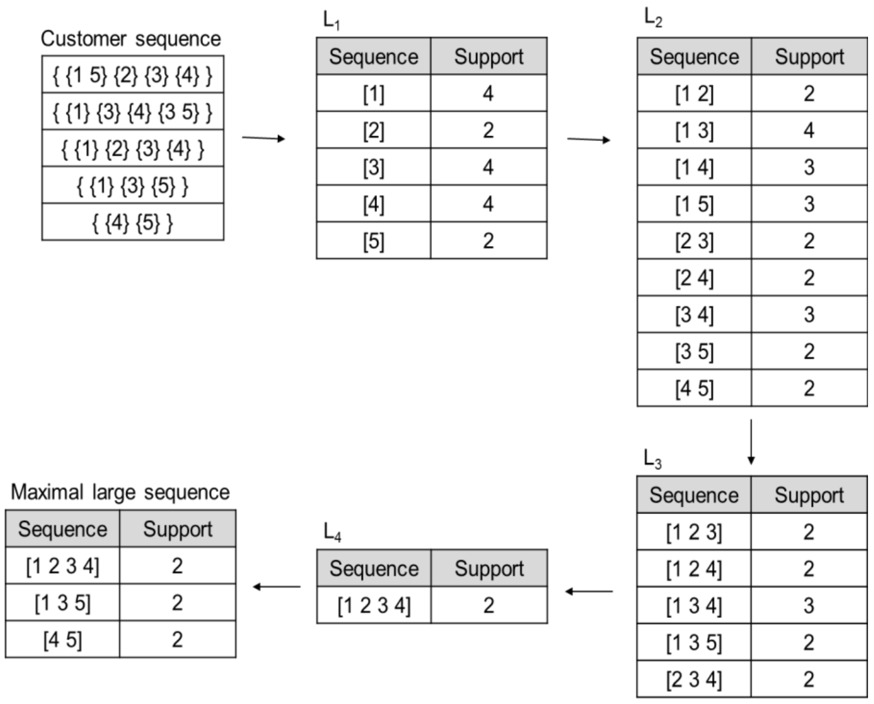
Sequence rule

A sequence is a time ordered list of objects, in which each object consists of an itemset, with an itemset consisting of all items that appear together in a transaction (Mitsa 2009). An example of a sequence is <(AB), (CD)>, where items (AB) indicate the items that were purchased together. In case of sequence rule, (CD) are purchased after (AB) were purchased however association rule does not considered the order of (AB) and (CD). Each transaction consists of sequence-id, transaction id, transaction-time, items and AprioriAll, AprioriSome, AprioriDynamic, GSP, SPADE, FreeSpan, and Frefix algorithms are considered.

AprioriAll: It is Count-All algorithm of the Apriori approaches. It uses the large sequences from the previous pass to generate the candidate sequences and then measure their support by making a pass over the database (Agrawal and Srikant 1995). After that the minimum support is used to determine the large sequences. Detailed procedure is described as below (Agrawal and Srikant 1995).

- Sort : transformas customer transactions into customer sequences
- Construct L-itemset : generate large itemsets
- Transformation : represents customer sequences based on large 1-length sequences
- Sequence : derive large k-length sequences based on large (k-1)-length sequences where k>= minimum support (threshold)
- Maximal: prunes non-maximal sequences.

Example of AprioriAll algorithm is displayed in Figure 2.18. At first 1-length sequences are generated using customer sequence. 2-length sequences are also generated using 1-length sequences and then prune the sequences whose support is less than minimum support. After generating 4-length sequences, maximal large sequences are generated. Maximal large sequences are generated using measure that certain sequence is contained in any other sequence or not. In case of 2-length, [4 5] is remained because the others were contained in 4-length [1 2 3 4] and in case of 3-length also remain only [1 3 5]. Therefore, final maximal large sequences are [1 2 3 4], [1 3 5], and [1 3].



Customer sequence

{ {1 5} {2} {3} {4} }
{ {1} {3} {4} {3 5} }
{ {1} {2} {3} {4} }
{ {1} {3} {5} }
{ {4} {5} }

L_1

Sequence	Support
[1]	4
[2]	2
[3]	4
[4]	4
[5]	2

L_2

Sequence	Support
[1 2]	2
[1 3]	4
[1 4]	3
[1 5]	3
[2 3]	2
[2 4]	2
[3 4]	3
[3 5]	2
[4 5]	2

L_3

Sequence	Support
[1 2 3]	2
[1 2 4]	2
[1 3 4]	3
[1 3 5]	2
[2 3 4]	2

L_4

Sequence	Support
[1 2 3 4]	2

Maximal large sequence

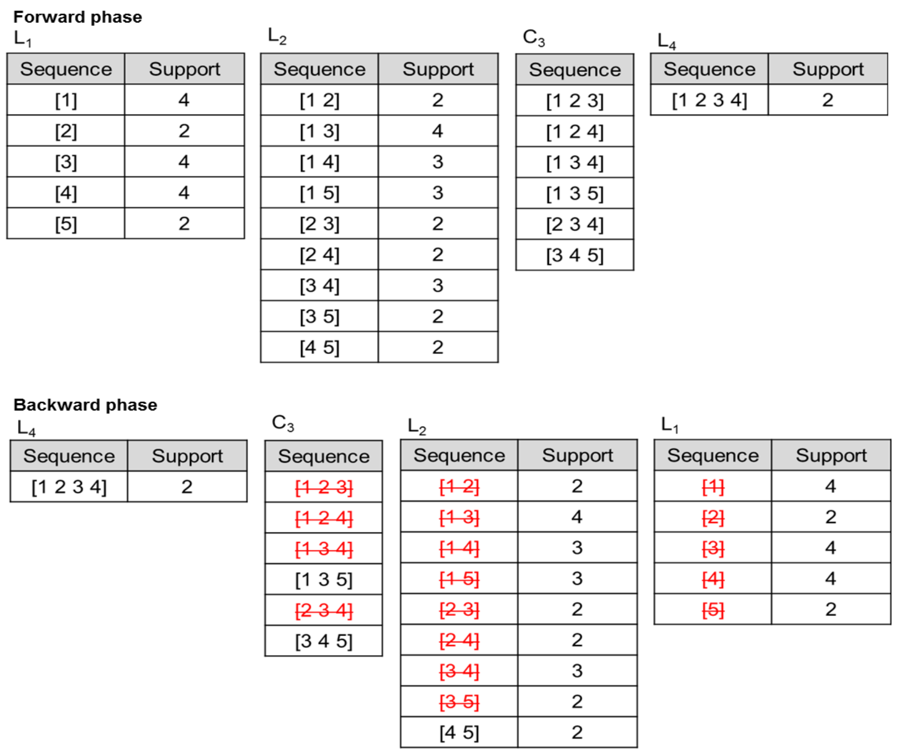
Sequence	Support
[1 2 3 4]	2
[1 3 5]	2
[4 5]	2

Figure 2.18: AprioriAll algorithm (minimum support =2) (Agrawal and Srikant 1995)

AprioriSome: This algorithm is the Count-Some algorithm. It can avoid counting sequences which are contained in a longer sequence if the first longer sequences is counted (Agrawal and Srikant 1995). This algorithm also helps to avoid counting a lot of longer sequences whose support is less than minimum support. AprioriSome algorithm consists of two phases as forward phase and backward phase. In forward phase, all large sequences of certain lengths are found, and then remaining large sequences are found in backward phase.

Considering data set which are same as AprioriAll algorithm, determines the length to count using next(k) function in AprioriSome algorithm. At first, 1-length sequences are extracted in forward phase. 2-length sequence and 4-length sequences are made when the k is two as displayed in Figure 2.19. In the backward phase, the skipped length sequences are counted and deleted which are contained in some large sequences as displayed in Figure 2.19. After two phases, only maximal sequences are remained which are same as result of the AprioriAll algorithm.

The feature of AprioriSome algorithm is to avoid counting many non-maximal sequences. However, the memory is consumed because candidate sets are generated to generate (k-1)-length sequences. The other Count-Some algorithm is DynamicSome algorithm. Its backward phase is same as AprioriSome algorithm however they will generate candidate sets (C_k).



Forward phase

L$_1$

Sequence	Support
[1]	4
[2]	2
[3]	4
[4]	4
[5]	2

L$_2$

Sequence	Support
[1 2]	2
[1 3]	4
[1 4]	3
[1 5]	3
[2 3]	2
[2 4]	2
[3 4]	3
[3 5]	2
[4 5]	2

C$_3$

Sequence
[1 2 3]
[1 2 4]
[1 3 4]
[1 3 5]
[2 3 4]
[3 4 5]

L$_4$

Sequence	Support
[1 2 3 4]	2

Backward phase

L$_4$

Sequence	Support
[1 2 3 4]	2

C$_3$

Sequence
~~[1 2 3]~~
~~[1 2 4]~~
~~[1 3 4]~~
[1 3 5]
~~[2 3 4]~~
[3 4 5]

L$_2$

Sequence	Support
~~[1 2]~~	2
~~[1 3]~~	4
~~[1 4]~~	3
~~[1 5]~~	3
~~[2 3]~~	2
~~[2 4]~~	2
~~[3 4]~~	3
~~[3 5]~~	2
[4 5]	2

L$_1$

Sequence	Support
~~[1]~~	4
~~[2]~~	2
~~[3]~~	4
~~[4]~~	4
~~[5]~~	2

Figure 2.19: AprioriSome backward phase(minimum support=2) (Agrawal and Srikant 1995)

GSP (Generalized Sequential Patterns): This algorithm is suggested to reduce the search space in AplriorAll and AprioriSome. It is a typical sequential pattern mining method and mines sequential patterns by adopting a candidate subsequence generation-and-test approach which are based on the Apriori principle (Han et al. 2001). GSP algorithm is consists of two phase as Join phase and detailed procedures are described as below.

- Initially, scan over the database to extract 1-length sequences
- Generate the set of all candidate k-length sequences, by joining two (k-1)-length sequences if only their first and last items are different
- Prune the sequences whose support is less than predefined minimum support
- Repeat until it cannot extract more candidate sets

Table 2.8: Candidate generation in GSP algorithm (Srikant and Agrawal 1996)

Frequent 3-length sequences	Candidate 4-length sequences	
	After join	After pruning
< (1,2) (3) >	< (1,2) (3,4) >	< (1,2) (3,4) >
< (1,2) (4) >	< (1,2) (3) (5) >	
< (1) (5,4) >		
< (1,3) (5) >		
< (2) (3,4) >		
< (2) (3) (5) >		

It is similar approach with Apriori algorithm. Table 2.8 displays an example of candidate generation. 3-length sequences are used to extract 4-length sequences. The sequence < (1,2) (3) > joins with < (2) (3,4) > to generate < (1,2) (3,4) > and with < (2) (3) (5) > to generate < (1,2) (3,5) >(Srikant and Agrawal 1996). After join phase, support of each sequences are calculated and compared with minimum support. If the support is less than minimum support, it will be deleted. It is repeated until there are no frequent patterns and then final frequent sequence patterns are extracted. Compared to the AprioriAll algorithm, GSP is faster because this algorithm regards fewer candidates. In addition, the AprioriAll has to find which frequent itemsets are presented in each element during the data transformation, and then determine which candidate sequences are presented (Srikant and Agrawal 1996). Due to this phase, the speed of the AprioriAll algorithm is slower than the GSP algorithm. However, many times of scanning is needed to generate a huge set of candidates.

SPADE (Sequential Pattern Discovery using Equivalence classes): To avoid repeated database scan and complex hash structures in GSP algorithm, SPADE algorithm is proposed (Zaki 2001). SPADE algorithm decomposes the original problem into smaller sub-problems and needs only three scans of database. Database for applying SPADE algorithm has a unique identifier as ‘SID (Sequence Identifier)’ and ‘EID (Event Identifier)’. ‘SID’ is a ordered sequence and ‘EID’ is a ordered event contained ‘SID’. It assumes that no sequence has more than one event with the same time stamp so ‘EID’ is used. 1-length sequence patterns are generated and each item should contain ‘SID’ and ‘EID’ as displayed in Table 2.9.

Based on this Table 2.9 (a), 2-length sequence patterns (‘ab’ and ‘ba’) can be generated. In case of ‘ab’, EID (a) should be faster than EID(b) in same SID condition because item ‘b’ can be generated after generating item ‘a’ as displayed in Table 2.9 (b). 3-length sequence patterns are also generated when EID condition is fitted in Table 2.8 (c). These patterns are also compared with minimum support and then items whose support is less than minimum support will be deleted. For enumerating frequent sequences of a class, memory to store intermediate id-list is needed. There are two types of memory as BFS (Breath First Search) and DFS (Depth First Search). BFS requires memory of all the classes on the two levels however DFS requires memory for two classes on the level.SPADE has some advantages. At first, it uses simple join operation using id-list which can reduce the memory using unique identifiers. Secondly, complicated hash tree structure does not need and only three scans are needed. Apriori, FP-Tree, and SPADE algorithms are the typical pattern mining algorithm and the summary of these algorithms are described in Table 2.10.

Table 2.9: Example of SPADE algorithm

SID(a)	EID(a)	SID(b)	EID(b)
1	1	1	2
1	2	2	3
1	3	3	2
2	1	3	5
2	4	4	5
3	2		
4	3		

(a) 1-length sequences: ‘a’ and ‘b’

SID	EID(a)	EID(b)	SID	EID(b)	EID(a)
1	1	2	1	2	3
2	1	3	2	3	4
3	2	5			
4	3	5			

(b) 2-length sequences: ‘ab’ and ‘ba’

SID	EID(a)	EID(b)	EID(a)
1	1	2	3
2	1	3	4

(c) 3-length sequences: ‘aba’

Table 2.10: The summary of the typical pattern mining algorithms

	Apriori	**FP-Tree**	**SPADE**
Rule	Association Rule	Association Rule	Sequence Rule
Related paper	(Agrawal and Srikant 1994)	(Han et al. 2004)	(Zaki 2001)
Motivation	-Improvement of AIS, SETM -Finding frequent itemsets using candidate generation -Use level-wise search -All nonempty subsets of a frequent itemset must also be frequent	-It is costly to handle a huge number of candidate sets in Apriori -Avoid explicit candidate generation -To solve multiple scans of transaction database -Use FP-Growth approach and FP –tree construction	-Solve GSP Problems (repeated database scans, use complex hash structures) -Decompose the original problem into smaller sub-problems using equivalence classes on frequent sequences -It can reduce main-memory size using lattice search techniques (DFS,BFS) and using simple join operation.
Scan	(k+1) scans	2 scans	3 scans
Algorithm	BFS(Breadth First order) i)Join step -C_k is generated by joining L_{k-1} with itself ii)Prune step -Any (k-1)-itemset that is not frequent cannot be a subset of a frequent k-itemset. *Generating strong association rules: -Using confidence and support_count	DFS(Depth First order) i)Scan DB once, find frequent 1-itemset-1 scan ii)Sort frequent items in frequency decending order iii)Scan DB again, construct FP-tree- 2 scan vi)For each frequent item, construct its conditional pattern-base, and then its conditional FP-tree vii)Repeat the process on each newly created conditional FP-tree vii)Until the resulting FP-tree is empty, or it contains only one path-single path will generate all the combinations of its sub-paths, each of which is a frequent pattern	BFS+DFS i)Computation of frequent 1-seq and frequent 2-seq(using vertical database format)<SID,EID> -1 scan, 2 scan ii)Enumerating frequent sequences of a class(BFS, DFS) iii)Id-list intersections(Prefix-based equivalence classes) -3 scan vi)Pruning sequences(check subsequence)
Advantages	-Use large itemset property	-Highly condensed but complete for frequent pattern mining -Avoid costly database scans during candidate generation -Usually faster than Apriori	-3 scans over the database -It can solve in-main memory using efficient seach, (Possible for in-memory computation and parallelization) -No overhead

	Apriori	**FP-Tree**	**SPADE**
Dis-advantages	-Huge candidate sets -Multiple scans of database	-In case every transaction has a unique set of items. As none of the transactions have any items in common, the same as the size of the original data.	-A huge set of candidates generated -Inefficient for mining long sequential patterns -FreefixSpan is better
Based-Approaches	*Association Rule -AprioriTID: Using $\widehat{C_k}$ to count the support of candidates in C_k, it reduces time spent on candidate generation that do not need. -Apriori Hybrid: Apriori(1) + AprioriTID(2) *Sequence Rule -AprioriAll: Count All(like-Apriori) Find Maximal Large Sequences -AprioriSome: Count Some Only count sequences of certain lengths -DynamicSome: Count Some Count determined by the variable step -GSP: Apriori minining(with sequence) Using hash-tree, multiple -SPADE	*Pattern-Growth-based Approaches *Sequence Rule -FreeSpan: S collects the support for each item, and finds the set of frequent items -PrefixSpan: It recursively project a sequence database into a set of smaller projected sequence databases and grows sequential patterns in each projected database by exploring only locally frequent fragments. -I-PrefixSpan(improvement of pattern growth-based PrefixSpan): instead of generating intermediate projected databases, we create the index set by registering all the position index of associated customer by means of a (pointer, offset) pair.	

2.3 Failure identification

Through two phases (failure detection and failure isolation), the causes of failure can be extracted. For stable system, suitable feedback also important as well as failure detection. In failure identification phase, information about the size, type, and time occurrence can be generated, and then suitable actions should be applied to eliminate the abnormal condition. The process used for identification may be unstable, where a feedback controller is necessary to stabilize the process (Maree 2009).

2.4 Failure prognosis

Failure prognosis is a task to determine whether a failure is impending and estimate how soon and how likely a failure will occur (Jardine et al. 2006). The result extracted from analysis phase is corresponding to prognosis phase. Using the causes of failures is basis to predict and prevent occurrences of failures. There are three different approaches for failure prognosis.

The first method can use the physical models. These are models which are developed by the experts in the component field, and then large data sets and these models are validated on about the accuracy (Brotherton et al. 2000). This method is similar MBFA (Model-Based Failure Analysis) methods as displayed in Figure 2.20. Based on the predefined mathematical model, residual can be calculated by comparing the input data and the output data which are extracted from the developed model. This residual is used to evaluate a failure state. This Model-Based Prognosis has some problem that is often impossible to collect data from all modes of machine operation and failure condition. In addition, unsuspected failures could not be extracted.

Expert systems can be applied as prognosis systems. PROMISE (Prognostic and Intelligent Monitoring Expert System) generates real-time information upon the existence of failures (Biagetti and Sciubba 2004). They can forecast the failures on the future time and suggest on how to solve the problems. Expert systems can be said as rule-based failure prognosis systems because the failures must be predicted compared with the constructed rules which is made from the experts. It has be successful to apply for failure prognosis however expert systems also have some problem same as failure analysis methods using expert systems. It is difficult to obtain the domain knowledge and the converting process is also difficult (Hao et al. 2009). Furthermore, if the number of rules is increased, it takes much time to compute.

Neural network-based failure prognosis is also broadly used. Dynamic wavelet neural network is applied for bearing failure (Wang and Vachtsevanos 2001) and data-driven neural network is applied for aircraft actuator components (Byington et al. 2004). This method is similar approach expert systems. As described before, similarity approaches are used to predict the occurrences of failures in general. In other words, occurrences of failures can be estimated based on probability basis. In reliability systems, prognosis phase is important procedure as well as analysis phase because appropriate reactions can make the accurate failure analysis.

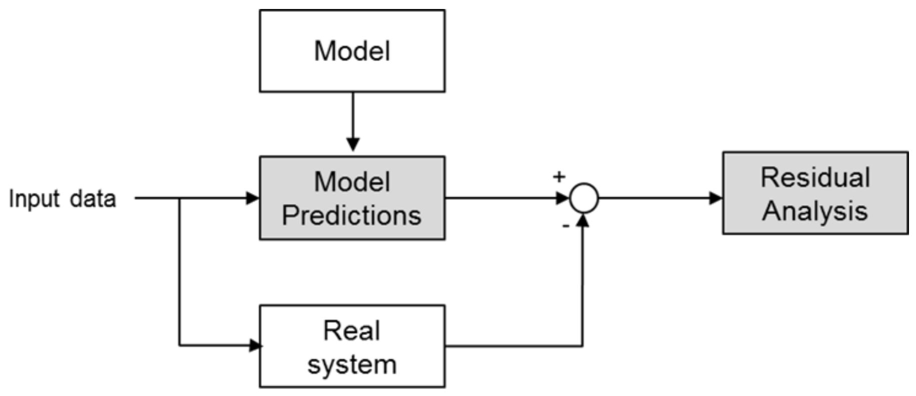


Figure 2.20: Model-based failure prognosis (Brotherton et al. 2000)

III. Event-Driven Failure Analysis

3.1 Problem statement

System techniques and natural domains are changing quickly, and the demand for accurate observations is increasing. Therefore, the importance of product quality has also increased. For these reasons, failure analysis is becoming a more important area and a critical issue in industrial applications(Efendic and Del Re 2005). The importance of analysis can be described in terms of economic areas. System failures may lead to financial losses, because they can cause unrecoverable losses in systems, even though they may be minor. Some examples are described in the following.

- Massive numbers of Toyota vehicles were recalled due to an accelerator pedal error. It damaged Toyota's economic value as shown in its stock prices. the stock price continued to fall after the vehicle recall, and Toyota reported a loss of 1.418 trillion yen, due to the effects of the reduced number of vehicles sold, and product composition, plus a loss of 760 billion yen due to the influence of exchange rate fluctuations (Sakurai).
- For Korail, KTX train accidents have occurred several times. These accidents were caused by defects of the train, rather than by driving mistakes. In fact, these problems may have occurred during the initial stabilization period involving several verification procedures. These problems occurred repeatedly, because the root causes could not be found. From these repeated failures, there was a lot of financial loss because of passengers demanding refunds due to delays, as well as repair costs.

As described in the examples above, accurate failure analysis is required in order to reduce the financial loss and customer's confidence problems. Though a failure analysis has to offer the precise causes, developed systems only offer whether the failures occur. In the case of a marine system, a mounted failure analysis does not provide the causes, and vehicle systems also provide only a warning of failures using the OBD (On-Board Diagnostics) system. For this reason, drivers cannot receive the failure causes. Therefore, the root causes should need to be provided to prevent the repeated failures.

To develop a failure analysis, many methods are contained in each procedure (failure detection – isolation – identification) as described in Chapter 2. All of the methods can be divided into MBFA (Model-Based Failure Analysis) and DDFA (Data-Driven Failure Analysis) depending on whether the system model can be represented or not. MBFA is mostly used because it is accurate. However, MBFA methods have some problems. One problem is it requires an accurate mathematical model. To get a mathematical model, input and output of the systems are required, thus failures can be analyzed after system operation is finished. For this reason, MBFA methods cannot interpret the interaction, which is an important factor of analysis. The other problem is that knowledge expertise is needed to accurately construct an analysis.

Because of those problems of MBFA, the DDFA method is suggested in this paper. DDFA is a flexible method because it does not need to construct a mathematical model. Therefore, it can be considered more efficient than MBFA when the system is complicated and a mathematical model cannot be constructed. DDFA can also be used to detect the failures that could not be extracted using MBFA. For example, sometimes failures occur from environmental problems; however, these factors cannot be considered in MBFA. In the case of DDFA, causes can be extracted that are affecting the failures from the stored historical data. The causes of failures can be extracted by analyzing historical data, and then proper action will be applied to prevent the failures.

Generally, DDFA uses the massive data collected from each sensor. For example, vehicle systems get data from the OBD system during driving, and other systems (marine, train, etc.) also contain the system to store data. Collected sensor data have some problems for accurate analysis, and these problems are considered from two aspects which are “sensor” and “storing”. In terms of sensor, all of the systems are composed of many types of sensors such as temperature, position and pressure sensors. These different types of sensors do not have common properties. For example, units and ranges to maintain normal state are different. Therefore, standard properties are required for accurate analysis. The other aspect is storing, which occurs when data are transferred from sensor to data acquisition device. The surrounding environment affects the analysis when data are measured by using the data acquisition system. For this reason, data can contain noise, empty values or missing values which are obstacles to analysis. In addition, historical data are too large, and are difficult to handle all at once. For accurate analysis, these data should be converted into comprehensive form. As a result, data filtering methods are applied to extract significant information from raw data while containing the information of the original data.

In the data filtering phase, statistical analysis and other data mining techniques are applied. In this paper, discretization techniques are applied in the filtering phase as well as statistical analysis and other data mining techniques. Many discretization techniques are described in Chapter 2. Some methods need much time to compute, however, it is faster than other methods. On the other hand, the other methods are faster, but the accuracy is decreased. In the failure analysis, both speed and accuracy are important factors. Therefore, the suitable discretization technique should be selected to efficiently reduce the data. In addition, suitable parameters are also estimated with selecting the discretization techniques. This selected discretization technique can help to increase the efficiency of the failure analysis.

If data is reduced, the information about the failures should be provided. Developed failure analysis gives notice about the failures without knowing causes of the failures in general. Using the filtered data, failure analysis should provide not only the notice of a failure, but also the causes of failure to resolve the abnormal phenomenon. These extracted results should contain the information about the interaction of sensor data. This means which interaction of sensor data affected the failure. Once this information is extracted, abnormal states should be predicted in real-time. From that type of failure analysis and prognosis system, failures can be diagnosed and prevented quickly.

3.2 The framework of event-driven failure analysis

In this thesis, event-driven failure analysis is developed. Event means the converted significant data from raw data. Broadly, failure diagnostics in this thesis consist of four steps (data cleaning – data reduction – failure pattern analysis – prognosis) as displayed in Figure 3.1. As mentioned before, raw data contain noise, redundancy values and missing values. Data cleaning methods are applied to resolve this problem. Missing values can be predicted and raw data can be transformed using the Kalman-filter and other signal processing methods. After the cleaning phase, data reduction techniques are applied. Statistical analyses are used to extract features from raw data. PCA (Principal Component Analysis) and Correlation analysis are used for selecting the significant sensors and analyzing relation of each sensor. Using these statistical analyses, the information about the sensor data can be known.

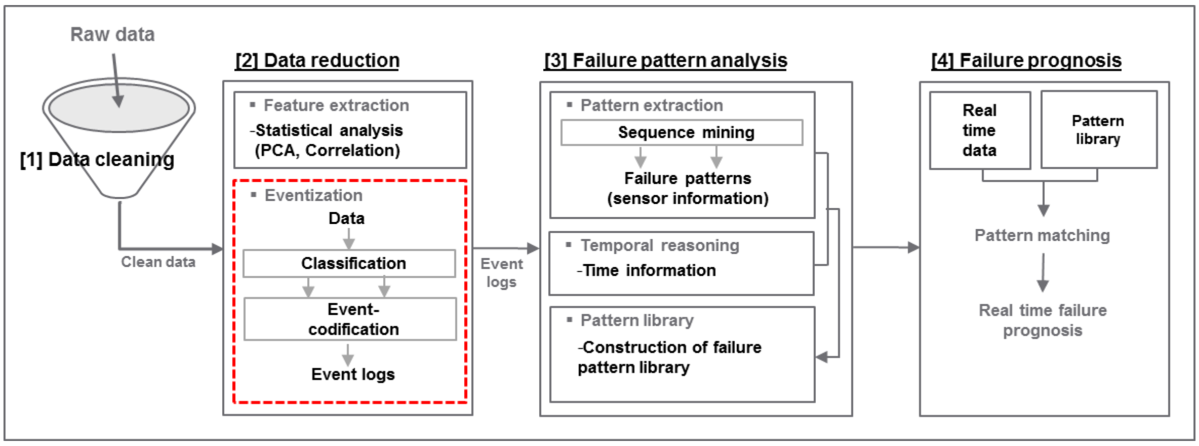


Figure 3.1: Event-driven failure analysis architecture

Eventization methods are applied which is the focus of this thesis. As mentioned before, raw data is difficult to handle directly because these collected data have many problems. In the cleaning phase, noise and missing values were processed; however, the dimension of the data could not be reduced. The eventization phase will be able to reduce the dimension of raw data while containing the information of raw data. Raw data can be represented as the event log, which is indicating the extracted significant data from raw data. The phase for extracting event logs is named "Eventization" and discretization techniques that can covert to the discrete data from the continuous data are used in this phase. Four different discretization techniques are applied; Equal-width binning, Entropy, Knowledge expert-based discretization, Estimation of probability distribution-based discretization. Along with these different discretization techniques, different results can be extracted. Therefore, performance of these results should be compared, and then the most suitable discretization technique has to be selected. Furthermore, parameters are needed for the data reduction phase. These parameters are carefully applied to the result. Proper parameters should be estimated for accurate analysis.

Once event logs are extracted from the eventization phase, the causes of failures would be extracted. Pattern mining methods can be applied to the filtered event log. Of all pattern mining algorithms, FP-Tree algorithm is selected to find certain patterns that are causes of failures because FP-Tree can quickly handle a large amount of data. The most frequent patterns which are extracted by applying this FP-Tree algorithm are the results and the causes of failure. The time information of the extracted event logs are also important factors. Therefore, the time intervals between event logs are considered using temporal reasoning techniques. After extracting cause patterns of failures, these results are stored in the constructed library, and then these stored data are used for the prognosis phase. Prognosis is possible through pattern matching between the input data and stored patterns. By calculating the probability, a warning can be issued before the occurrence of a failure, and then the failure can be predicted. All of these procedures will be presented in the following chapters, and terminal data collected from marine diesel engines are used for case study.

IV. Failure Analysis Procedure

4.1 Data cleaning

Data quality is the most crucial factor for data analysis. As mentioned before, real data contain noise, redundant values, and missing values, and these have a strong influence on data analysis and increase the need for data cleaning. Data cleaning, also called data cleansing or scrubbing, deals with detecting and removing errors and inconsistencies from data in order to improve the quality of data (Rahm and Do 2000). There are many issues in data cleaning as described in Chapter 2, however, the Kalman filter method is suggested in this paper. This is because the Kalman filter is a set of mathematical equations that provides an efficient computational means to estimate the state of a process, in a way that minimizes the mean of the squared error (Welch and Bishop 1995). A Kalman filter is a probability-based digital filter that is capable of filtering noise, and this method consists of two phases as below (Zhuangt et al. 2007a).

- Predict: estimate of the current state from the previous timestamp
- Update: a new measurement of the current timestamp is used to refine this prediction

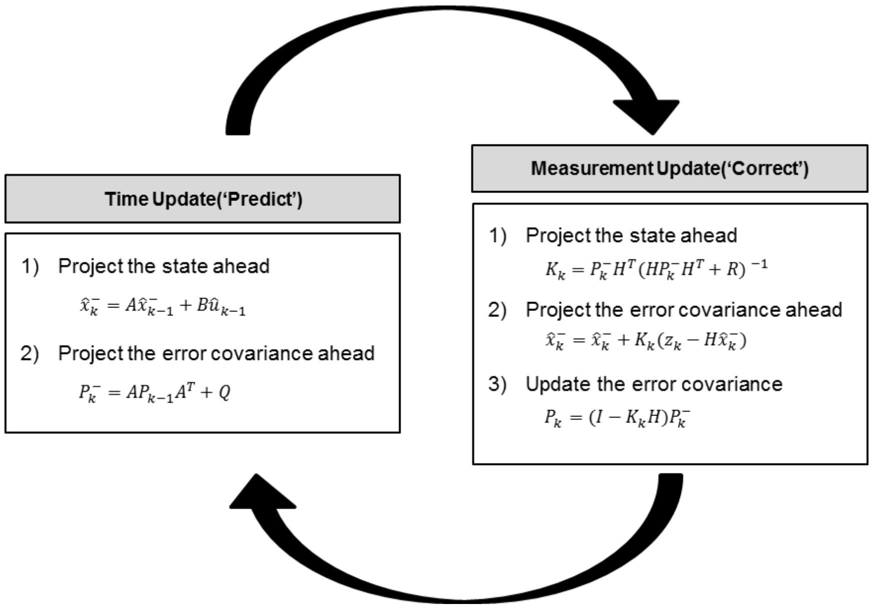

Time Update('Predict')
1) Project the state ahead
$\hat{x}_k^- = A\hat{x}_{k-1}^- + B\hat{u}_{k-1}$
2) Project the error covariance ahead
$P_k^- = AP_{k-1}A^T + Q$
Measurement Update('Correct')
1) Project the state ahead
$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1}$
2) Project the error covariance ahead
$\hat{x}_k^- = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$
3) Update the error covariance
$P_k = (I - K_k H)P_k^-$

Figure 4.1: Kalman filter diagram (Welch and Bishop 1995)

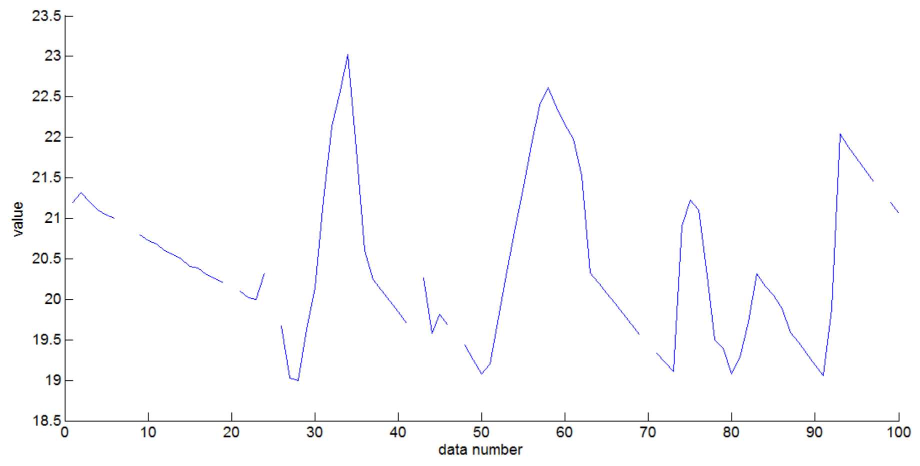
This method is limited to a linear assumption and its linear stochastic difference equations are described below. Based on these equations, a prediction can be made of the next value based on a few of the most recent estimates, and then the next actual data point is read, and a compromise value between the predicted and actual value is calculated based on the noise. From these two phases, the Kalman filter maintains an estimate of the accuracy of the prediction, which is based on its historical accuracy, and is reported as the standard deviation of the prediction (Vijayakumar and Plale 2007b). These two phases are described in Figure 4.1.

$$x_k = Ax_{k-1} + Bu_k + w_{k-1}$$
$$z_k = H_k x_{k-1} + v_k$$

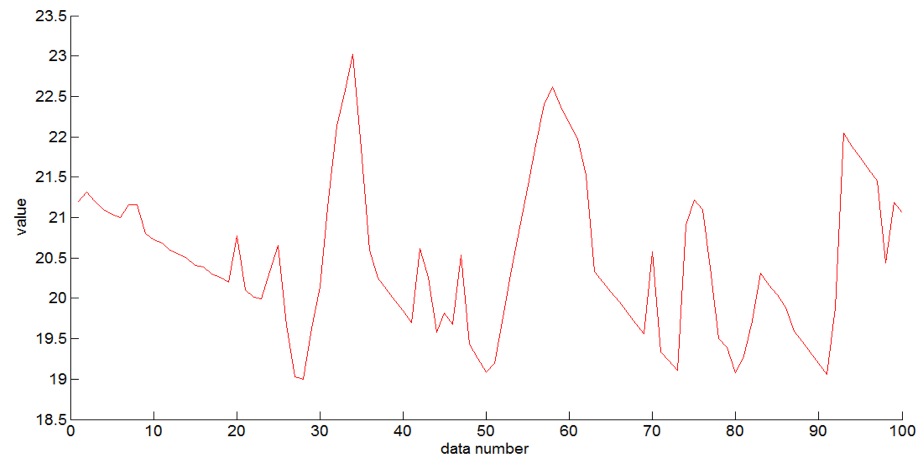
Where:

- x_k= estimate the state
- z_k= a measurement in certain time (k)
- A = state transition model
- B = control-input model applied to the control vector u_k
- w_k= the process noise, zero mean with covariance Q_k
- H_k= the observation model
- v_k= the observation noise with zero mean and covariance R_k

In this method, state estimation model (A) and control-input model (B) are determined by using CATS benchmark competition (Lendasse et al. 2007) which can be used to determine the predicted values. In this paper, three different results could be extracted using the Kalman filter. At first, only missing values are treated and then filled in with the predicted values. For example, Figure 4.2(a) is original data; however, there are some missing values. Using only time update phase, missed values are processed because it is just filling in the missing value and remaining value do not need to change. Therefore, the result is displayed in Figure 4.3 (b). This original data can be also transformed as displayed in Figure 4.3. This result is predicted by using previous values so data flow is similar after it is transformed. The last result is displayed in Figure 4.4. If the previous data were available in Figure 4.3, this result is used for all the data. For smoothing, the Rauch-Tung-Striebel two pass smoother method is applied (Grewal and Andrews 2008). As displayed in each figure, smoothed values cannot be extracted from certain features compared with Figure 4.3. From the Kalman filter, raw data can be transformed into filtered data that are able to analyze the data accurately; however, specific features can be removed that may have a bad influence.



(a) Original data (Kalman filter)



(b) Result after processing missing values

Figure 4.2: Process missing values using Kalman filter

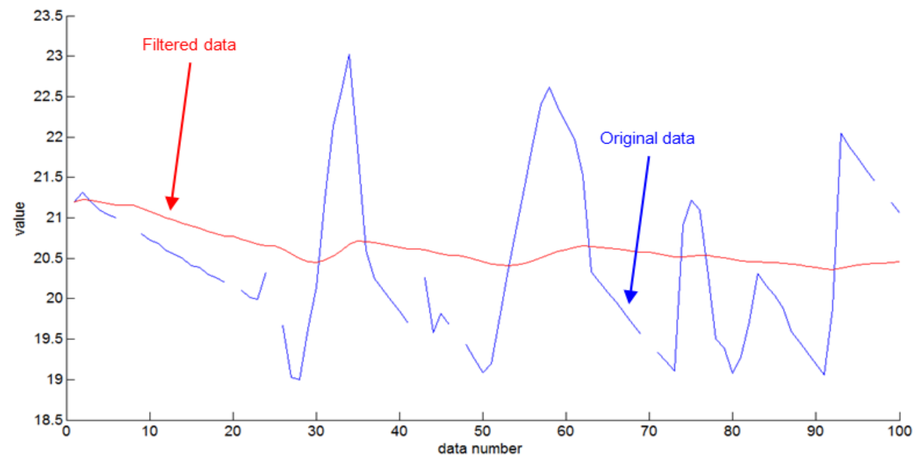


Figure 4.3: Data transformation using Kalman filter

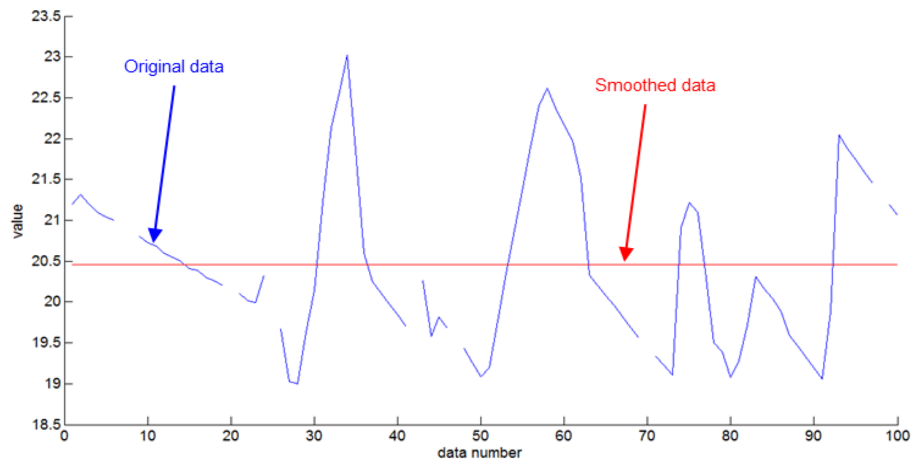


Figure 4.4: Smoothed data using Kalman filter

4.2 Data reduction

In the data cleaning phase, noise or the other problems for data quality are removed. After the data cleaning phase, data reduction methods are needed to analyze the data efficiently. As mentioned before, collected data are too large because system failures are usually caused by long-term mechanical problems. For this reason, data are collected from several weeks to several years, and the amount is very large. This paper applies suitable data reduction techniques to data reduction and feature extraction, and then converts the reduced data log into a new format of event sequence information. Statistical data reduction, discretization and eventization techniques are described.

4.2.1 Feature extraction from raw data

PCA (Principal Component Analysis) and correlation analysis are applied to extract features from raw data. From these statistical techniques, information about the sensors is extracted that is useful for failure analysis.

PCA (Principal Component Analysis)

Data from different types of sensors are collected from the system in a range of different unit scales. If these data are used directly, it would take much time to compute the all. Therefore, important information can be extracted from raw data and then extracted data are used for failure analysis. PCA applies in order to extract important information from raw data in this thesis.

PCA is appropriate when measures have been obtained on a number of observed variables, and the goal is to develop a smaller number of principal components that will account for most of the variance in the observed variables (Hatcher 1994). Principal component is defined as a linear combination of variables, and it can be determined by the eigenvectors of the covariance matrix corresponding to the largest eigenvalues. At first, original data are normalized because they have different types of unit scales. After normalizing data, the covariance matrix is calculated and then original data can be converted to a representative data set of original data using the following equation (Shlens 2005).

$$Y = PX$$

Where:

- X: original data set
- Y: a representation of data set
- P: a set of new basis vectors for expressing the X

The result which is a representation of original data set (Y), is a projection on the basis of principal components (PC) which are displayed as perpendicular lines in Figure 4.5. Calculated PC are arranged along with eigenvalues in descending order. The first PC is the most describable component about the relationships between all of the variables consisting of raw data. The second PC can explain well the relationships between each variable secondarily. Along with the selected number of PC, original data can be reduced as the number of PC dimensional data. Significant variables are also revealed using PCA as in Figure 4.6. Collected data in systems consist of many types of sensors (variables). All of the sensors do not need to be analyzed for efficiency. If several significant sensors are only used for failure analysis, it is better in terms of efficiency. Several significant sensors can be extracted using PCA. In the PCA result, all of variables are drawn along the directions most corresponding to the principal component. For example, variable2 is corresponding to PC2 better than PC1, and variable3 is corresponding to two PCs. Therefore, efficiency of failure analysis would be increased by selecting several significant variables.

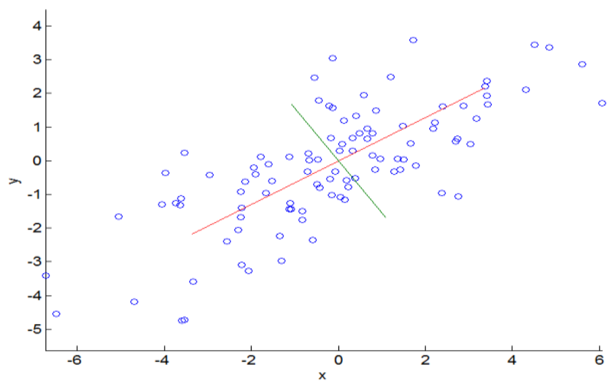


Figure 4.5: Principal Component Analysis

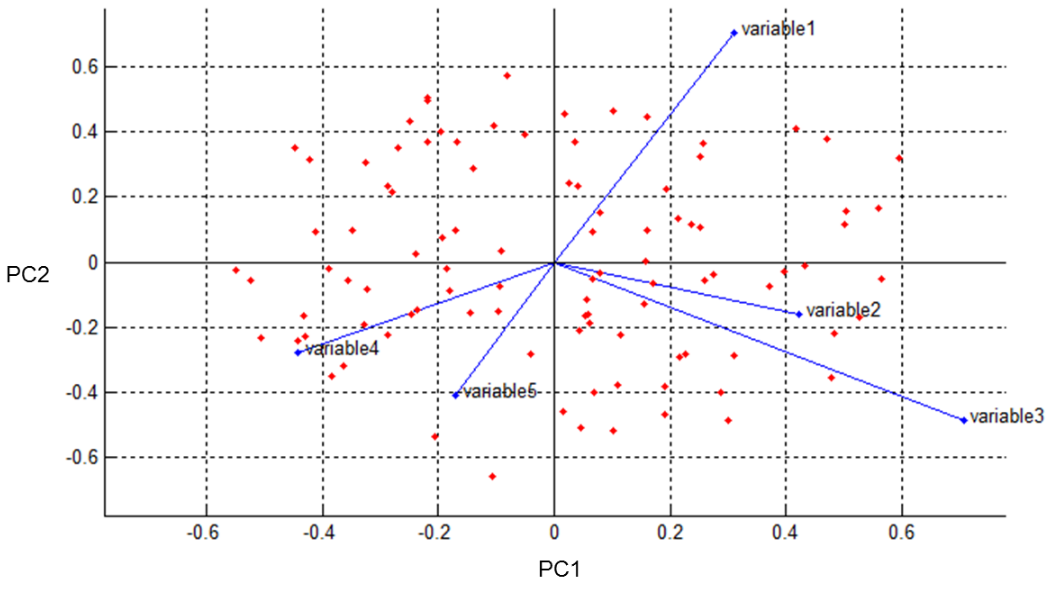


Figure 4.6: Variables information from PCA

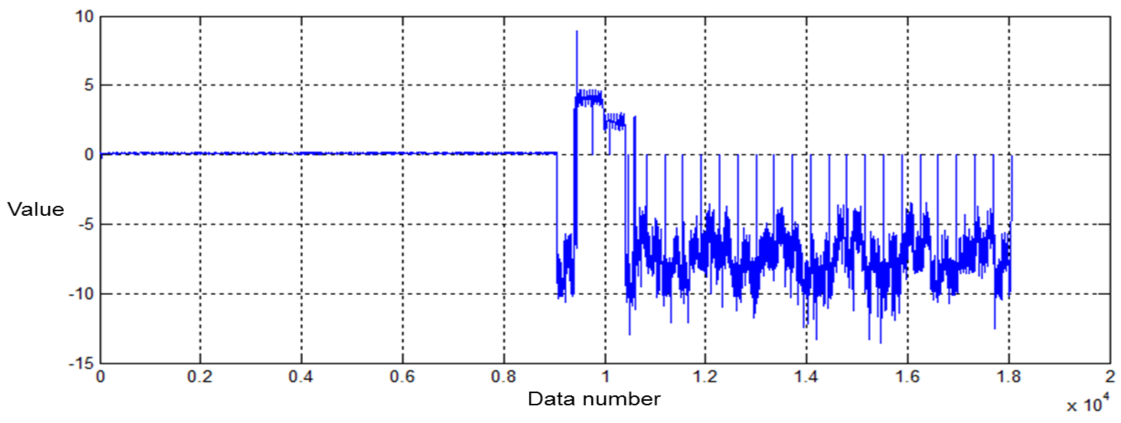
Correlation analysis

In PCA, significant sensors can be determined; however, information about relationships between each sensor can be known using correlation analysis. In general, most systems have many types of sensors. For example, a vehicle contains more than 40 sensors, such as TPS (Throttle position sensor), AFS (Air-Flow sensor), and Knock sensor, etc. If other sensors related to a certain sensor to be considered are determined before analyzing the failure, not as much time is needed to compute. Therefore, correlation analysis is used to identify the relationships between all of the sensors.

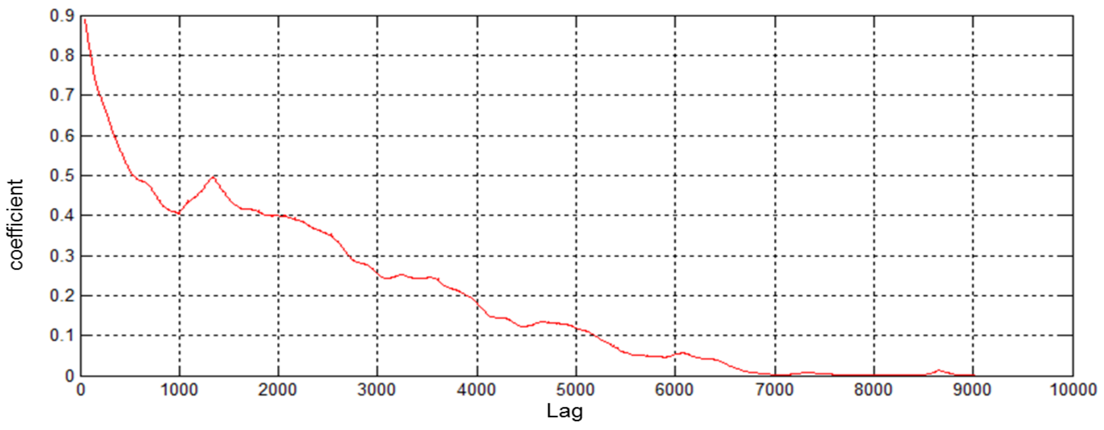
The correlation analysis is a tool used to evaluate the similarity of two sets of measurements and it indicates how much information is shared by two variables, or in other words, how much two variables have in common. Two types of correlation analysis are considered, which are auto-correlation and cross-correlation. Auto-correlation measures the dependence of the values of the sensors at one time on the values at another time, whereas cross-correlation measures the interdependence between two different sensors. The measure of correlation is called the correlation coefficient, which is also referred to as Pearson's coefficient of correlation. It was first introduced by Galton (Galton 1886), and Pearson's correlation coefficient can be calculated using following equation when x is a certain sensor and y is the other sensor to compute the correlation coefficient.

$$r\ (correlation\ coefficient) = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2 \sum_{i=1}^{n}(y_i - \bar{y})^2}}$$

From this equation, the interdependence between each sensor can be measured. Examples are displayed in Figure 4.7 and 4.8. At first, the correlation coefficient is calculated as displayed in Figure 4.7(a) when original data is as Figure 4.7 (b) in auto-correlation analysis. The peak point is the correlation coefficient, so this original data (Figure 4.7 (a)) has high periodicity. Secondly, an example of cross-correlation is displayed in Figure 4.8. The correlation coefficient is generated as Figure 4.8 (c) when two different sensors (Figure 4.8 (a) and Figure 4.8 (b)) are considered. The peak correlation coefficient is very low so it means that there is no relationship between these two different sensors. Correlation analysis helps to interpret relationships between various sensors; therefore it is possible to understand the collected data more easily. After applying these statistical methods such as PCA and correlation analysis, system failures can be diagnosed more accurately and efficiently.

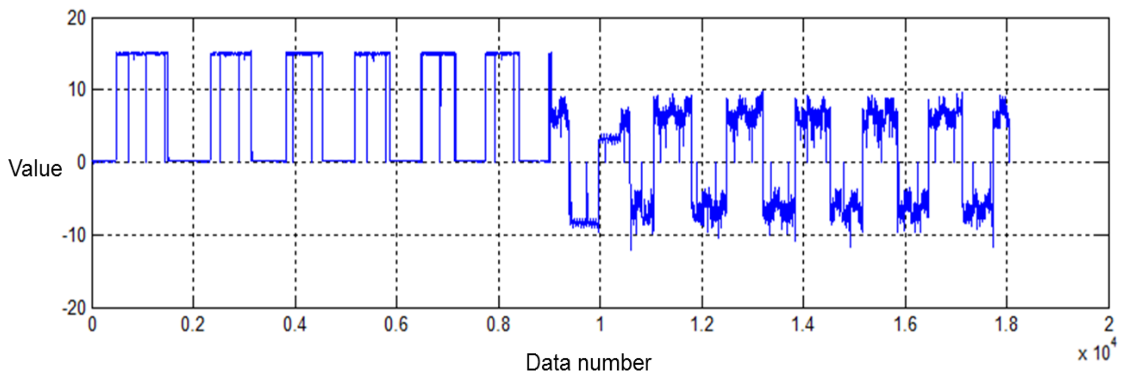


(a) Original data

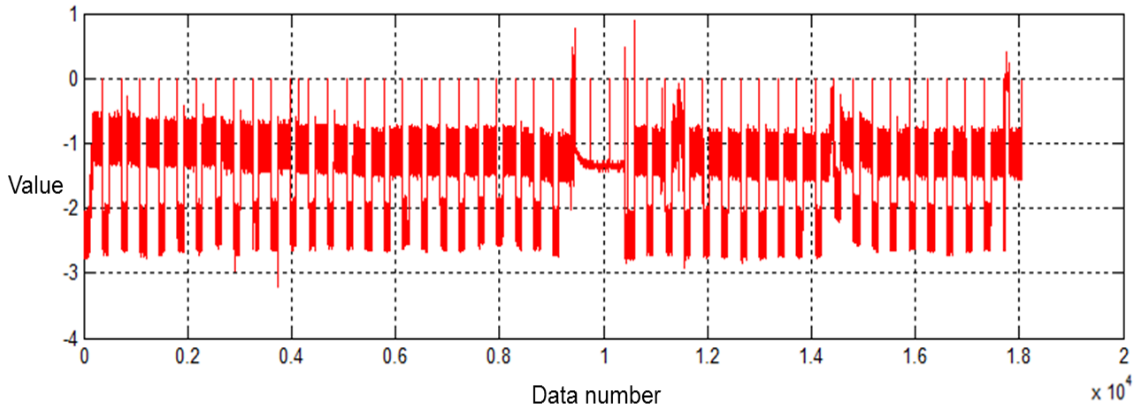


(b) Result of Auto-correlation analysis

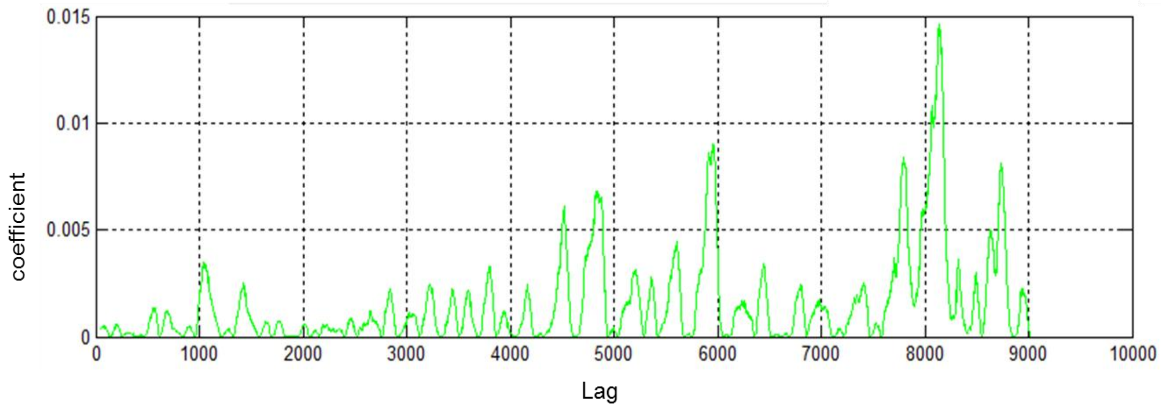
Figure 4.7: Example of Auto-correlation



(a) Original data (sensor1)



(b) Original data (sensor2)



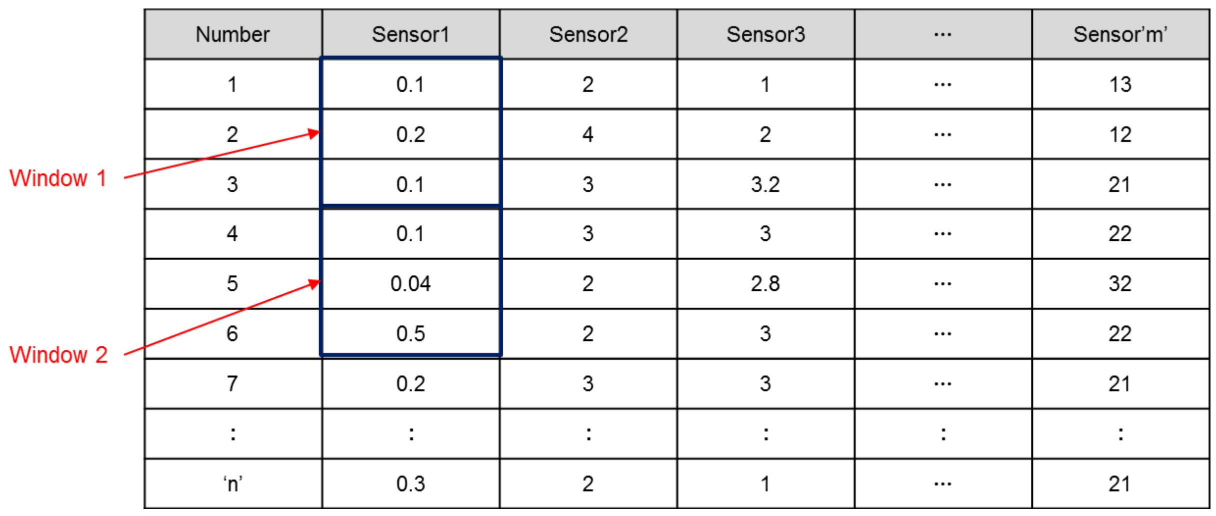
(c) Result of Cross-correlation

Figure 4.8: Example of Cross-correlation analysis

4.2.2 Eventization

Statistical information can be extracted by applying PCA and correlation analysis, and then the eventization phase will be described in this chapter. Eventization means the procedure which can extract significant information from raw data. For this eventization phase, proper discretization methods are applied to convert raw data into a reduced data format while retaining significant information. Equal-width binning, Entropy, Knowledge expert-based discretization, and Estimation of probability distribution-based discretization techniques are applied, and these techniques are compared.

The eventization phase has two steps which are “Classification” and “Event-codification”. Raw data that are collected directly from sensors convert to certain event logs in the classification phase. In this paper, an event log means significant information that is extracted from raw data. Before the classification phase, raw data have to be represented as the other smoothed value because these raw data are too large. To represent the other smoothed value, raw data should be divided as the same size. In this thesis, window size indicates this size to divide the raw data. As displayed in Figure 4.9, raw data are grouped along the certain window size, and then these data are representative as mean value. This computation is performed for each sensor individually, and then the size of raw data is reduced form of original data. Classification and Event-codification phases are processed based on these representative data.


Window 1
Window 2

Number	Sensor1	Sensor2	Sensor3	···	Sensor'm'
1	0.1	2	1	···	13
2	0.2	4	2	···	12
3	0.1	3	3.2	···	21
4	0.1	3	3	···	22
5	0.04	2	2.8	···	32
6	0.5	2	3	···	22
7	0.2	3	3	···	21
:	:	:	:	:	:
‘n’	0.3	2	1	···	21

Figure 4.9: Data grouping along with the specified window size

Table 4.1: Data eventization procedures ((a)→(b): classification, (b)→(c): event-codification)

Number	Sensor1	Sensor2	Sensor3	…	Sensor'm'
1	0.1	2	1	…	13
2	0.2	4	2	…	12
3	0.1	3	3.2	…	21
:	:	:	:	:	:
'n'	0.3	2	1	…	21

(a) Original sensor data

Window	Sensor1	Sensor2	Sensor3	…	Sensor'm'
w1	e11	e21	e32	…	em1
w2	e12	e21	e32	…	em2
:	:	:	:	:	:
'wn'	e11	e22	e31	…	em2
			Failure 1		
w1	e12	e22	e32	…	em1
w2	e11	e22	e32	…	em1
:	:	:	:	:	:
'wn'	e12	e21	e31	…	em2
			Failure 2		
:	:	:	:	:	:

(b) Classified data

Number	w1	w2	w3	…	w'k'
Failure1	A011	A003	A111	…	A182
Failure2	A212	A001	A122	…	
Failure3	A211	A122	A112	…	A121
:	:	:	:	:	:
Failure'j'	A212	A121	A187	…	

(c) Eventized data

An example of the data eventization phase is displayed in Table 4.1. In classification, data are compared with standard values that are determined from each discretization technique. By comparing with standard values, certain event logs are contributed. As displayed in Table 4.1, original sensor data (a) can be converted into event logs (b). For this classification phase, discretization techniques are applied and will be described in detail. Once data are represented as certain event logs, these are converted into new event logs, again along the certain failures as displayed (c) in the event-codification phase. These eventized data are the result of the eventization phase, and then failure pattern will be analyzed based on these eventized data.

If these converted data are used to pattern the extraction phase, the computation time would not take much time and space to store the data can be also reduced. The focus of this chapter is the discretization techniques that are used for converting to event logs in the classification phase. A standard is needed to determine the ranges of various events, and this standard can be represented as various ranges. Therefore, certain techniques are needed and various results of pattern analysis are extracted from these different discretization techniques.

Eventization based on the Equal-width binning discretization

As described in Chapter 2, equal-width binning divides the data of different lengths and contains these different numbers of data in each bin. By using this criterion, the size of original data can be reduced and converted as certain representative values. This technique can be used to define the range of events. In failure analysis, normal data and abnormal data are collected and these data are used. However, in order to increase efficiency normal data do not need to be analyzed in the data analysis phase. Actually, normal data do not have any features to analyze as failures, and abnormal data are enough to extract the causes of failures. Therefore, the values that are containing in the normal state are not distributed specified events and only events of abnormal data are used in equal-width binning eventization.

An example of equal-width binning-based eventization is displayed in Figure 4.10. This example is the same as in Figure 2.11, and each bin is a specified certain event log. If original attribute values are [0 4 12 16 16 18 24 26 28], these data can be divided into three groups as displayed in Figure 4.10. “Bin 1” and “Bin 3” are specified event logs, and “Bin 2” is regarded as normal state. If “26” is contained in original data, this data would be represented as event log 2 along with already specified event logs. In addition, if “16” is contained, this value would be represented as zero value because this value is normal state value.

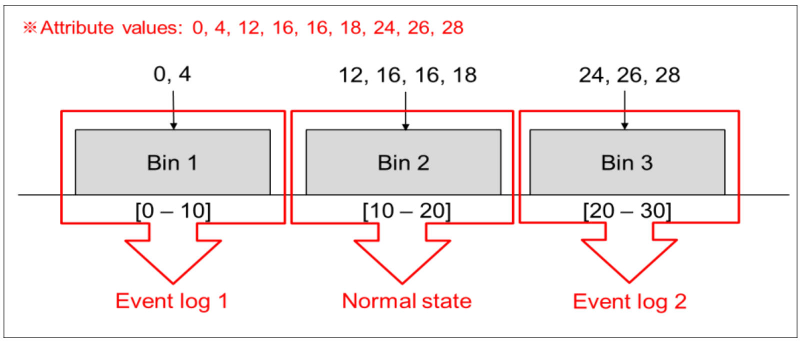


Figure 4.10: Example of equal-width binning-based eventization

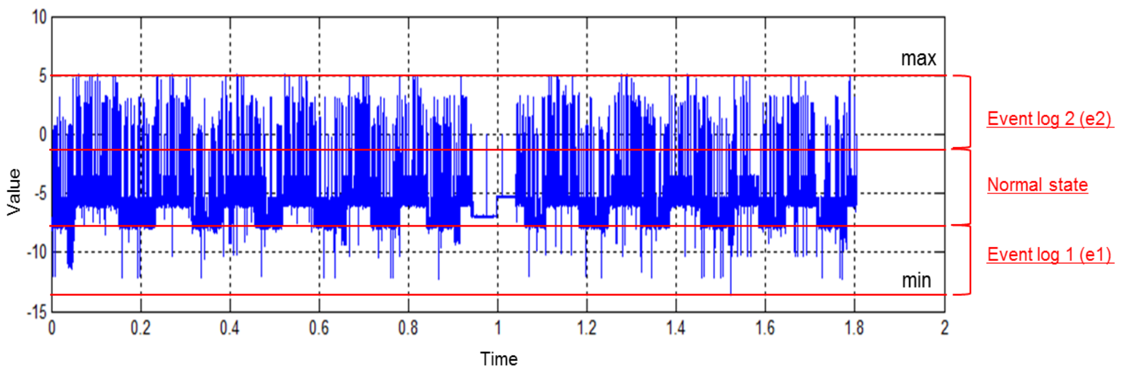


Figure 4.11: Event logs from equal-width binning-based eventization

If this technique is applied in real sensor data, three different types of event logs can be determined from minimum and maximum values of each sensor. Each sensor has different minimum and maximum values; therefore they have different ranges to determine the events. As displayed in Figure 4.11, if the number of events is three, three different event logs are distributed to each range. For example, if a certain sensor value is contained in a certain range, this value would be converted to certain event logs. As a result, this continuous data can be converted as discrete values. By applying this discretization technique, different ranges of values can be changed to only three different event logs.

Eventization based on the Entropy discretization

In this analysis, objective attribute is indicating whether failure occurs or not. Entropy-based discretization needs this objective attribute because this technique is supervised discretization. As mentioned in Chapter 2, supervised discretization is more accurate than unsupervised discretization (Liu et al. 2002). Due to this objective attribute, this supervised discretization is not suitable in failure analysis. Failure cannot be predicted before analysis in failure analysis, therefore failure information should be assumed. In this paper, Entropy discretization is applied by assuming the failure.

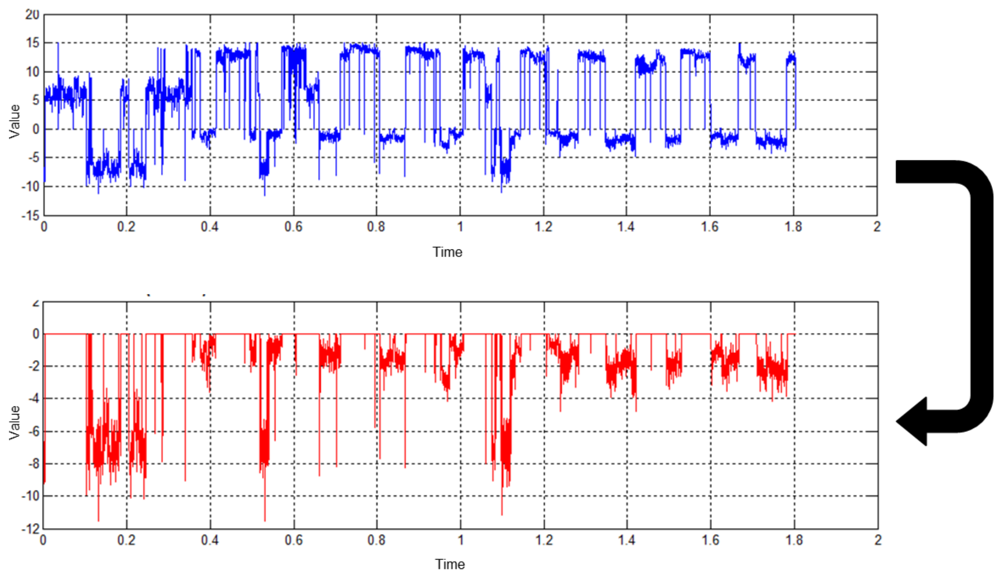


Figure 4.12: Event logs from entropy-based eventization

If Entropy discretization is applied, continuous data would be divided into small intervals. For example, continuous data can be converted to discrete values including very small intervals as displayed in Figure 4.12. After converting discrete data, event logs are distributed to all intervals which are defined as different values. As a result, many event logs are extracted in Entropy discretization as compared with binning discretization. Actually, run time of entropy discretization is very long because entropy values of all intervals are calculated. Some papers suggest a stop point to solve these problems (Bay 2001; Elomaa and Rousu 2004). To define the stop point accurately, training data are needed. This training data are not suitable for our failure analysis because there are some cases that cannot get training data. Therefore, entropy is calculated for all intervals in this thesis. For this reason, small patterns are extracted because too many event logs are created. The other problem is that operation cannot be completed when sensor data is too large. On the other hand, small numbers of event logs are extracted when the stop point is low. Therefore, a suitable stop point is needed for accurate analysis.

Eventization based on the knowledge expert

In addition to developed discretization techniques, system experts can determine the ranges of event logs. A definite range of intervals is provided in developed discretization techniques; however, an expert can determines this range of event logs in this technique. Knowledge experts can define the range of normal state from their knowledge of sensor data. The standard normal state of sensor data is different from certain failure situations. For this reason, non-experts cannot define the ranges accurately; however, knowledge experts should be able to define the more accurate percentage of range. Along with this defined range, different types of event logs are determined.

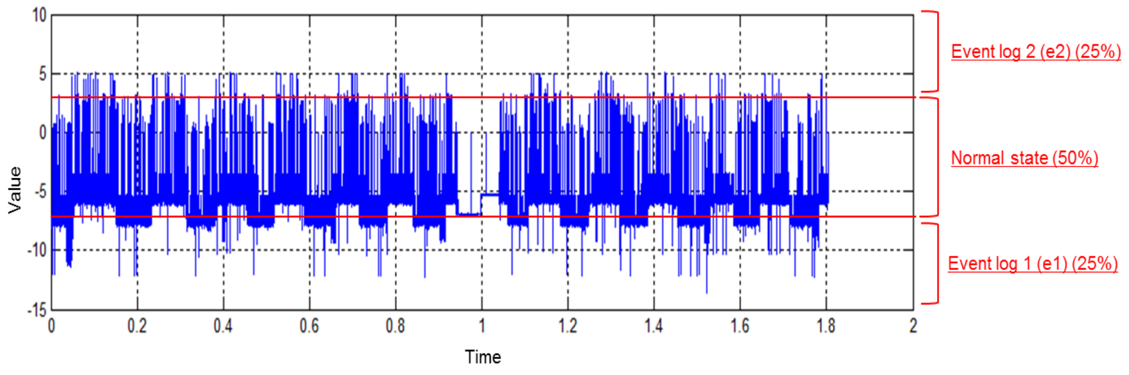


Figure 4.13: Event logs from the knowledge expert-based eventzation

In real sensor data, a parameter is needed to determine the range of normal state, and this parameter is displayed as normal state in this paper. Along with this range of normal state, different ranges of event logs are distributed, and this normal state is given as a percentage. All sensors have a different range of value and normal state because sensors have different features in a system. The normal state of each sensor which was already defined is based on dividing event logs. Figure 4.13 displays a real sensor signal and distributed event logs. If the percentage of normal state is defined as 50%, it means that 50% of all values whose range is the difference between maximum normal value and minimum normal value are regarded as normal state. Other ranges are distributed event logs except the values which are contained in the range of normal state.

For these procedures, three different event logs are defined. As described, representative value which is calculated means by grouping is compared to each range of event log, and then proper event log is distributed. If grouped mean value is bigger than the range of normal state, event log number is contributed as “1”. Otherwise, if grouped mean value is lower than the range of normal state, event log number is contributed as “2”. Finally, if grouped mean value is containing in the range of normal state, event log number is not contributed, and it is defined as empty event log. As a result, all sensor data are converted as “e1”, “e2”, or “0” in the classification phase, and then these event logs are converted to the other event logs in the event-codification phase. The most sensitive effect of this knowledge expert-based discretization is to determine the range of normal state. If the range is large, a very small number of event logs are extracted. On the other hand, if the range is small, too many event logs are extracted. Therefore, experts have to provide a suitable range of normal state.

Eventization based on the estimation of probability distribution

Each sensor has different features which are different normal state, maximum value, minimum value, and average. Mentioned eventization techniques of Equal-width binning, Entropy, and Knowledge-expert-based eventization cannot consider these different features, and then these techniques just divide the ranges regardless of sensor features. However, sensor features should be considered for accuracy. In Estimation of probability distribution-based eventization, sensor features are considered by applying different distribution of each sensor, and then ranges of event logs can be divided based on this probability distribution of sensor.

When applying Estimation of probability distribution, Equal-frequency binning is applied for the eventization phase. Equal-frequency binning eventization is similar as to equal-width binning eventization; however, equal-frequency binning divides into the same size of bins. In Figure 4.14, the same original data are given [0 4 12 16 16 18 24 26 28] as in Figure 4.10, and each bin contains the same number of data. Therefore, “Bin 1” and “Bin 3” are distributed event logs, and “Bin 2” is regarded as normal state. To apply this technique, the same percentage of ranges is specified with Estimation of probability distribution. Actually, it is difficult to divide the same number of values when there are redundant values in real sensor data. If the number of redundant data is higher than the number of bins, it would be difficult to determine the cut point. Therefore, same percentage of range is specified to divide the same number of data.

Figure 4.15 displays an example of determining the most suitable probability distribution estimation of certain sensor. Suitable probability distribution is determined at each sensor before discretization phase, and then ranges of event logs are divided along with determined probability distribution. To determine the most suitable probability distribution of sensor, “goodness of fit” criterion is used. Goodness of fit is defined along with four different standards, which are NLL (Negative of the Log Likelihood), BIC (Bayesian Information Criterion), AIC (Akaike Information Criterion) and AICc (AIC with a correction of for finite sample sizes). For example, t-location-scale is selected as the most suitable probability distribution in Figure 4.15. If the most suitable probability distribution is selected at each sensor, Equal-frequency binning technique would be applied. Figure 4.16 is the same data from equal-frequency binning, and the percentage of normal state range is given 33%. By applying the 33% parameter value, data can be divided as the same number of data. Along with these event logs, this technique is also divided into three different event logs which are “e1”, “e2”, and “0”. The result of applying this discretization technique is different from the result of applying equal-width binning definitely because the range of event logs is different from each sensor. As a result, this probability distribution estimation can be said to be the most accurate technique of the four different eventization techniques.

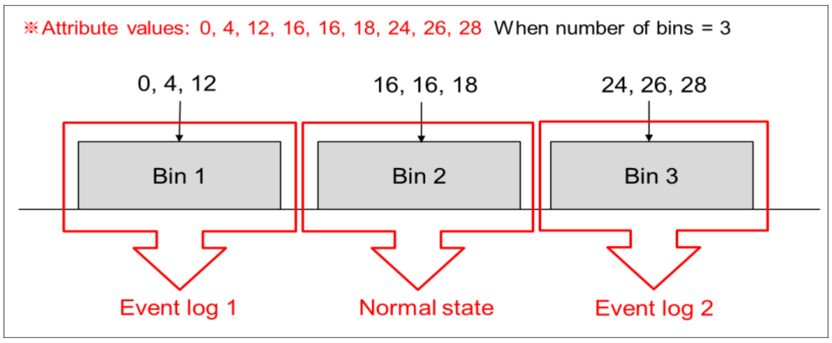


Figure 4.14: Example of equal-frequency binning discretization

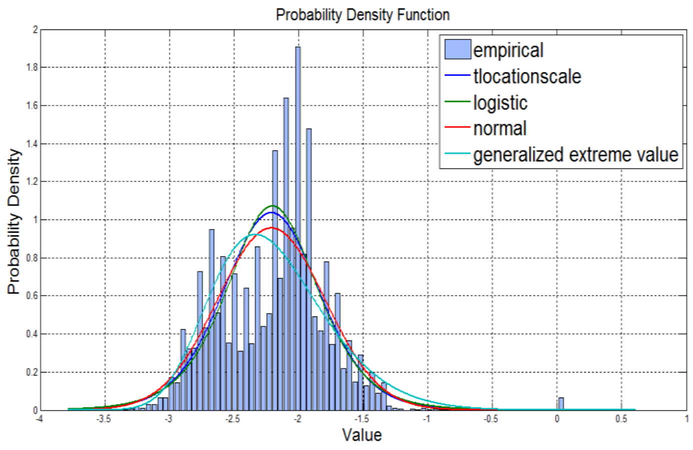


Figure 4.15: Probability distribution estimation

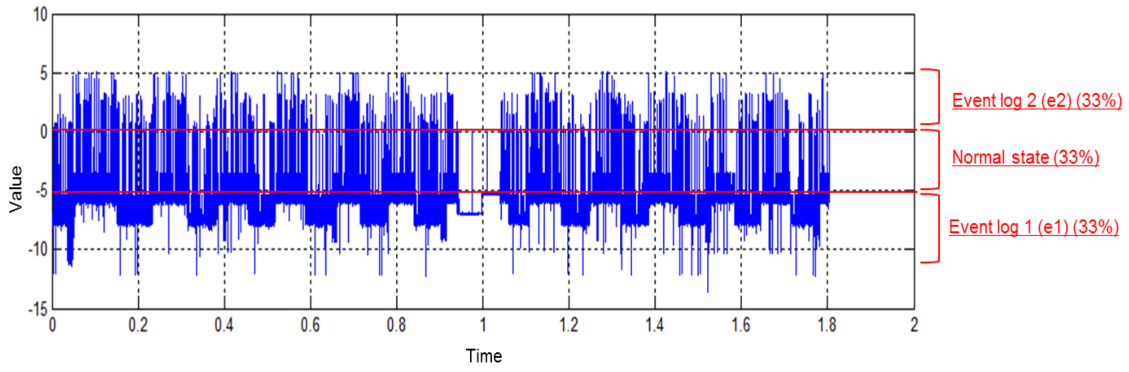


Figure 4.16: Event logs from the estimation of probability distribution-based eventization

4.3 Failure pattern analysis

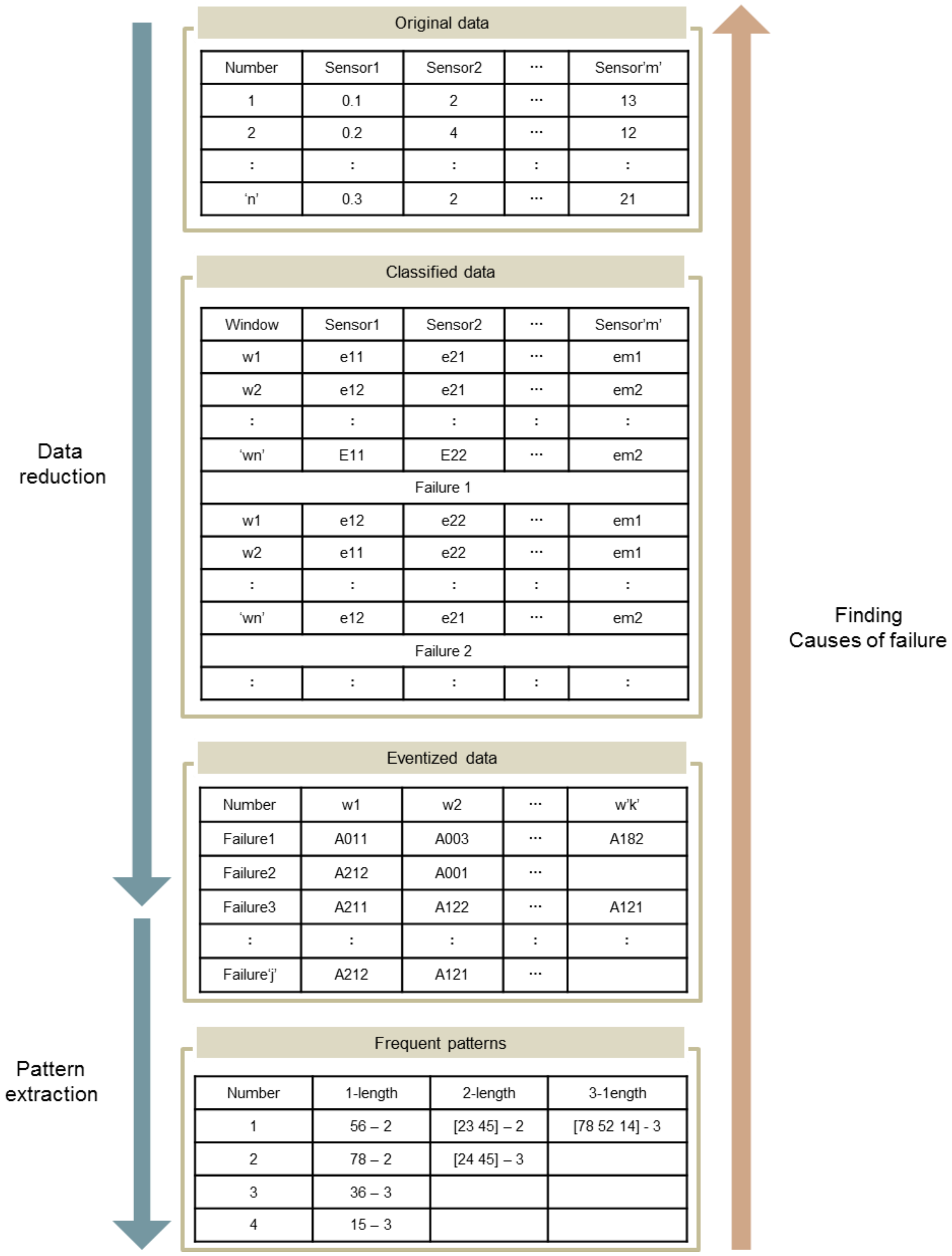
The goal of failure analysis is to find the causes of failures. If event logs are extracted from data reduction phases, failure patterns about the causes of failures would be extracted and then proper action should be applied to the systems. In this section, the process for extracting failure patterns is described, and then the relation between each parameter and extracted result is analyzed. Temporal reasoning technique is also applied to consider the interval of event logs. Extracted failure patterns are stored in the pattern library.

4.3.1 Pattern extraction

Raw data is processed from data cleaning and reduction phases and then converted certain event logs. Using these event logs that are converted from four different eventization techniques, causes of failure will be extracted. Different operations of sensors are considered when the system is operating. If system has a certain failure state, specific sensor data and interaction information of each sensor would be stored. Failure pattern means the relation information about each sensor before failure occurs. Data mining technique is applied to get this sensor information, and then interaction between sensors would be provided.

Of all data mining methods as described in Chapter 2, FP-Tree mining is applied because it can extract the patterns more quickly as compared with other sequence mining methods. As displayed in Figure 4.17, the blue arrow indicates procedures which are data reduction and pattern extraction. If original data is used for FP-Tree analysis, it may take much time. On the other hand, reduced event logs that are processed in the eventization phase can help to make analysis more efficient. Based on eventized data, patterns are extracted as in the last table in Figure 4.17. In this table, different lengths of patterns are extracted. First values indicate patterns and second value indicates a frequency, which means the occurrence of patterns. For example, “[78 52 14] – 3” means that [78 52 14] is a three length pattern and this pattern occurred three times in event logs. Pattern [78 52 14] means [event 78, event 52, event 14], and they contain certain sensor information. Of all these extracted patterns, the result is specified as the longest patterns in this thesis.

Only frequent patterns can give information about the causes of failure. Therefore, these patterns should be interpreted sensor information. The red arrow in Figure 4.17indicates a procedure for finding sensor information related to frequent patterns. Accordance with extracted patterns, procedures would be back in reverse order. For example, pattern [78 52 14] contains information of sensor state in each event. Event “78” can contain “e12e21e…” which is interpreted as sensor 1 value has a higher value as compared to standard value, and sensor 2 value has a smaller value as compared to standard value. This sensor information is contained in each event. The result can be said that if these state of sensor state occurred together, certain failure can occur. The example of the result is displayed in Table 4.2. Two sequences are generated, such as [29 15 1] and [2 5 30]. Each event in the pattern contains sensor information in comparison to standard value.



Original data

Number	Sensor1	Sensor2	···	Sensor'm'
1	0.1	2	···	13
2	0.2	4	···	12
:	:	:	:	:
'n'	0.3	2	···	21

Classified data

Window	Sensor1	Sensor2	···	Sensor'm'
w1	e11	e21	···	em1
w2	e12	e21	···	em2
:	:	:	:	:
'wn'	E11	E22	···	em2
Failure 1				
w1	e12	e22	···	em1
w2	e11	e22	···	em1
:	:	:	:	:
'wn'	e12	e21	···	em2
Failure 2				
:	:	:	:	:

Eventized data

Number	w1	w2	···	w'k'
Failure1	A011	A003	···	A182
Failure2	A212	A001	···	
Failure3	A211	A122	···	A121
:	:	:	:	:
Failure'j'	A212	A121	···	

Frequent patterns

Number	1-length	2-length	3-1ength
1	56 – 2	[23 45] – 2	[78 52 14] - 3
2	78 – 2	[24 45] – 3	
3	36 – 3		
4	15 – 3		

Figure 4.17: Pattern extraction and finding causes of failure procedure

This standard value is calculated from the eventization phase using discretization techniques. Users can know about the causes of failure, when certain failures occurred and then suitable action will be processed. In the pattern extraction phase, the causes of failure could be generated along with the event logs that are processed in the eventization phase. These results are different by applying four different discretization techniques because the ranges of event logs are different. In this paper, the best result is indicating that patterns should be long and have many frequencies and also have appropriate sensor information. Extracted patterns are stored in a library, and failure will be predicted based on these collected results.

Table 4.2: Example of pattern extraction

Sequence	Event number	Sensor name	Comparison	Standard value	Normal (min)	Normal (max)
#1	Event id 29	Sensor1	>	3.2	0	2.5
		Sensor3	>	2.55	0	2
		Sensor25	<	12	0	11
	Event id 15	Sensor4	>	13.5	0	8
		Sensor13	>	12	0	9
		Sensor2	<	1.22	0	0.5
	Event id 1	Sensor17	<	0.25	0	0.27
		Sensor18	>	0.2	0	0.1
#2	Event id 2	Sensor2	>	0.7	0	0.5
		Sensor3	>	3.2	0	2
	Event id 15	Sensor22	>	2.7	0	2.5
		Sensor11	>	12	0	10
		Sensor3	>	25	0	22.2
	Event id 30	Sensor41	<	-2	0	13
		Sensor38	>	0.8	0	0.8

4.3.2 Temporal reasoning

Actually, extracted patterns in the pattern extraction phase have much information when we analyze in detail. Extracted patterns in the pattern extraction phase could not be considered the temporal relationship in that they just could be interpreted as associations between each event. For example, a typical would be rule “$X \rightarrow Y$” which states if X occurs then Y occurs when temporal relationship is not considered. On the other hand, “$X \rightarrow^{T} Y$” states if X occurs then Y will occur within time T (Das et al. 1998). As described in the example, sequence mining cannot tell us the time gaps between successive items(Chen et al. 2003); however, information of time intervals leads to useful knowledge (Verma and Vyas 2005).

As previously described, the information of sensors’ relation is given in the failure extraction phase, and it just gives the value of each sensor as causes of failures. In failure analysis, many types of sensors interact in different time domains and different ordered patterns in the same pattern give different meanings. In detail, different meanings are extracted when the time intervals between two different sensors' states are different. We can assume that "event B" will happen a year from the occurrence of "event A." The extracted result could not be determined as significant information though the relation between "event A" and "event B" is extracted as a frequent pattern. Therefore, detailed time interval analysis is needed for accurate analysis.

In general, methods about temporal reasoning use extending of developed data mining algorithms. At first, a sliding window can be used by limiting specific size (Mannila et al. 1997) that different lengths of patterns are extracted within a specified size of sliding window as displayed in Figure 4.18. This method has a limitation that longer size patterns than the specified window size cannot be extracted but it is easy to extract patterns while considering the time interval between items. The famous temporal sequence mining technique is temporal apriori approach (Ale and Rossi 2000). The purpose of this method is to include a new product which is popular recently. Actual significant information is the most popular product in the recent in marketing area; however, some of these new products may not be included in a developed sequence mining rule. In temporary apriori approach, temporary information can be extracted by adding a temporal threshold. This method needs two thresholds which are minimum support and minimum temporal support. After pruning using minimum support, temporal support is applied to prune more restricted patterns whose temporal support is lower than specified temporal support. This temporal apriori approach is also slow to generate results because it scans multiply.

As mentioned before, temporal apriori approach is not useful in large data, so another approach that can generate the result quickly is needed. Temporal association rules is applied using FP-tree approach (Verma and Vyas 2005). This is same approach as the original FP-tree approach, only it constructs the time calendar pattern. Calendar pattern means that time information is divided, such as months or days. An example is displayed in Figure 4.19and the result can be interpreted separately as different periods separately. Therefore, we can know which patterns are extracted frequently in each period.

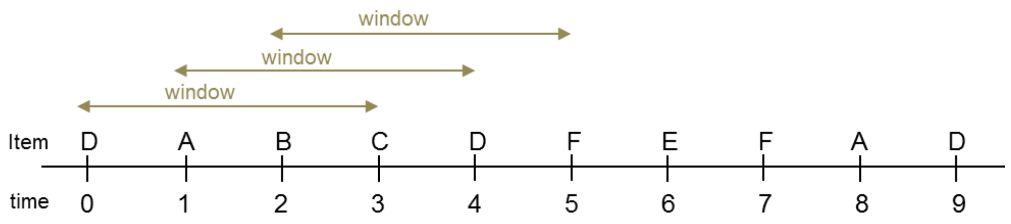


Figure 4.18: Frequency episode extraction (Mannila et al. 1997)

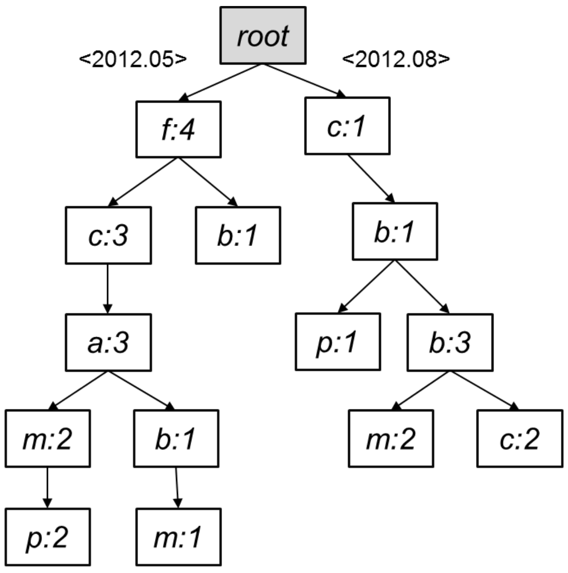


Figure 4.19: A temporal association rule by extending FP-Tree (Verma and Vyas 2005)

Table 4.3: Extracted temporal patterns

Patterns	Item1	Interval	Item2	Interval	Item3
[29 15 1]	29	short	15	long	1
	29	short	15	short	1
	29	long	15	short	1
[2 15 30]	2	long	15	long	30
	2	long	15	short	30

The two methods described, temporal apriori and FP-Tree, are not suitable for our problem because our interest is how much time until the next incident would take place among sequential patterns (Chen et al. 2003). I-Apriori algorithm and I-PrefixSpan algorithm are suggested (Chen et al. 2003), which consider time interval sequential pattern mining and then find all the time interval sequential sequences whose supports are more than a specified temporal threshold. They divide a certain number of time intervals and then calculate time intervals between each item. As a result, the extracted patterns have information about time interval between two items. To divide time intervals, fuzzy time interval approach is also proposed which can discover the time interval information between successive items in the pattern (Chen and Huang 2005).

In this thesis, temporal reasoning is applied by adding time interval threshold. FP-Tree algorithm is used in the pattern extraction phase, so information about the time intervals is added by using extracted frequency patterns. This procedure can be described using Figure 4.19. The information of time interval can be discovered from an eventized data table and then the major two states of time intervals are grouped as “short” and “long” based on the eventized data table. These two states of time intervals are determined by using specified temporal minimum support so we can know the state between a certain item and successive item. An example of extracted temporal patterns is displayed in Table 4.3. A pattern [2 15 30] has detailed information which is [2 – long – 15 – long -30] and [2– long –15 – short -30]. In temporal reasoning results, more detailed information is extracted because one pattern contains several of combinations. Using this detailed information, causes of failures can be extracted more accurately and give proper solutions.

The information about time intervals of items give more detailed causes of failures and it also give sequenced events. In FP-Tree mining, extracted patterns are just the association of items; however, ordered sequence patterns can be extracted in the temporal reasoning method as well as the relation between events. This sequence pattern information is more useful to the analysis and prognosis phases.

4.4 Parameter specifications of the pattern extraction procedure

Failure patterns can be extracted from the data reduction and failure pattern analysis phases. Along with the different discretization techniques, the different patterns can be extracted. The result can also be changed along with parameters that are specified by the users. As mentioned before, the most significant pattern is indicating the longest patterns of all extracted patterns in this paper. Therefore, proper parameters should be selected to get more accurate results. In these failure analysis procedures, three parameters are needed as follows.

- Window size: the number of data contained in one window in eventization phase.
- Normal state (%): the standard values when an event is contributed in the Knowledge expert-based discretization phase. It is given percentage values and then period of standard value is determined by distributing the percentage to calculated standard value.
- Threshold (minimum support): Frequent patterns are determined when the frequency of pattern is higher than a fixed threshold. It is used in the pattern extraction phase.

The result is based on the above three parameters precisely. At first, if window size is large, the number of extracted patterns can be reduced; however, patterns contain a lot of information. It is difficult to justify accurately. Appropriate window size can be determined by considering the time of collected data. User should determine the window size by calculating the number of data before failures occurred. Secondly, if normal state is large, the number of extracted patterns would be reduced because event is determined when the value of event is not containing a specified standard period. If percentage of normal state is large, period of standard values would become large. It can help to extract patterns quickly; however, information about the sensor state could be lost. Lastly, the threshold is also determined by the user when FP-Tree algorithm is applied. In pattern mining, research is being done to determine the suitable threshold. The number of patterns is reduced when the threshold is large. However if too large a threshold is applied, extracted patterns would not be enough to interpret the causes of failure. On the other hand, too many patterns would be generated when the threshold is too small. In that case, it takes much time for processing because large numbers of patterns are generated. As a result, the three parameters have to be determined appropriately by users because the result is sensitive to these parameters.

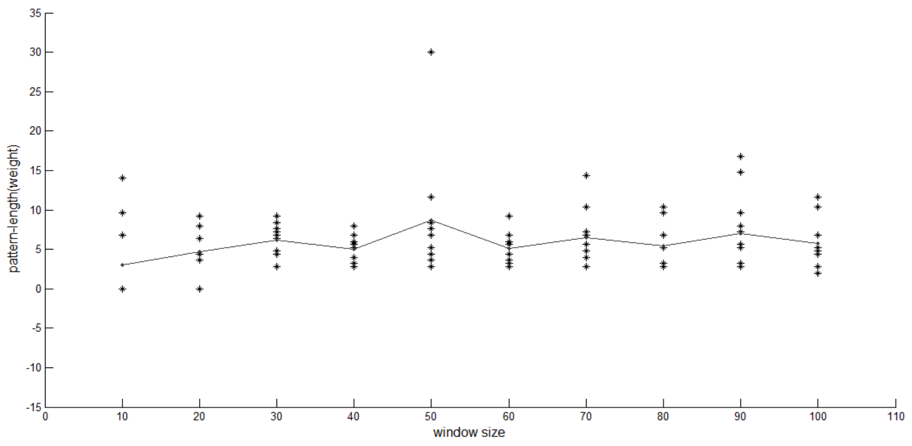


Figure 4.20: Pattern lengths – Window size (Equal-width binning)

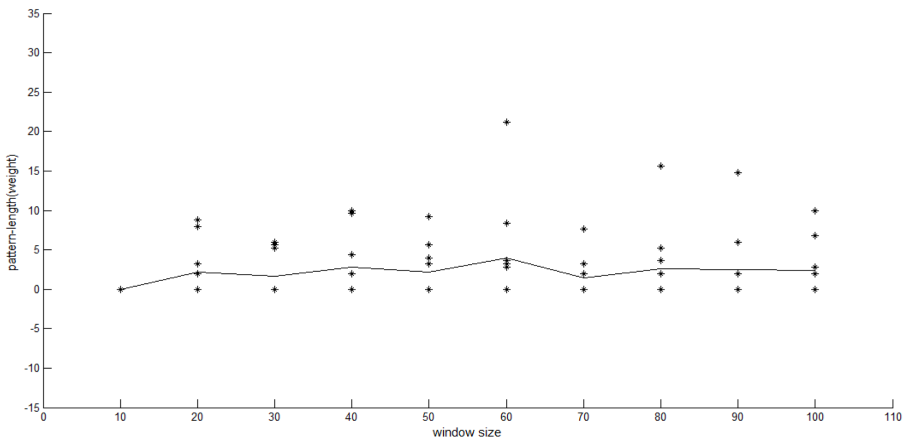


Figure 4.21: Pattern lengths – Window size (Entropy)

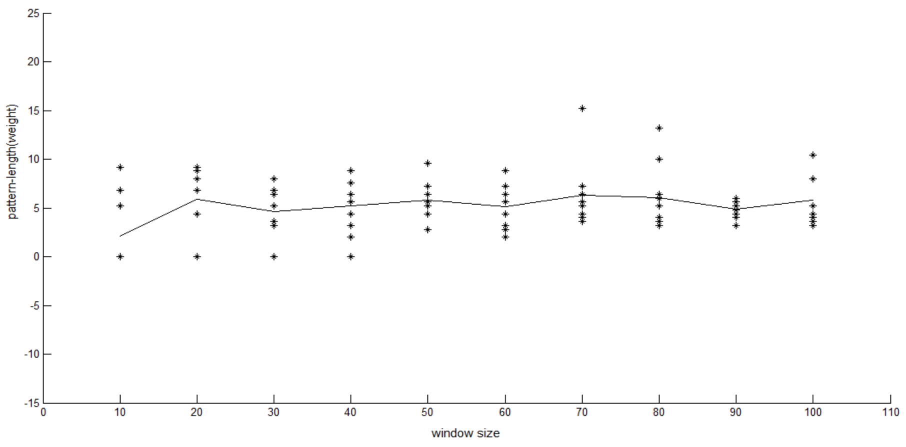


Figure 4.22: Pattern lengths – Window size (Knowledge expert)

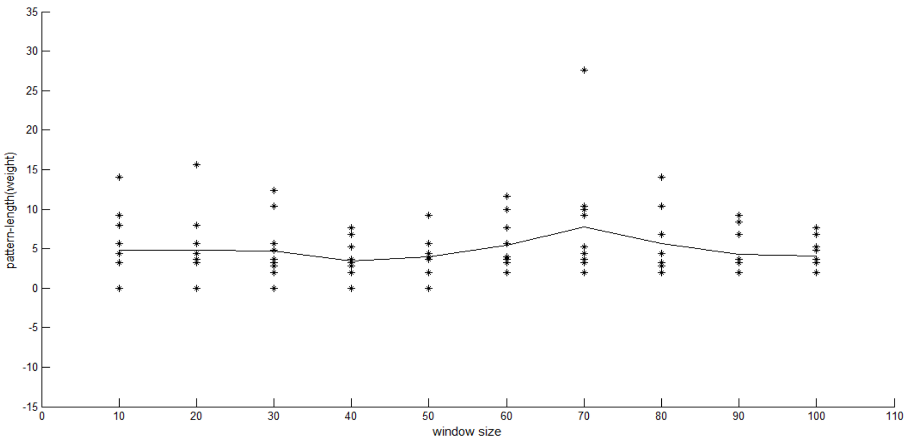


Figure 4.23: Pattern lengths – Window size (Estimation of probability distribution)

Of these three parameters, a suitable window size is estimated in this paper with four different discretization techniques. Experiments are analyzed as increasing window size, and this window size is applied to 10 different data sets. The result can be extracted by applying a weighting method to pattern length and number of patterns. Weight of pattern length is 60%, and weight of number of patterns is 40%. Based on this weighting criterion, the result is displayed in Figure 4.20, 21, 22 and 23 when these different window sizes are applied in equal-width binning, entropy, knowledge expert-based discretization and estimation of probability distribution-based discretization. As displayed in these graphs, good results are extracted when window size is between 50 and 70. A small window size gives long patterns; however this presents some problems. Sometimes any event logs cannot be extracted when a small window size is used. Therefore, the result patterns also cannot be extracted. The other problem is also displayed in Figure 4.25. Time consumption is very long when window size is small. The time is the significant attribute for efficient analysis. As a result, the suitable window size should be selected for accuracy and efficiency of analysis.

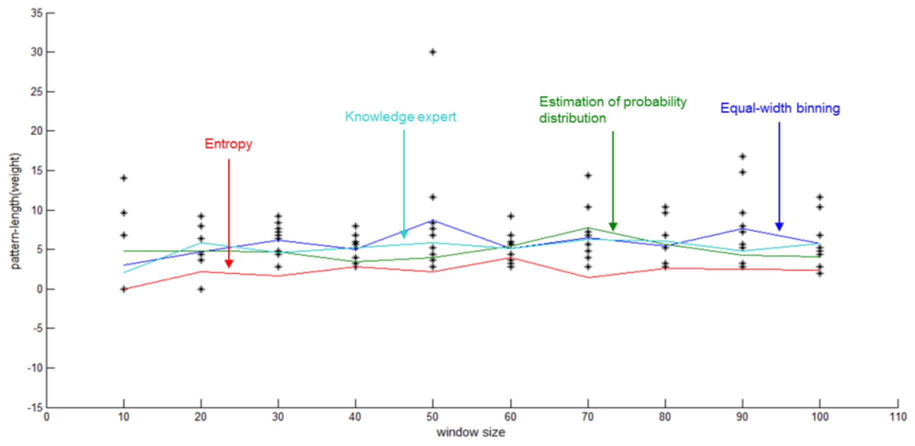


Figure 4.24: Pattern lengths – Window size

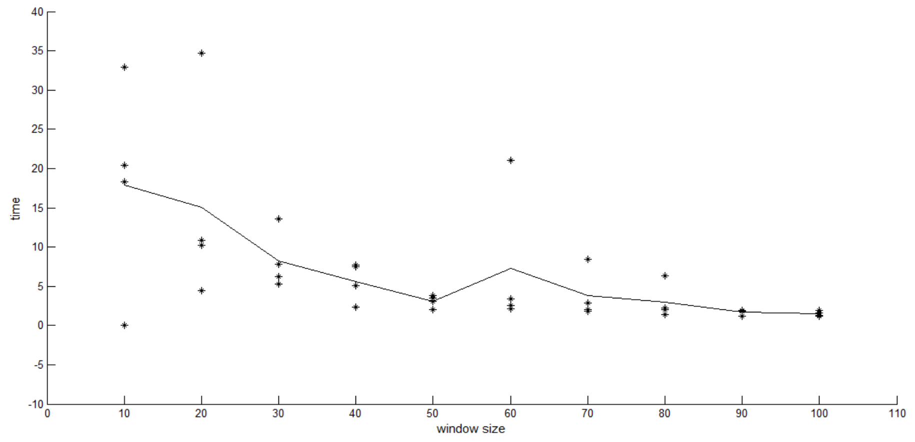


Figure 4.25: Run time – Window size

Table 4.4: ANOVA Test (Pattern length – Window size)

Source	Sum of Squares	Degree of freedom	Mean squares	Prob>F
Columns	77.26	3	25.7533	1.09418e-006
Error	59.11	36	1.642	
Total	136.37	39		

When weighting method is applied, the relation between pattern length and number of patterns should be considered. If the pattern length is short, the number of patterns is usually higher. In certain cases, the relation between pattern length and the number of patterns is an inverse proportion to each other. In this paper, the weighting method for this relation is not considered and different weighting percentages are given to pattern length and number of pattern attributes. When these weighting percentages are given, the result is displayed in Figure 4.24. As displayed, the pattern length is the longest when window size is 50 to 70. However, a similar trend is displayed when weighting percentage of pattern length is 60. If the gap of percentage is larger, the pattern length would be changed more radically.

To verify the relation between pattern length and window size, ANOVA test is analyzed. The result is displayed in Table 4.4 when pattern lengths of four different discretization techniques are used for analysis. The p-value is 1.09418e-006, which is a very low value. Therefore, the pattern length is affected by window size. In addition, T-test is applied to verify the hypothesis that window size 70 is more effective on the pattern length than window size 10. P-value is 0.01 when T-test is applied, and then this hypothesis is said to be valid.

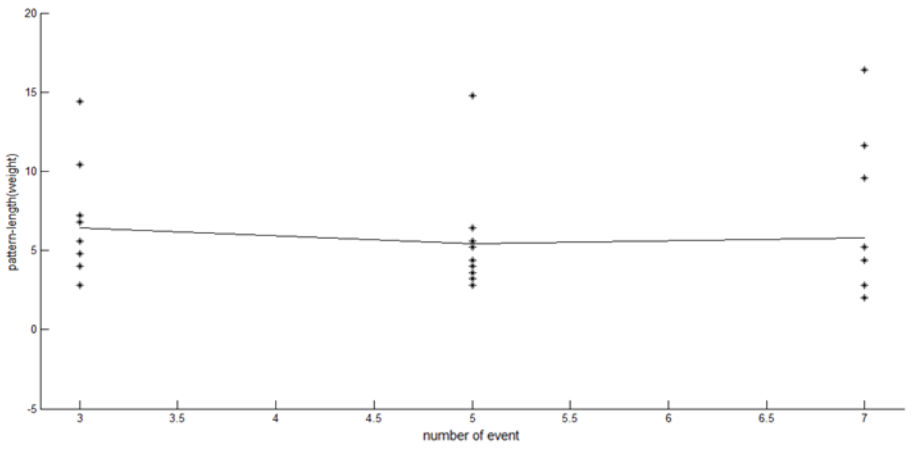


Figure 4.26: Pattern length – Number of event (Equal-width binning)

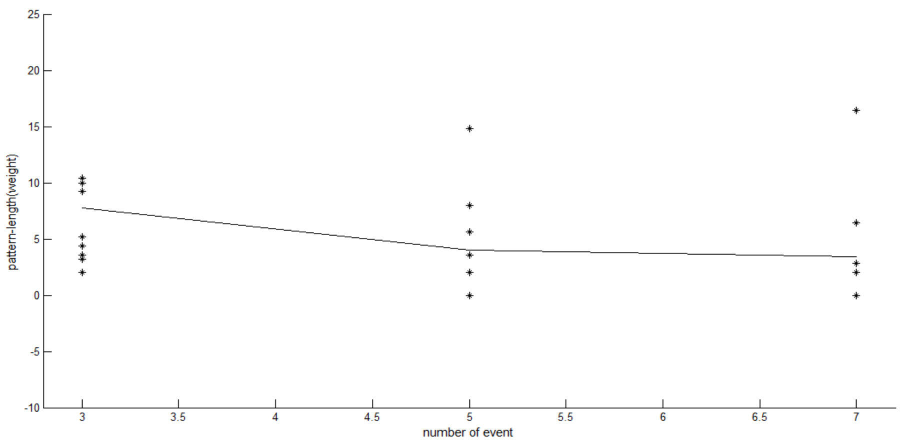


Figure 4.27: Pattern length – Number of event (Estimation of probability distribution)

In addition, the relation between pattern length and the number of events is considered in this paper. Equal-width binning and estimation of probability distribution-based discretization of four discretization techniques can make different length patterns, which are three, five and seven. To verify this effect of the number of events, window size is fixed as 70 and the experiment is repeated by increasing the number of events. One result is displayed in Figure 4.26 when equal-width binning discretization is used. Similar results are displayed; however, the pattern length is slightly higher when the number of events is three. The other result is displayed in Figure 4.27 when estimation of probability distribution-based discretization is applied. As compared with the result of equal-width binning discretization, the more obvious result can be extracted. As shown in this result, the lowest number of events can extract more good results because the lower number of events has redundant event logs. For this reason, redundant patterns can be extracted as compared with other numbers of events.

Through this parameter estimation, a suitable window size and the number of events can be known. Based on this result, accuracy should also be considered above pattern length. For failure analysis, the accuracy of these extracted patterns is also a significant factor. After this research, the accuracy needs to be studied with fixing these suitable parameters.

4.5 Prognosis

Modern industry is concerned about extending the lifetime of its critical processes and maintaining them only when required (Wang and Vachtsevanos 2001). To extend the lifetime of a system and improve the quality of products, proper prognosis techniques are needed. As previous failure diagnostics methods, prognosis methods also have two different methods, which are data-driven methods and model-based methods. In our case, data-driven failure prognosis is more suitable because we just extract the certain patterns without a mathematical model. As described before, there are many methods of data-driven failure prognostics which are neural network, wavelet and similarity approaches (Schwabacher 2005). Similarity approach, which uses average values, is close to our method developed in this paper. A detailed method is described as follows. If causes of failures were extracted, suitable reaction should be provided for reliable systems. As described in Chapter 2, probability basis is used to predict the failures using signals or certain rules. For using these methods, a large amount of previous information should be stored in a pattern construction library because more historical data increases the probability of failure prediction. In this paper, historical data that are stored in pattern construction are indicating extracted patterns as causes of failure. Therefore, extracted patterns should essentially be stored in the failure analysis phase.

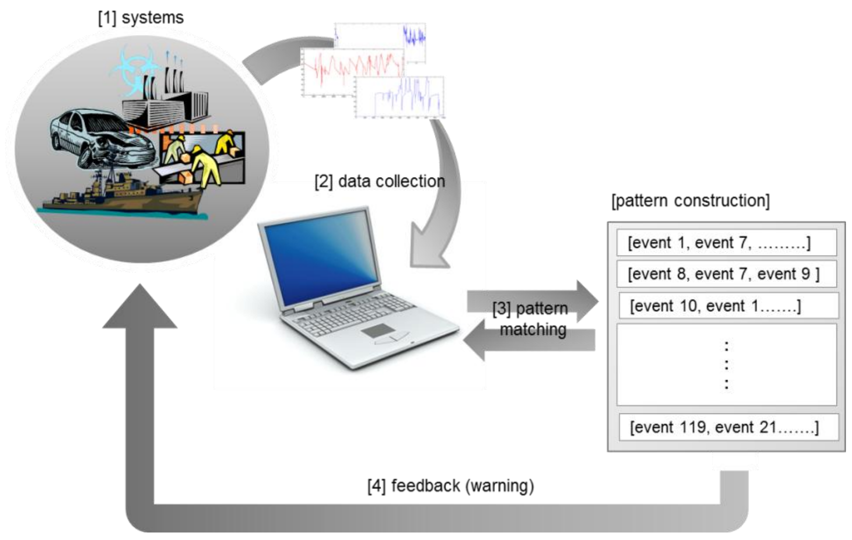
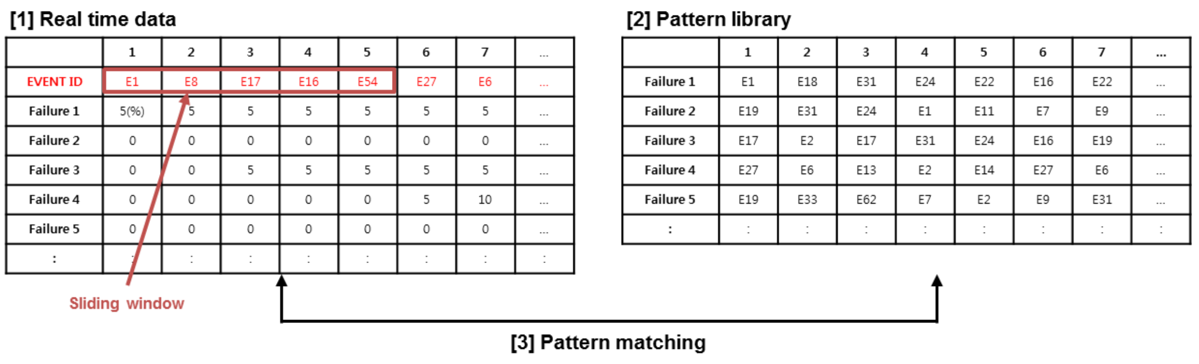

[1] systems
[2] data collection
[pattern construction]
[event 1, event 7, ...]
[event 8, event 7, event 9]
[event 10, event 1.......]
:
:
:
[event 119, event 21.......]
[3] pattern matching
[4] feedback (warning)

Figure 4.28: Procedure of failure prognosis



	1	2	3	4	5	6	7	...
EVENT ID	E1	E8	E17	E16	E54	E27	E6	...
Failure 1	5(%)	5	5	5	5	5	5	...
Failure 2	0	0	0	0	0	0	0	...
Failure 3	0	0	5	5	5	5	5	...
Failure 4	0	0	0	0	0	5	10	...
Failure 5	0	0	0	0	0	0	0	...
:	:	:	:	:	:	:	:	:

	1	2	3	4	5	6	7	...
Failure 1	E1	E18	E31	E24	E22	E16	E22	...
Failure 2	E19	E31	E24	E1	E11	E7	E9	...
Failure 3	E17	E2	E17	E31	E24	E16	E19	...
Failure 4	E27	E6	E13	E2	E14	E27	E6	...
Failure 5	E19	E33	E62	E7	E2	E9	E31	...
:	:	:	:	:	:	:	:	:

Figure 4.29: Pattern matching process in failure prognosis

Figure 4.28 shows procedures of failure prognosis. As described before, data can be collected from systems and these data would be diagnosed. After analyzing data, extracted patterns that are causes of previous failures are stored in the pattern construction library. Many different failures are stored in the pattern library, and this data is based on failure prognosis. Based on these stored patterns, a pattern matching process is started. In detail, accepted data in real-time must be converted to certain event logs, and then these converted event logs are compared with stored event patterns as displayed in Figure 4.29. The probability of failure occurrence is calculated by comparing real-time event logs and stored event logs. If two event logs are the same, the probability of failure occurrence would be increased. On the other hand, if two patterns are different, the probability of failure occurrence would be not changed.

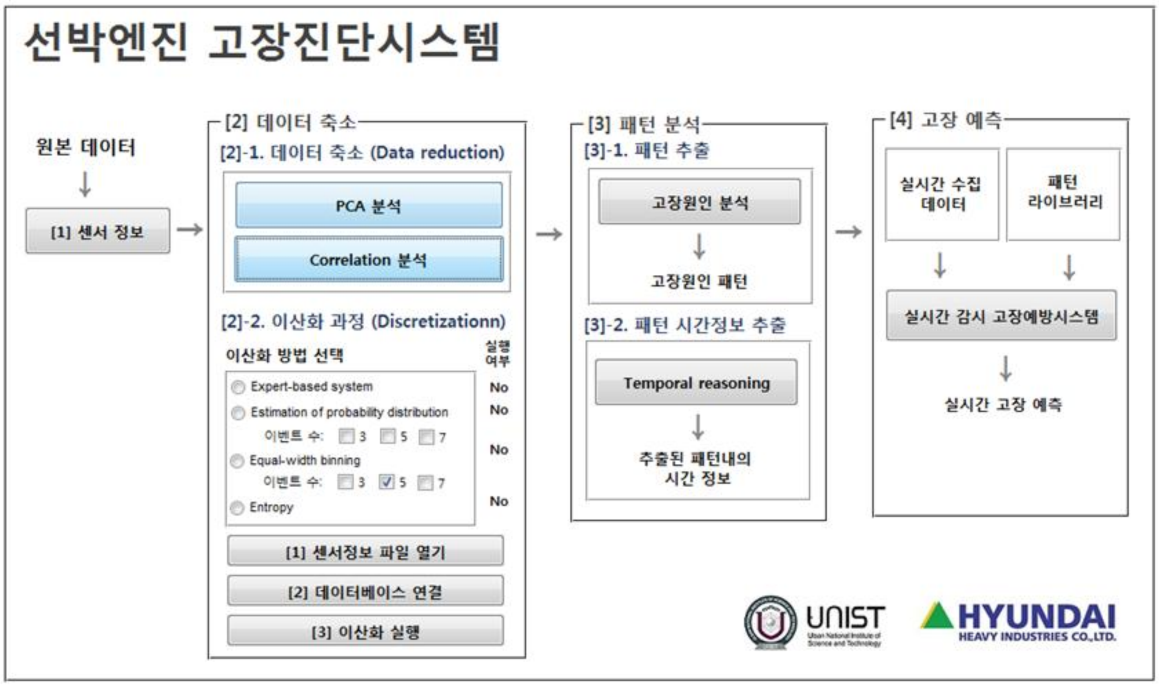
In a real prognosis system, a warning condition exists, and the compared event logs are included in a specified sliding window. This sliding window is moving at one second, and then these event logs included in the window are compared with all event logs that are stored in the pattern library. If the probability of the occurrence is higher than the specified warning condition, the system would give the alarm. Users can give proper reaction from these procedures, and then failure can be predicted and solved.

In this thesis, we find frequent long patterns where the longest patterns are regarded as important information of failure analysis. Pattern discovery in long sequences is of a great importance in many applications including computational biology study, consumer behavior analysis, and system performance analysis (Benson and Waterman 1994; Brāzma et al. 1998; Yang et al. 2002). Given a large data set, there have been many researches in mining a frequent long pattern. These long patterns contain many frequently occurring items and have a wide average record length (Bayardo Jr 1998). Extracted long pattern in failure analysis also contain most sensor information about failures. In addition, the longest pattern concisely represents all frequent itemsets because they contain all subsets of frequent itemsets (Bayardo Jr 1998). It means that the longest failure pattern contains all subset information of patterns. Because of these reasons, the longest pattern is defined as important information of failure analysis. For the future, condition-based statistical analysis will be used to analyze a completeness and soundness of the proposed failure analysis method.

This Page Intentionally Left Blank

V. Case study: marine diesel engines

Failure analysis procedures are described in Chapter 4. These procedures are applied to terminal box data which are collected from marine diesel engine by 56 different sensors. Data can be converted as filtered data by applying data cleaning, and some features are extracted in statistical analysis. After the data reduction phase, four different eventization techniques are applied, and results are compared. At the last phase, failures are predicted in prognosis phase. Software is developed to execute these procedures therefore the case study is described with software description. As displayed Figure 5.1, the developed software consists of pre-processing and pattern analysis phase, and all analysis phases can nr applied using this software.



이산화 방법 선택	실행 여부
Expert-based system	No
Estimation of probability distribution (이벤트 수: ☐ 3 ☐ 5 ☐ 7)	No
Equal-width binning (이벤트 수: ☐ 3 ☑ 5 ☐ 7)	No
Entropy	No

Figure 5.1: Failure analysis (main software)

5.1 Data cleaning of terminal box data

Terminal box data which are collected from marine engine do not have missing values. To convert into filtered and smoothed data, Kalman filter is applied. Engine RPM and Turbo Charger RPM sensors are used in this data cleaning phase. Figure 5.2 and 5.5 show the original Engine RPM sensor and Turbo Charger sensor signal, and all of data have similar values; however, certain feature cannot be found. Some features are extracted when Kalman filter is applied. Figure 5.3 and 5.6 display the filtered data which are predicted from forward data. In addition, Figure 5.4 and 5.7 display the smoothed data which are predicted from entire sensor data.

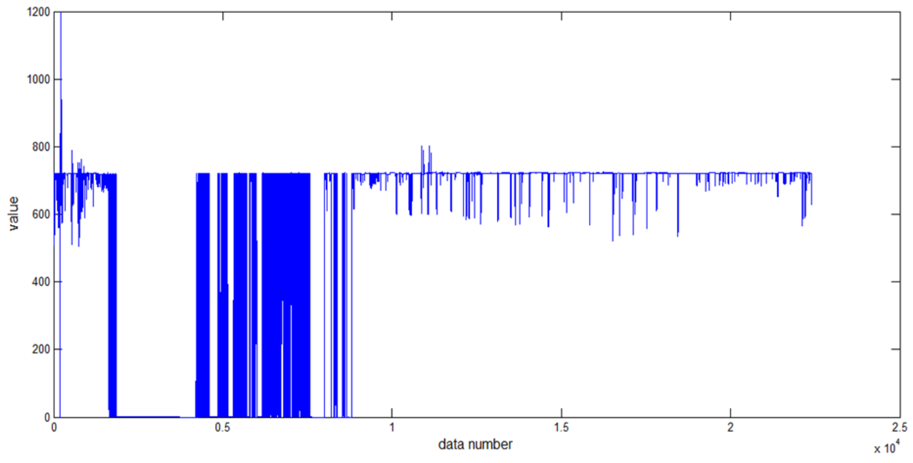


Figure 5.2: Engine RPM sensor signal

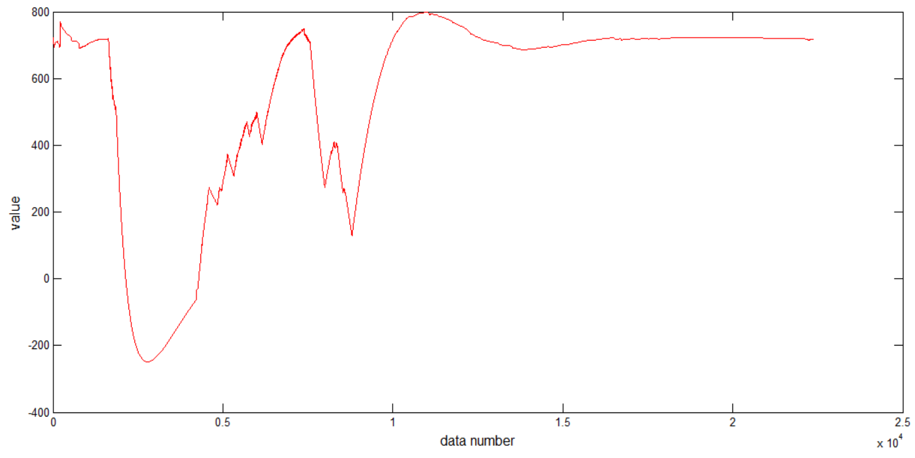


Figure 5.3: Filtered data (Engine RPM sensor)

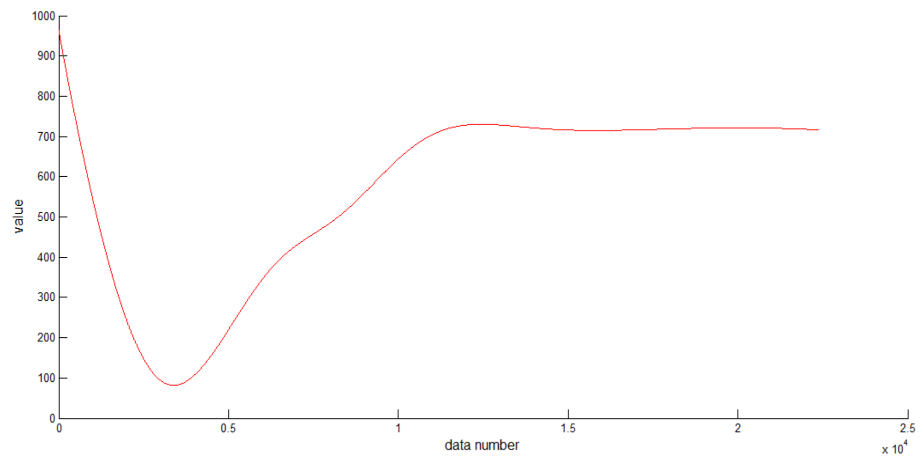


Figure 5.4: Smoothed data (Engine RPM sensor)

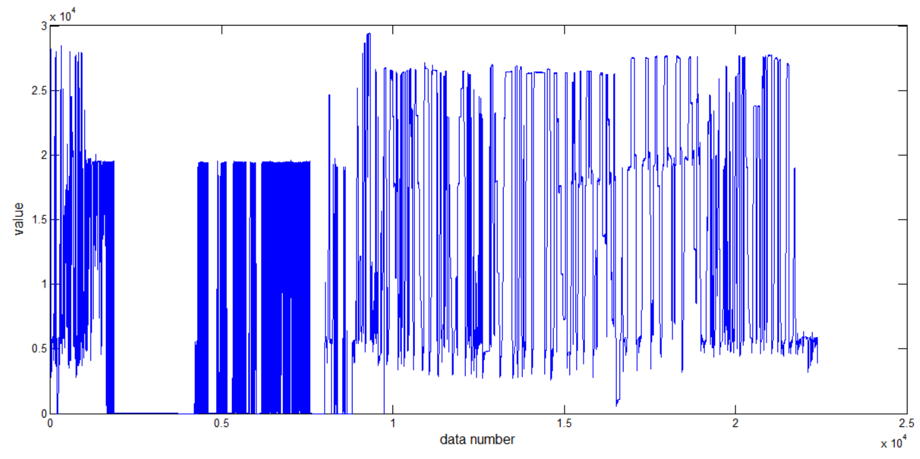


Figure 5.5: Turbo charger sensor signal

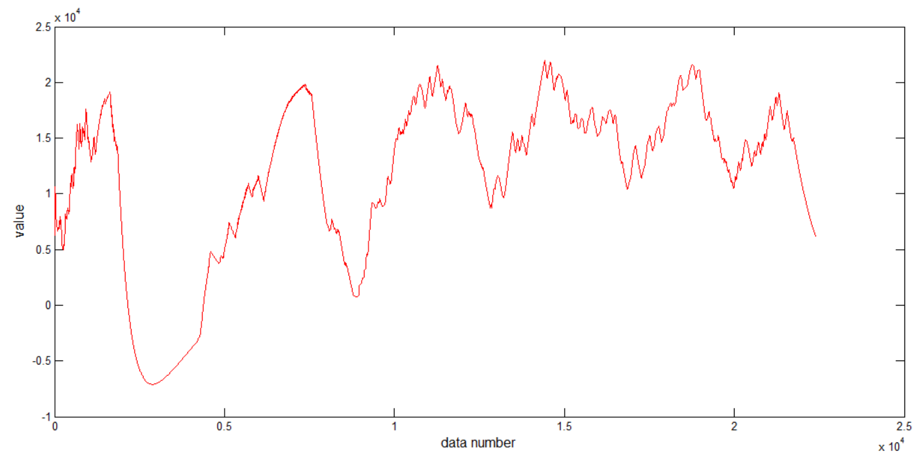


Figure 5.6: Filtered data (Turbo charger sensor)

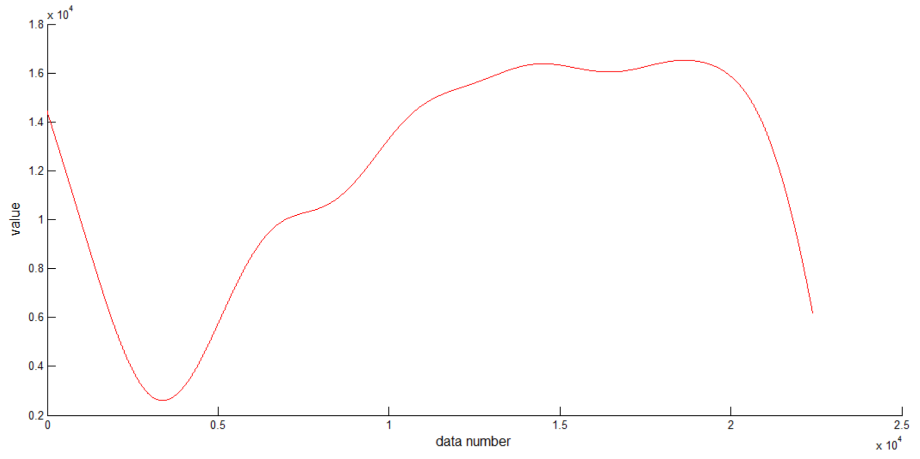


Figure 5.7: Smoothed data (Turbo charger sensor)

In this case, more obvious features can be extracted by applying Kalman filter as compared with original sensor data. Filtered data give more obvious features, and then sensor trends can be more easily known. On the other hand, smoothed data can lose some information of original sensor data. Therefore, filtered data is more helpful for analysis than smoothed data. If these filtered data are used for analysis, it would be more efficient.

5.2 Data reduction and Failure pattern analysis of terminal box data

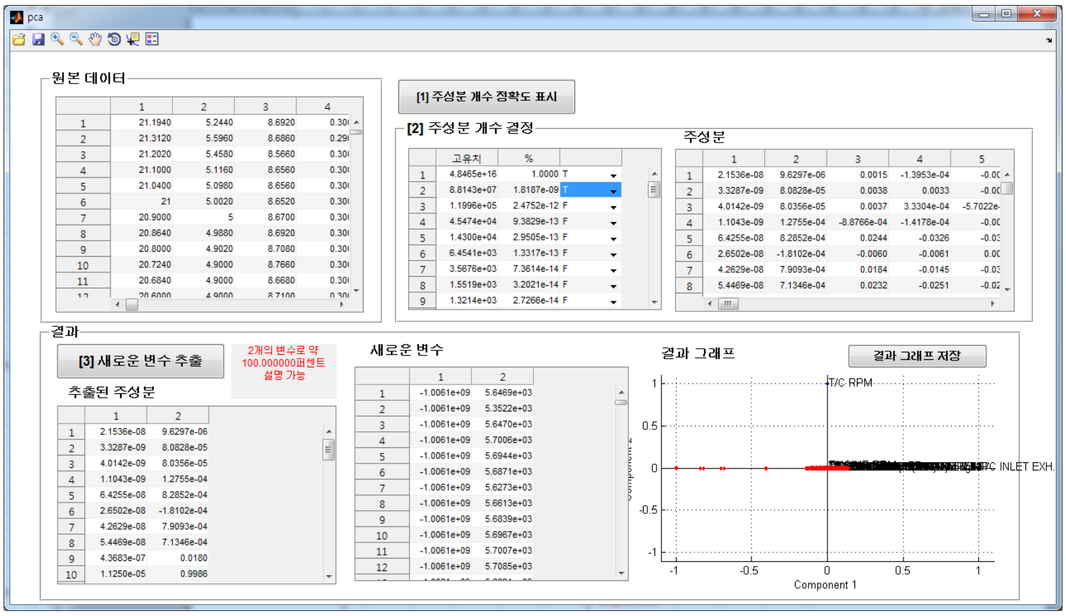
As described in Chapter 4, data reduction phase consists of two phases which are feature extraction and eventization phases. Terminal box data are analyzed by using statistical analyses, and then four different eventization techniques are applied. At results, failure patterns are extracted using converted data from eventization phase, and temporal reasoning technique is also applied.

5.2.1 Feature extraction from terminal box data

Terminal box data are collected by 56 different types of sensors. The most significant sensors and their relation are analyzed using PCA and Correlation analysis.

PCA (Principal Component Analysis)

The most significant sensors can be verified by applying PCA technique. Developed software for PCA is displayed in Figure 5.8. User can select the number of principal components, and then significant principal components are extracted along with the previously selected number of principal components.


pca

원본 데이터

	1	2	3	4
1	21.1940	5.2440	8.6920	0.30
2	21.3120	5.5960	8.6860	0.29
3	21.2020	5.4580	8.5660	0.30
4	21.1000	5.1160	8.6560	0.30
5	21.0400	5.0980	8.6560	0.30
6	21	5.0020	8.6520	0.30
7	20.9000	5	8.6700	0.30
8	20.8640	4.9880	8.6920	0.30
9	20.8000	4.9020	8.7080	0.30
10	20.7240	4.9000	8.7660	0.30
11	20.6840	4.9000	8.6680	0.30
12	20.6000	4.9000	8.7100	0.30

[1] 주성분 개수 정확도 표시

[2] 주성분 개수 결정

	고유치	%	
1	4.8465e+16	1.0000	T
2	8.8143e+07	1.8187e-09	T
3	1.1996e+05	2.4752e-12	F
4	4.5474e+04	9.3829e-13	F
5	1.4300e+04	2.9505e-13	F
6	6.4541e+03	1.3317e-13	F
7	3.5676e+03	7.3614e-14	F
8	1.5519e+03	3.2021e-14	F
9	1.3214e+03	2.7266e-14	F

주성분

	1	2	3	4	5
1	2.1536e-08	9.6297e-06	0.0015	-1.3953e-04	-0.00
2	3.3287e-09	8.0828e-05	0.0038	0.0033	-0.00
3	4.0142e-09	8.0356e-05	0.0037	3.3304e-04	-5.7022e-
4	1.1043e-09	1.2755e-04	-8.8766e-04	-1.4178e-04	-0.00
5	6.4255e-08	8.2852e-04	0.0244	-0.0326	-0.03
6	2.6502e-08	-1.8102e-04	-0.0060	-0.0061	0.00
7	4.2629e-08	7.9093e-04	0.0184	-0.0145	-0.03
8	5.4469e-08	7.1346e-04	0.0232	-0.0251	-0.02

결과

[3] 새로운 변수 추출

2개의 변수로 약
100.000000퍼센트
설명 가능

추출된 주성분

	1	2
1	2.1536e-08	9.6297e-06
2	3.3287e-09	8.0828e-05
3	4.0142e-09	8.0356e-05
4	1.1043e-09	1.2755e-04
5	6.4255e-08	8.2852e-04
6	2.6502e-08	-1.8102e-04
7	4.2629e-08	7.9093e-04
8	5.4469e-08	7.1346e-04
9	4.3683e-07	0.0180
10	1.1250e-05	0.9986

새로운 변수

	1	2
1	-1.0061e+09	5.6469e+03
2	-1.0061e+09	5.3522e+03
3	-1.0061e+09	5.6470e+03
4	-1.0061e+09	5.7006e+03
5	-1.0061e+09	5.6944e+03
6	-1.0061e+09	5.6871e+03
7	-1.0061e+09	5.6273e+03
8	-1.0061e+09	5.6613e+03
9	-1.0061e+09	5.6839e+03
10	-1.0061e+09	5.6967e+03
11	-1.0061e+09	5.7007e+03
12	-1.0061e+09	5.7085e+03

결과 그래프

결과 그래프 저장

Figure 5.8: PCA analysis (software)

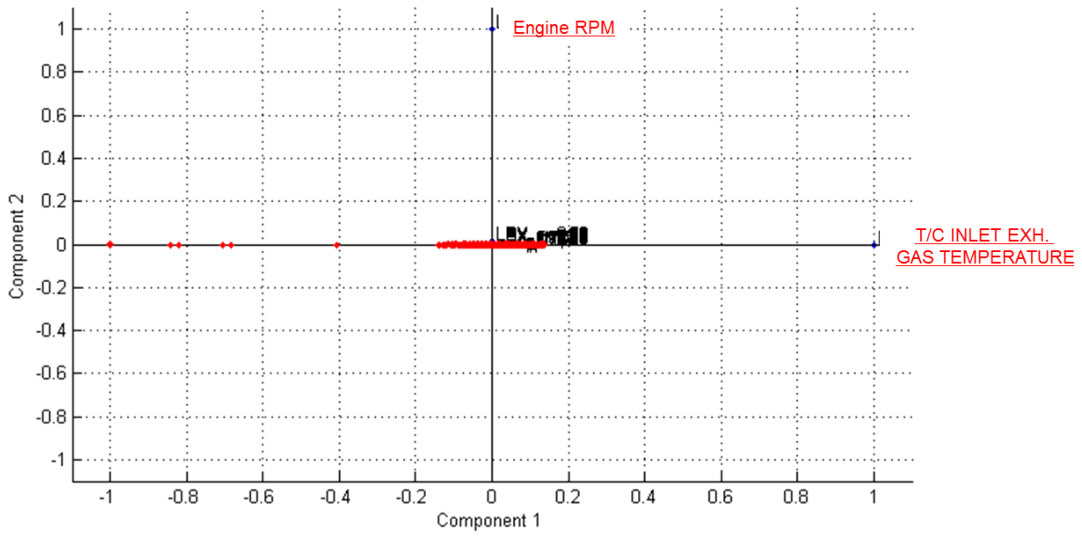


Figure 5.9: PCA result of terminal box data

When terminal box data collected by 56 sensors are used for PCA, two principal components are selected. In that case, extracted two variables can describe the all sensor data about 100 percent and the result is almost good. The reduced new variables containing two variables can represent all sensor data. The most significant sensors are also can extract as described Figure 5.9. Engine RPM and Turbo Charger Inlet gas temperature sensors are extracted the significant sensors. Engine RPM sensor is affected on components 1 and Turbo Charger Inlet gas temperature sensor is affected on component 2. Through this result, data which are consists of 56 sensors can be reduced as only two variables data.

Correlation analysis

The relation between sensors can be extracted in correlation analysis. As described before, correlation analysis has two different analyses which are auto correlation and cross correlation. Developed correlation software is displayed in Figure 5.10. The starting point can be specified by user because the result has no meaning when starting point is zero. Therefore, user should specify the suitable starting point.

Lube-Oil Inlet Temperature sensor has the highest correlation coefficient of all 56 sensors when auto correlation is executed. As displayed in Figure 5.12, the result of correlation coefficient is about 0.96. This result said that this sensor has a certain periodicity as shown in Figure 5.11. Cross correlation analysis is also applied to terminal box data. All pairs of sensors are used for cross correlation analysis, and the highest correlation coefficient is about 0.96 when Cylinder Temperature Main Bearing 8 and 9 sensors are applied as displayed in Figure 5.15. The original sensor signals are displayed in Figure 5.13 and 5.14. As shown in these Figures, these two sensor signals are almost similar. Therefore, correlation coefficient between these two sensors is also high. Through this correlation analysis, some significant sensors of all different sensors can be selected. Number of sensors and entire data size can be also reduced. These statistical analyses can help to analyze the large data more efficiently.

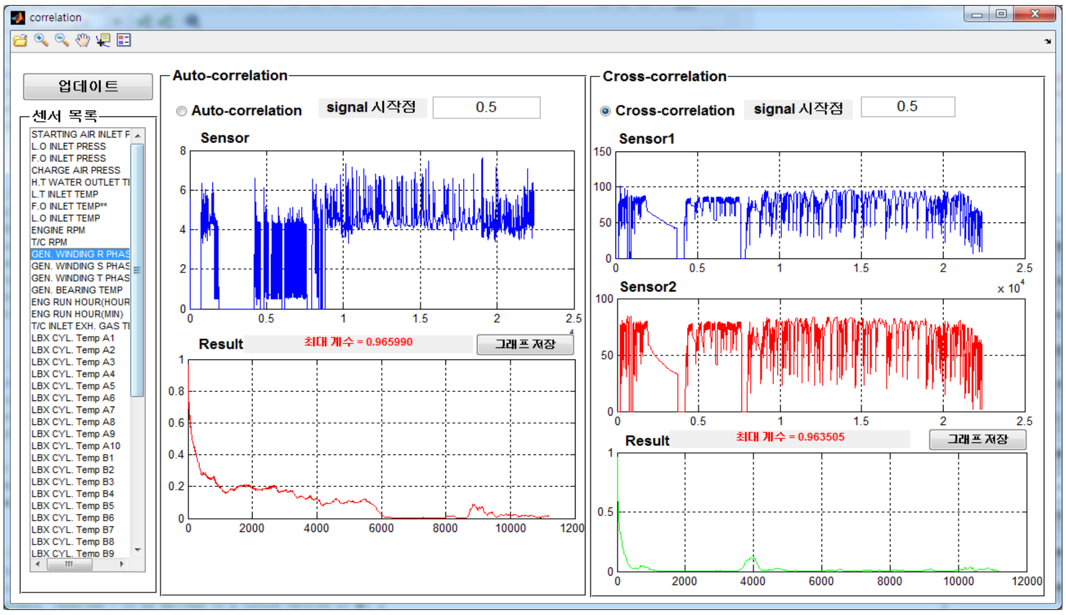


Figure 5.10: Correlation analysis (software)

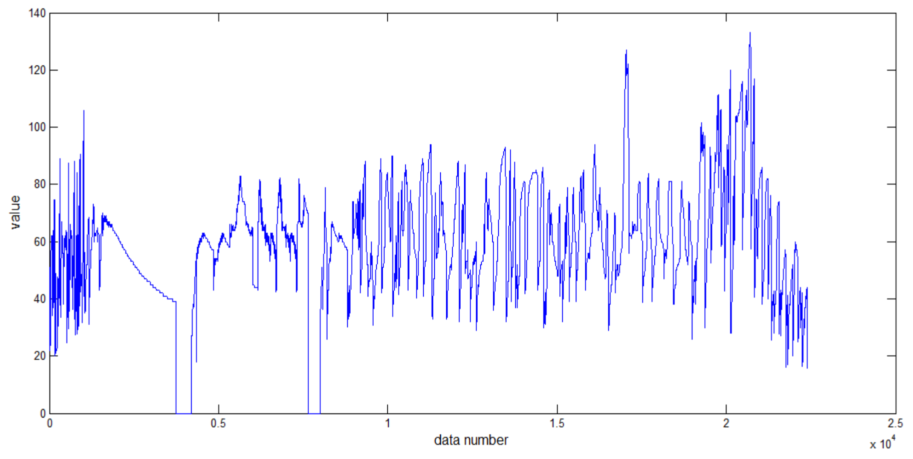


Figure 5.11: Signal of Lube-Oil Inlet Pressure sensor

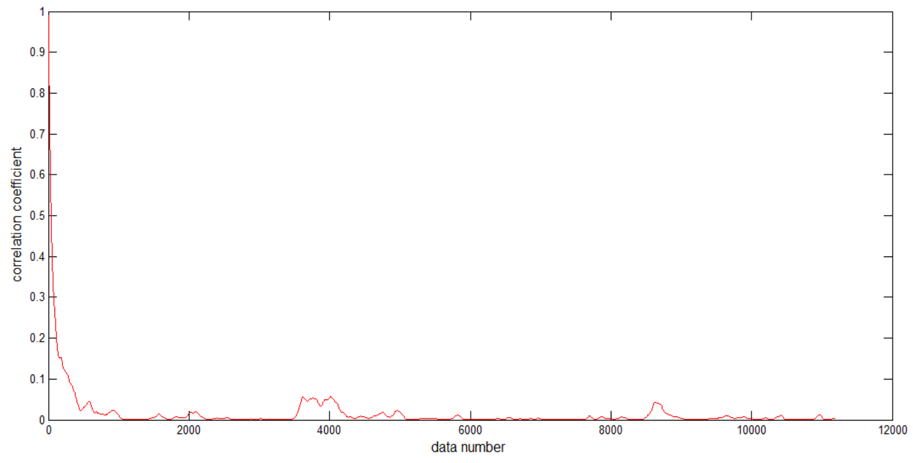


Figure 5.12: The result of Auto correlation (Lube-Oil Inlet Pressure sensor)

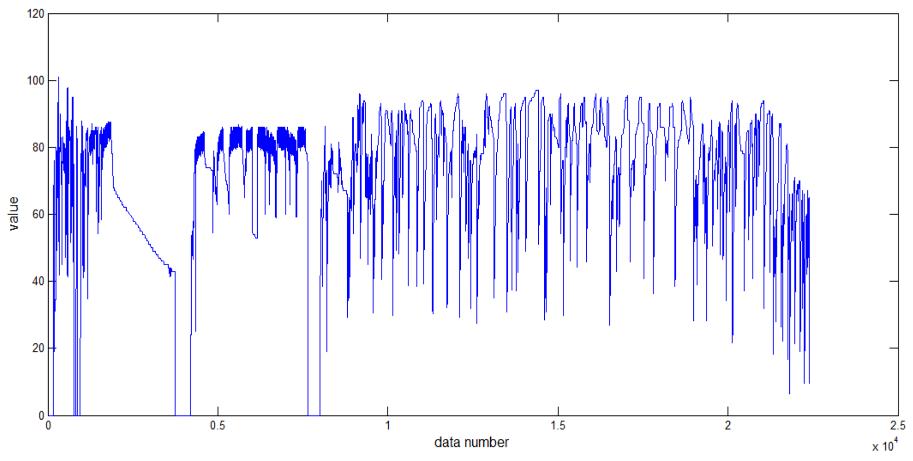


Figure 5.13: Cylinder Temperature Main Bearing 8

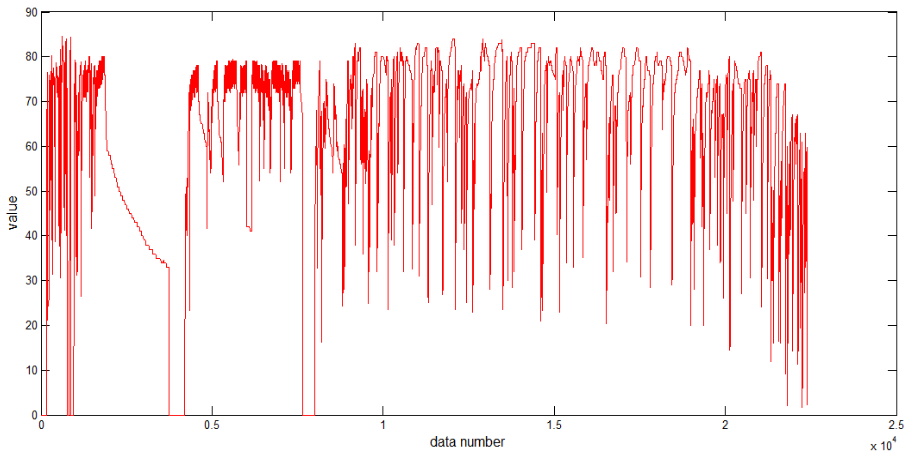


Figure 5.14: Cylinder Temperature Main Bearing 9

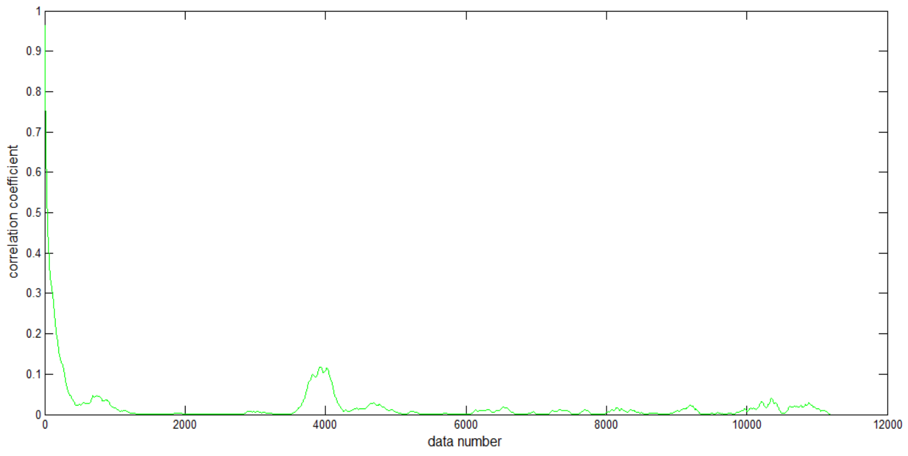
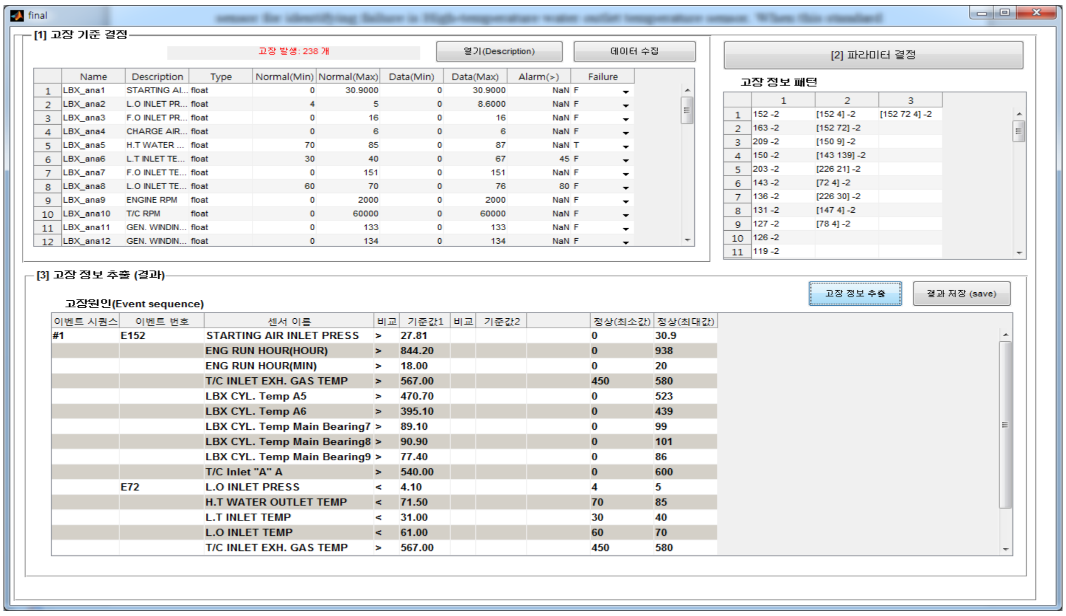


Figure 5.15: The result of Cross correlation (Cylinder Temperature Main Bearing 8 and 9)

5.2.2 Eventization and Failure pattern analysis

After extracting features, eventization phase is applied. Four different discretization techniques which are described in Chapter 2 are applied, and then failure patterns can be extracted. The standard sensor for identifying failure is High-temperature water outlet temperature sensor. When this standard sensor is selected, the experiment data have 238 failures. These data extract different event logs along with different eventization techniques, and the results are also different. Event logs create before extracting patterns, these event logs can make in data pre-processing phase. In Figure 5.1, developed software has functions to divide the event logs along with three different eventization techniques. Using these extracted event logs, patterns are extracted in pattern analysis functions. Developed software is displayed in Figure 5.16including parameter option. Users can select the discretization option in parameter option windows, and then result patterns can be extracted.



고장 발생: 238 개

	Name	Description	Type	Normal(Min)	Normal(Max)	Data(Min)	Data(Max)	Alarm(>)	Failure
1	LBX_ana1	STARTING AI...	float	0	30.9000	0	30.9000	NaN	F
2	LBX_ana2	L.O INLET PR...	float	4	5	0	8.6000	NaN	F
3	LBX_ana3	F.O INLET PR...	float	0	16	0	16	NaN	F
4	LBX_ana4	CHARGE AIR...	float	0	6	0	6	NaN	F
5	LBX_ana5	H.T WATER ...	float	70	85	0	87	NaN	T
6	LBX_ana6	L.T INLET TE...	float	30	40	0	67	45	F
7	LBX_ana7	F.O INLET TE...	float	0	151	0	151	NaN	F
8	LBX_ana8	L.O INLET TE...	float	60	70	0	76	80	F
9	LBX_ana9	ENGINE RPM	float	0	2000	0	2000	NaN	F
10	LBX_ana10	T/C RPM	float	0	60000	0	60000	NaN	F
11	LBX_ana11	GEN. WINDIN...	float	0	133	0	133	NaN	F
12	LBX_ana12	GEN. WINDIN...	float	0	134	0	134	NaN	F

고장 정보 패턴

	1	2	3
1	152 -2	[152 4] -2	[152 72 4] -2
2	163 -2	[152 72] -2	
3	209 -2	[150 9] -2	
4	150 -2	[143 139] -2	
5	203 -2	[226 21] -2	
6	143 -2	[72 4] -2	
7	136 -2	[226 30] -2	
8	131 -2	[147 4] -2	
9	127 -2	[78 4] -2	
10	126 -2		
11	119 -2		

고장원인(Event sequence)

이벤트 시퀀스	이벤트 번호	센서 이름	비교	기준값1	비교	기준값2		정상(최소값)	정상(최대값)
#1	E152	STARTING AIR INLET PRESS	>	27.81				0	30.9
		ENG RUN HOUR(HOUR)	>	844.20				0	938
		ENG RUN HOUR(MIN)	>	18.00				0	20
		T/C INLET EXH. GAS TEMP	>	567.00				450	580
		LBX CYL. Temp A5	>	470.70				0	523
		LBX CYL. Temp A6	>	395.10				0	439
		LBX CYL. Temp Main Bearing7	>	89.10				0	99
		LBX CYL. Temp Main Bearing8	>	90.90				0	101
		LBX CYL. Temp Main Bearing9	>	77.40				0	86
		T/C Inlet "A" A	>	540.00				0	600
	E72	L.O INLET PRESS	<	4.10				4	5
		H.T WATER OUTLET TEMP	<	71.50				70	85
		L.T INLET TEMP	<	31.00				30	40
		L.O INLET TEMP	<	61.00				60	70
		T/C INLET EXH. GAS TEMP	>	567.00				450	580

Figure 5.16: Pattern analysis (software)

Table 5.1: Extracted event logs (Equal-width binning)

	Event id 1	Event id 2	Event id 3	Event id 4	Event id 5
Failure 1	E231	E229	E222	E89	E87
Failure 2	E138	E124	E89	E88	E87
Failure 3	E178	E170	E157	E102	E100

Table 5.2: Extracted sensor information (Equal-width binning)

Number	Event ID	Sensor name	Comparison	Standard value
#1	E231	Starting air inlet pressure	>	20.6
		Fuel oil inlet pressure	>	8.35
		Charge air pressure	<	2
		High-temperature water outlet temperature	>	58
		Fuel oil inlet temperature	>	100.67
		Lube oil inlet temperature	>	50.67
		Gen. Bearing temperature	>	53.33
		Engine run hour (hour)	>	638.67
		Engine run hour (min)	<	19.69
		T/c inlet exh. Gas temperature	>	7.62E+08
		Cylinder. Temperature a1	>	306.89
		Cylinder. Temperature a2	>	301.61
		Cylinder. Temperature a3	>	304.6
		Cylinder. Temperature a4	>	303.61
		Cylinder. Temperature a5	>	347.92
		Cylinder. Temperature a6	>	291.73
		Cylinder. Temperature a7	>	344.8
		Cylinder. Temperature a8	>	348.49
		Cylinder. Temperature b1	>	323.07
		Cylinder. Temperature b2	>	343.07
		Cylinder. Temperature b3	>	330.04
		Cylinder. Temperature b4	>	342.28
		Cylinder. Temperature b5	>	333.33
		Cylinder. Temperature b6	>	339.21
		Cylinder. Temperature b7	>	304.43
		Cylinder. Temperature b8	>	327.55
		Cylinder. Temperature main bearing1	>	62.67
		Cylinder. Temperature main bearing2	<	126.33

Equal-width binning discretization is applied at first, and FP-Tree algorithm is applied to extract frequent patterns. The best result is displayed in Table 5.1. The longest pattern length is five, and the number of patterns is three. These extracted patterns contain much sensor information. For example, event id 231 contains sensor information which is described in Table 5.2. As shown in this table, certain sensor value is higher or lower than standard value. Other event ids also contain similar sensor information, and the failures occurred due to interaction of these sensor actions from this result.

When Entropy eventization technique is applied, many ranges of event logs are extracted. The maximum number of range is 42 in Turbo Charger Inlet Gas Temperature sensor. Based on these ranges, patterns can be extracted, and result is displayed in Table 5.3. Only two lengths is the longest pattern, and their sensor information is described in Table 5.3. This extracted pattern is the shortest pattern as compared with other eventization techniques because too many event logs are created in Entropy discretization. On the other hand, these events contain too much sensor information as displayed in Table 5.4.

Table 5.3: Extracted event logs (Entropy)

	Event id 1	Event id 2
Failure 1	E600	E263

Table 5.4: Extracted pattern information (Entropy)

Number	Event ID	Sensor name	Comparison	Standard value1	Comparison	Standard value2
#1	E600	Starting air inlet pressure	>	28.5	<	29.5
		Lube.oil inlet pressure	>	3.5	<	4.5
		Fuel.oil inlet pressure	>	3.5	<	4.5
		Charge air pressure	<	1.5		
		High-temperature water outlet temperature	>	52.5		
		Lube oil inlet temperature	>	39.5	<	40.5
		Fuel.oil inlet temperature**	>	39.5	<	40.5
		Lub.oil inlet temperature	>	54.5		
		Engine rpm	>	59.5		
		T/c rpm	>	16017.5		
		Generator Winding r phase temperature	>	55.5		
		Generator. Winding s phase temperature	>	56.5		
		Generator. Winding t phase temperature	>	55.5		
		Generator. Bearing temperature	>	51.5		
		Engine run hour(hour)	>	726.5		
		Engine run hour(min)	>	30.5	<	31.5
		T/c inlet. Gas temperature	>	1.04E+09		
		Cylinder. Temperature a1	>	68.5		
		Cylinder. Temperature a2	>	66.5		
		Cylinder. Temperature a3	>	66.5		
		Cylinder. Temperature a4	>	65.5		
		Cylinder. Temperature a5	>	65.5		
		Cylinder. Temperature a6	>	65.5		
		Cylinder. Temperature a7	>	68.5		
		Cylinder. Temperature a8	>	63.5		
		Cylinder. Temperature b1	>	66.5		
		Cylinder. Temperature b2	>	68.5		
		Cylinder. Temperature b3	>	70.5		
		Cylinder. Temperature b4	>	69.5		
		Cylinder. Temperature b5	>	68.5		

The third eventization technique is Knowledge expert-based discretization when normal percentage is 80. 80 normal percentage means that 20 percentages of ranges is not contributed event logs, and remained 50 percentages of ranges is distributed certain event logs. Number of the longest patterns is five as displayed in Table 5.5, and they contain much sensor information as displayed in Table 5.6. If the normal percentage is 100, sensors information is critically reduced. This percentage of normal state should be controlled properly from the system experts.

At the last, Estimation of probability distribution-based discretization technique is applied. As compared with other discretization techniques, this technique can get the longest pattern. The extracted pattern is displayed in Table 5.7, and they have only one pattern. These event ids which are containing a pattern have also much information, sensor information of only one event id is displayed in Table 5.8.

Table 5.5: Extracted patterns (Knowledge expert)

	Event id 1	Event id 2	Event id 3	Event id 4	Event id 5
Failure 1	E422	E46	E314	E288	E10

Table 5.6: Extracted sensor information (Knowledge expert)

Number	Event ID	Sensor name	Comparison	Standard value
#1	E422	Starting air inlet pressure	>	27.81
		Lube.oil inlet pressure	<	4.1
		Engine run hour(hour)	>	844.2
		Engine run hour(min)	>	18
		T/C inlet exh. Gas temp	>	567
		Lbx cylinder. Temp a6	>	395.1
		Cylinder. Temperature main bearing7	>	89.1
		Cylinder. Temperature main bearing8	>	90.9
		Cylinder. Temperature main bearing9	>	77.4
	E346	Starting air inlet press	>	27.81
		Low Temperature inlet temp	<	31
		Engine run hour(hour)	>	844.2
		Engine run hour(min)	>	18
		T/c inlet exh. Gas temp	>	567
		Cylinder. Temperature main bearing1	>	84.6
		Cylinder. Temperature main bearing9	>	77.4
	E314	Starting air inlet press	>	27.81
		Gen. Bearing temp	>	71.1
		Engine run hour(hour)	>	844.2
		Engine run hour(min)	>	18
		T/c inlet exh. Gas temp	>	567
		Cylinder. Temperature main bearing1	>	84.6
		Cylinder. Temperature main bearing9	>	77.4
	E288	Starting air inlet pressure	>	27.81
		Engine run hour(hour)	>	844.2
		Engine run hour(min)	>	18
		T/c inlet exh. Gas temp	>	567
	E10	Eng run hour(hour)	>	844.2
		T/c inlet exh. Gas temperature	>	567

Table 5.7: Extracted patterns (Estimation of probability distribution)

	Event id 1	Event id 2	Event id 3	Event id 4	Event id 5	Event id 6	Event id 7
Failure 1	E989	E967	E928	E573	E553	E483	E130

Table 5.8: Extracted sensor information (Estimation of probability distribution)

Number	Event ID	Sensor name	Comparison	Standard value
#1	E989	Starting air inlet pressure	<	27.11
		Fuel oil inlet pressure	>	5.25
		Charge air pressure	>	2.95
		High-temperature water outlet temperature	>	78.44
		Lube oil inlet temperature	<	34.73
		Fuel oil inlet temperature**	>	39.64
		Engine rpm	<	721.72
		T/c rpm	>	17085.84
		Generator. Winding r phase temperature	>	65.99
		Generator. Winding s phase temperature	>	66.28
		Generator. Winding t phase temperature	>	66.58
		Generator. Bearing temperature	>	68.77
		Engine run hour (hour)	>	864.51
		Engine run hour (min)	>	38.6
		Cylinder. Temperature a1	>	365.88
		Cylinder. Temperature a2	>	351.51
		Cylinder. Temperature a3	>	359.01
		Cylinder. Temperature a4	>	321.27
		Cylinder. Temperature a5	>	342.2
		Cylinder. Temperature a6	>	330.67
		Cylinder. Temperature a7	>	341.56
		Cylinder. Temperature a8	>	366.49
		Cylinder. Temperature b1	>	363.78
		Cylinder. Temperature b2	>	368.32
		Cylinder. Temperature b3	>	343.78
		Cylinder. Temperature b4	>	384.22
		Cylinder. Temperature b5	>	353.55
		Cylinder. Temperature b6	>	392.55
		Cylinder. Temperature b8	>	350.04
		Cylinder. Temperature main bearing1	>	80.19
		Cylinder. Temperature main bearing2	>	76.31
		Cylinder. Temperature main bearing3	>	81.63
		Cylinder. Temperature main bearing4	>	79.26
		Cylinder. Temperature main bearing5	>	81.05
		Cylinder. Temperature main bearing6	>	77.06
		Cylinder. Temperature main bearing7	>	83.89
		Cylinder. Temperature main bearing8	>	81.42
		Cylinder. Temperature main bearing9	>	74.49
		T/c inlet "a" a	>	442.35
		T/c inlet "b" a	>	431.96

As shown their results, different results are extracted along with different discretization techniques. Using this terminal box data, patterns are extracted while increasing the window size. The result is displayed in Figure 5.17. Estimation of probability distribution-based discretization has the best result because this technique considers the features of sensor as mentioned before. Entropy has the good result secondly; however, this technique has a disadvantage which is necessary objective attribute essentially. Knowledge expert-based discretization is the shortest pattern length because this technique depends on the percentage of normal state. Therefore, a proper discretization technique should be selected along with features of data.

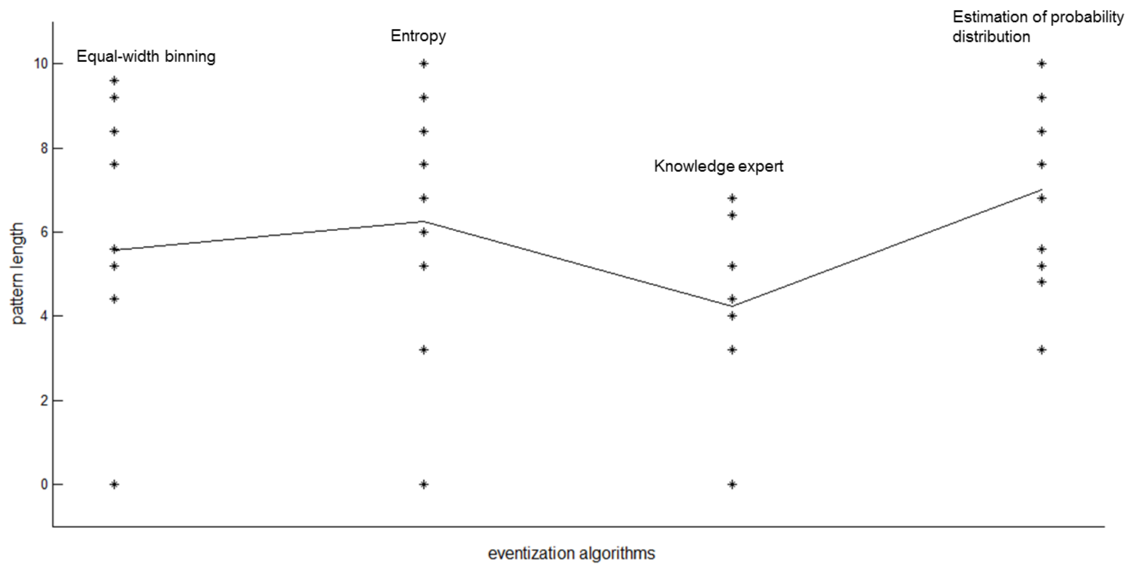
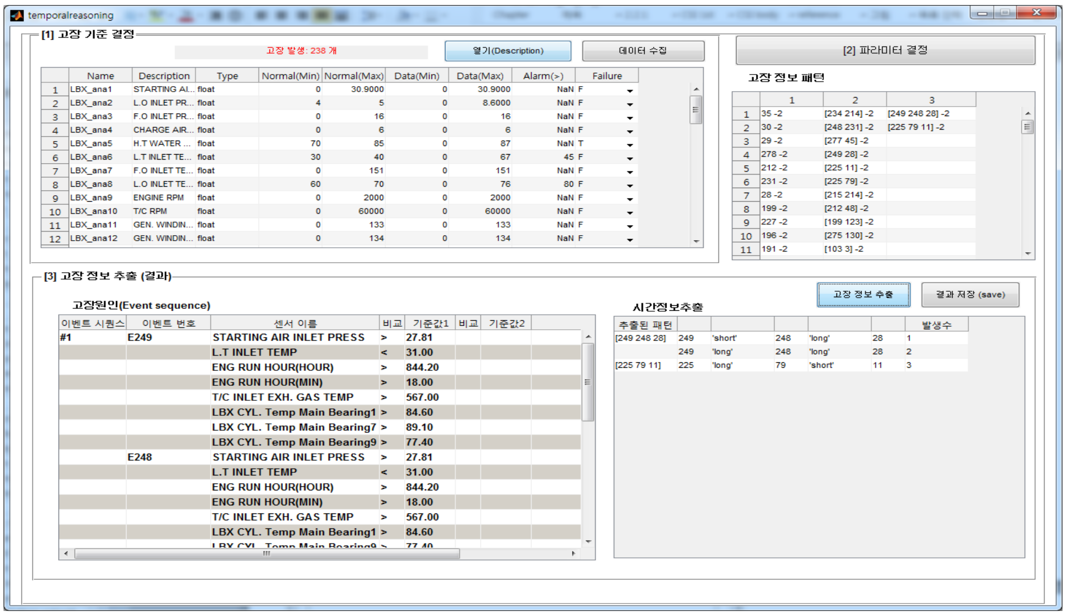


Figure 5.17: Pattern lengths – Four different eventization techniques

5.2.3 Temporal reasoning

Patterns are extracted by applying different window size to determine a suitable window size. Developed software is displayed in Figure 5.18. This software is similar as pattern analysis software. Based on these pattern analysis procedures, temporal reasoning technique is added, and this software gives time interval information of event logs. Some parameter options are also controlled using the other parameter control window. In this software, the threshold of time interval should be entered, and patterns are analyzed along this specified threshold. The result display both original extracted patterns and the information of time intervals.

In temporal reasoning phase, the time interval information is given based on extracted patterns in pattern extraction phases. The threshold of time interval is two, and then two different comparisons are extracted which are “short” and “long”. The results of temporal reasoning are displayed in Table 5.9, 10, and 11, and 12. The results of temporal reasoning have more information than already extracted patterns because extracted patterns have many combinations of event logs. Actually, the threshold of time interval is too small. If this threshold of time intervals is large, the many types of temporal reasoning result would be reduced.


temporalreasoning

[1] 고장 기준 결정

고장 발생: 238 개

열기(Description) | 데이터 수집

	Name	Description	Type	Normal(Min)	Normal(Max)	Data(Min)	Data(Max)	Alarm(>)	Failure
1	LBX_ana1	STARTING AI...	float	0	30.9000	0	30.9000	NaN	F
2	LBX_ana2	L.O INLET PR...	float	4	5	0	8.6000	NaN	F
3	LBX_ana3	F.O INLET PR...	float	0	16	0	16	NaN	F
4	LBX_ana4	CHARGE AIR...	float	0	6	0	6	NaN	F
5	LBX_ana5	H.T WATER ...	float	70	85	0	87	NaN	T
6	LBX_ana6	L.T INLET TE...	float	30	40	0	67	45	F
7	LBX_ana7	F.O INLET TE...	float	0	151	0	151	NaN	F
8	LBX_ana8	L.O INLET TE...	float	60	70	0	76	80	F
9	LBX_ana9	ENGINE RPM	float	0	2000	0	2000	NaN	F
10	LBX_ana10	T/C RPM	float	0	60000	0	60000	NaN	F
11	LBX_ana11	GEN. WINDIN...	float	0	133	0	133	NaN	F
12	LBX_ana12	GEN. WINDIN...	float	0	134	0	134	NaN	F

[2] 파라미터 결정

고장 정보 패턴

	1	2	3
1	35 -2	[234 214] -2	[249 248 28] -2
2	30 -2	[248 231] -2	[225 79 11] -2
3	29 -2	[277 45] -2	
4	278 -2	[249 28] -2	
5	212 -2	[225 11] -2	
6	231 -2	[225 79] -2	
7	28 -2	[215 214] -2	
8	199 -2	[212 48] -2	
9	227 -2	[199 123] -2	
10	196 -2	[275 130] -2	
11	191 -2	[103 3] -2	

[3] 고장 정보 추출 (결과)

고장 정보 추출 | 결과 저장 (save)

고장원인(Event sequence)

이벤트 시퀀스	이벤트 번호	센서 이름	비교	기준값1	비교	기준값2
#1	E249	STARTING AIR INLET PRESS	>	27.81		
		L.T INLET TEMP	<	31.00		
		ENG RUN HOUR(HOUR)	>	844.20		
		ENG RUN HOUR(MIN)	>	18.00		
		T/C INLET EXH. GAS TEMP	>	567.00		
		LBX CYL. Temp Main Bearing1	>	84.60		
		LBX CYL. Temp Main Bearing7	>	89.10		
		LBX CYL. Temp Main Bearing9	>	77.40		
	E248	STARTING AIR INLET PRESS	>	27.81		
		L.T INLET TEMP	<	31.00		
		ENG RUN HOUR(HOUR)	>	844.20		
		ENG RUN HOUR(MIN)	>	18.00		
		T/C INLET EXH. GAS TEMP	>	567.00		
		LBX CYL. Temp Main Bearing1	>	84.60		
		LBX CYL. Temp Main Bearing9	>	77.40		

시간정보추출

추출된 패턴						발생수
[249 248 28]	249	'short'	248	'long'	28	1
	249	'long'	248	'long'	28	2
[225 79 11]	225	'long'	79	'short'	11	3

Figure 5.18: Temporal reasoning (software)

Table 5.9: The result of temporal reasoning (Equal-width binning)

Patterns	Event id 1		Event id1 2		Event id 3		Event id 4		Event id 5
[231 229 222 89 87]	231	'short'	229	'long'	222	'long'	89	'long'	87
	231	'long'	229	'short'	222	'long'	89	'long'	87
	231	'long'	229	'long'	222	'long'	89	'long'	87
	231	'short'	229	'short'	222	'long'	89	'long'	87
	231	'short'	229	'long'	222	'long'	89	'short'	87
[138 124 89 88 87]	138	'short'	124	'short'	89	'short'	88	'short'	87
	138	'long'	124	'long'	89	'long'	88	'long'	87
	138	'long'	124	'long'	89	'long'	88	'short'	87
	138	'long'	124	'long'	89	'short'	88	'long'	87
	138	'long'	124	'long'	89	'short'	88	'short'	87
	138	'short'	124	'long'	89	'long'	88	'short'	87
	138	'short'	124	'long'	89	'long'	88	'long'	87
[178 170 157 102 100]	178	'short'	170	'short'	157	'long'	102	'short'	100
	178	'short'	170	'long'	157	'short'	102	'short'	100
	178	'short'	170	'short'	157	'short'	102	'short'	100
	178	'long'	170	'long'	157	'short'	102	'long'	100
	178	'long'	170	'long'	157	'short'	102	'short'	100
	178	'short'	170	'long'	157	'long'	102	'short'	100
	178	'short'	170	'long'	157	'long'	102	'long'	100

Table 5.10: The result of temporal reasoning (Entropy)

Patterns	Event id 1		Event id1 2
[600 263]	600	'short'	263

Table 5.11: The result of temporal reasoning (Knowledge expert)

Patterns	Event id 1		Event id 2		Event id 3		Event id 4		Event id 5
[422 346 314 288 10]	422	'long'	46	'long'	314	'long'	288	'long'	10
	422	'long'	46	'long'	314	'long'	288	'short'	10

Table 5.12: The result of temporal reasoning (Estimation of probability distribution)

Patterns	Event id 1		Event id 2		Event id 3		Event id 4		Event id 5		Event id 6		Event id 7
[989 967 928	989	'long'	967	'long'	928	'long'	573	'long'	553	'long'	483	'long'	130
573 553 483 130]	989	'short'	967	'long'	928	'long'	573	'long'	553	'long'	483	'long'	130

5.3 Parameter specifications

In Chapter 4, the best result can be extracted when the range of window size is from 50 to 70. The result which applied these window sizes is displayed in Figure 5.19. Figure 5.19shows that four different eventization techniques are applied while window size is increasing. As described, the longest pattern is in window size 50, and the lowest pattern is in window size 10 when the same weighting percentages are given. Based on a suitable window size, more good result can be extracted.

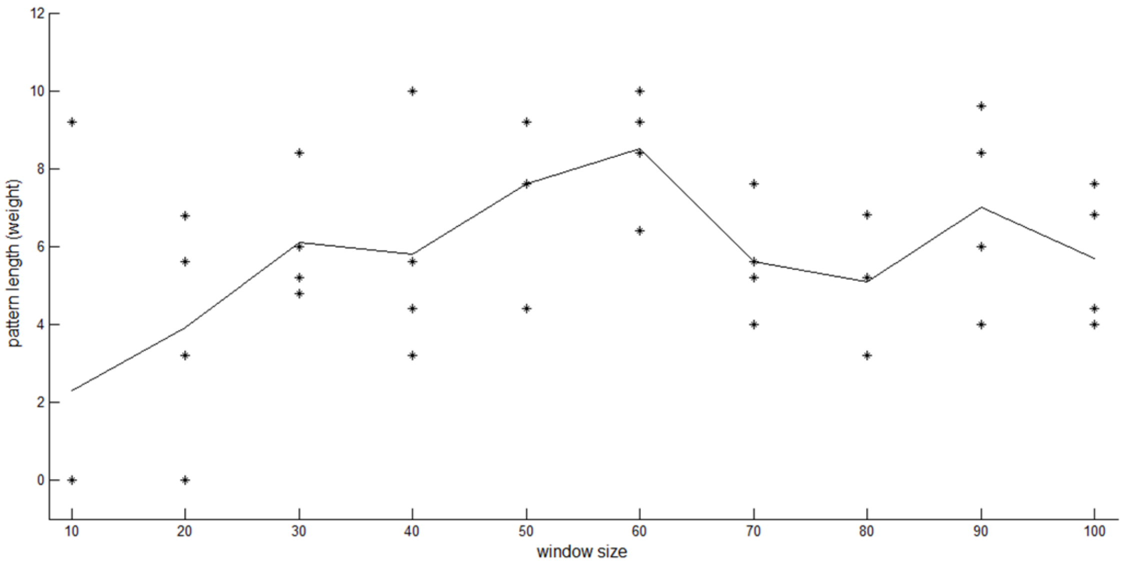
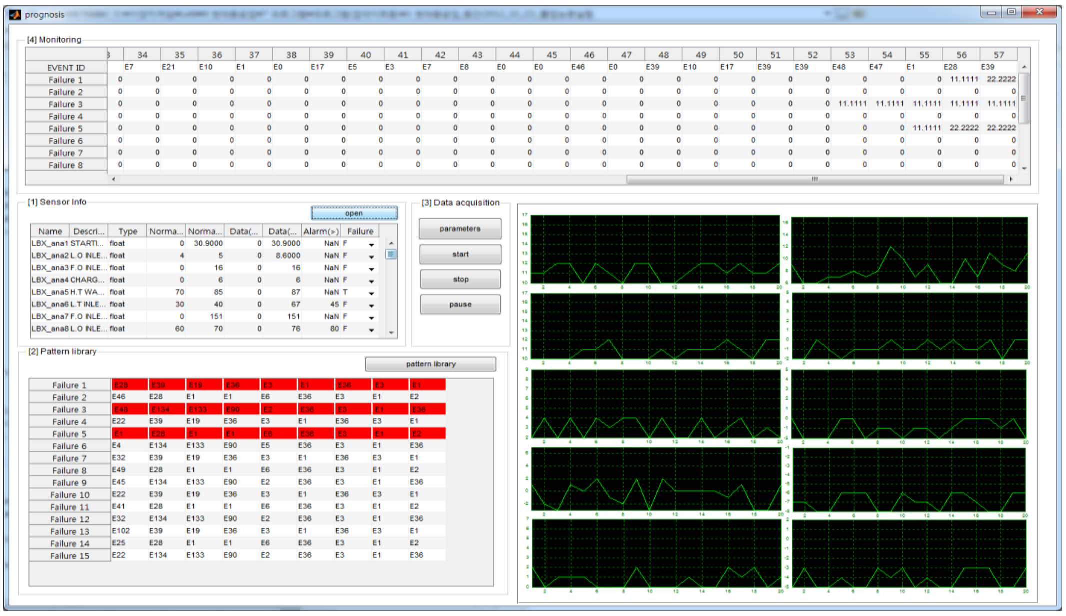


Figure 5.19: Pattern lengths – Window size of terminal box data

5.4 Prognosis of terminal box data

The result which is extracted in pattern analysis phase are stored pattern library. After pattern analysis procedures, the last phase is failure prognosis. Developed software is displayed in Figure 5.20. Extensively, there are two parts which are pattern library data and monitoring data in prognosis software and monitoring signals are also displaying in real-time. Only one parameter is needed for prognosis procedure and that is warning condition. This parameter is specified as percentage.

To do experiment using terminal box data, random data is generating as monitoring data. One random event id is generating per second, and then these event ids are matched with pattern library event ids. Patterns are stored along the certain failure types when extracted patterns are stored in pattern library. Monitoring event ids are compared with these stored patterns which are ordered along the failure types, and then the probability of failure occurrences is calculated. In this terminal box data, warning condition is 80%; however, there is no warning. This problem is because monitoring data is using the random data. Therefore, warning condition is changed as 20%, and then some warnings are occurred. If red window is occurred in pattern library, the certain failure would occur. From these procedures, failures can be predicted, and a proper action should be provided.



[4] Monitoring

	3	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53	54	55	56	57
EVENT ID		E7	E21	E10	E1	E0	E17	E5	E3	E7	E8	E0	E0	E46	E0	E39	E10	E17	E39	E39	E48	E47	E1	E28	E39
Failure 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11.1111	22.2222
Failure 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Failure 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11.1111	11.1111	11.1111	11.1111	11.1111
Failure 4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Failure 5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11.1111	22.2222	22.2222
Failure 6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Failure 7	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Failure 8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

[1] Sensor Info

Name	Descri...	Type	Norma...	Norma...	Data(...	Data(...	Alarm(>)	Failure
LBX_ana1	STARTI...	float	0	30.9000	0	30.9000	NaN	F
LBX_ana2	L.O INLE...	float	4	5	0	8.6000	NaN	F
LBX_ana3	F.O INLE...	float	0	16	0	16	NaN	F
LBX_ana4	CHARG...	float	0	6	0	6	NaN	F
LBX_ana5	H.T WA...	float	70	85	0	87	NaN	T
LBX_ana6	L.T INLE...	float	30	40	0	67	45	F
LBX_ana7	F.O INLE...	float	0	151	0	151	NaN	F
LBX_ana8	L.O INLE...	float	60	70	0	76	80	F

[3] Data acquisition

[2] Pattern library

Failure 1	E28	E39	E19	E36	E3	E1	E36	E3	E1
Failure 2	E46	E28	E1	E1	E6	E36	E3	E1	E2
Failure 3	E48	E134	E133	E90	E2	E36	E3	E1	E36
Failure 4	E22	E39	E19	E36	E3	E1	E36	E3	E1
Failure 5	E1	E28	E1	E1	E6	E36	E3	E1	E2
Failure 6	E4	E134	E133	E90	E5	E36	E3	E1	E36
Failure 7	E32	E39	E19	E36	E3	E1	E36	E3	E1
Failure 8	E49	E28	E1	E1	E6	E36	E3	E1	E2
Failure 9	E45	E134	E133	E90	E2	E36	E3	E1	E36
Failure 10	E22	E39	E19	E36	E3	E1	E36	E3	E1
Failure 11	E41	E28	E1	E1	E6	E36	E3	E1	E2
Failure 12	E32	E134	E133	E90	E2	E36	E3	E1	E36
Failure 13	E102	E39	E19	E36	E3	E1	E36	E3	E1
Failure 14	E25	E28	E1	E1	E6	E36	E3	E1	E2
Failure 15	E22	E134	E133	E90	E2	E36	E3	E1	E36

Figure 5.20: Failure prognosis (software)

VI. Conclusion and Future research

A framework for event-driven failure analysis was proposed, through which failure patterns can be efficiently analyzed from a large data set. In particular, a systematic eventization procedure was developed which makes it possible to reduce the original data size into a manageable one in the form of event logs, and eventually to extract failure patterns efficiently from the reduced data.

The proposed failure analysis procedure consists of five main phases: data cleaning, reduction, failure pattern analysis and failure prognosis. For the data cleaning phase, Kalman filter is adopted to handle noise and redundant values of raw data, and then PCA and correlation analysis are conducted to reduce the dimension of data. The four eventization strategies (Equal-width binning, Entropy, Knowledge expert-based eventization and Estimation of probability distribution-based eventization) are examined, and performance comparisons are made. Failure pattern analysis was accelerated when we used 'probability distribution-based eventization'. Failure patterns are extracted in the form of event sequences by sequence-mining algorithms, (e.g. FP-Tree algorithm). Extracted patterns are stored in a failure pattern library, and eventually, we use the stored failure pattern information to predict potential failures. The two practical case studies (marine diesel engine and SIRIUS-Ⅱ car engine) provide empirical support for the performance of the proposed failure analysis procedure. This procedure can be easily extended for wide application fields of failure analysis such as vehicle and machine diagnostics. Furthermore, it can be applied to human health monitoring & prognosis, so that human body signals could be efficiently analyzed.

The directions of future research can be summarized as follows: first, intelligent (adaptive) discretization strategies should be developed with respect to the characteristics of sensor signals. Second, the Entropy-based discretization method must be tuned appropriately for the case of rare occurrences of failures. Finally, the completeness and the soundness of the proposed failure analysis must be analyzed along with more empirical evidence.

Reference

1. Agrawal, R., T. Imieliński and A. Swami (1993). "Mining association rules between sets of items in large databases". Proceedings of the 1993 ACM SIGMOD international conference on Management of data, ACM.
2. Agrawal, R. and R. Srikant (1994). "Fast algorithms for mining association rules". Proceedings of the 20th International Conference on Very Large Data Bases.
3. Agrawal, R. and R. Srikant (1995). "Mining sequential patterns". Proceedings of the 5th International Conference on Extending Database Technology, IEEE.
4. Ale, J. M. and G. H. Rossi (2000). "An approach to discovering temporal association rules", ACM.
5. Allison, P. D. (2002). "Missing data: Quantitative applications in the social sciences." British Journal of Mathematical and Statistical Psychology 55: 193-196.
6. Angeli, C. (2010). "Diagnostic Expert Systems: From Expert's Knowledge to Real-Time Systems." Advanced Knowledge Based Systems: Model, Applications & Research, Eds. Sajja & Akerkar 1: 50-73.
7. Bauer, B., D. George, B. Geropp and M. Poschmann (1996). "Lathe tool wear monitoring with neural networks". In proceeding of European Congress on Fuzzy and Intelligent Techniques, Germany.
8. Bay, S. D. (2001). "Multivariate discretization for set mining." Knowledge and Information Systems 3(4): 491-512.
9. Bayardo Jr, R. J. (1998). "Efficiently mining long patterns from databases". ACM SIGMOD Record, ACM.
10. Becraft, W. R. and P. L. Lee (1993). "An integrated neural network/expert system approach for fault diagnosis." Computers & chemical engineering 17(10): 1001-1014.
11. Benson, G. and M. S. Waterman (1994). "A method for fast database search for all k-nucleotide repeats." Nucleic Acids Research 22(22): 4828-4836.
12. Biagetti, T. and E. Sciubba (2004). "Automatic diagnostics and prognostics of energy conversion processes via knowledge-based systems." Energy 29(12): 2553-2572.
13. Bingzhen Chen, J. Z. and J. Shen (1997). "A hybrid ANN-ES system for dynamic fault diagnosis of hydrocracking process." Computers & chemical engineering 21: S929-S933.
14. Boulle, M. (2005). "Optimal bin number for equal frequency discretizations in supervized learning." Intelligent Data Analysis 9(2): 175-188.
15. Brāzma, A., I. Jonassen, J. Vilo and E. Ukkonen (1998). "Pattern discovery in biosequences." Grammatical Inference: 257-270.
16. Brin, S., R. Motwani and C. Silverstein (1997). "Beyond market baskets: Generalizing association rules to correlations". Proceedings of the 1997 ACM SIGMOD international conference on Management of data ACM.
17. Brotherton, T., G. Jahns, J. Jacobs and D. Wroblewski (2000). "Prognosis of faults in gas turbine engines", IEEE.
18. Byington, C. S., M. Watson and D. Edwards (2004). "Data-driven neural network methodology to remaining life predictions for aircraft actuator components". Proceeding of IEEE Aerospace Conference Ieee.
19. Chen, B. H., X. Z. Wang, S. H. Yang and C. McGreavy (1999). "Application of wavelets and neural networks to diagnostic system development, 1, feature extraction." Computers &amp; Chemical Engineering 23(7): 899-906.
20. Chen, Y. L., M. C. Chiang and M. T. Ko (2003). "Discovering time-interval sequential patterns in sequence databases." Expert Systems with Applications 25(3): 343-354.
21. Chen, Y. L. and T. C. K. Huang (2005). "Discovering fuzzy time-interval sequential patterns in sequence databases." Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 35(5): 959-972.
22. Cheung, J. T. Y. and G. Stephanopoulos (1990). "Representation of process trends—Part I. A formal representation framework." Computers & chemical engineering 14(4): 495-510.
23. Cios, K. J., W. Pedrycz, R. Świniarski and R. Swiniarski (1998). "Data mining methods for knowledge discovery", Springer.

24. Cool, A. L. (2000). "A review of methods for dealing with missing data". Educational Research and Evaluation.
25. Das, G., K. I. Lin, H. Mannila, G. Renganathan and P. Smyth (1998). "Rule discovery from time series." Knowledge Discovery and Data Mining: 16-22.
26. de Queiroz Souza, R. and A. J. Á lvares (2008). "FMEA and FTA Analysis for Application of the Reliability-Centered Maintenance Methodology: Case Study on Hydraulic Turbines". Proceedings of COBEM.
27. Derksen, S. and H. Keselman (1992). "Backward, forward and stepwise automated subset selection algorithms: Frequency of obtaining authentic and noise variables." British Journal of Mathematical and Statistical Psychology 45(2): 265-282.
28. Dougherty, J., R. Kohavi and M. Sahami (1995). "Supervised and unsupervised discretization of continuous features". International conference on machine learning, Morgan Kaufmann Publishers, Inc.
29. Efendic, H. and L. Del Re (2005). "Iterative multilayer fault diagnosis approach for complex systems". ICECS'05 Proceedings of the 4th WSEAS international conference on Electronics, control and signal processing World Scientific and Engineering Academy and Society (WSEAS).
30. Elomaa, T. and J. Rousu (2004). "Efficient multisplitting revisited: Optima-preserving elimination of partition candidates." Data Mining and Knowledge Discovery 8(2): 97-126.
31. Escobet, T. and L. Travé-Massuyès (2001). "Parameter estimation methods for fault detection and isolation." Control System Technology 40.
32. Fujimaki, R., T. Yairi and K. Machida (2005). "An anomaly detection method for spacecraft using relevance vector learning." Advances in Knowledge Discovery and Data Mining: 785-790.
33. Galton, F. (1886). "Regression towards mediocrity in hereditary stature." The Journal of the Anthropological Institute of Great Britain and Ireland 15: 246-263.
34. Gelman, A., J. B. Carlin, H. S. Stern and D. B. Rubin (2004). "Bayesian data analysis", CRC press.
35. Gertler, J. (1991). "Analytical redundancy methods in fault detection and isolation". IFAC-Symposium SAFEPROCESS.
36. Gertler, J. (1998). "Fault detection and diagnosis in engineering systems", CRC.
37. Gertler, J. and D. Singer (1990). "A new structural framework for parity equation-based failure detection and isolation." Automatica 26(2): 381-388.
38. Grahne, G. and J. Zhu (2005). "Fast algorithms for frequent itemset mining using fp-trees." Knowledge and Data Engineering, IEEE Transactions on 17(10): 1347-1362.
39. Grewal, M. S. and A. P. Andrews (2008). "Kalman Filtering: Theory and Practice Using MATLAB", Wiley.
40. Grossmann, A., R. Kronland-Martinet, J. Morlet, J. Combes, A. Grossman and P. Tchamitchian (1989). "Reading and understanding continuous wavelet transforms, Wavelets: Time-Frequency Methods and Phase Space."
41. Guyon, I. and A. Elisseeff (2003). "An introduction to variable and feature selection." The Journal of Machine Learning Research 3: 1157-1182.
42. Han, J. (2006). "Data mining concepts and techniques". Elsevier ;San Francisco CA : Morgan Kaufmann, Amsterdam ;Boston.
43. Han, J., H. Cheng, D. Xin and X. Yan (2007). "Frequent pattern mining: current status and future directions." Data Mining and Knowledge Discovery 15(1): 55-86.
44. Han, J., J. Pei, B. Mortazavi-Asl, H. Pinto, Q. Chen, U. Dayal and M. Hsu (2001). "PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth". ICDE '01 Proceedings of the 17th International Conference on Data Engineering Citeseer.
45. Han, J., J. Pei, Y. Yin and R. Mao (2004). "Mining frequent patterns without candidate generation: A frequent-pattern tree approach." Data Mining and Knowledge Discovery 8(1): 53-87.
46. Hao, L., Y. Jinsong, Z. Ping and L. Xingshan (2009). "A review on fault prognostics in integrated health management". The Ninth International Conference on Electronic Measurement and Instruments, IEEE.
47. Hatcher, L. (1994). "A step-by-step approach to using the SAS system for factor analysis and structural equation modeling", SAS Publishing.

48. Hayes-Roth, F., D. A. Waterman and D. B. Lenat (1983). "An overview of expert systems." Building expert systems 1.
49. Hellerstein, J. M., W. Hong, S. Madden and K. Stanek (2003). "Beyond average: Toward sophisticated sensing with queries". IPSN'03 Proceedings of the 2nd international conference on Information processing in sensor networks Springer-Verlag.
50. Ho, K. and P. Scott (1997). "Zeta: A Global Method for Discretization of Cotitinuous Variables."
51. Hong, T. P., C. W. Lin and Y. L. Wu (2008). "Incrementally fast updated frequent pattern trees." Expert Systems with Applications 34(4): 2424-2435.
52. Houtsma, M. and A. Swami (1995). "Set-oriented mining for association rules in relational databases". ICDE '95 Proceedings of the Eleventh International Conference on Data IEEE.
53. Isermann, R. (2006). "Fault-Diagnosis Systems an Introduction from Fault Detection to Fault Tolerance". Berlin, Heidelberg, Springer-Verlag Berlin Heidelberg.
54. Jardine, A. K. S., D. Lin and D. Banjevic (2006). "A review on machinery diagnostics and prognostics implementing condition-based maintenance." Mechanical Systems and Signal Processing 20(7): 1483-1510.
55. Jeffery, S., G. Alonso, M. Franklin, W. Hong and J. Widom (2006). "Declarative support for sensor data cleaning." Pervasive Computing: 83-100.
56. Jiang, N. and Z. Chen (2010). "Model-driven data cleaning for signal processing system in sensor networks". Signal Processing Systems (ICSPS), IEEE.
57. Jiang, S., X. Li, Q. Zheng and L. Wang (2009). "Approximate equal frequency discretization method". GCIS '09 Proceedings of the 2009 WRI Global Congress on Intelligent Systems, IEEE.
58. Kerber, R. (1992). "Chimerge: Discretization of numeric attributes". AAAI'92 Proceedings of the tenth national conference on Artificial intelligence AAAI Press.
59. Kim, K. O. and M. J. Zuo (2007). "Two fault classification methods for large systems when available data are limited." Reliability Engineering & System Safety 92(5): 585-592.
60. Kivikunnas, S. (1998). "Overview of process trend analysis methods and applications". ERUDIT Workshop on Applications in Pulp and Paper Industry, page CD ROM, Citeseer.
61. Konstantinov, K. B. and T. Yoshida (1992). "Real-time qualitative analysis of the temporal shapes of (bio) process variables." AIChE Journal 38(11): 1703-1715.
62. Koo, I. S. and W. W. Kim (2000). "The development of reactor coolant pump vibration monitoring and a diagnostic system in the nuclear power plant." ISA transactions 39(3): 309-316.
63. Kotsiantis, S. and D. Kanellopoulos (2006). "Discretization techniques: A recent survey." GESTS International Transactions on Computer Science and Engineering 32(1): 47-58.
64. Kurgan, L. A. and K. J. Cios (2004). "CAIM discretization algorithm." Knowledge and Data Engineering, IEEE Transactions on 16(2): 145-153.
65. Lendasse, A., E. Oja, O. Simula and M. Verleysen (2007). "Time series prediction competition: The CATS benchmark." Neurocomputing 70(13-15): 2325-2329.
66. Leonard, J. A. and M. A. Kramer (1991). "Radial basis function networks for classifying process faults." Control Systems Magazine, IEEE 11(3): 31-38.
67. Leonhardt, S. and M. Ayoubi (1997). "Methods of fault diagnosis." Control engineering practice 5(5): 683-692.
68. Liang, Y., Y. Zhang, H. Xiong and R. Sahoo (2007). "Failure prediction in ibm bluegene/l event logs". ICDM '07 Proceedings of the 2007 Seventh IEEE International Conference on Data Mining IEEE.
69. Lin, J. and L. Qu (2000). "Feature extraction based on morlet wavelet and its application for mechanical fault diagnosis." Journal of Sound and Vibration 234(1): 135-148.
70. Little, R. J. A. and D. B. Rubin (1987). "Statistical analysis with missing data", Wiley New York.
71. Liu, H., F. Hussain, C. Tan and M. Dash (2002). "Discretization: An enabling technique." Data Mining and Knowledge Discovery 6(4): 393-423.
72. Liu, H. and R. Setiono (1995). "Chi2: Feature selection and discretization of numeric attributes". In Proceedings of the Seventh International Conference on Tools with Artificial Intelligence, IEEE.
73. Liu, H. and L. Yu (2005). "Toward integrating feature selection algorithms for classification and clustering." IEEE Transactions on Knowledge and Data Engineering 17(4): 491-502.

74. Lu, C.-J. and Y.-T. Hsu (2000). "Application of wavelet transform to structural damage detection " Shock and Vibration Digest 50.
75. Luo, M. (2006). Data-driven fault detection using trending analysis, [Baton Rouge, La. : Louisiana State University.
76. Ma, S. and J. L. Hellerstein (2002). "Mining mutually dependent patterns for system management." Selected Areas in Communications, IEEE Journal on 20(4): 726-735.
77. Mannila, H., H. Toivonen and A. Inkeri Verkamo (1997). "Discovery of frequent episodes in event sequences." Data Mining and Knowledge Discovery 1(3): 259-289.
78. Maree, J. P. (2009). Fault Detection for the Benfield Process Using a Closed-loop Subspace Re-identification Approach, University of Pretoria.
79. McInerny, S. and Y. Dai (2003). "Basic vibration signal processing for bearing fault detection." Education, IEEE Transactions on 46(1): 149-156.
80. Miljkovic, D. (2011). "Fault detection methods: A literature survey". MIPRO, 2011 Proceedings of the 34th International Convention, IEEE.
81. Mitsa, T. (2009). "Temporal data mining", CRC Press.
82. Naik and Amol.S. (2010). Subspace Based Data-driven Designs of Fault Detection Systems, University of Duisburg-Essen**:** 1-128.
83. Namburu, S. M., S. Chigusa, D. Prokhorov, L. Qiao, K. Choi and K. Pattipati (2007). "Application of an effective data-driven approach to real-time fault diagnosis in automotive engines". Aerospace Conference.
84. Naughton, J., Y. Chen and J. Jiang (1996). "A Neural Network Application to Fault Diagnosis for Robotic Manipulato". Proceedings of the 1996 IEEE International Conference, IEEE.
85. Ortmeier, F. and G. Schellhorn (2007). "Formal fault tree analysis-practical experiences." Electronic Notes in Theoretical Computer Science 185: 139-151.
86. Peng, Z. K. and F. L. Chu (2004). "Application of the wavelet transform in machine condition monitoring and fault diagnostics: a review with bibliography." Mechanical Systems and Signal Processing 18(2): 199-221.
87. Pouliezos, A., G. Stavrakakis and C. Lefas (1989). "Fault detection using parameter estimation." Quality and reliability engineering international 5(4): 283-290.
88. Price, C. and N. Taylor (2002). "Automated multiple failure FMEA." Reliability Engineering & System Safety 76(1): 1-10.
89. QIN, J. and H. SONG (2004). "Study and Improvement of AprioriHybrid Algorithm in Mining Association Rules [J]." Computer Engineering 17.
90. Rabatel, J., S. Bringay and P. Poncelet (2009). SO-MAD: SensOr mining for anomaly detection in railway data. 5633 LNAI**:** 191-205.
91. Rahm, E. and H. H. Do (2000). "Data cleaning: Problems and current approaches." IEEE Data Engineering Bulletin 23(4): 3-13.
92. Rausand, M. and A. Hoylanc (2004). "System Reliability Theory: Models and Statistical Method>." Annals of Statistics 6: 701-726.
93. Richard, O. D., E. H. Peter and G. S. David (2001). "Pattern classification." A Wiley-Interscience Publication 373378.
94. Rodrigues, P. P. and J. Gama (2006). "Online prediction of streaming sensor data". Proceedings of the 3rd international workshop on knowledge discovery from data streams (IWKDDS).
95. Rubin, D. B. (1987). "Multiple imputation for nonresponse in surveys", Wiley Online Library.
96. Rubin, D. B. (1996). "Multiple imputation after 18+ years." Journal of the American Statistical Association: 473-489.
97. Rumelhart, D. E. and J. L. McClelland (1986). "Parallel distributed processing: Psychological and biological models", The MIT press.
98. Sakurai, M. "Impact of Toyota Recall on Corporate Reputation."
99. Samy, I. and D. W. Gu (2011). "Fault Detection and Isolation (FDI)." Fault Detection and Flight Data Measurement: 5-17.
100. Sarkar, S., X. Jin and A. Ray (2011). "Data-Driven Fault Detection in Aircraft Engines With Noisy Sensor Measurements." Journal of Engineering for Gas Turbines and Power 133: 081602.
101. Schwabacher, M. (2005). "A survey of data-driven prognostics".

102. Shlens, J. (2005). "A tutorial on principal component analysis." Systems Neurobiology Laboratory, University of California at San Diego.
103. Siciliano, R. and C. Conversano (2006). "Decision tree induction." Encyclopedia of data warehousing and mining, Idea Group, USA: 353.
104. Srikant, R. and R. Agrawal (1996). "Mining sequential patterns: Generalizations and performance improvements." Advances in Database Technology—EDBT'96: 1-17.
105. Stearns, S. D. and D. R. Hush (1990). "Digital signal analysis." Englewood Cliffs, NJ, Prentice Hall, 1990, 460 p. 1.
106. Subramaniam, S., T. Palpanas, D. Papadopoulos, V. Kalogeraki and D. Gunopulos (2006). "Online outlier detection in sensor data using non-parametric models". VLDB '06 Proceedings of the 32nd international conference on Very large data bases, VLDB Endowment.
107. Sui, Y., F. J. Shao, R. C. Sun and J. L. Wang (2008). "A sequential pattern mining algorithm based on improved FP-tree". SNPD '08 Proceedings of the 2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing IEEE.
108. Tan, P. N., M. Steinbach and V. Kumar (2005). "Association Analysis: Basic Concepts and Algorithms." Introduction to Data Mining: 327-414.
109. Tan, P. N., M. Steinbach and V. Kumar (2006). "Introduction to Data Mining", Posts & Telecom Press.
110. Tsai, C. J., C. I. Lee and W. P. Yang (2008). "A discretization algorithm based on Class-Attribute Contingency Coefficient." Information Sciences 178(3): 714-731.
111. Vatn, J. (1992). "Finding minimal cut sets in a fault tree." Reliability Engineering & System Safety 36(1): 59-62.
112. Vedam, H. and V. Venkatasubramanian (1997). "A wavelet theory-based adaptive trend analysis system for process monitoring and diagnosis". American Control Conference, IEEE.
113. Venkatasubramanian, V. and K. Chan (1989). "A neural network methodology for process fault diagnosis." AIChE Journal 35(12): 1993-2002.
114. Venkatasubramanian, V., R. Rengaswamy, S. N. Kavuri and K. Yin (2003). "A review of process fault detection and diagnosis:: Part III: Process history based methods." Computers & chemical engineering 27(3): 327-346.
115. Verma, K. and O. Vyas (2005). "Efficient calendar based temporal association rule." ACM SIGMOD Record 34(3): 63-70.
116. Vijayakumar, N. and B. Plale (2007a). "Prediction of missing events in sensor data streams using kalman filters". 1st Int'l Workshop on Knowledge Discovery from Sensor Data, in conjunction with ACM 13th Int'l Conference on Knowledge Discovery and Data Mining.
117. Vijayakumar, N. and B. Plale (2007b). "Prediction of missing events in sensor data streams using kalman filters".
118. Wang, H., T.-Y. Chai, J.-L. Ding and M. Brown (2009). "Data Driven Fault Diagnosis and Fault Tolerant Control: Some Advances and Possible New Directions." Acta Automatica Sinica 35(6): 739-747.
119. Wang, P. and G. Vachtsevanos (2001). "Fault prognostics using dynamic wavelet neural networks." Artificial Intelligence for Engineering Design, Analysis and Manufacturing 15(4): 349-365.
120. Welch, G. and G. Bishop (1995). "An introduction to the Kalman filter." University of North Carolina at Chapel Hill, Chapel Hill, NC 7(1).
121. Wilkinson, L. (1999). "the Task Force on Statistical Inference, American Psychological Association Board of Scientific Affairs." Statistical methods in psychology journals 54: 594-604.
122. Witten, I. H., E. Frank and M. A. Hall (2011). "Data Mining: Practical machine learning tools and techniques", Morgan Kaufmann.
123. Wowk, V. (1991). "Machinery vibration: measurement and analysis", McGraw-Hill Professional.
124. Wu, J. D. and C. H. Liu (2009). "An expert system for fault diagnosis in internal combustion engines using wavelet packet transform and neural network." Expert Systems with Applications 36(3): 4278-4286.

125. Xue, W., Q. Luo, L. Chen and Y. Liu (2006). "Contour map matching for event detection in sensor networks". SIGMOD '06 Proceedings of the 2006 ACM SIGMOD international conference on Management of data ACM.
126. Yairi, T., Y. Kato and K. Hori (2001). "Fault detection by mining association rules from house-keeping data". In Proceedings of International Symposium on Artificial Intelligence, Robotics and Automation in Space, Citeseer.
127. Yang, J., W. Wang, P. S. Yu and J. Han (2002). "Mining long sequential patterns in a noisy environment". Proceedings of the 2002 ACM SIGMOD international conference on Management of data, ACM.
128. Zaki, M. J. (2001). "SPADE: An efficient algorithm for mining frequent sequences." Machine Learning 42(1): 31-60.
129. Zhuangt, Y., L. Chen, X. S. Wang and J. Lian (2007a). "A weighted moving average-based approach for cleaning sensor data", IEEE.
130. Zhuangt, Y., L. Chen, X. S. Wang and J. Lian (2007b). "A weighted moving average-based approach for cleaning sensor data". ICDCS '07 Proceedings of the 27th International Conference on Distributed Computing Systems IEEE.

Acknowledgement

First of all, I would like to express the deepest appreciation to my committee chair, Professor Duck-Young Kim, for his constant guidance and support. Without his inspirational guidance, enthusiasm, and encouragements to accomplish this research, I would not be able to finish my research. I also want to show my sincere gratitude to my other committee members, Gwan-Seob Shin and Min-Seok Song, for taking their precious time to consider my work. I would like to thank all people in the UNIST who assisted me by providing technical support and checking of this thesis.

In addition, I thanks to my CSI lab member, Hyun-Ki, Amit, Prerna, and Su-Jung who work together for two years. I would especially like to thanks Hyun-Ki for his support and kindness. I also thanks to all DHE member, Hae-Seung, Young-Gwang, Moise, Aream and all other members. They always help me in give the enjoyable studying environment and their friendships are invaluable to me. My grateful thanks also go to the other friends (too many to list here but you know you are) for the friendship and memories.

Finally, and most importantly, I would like to thank my family members, father, mother, and brother, for supporting to pursue this master degree. This thesis would not have been possible without the help and support. Their endless love and support made it possible to complete this thesis. Thanks for giving me opportunity to take master degree.