Institutionen för systemteknik
Department of Electrical Engineering

Examensarbete

Investigation of a troubleshooting procedure
By assessing fault tracing algorithms

Examensarbete utfört i Fordonssystem
vid Tekniska högskolan vid Linköpings universitet
av

Lukas Lorentzon

LiTH-ISY-EX--14/4797--SE

Linköping 2014

Linköpings universitet
TEKNISKA HÖGSKOLAN

Department of Electrical Engineering
Linköpings universitet
SE-581 83 Linköping, Sweden

Linköpings tekniska högskola
Linköpings universitet
581 83 Linköping

Investigation of a troubleshooting procedure

By assessing fault tracing algorithms

Examensarbete utfört i Fordonssystem
vid Tekniska högskolan vid Linköpings universitet
av

Lukas Lorentzon

LiTH-ISY-EX--14/4797--SE

Handledare: **Neda Nickmehr**
ISY, Linköpings universitet
Daniel Jung
ISY, Linköpings universitet
Henrik Fagrell
Diadrom

Examinator: **Erik Frisk**
ISY, Linköpings universitet

Linköping, 12 september 2014

Avdelning, Institution
Division, Department

Avdelningen för Fordonssytem
Department of Electrical Engineering
SE-581 83 Linköping

Datum
Date

2014-09-12

Språk
Language

☐ Svenska/Swedish
☒ Engelska/English
☐ ________________

Rapporttyp
Report category

☐ Licentiatavhandling
☒ Examensarbete
☐ C-uppsats
☐ D-uppsats
☐ Övrig rapport
☐ ________________

ISBN
—

ISRN
LiTH-ISY-EX--14/4797--SE

Serietitel och serienummer
Title of series, numbering

ISSN
—

URL för elektronisk version

http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-XXXXX

Titel
Title

Undersökning av olika felsökningsmetoder
Investigation of a troubleshooting procedure

Författare
Author

Lukas Lorentzon

Sammanfattning
Abstract

Nyckelord
Keywords

Troubleshooting procedures, Algorithms

Abstract

The thesis delves into the area of troubleshooting procedures, an interesting area for industry. Many products in industry tend to be complex, which in turn makes troubleshooting procedures trickier. A fast and efficient repair process is often desired, since customers want the product to be repaired as fast as possible.

The purpose of a troubleshooting procedure is to find a fault in a broken product, and to choose proper repair actions in a workshop. Such a procedure can be simplified by diagnosis tools, for example software programs that make fault conclusions based on fault codes. These tools can make such conclusions with the help of algorithms, i.e. fault tracing algorithms.

Before a product release, it is hard to specify all faults and connections in the system. New unknown fault cases are likely to arise after release, and somehow this need to be taken into account in the troubleshooting scenario. The troubleshooting procedure can be made more robust, if new data could be easily incorporated in the current structure. This work seek to answer how new data can be incorporated in trouble shooting procedures.

A good and reliable fault tracing algorithm is essential in the process of finding faults and repair actions, which is the reason behind the focus of this thesis. The presented problem asks how a fault can be identified from fault codes and symptoms, in order to recommend suitable repair actions. Therefore, the problem is divided into two parts, finding the fault and recommending repair actions. In the first part, three candidate algorithms for finding the faults are investigated, namely Bayesian networks, neural networks, and a method called matrix correlation inspired from latent semantic indexing. The investigation is done by training each algorithm with data, and evaluating the results. The second part consists of one method proposal for repair action recommendations and one example. The theoretical investigation is based on the Servo unit steering (SUS), which reside in the IPS system of Volvo Penta.

The primary contribution of the thesis is the evaluation of three different algorithms and a proposal of one strategy to recommend suitable repair actions. In this study Bayesian networks are found to conform well with the desired attributes, which in turn lead to the conclusion that Bayesian networks is well suited for this problem.

Acknowledgments

I would like to thank the staff at Diadrom and my examiner Erik Frisk that made the master thesis possible. During the work I received support from my supervisors Neda Nickmehr and Daniel Eriksson at Linköping university, and from Henrik Fagrell at Diadrom, and I am very grateful for their support. The discussions regarding issues with them were fruitful and made the thesis progress onward when setbacks occurred. I would also like to thank the staff at Volvo Penta and Aros that showed interest in the thesis and contributed with data.

Linköping, June 2014
Lukas Lorentzon

Contents

Notation

Abbreviations	Meaning
MC	Matrix correlation
BN	Bayesian network
NN	Neural network
M_i	Malfunction $i = 1, \ldots, 15$
FMI_i	Fault mode identifier, $i = 1, \ldots, 15$ (J1587 standard for codes)
S_i	Symptoms $i = 1, \ldots, 16$
PID_i	Parameter identification description, i = code number (J1587 standard for codes)
SID_i	Subsystem identification description, i = code number (J1587 standard for codes)
$PPID_I$	Proprietary PID, i = code number (manufacturer specific code)
$SSID_i$	Proprietary SID, i = code number (manufacturer specific code)

1 Introduction

1.1 Background

Products such as boats, cars, and trucks are expected to be operational for a long period, due to their long life span. A long period of usage often causes wearing of the components, which could lead to faulty components that need to be repaired. The troubleshooting procedures of such products can be challenging due to their complexity. Products like these are therefore likely to end up in a workshop sooner or later. When this occurs, the repair process should go as smoothly as possible, by finding faults and suitable repair actions without too many iterations. This procedure can be made more efficient by a fault tracing algorithm, which refer to an algorithm that point out the most probable fault and suggest suitable repair actions. The trouble shooting procedure could become less time consuming and fewer spare parts might need to be replaced. In this way the right conclusion could be reached faster. Time and material savings can therefore be achieved by improving diagnosis tools, that aid mechanics by specifying likely faults and suitable repair action recommendations. The focus of this thesis is to investigate fault tracing algorithms.

The thesis is performed at Diadrom with Volvo Penta as Partner. Diadrom specializes in diagnostics and high technology products. Volvo Penta is a supplier of engines and propulsion systems in the marine and industrial area.

The system focused on in the thesis is the Servo Unit Steering (SUS), which is a part of the Volvo Penta propulsion system for boats (IPS). The SUS is located in the upper gear of the IPS system, i.e the modern inboard system, see Figure 1.1. The SUS consists of control units and one electric motor. SUS units control the propellers in a way that makes lateral steering in a boat possible by using a joystick.

Today a repair manual exists to aid mechanics in troubleshooting scenarios, regarding the IPS system. The repair manual contain all relations between faults, faults codes, symptoms, and repair actions. In total there are 15 faults, 30 fault codes, and around 23 symptoms. Repair actions in the repair manual are represented as lists which specify suitable actions. Each list specify suitable repair actions for a certain combination of fault codes and faults. The repair manual is sufficient in certain cases where the problem is easily identified. If the repair manual does not point out a specific fault during a troubleshooting scenario, the identification of a fault and the choice of a repair action rely mostly on the experience of the mechanic. It is hard to cover all possible troubleshooting scenarios before a product release. Therefore, much manual labour is required to update the repair manual when new fault scenarios arise. The data from new fault scenarios can be used as training data for the algorithms in order to reflect new dependencies. The manual labour of updating the algorithm would presumably decrease, due to the feedback of data. In other words the algorithm would be responsible for identifying patterns in a data set in order to update itself.

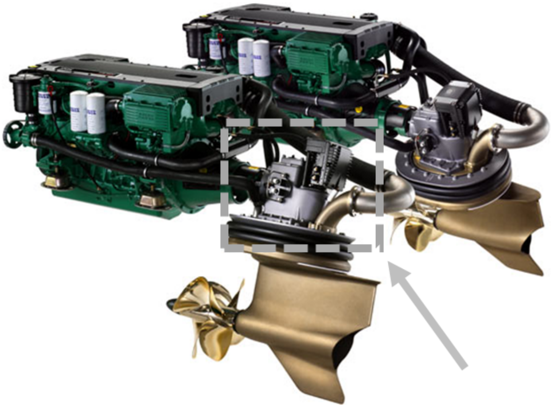


Figure 1.1: The IPS system, in which the black arrow and rectangle indicate where the SUS unit resides.

1.2 Diagnostic Process Example

A faulty boat system produces fault codes that are downloaded to a diagnosis tool on workshop arrival. The tool also receives inputs such as observed symptoms from the mechanic or the user, for example strange engine sounds, engine start up failure, and steering problems. A troubleshooting algorithm assesses information in order to reach a conclusion of which fault is present, and to recommend suitable repair actions. The diagnosis tool will then display a list of suitable repair actions that are likely to fix the problem. Note that observations from the mechanic can be used as feedback to the troubleshooting procedure to further isolate the faulty component. Figure 1.2 shows one possible scenario at a workshop with a faulty product.

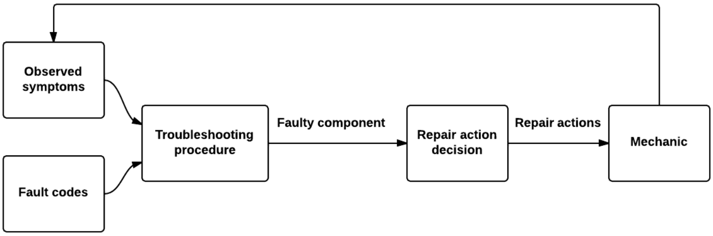


Figure 1.2: Example process for finding fault and suitable repair actions.

1.3 Problem Formulation

Fault tracing/troubleshooting in this concept aims at finding the faulty components and suitable repair actions. *Operating profile* includes usage e.g mileage, product location in the world and so on. See Figure 1.3 for a clearer view of the problem. A *fault code* is a code that signals for a certain error and a *symptom* describes a certain behaviour of the system.

The fault model in Figure 1.3 is based on the repair manual [Penta, 2006]. The repair manual describes connections between fault codes, symptoms, faults and repair actions. Note that the repair manual contains ideal cases constructed by experts that do not always represent reality in a good way. The repair manual is used by reading fault codes from the system, and by identifying cases that are similar in the repair manual.

The goal here is to investigate diagnostic algorithms that can be of assistance to a mechanic during troubleshooting. Another desired outcome is to see how different fault cases from workshops can update model dependencies in order to improve the troubleshooting algorithms, for example between fault codes and faults. A fault case can include fault codes, symptoms and faults. Fault case data can be used as feedback data, to update dependencies between, e.g., symptoms and faults, both manually and automatically. For example, a new set of fault cases, which indicates a stronger connection between one fault code and a fault, should update the connection between these.

The research question that reflect these desires is stated below:

How can fault tracing be implemented with use of system data, e.g. fault codes, observed symptoms, and operating profile to recommend and rank suitable repair actions?

Problem outline

The problem will be divided into two parts, and the focus of both is to study and evaluate different methods.

- *The first part:* The first part involves investigating algorithms that can find the most likely fault in a troubleshooting scenario, which is done in a case study.
- *The second part:* The second part covers how to find suitable repair actions.

In Figure 1.3, the fault model represents how all the components of the model relate to each other. The fault tracing algorithm represents how a fault is chosen. The outputs from the algorithm are faults and a set of recommended repair actions. The feedback loop demonstrates how data from repair cases can be used to update the model to reflect new situations, for example, successful repair actions and fault identifications.

Input and output in Figure 1.3:

Input: Observed symptoms, fault codes and operating profile.
Output: Fault and a ranked list of suitable repair actions. The list does not represent a sequence of repair actions that should be performed.

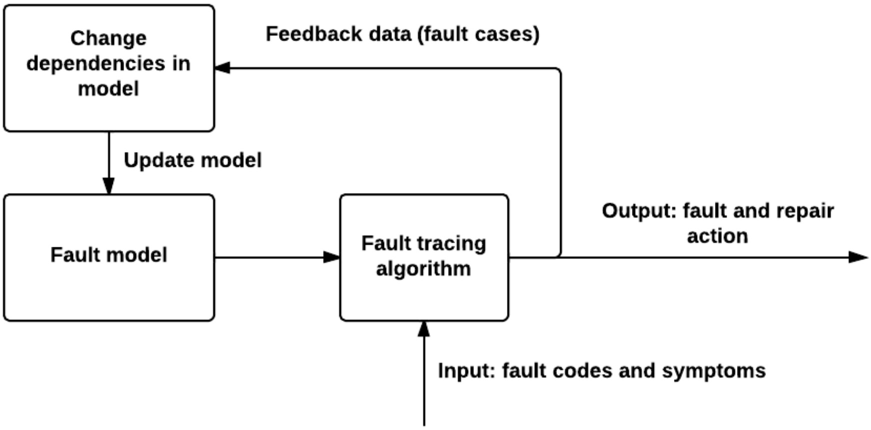

Change dependencies in model
Feedback data (fault cases)
Update model
Fault model
Fault tracing algorithm
Output: fault and repair action
Input: fault codes and symptoms

Figure 1.3: A flowchart of the troubleshooting procedure representing the problem formulation.

The SUS unit is one component in a bigger system and faults in other components might affect it. The problem in the case study (Chapter 4) is narrowed down to only the SUS, and external effects from other components are not taken into account. This problem is considered to some extent in Chapter 5.

1.4 Related Research

Similar problems have been considered before in areas like automotive industry. In the case of automotive industry, the dissertation [Pernestål, 2009] addresses ways to do troubleshooting, making repair action and repair strategy choices through Bayesian networks. The paper [Warnquist, 2011], addresses a way to do off-board diagnosis with the goal of suggesting repair actions as in the previously mentioned case. These two papers deal with troubleshooting through non stationary Dynamic Bayesian Networks (nsDBN). In short nsBDN are event driven and consist of different epochs in regard to the actions performed. In contrast to this an ordinary Bayesian network is static, since the network structure do not change. Other works that concentrate on a similar topic regarding repair strategies are [Heckerman et al., 1995] and [Langseth and Jensen, 2003].

A research article [Yingping Huang and Zhang, 2014], handles troubleshooting with Bayesian networks and multicriteria decision analysis (MCDA). By combining these methods, the paper incorporates for example cost and repair times values into the process of making a repair action decision. More accurately the task of the Bayesian network is to supply fault probabilities, while the MCDA method choose a repair action.

In the paper [Shatnawi and Al-khassaweneh, 2014], an extended neural network (ENN) is used for classification of faults from all features in the troubleshooting procedure of an internal combustion engine. Neural network implementations also exist within the area of medicine, for example, a breast cancer classification problem [Azar and El-Said, 2012]. Three classification algorithms were compared to each other, and one of them was probabilistic neural networks (PNN).

The paper [Li and Han, 2013], studies comparison of vectors with similarity measures, and the focus lies on the cosine similarity measure. A number of different similarity measures are presented and evaluated.

1.5 Method

The work during this thesis is divided into three main parts: gathering information, choosing candidates, and evaluating methods. The reason behind this is to get a structured thesis procedure. Information gathering was needed in order to get a good understanding of the area and to learn new theory. The choice of methods are based on the gathered information, and the purpose of the tests is to find the most suitable method for the problem. In more detail the steps can be divided into the following structure:

- *Literature study*
    - The literature study involves finding similar works in regard to the research question, for theory and inspiration. This step also includes learning the basic theory behind the methods that where considered as candidates for the case study. In order to assess how well they suit the problem.
- *Interviews* with SUS experts. The interview occasions:
    - Discussion of possible solution methods with two diagnosis experts at Diadrom.
    - Discussion of problem and data with diagnosis and after market employees at Volvo Penta.
    - One product quality employee at Volvo Penta describing relations in more complicated cases.
    - A product specific interview about data with two employees at Aros, a key supplier of the motor components.
- *Acquiring data*
    - Familiarization with repair manual
- *Choosing methods*
- *Simulation of data*
    - Generation of data for the case study.
- *Case study,* i.e. testing the methods
- *Further Analysis* of chosen method
- Suggestion of possible *implementation strategy*

1.6 Outline

In Chapter 2, the system and available data is presented. In Chapter 3, candidate methods are explained briefly. A case study consisting of all methods is presented in Chapter 4, where the results are discussed in order to choose one method for further analysis in Chapter 5. Finally the conclusion and future work, in the same area, is discussed in Chapter 6.

2
System

2.1 Description

The IPS system from Volvo Penta includes a distributed system, which consists of small electronic nodes. These nodes are called: PCU (Power control unit), SHCU (Steering helm control unit), and SUS (Servo steering unit), see Figure 2.1. All these components work together in the system, and because of this a fault in one component could cause false faults in the other components. These relations makes them closely related in a fault tracing scenario. As mentioned before, the focus here is on the SUS unit.

2.2 Data Definitions

The repair manual[Penta, 2006] presents a couple of data definitions that follow below:

- *Malfunction:* Describes a system failure.

- *Symptom:* Describes behavior of the system.

- *Fault code:* Is a code that signals for a certain fault, which can consist of fault mode identifier (FMI), parameter identification description (PID), subsystem identification description (SID), and message identification description (MID).
  - *FMI:* Indicate type of fault

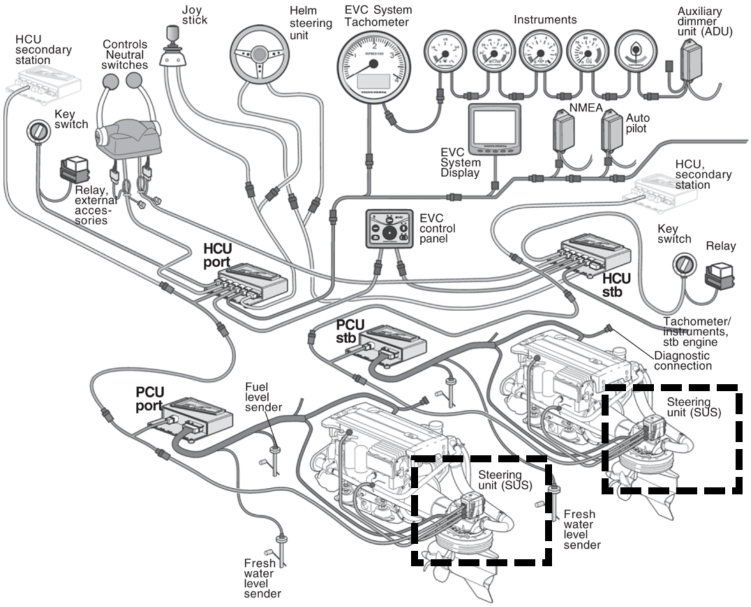


Figure 2.1: The whole system and all the components [Penta, 2006]. The location of the SUS unit is shown by the black dashed rectangles.

- *PID and PPID (proprietary PID):* Points out the parameter to which the fault code relates to.
- *SID and SSID (proprietary SID):* Points out the component to which the fault code relates to.
- *MID:* Designates the control unit that sends the fault code, i.e SUS, PCU or SCHU.

Fault codes are set by the system while symptoms are observed by the mechanic or the user. An example of a symptom is a boat that cannot be steered properly. Example 2.1 show how a combination of fault codes and one observed symptom are used to find the fault according to the repair manual. The repair manual states the connections between malfunctions, fault codes, and symptoms but do not evaluate their strength.

Example 2.1

The Tables 2.2, 2.3, and 2.4 state all relations between faults, fault codes, and symptoms. The crosses in the tables represent connections between faults (rows) and fault codes or symptoms (columns). These tables can be used in a fault case scenario to deduce which fault is likely to be present.

Consider a case there the fault codes MID_{250}, FMI_0, $PSID_3$, and symptom S_{11} are active. The MID_{250} code indicates that the SUS unit has sent the fault codes FMI_0 and $PSID_3$. Table 2.2 shows that FMI_0 is connected to M_{11} and M_{13}. Table 2.3 shows that $PSID_3$ is connected to M_{13}. Table2.4 shows that S_{11} is connected to M_{10}, M_{11}, and M_{13}. The fault that has most in common with the fault codes and the symptom in this case is M_{13}. Therefore the conclusion is that the fault M_{13} (servo motor fault) is likely to be present.

Note that the tables represent connections stated in the repair manual.

2.3 Available Data

In this section, all available data to the troubleshooting algorithm will be presented. However, all data here have not been used in this thesis. The reason for is that much data exists in the form of work cards, i.e usage, location, mileage, configuration, and so on. The data on the work cards cannot be easily accessed, and it is not tractable to extract all of this information, since it would have been very time consuming. Another reason is that some of the data became available very late in the thesis work, which made it harder to incorporate in the study. The list below presents all data types briefly. The data with the available tags is easy to access while the data without is stored on work cards.

Vehicle data

- Fault codes (available)
- Symptoms (available)
- Repair Actions (available)
- Connections between fault codes, symptoms and repair actions (available)
- Product configuration
- Location, where in the world the product is used
- How long has the product been in use
- Product usage, for example easy or hard
- Earlier service and reparations
- Reparation and service manuals

Feedback data

- Feedback data refer to data that can update the fault model of the system. Feedback data consists of one or many fault cases, which contain fault codes, symptoms, and repair actions.

Data like symptoms, FMI's, SID's and PID's are binary. In other words, they are either present or not. Data such as mileage are not binary. This presents a more complex problem, since both binary and continuous data could be in the same model. This is due to the fact that continuous data can be harder to interpret compared discrete data.

A set of defined malfunctions (faults), FMI's, and symptoms exist in the repair manual [Penta, 2006]. See Table 2.1 for FMI definitions. See Tables 2.2, 2.3, and 2.4 for malfunction definitions and their connections to FMI's, SID's, PID's, and symptoms. Note that a malfunction might be present even if all fault codes connected to it are not active. Table 2.4 shows symptoms for all malfunctions that have such. There are lists of repair actions corresponding to one FMI and one malfunction in the repair manual [Penta, 2006]. The SUS has approximately 40 repair actions in the repair manual. An example of a repair action list is:

Servo Motor faulty and $\mathbf{FMI_0}$ is active

1. Check if other error codes exist, that imply error in electrical system
2. Check battery connection
3. Measure battery voltage
4. Check power cable connection between SUS and engine
5. Measure the voltage on B+ and B– on the SUS

Table 2.1: FMI definitions

FMI	Display text
0	"Value too high"
1	"Value too low"
2	"Faulty data"
3	"Electrical fault"
4	"Electrical fault"
5	"Electrical fault"
6	"Electrical fault"
7	"Mechanical fault"
8	"Mechanical or electrical fault"
9	"Communication fault"
10	"Mechanical or electrical fault"
11	"Unknown fault"
12	"Component fault"
13	"Faulty calibration"
14	"Unknown fault"
15	"Unknown fault"

Table 2.2: Dependencies between malfunctions and the FMI fault codes.

Fault codes		FMI_0	FMI_1	FMI_2	FMI_3	FMI_4	FMI_5	FMI_6	FMI_7	FMI_8	FMI_9	FMI_{10}	FMI_{11}	FMI_{12}	FMI_{13}	FMI_{14}	FMI_{15}
Malfunctions	M_i																
Program memory fault	M_1			x													
Calibration Memory Failure	M_2			x							x			x	x		
Internal CPU faults	M_3			x									x	x			
Data bus communication warning	M_4			x			x										
Data bus network configuration fault	M_5													x			
Data bus power input	M_6					x							x				
Battery input	M_7					x							x				
ECU temperature	M_8													x			
Steering wheel position	M_9			x										x			
Rudder angle	M_{10}			x			x	x	x								
Servo motor temperature	M_{11}	x	x														
Data bus power output	M_{12}					x											
Servo motor	M_{13}	x	x		x	x	x	x	x			x					
Electro mechanical rudde brake	M_{14}						x	x	x							x	
Data bus communicating with active helm failure	M_{15}										x	x					

Table 2.3: Dependencies between malfunctions and the SID and PID fault codes.

Fault codes		SID_{240}	SID_{253}	SID_{254}	$PSID_{234}$	$PSID_1$	$PPID_{393}$	PID_{55}	$PPID_{55}$	$PPID_{424}$	$PPID_{426}$	$PPID_{427}$	$PSID_2$	$PSID_3$	$PSID_4$	$PSID_6$
Malfunctions	M_i															
Program memory fault	M_1	x														
Calibration Memory Failure	M_2		x													
Internal CPU faults	M_3			x												
Data bus communication warning	M_4				x											
Data bus network configuration fault	M_5					x										
Data bus power input	M_6						x									
Battery input	M_7							x								
ECU temperature	M_8								x							
Steering wheel position	M_9									x						
Rudder angle	M_{10}										x					
Servo motor temperature	M_{11}											x				
Data bus power output	M_{12}												x			
Servo motor	M_{13}													x		
Electro mechanical rudde brake	M_{14}														x	
Data bus communicating with active helm failure	M_{15}															x

Table 2.4: Possible symptoms related to different malfunctions

Symptoms		S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}
Malfunctions	M_i																
Program memory fault	M_1	x															
Calibration Memory Failure	M_2		x	x													
Internal CPU faults	M_3	x															
Data bus communication warning	M_4																
Data bus network configuration fault	M_5				x	x	x										
Data bus power input	M_6																
Battery input	M_7							x	x	x							
ECU temperature	M_8										x						
Steering wheel position	M_9	x								x							
Rudder angle	M_{10}											x	x				
Servo motor temperature	M_{11}											x					
Data bus power output	M_{12}																
Servo motor	M_{13}											x					
Electro mechanical rudde brake	M_{14}													x			
Data bus communicating with active helm failure	M_{15}														x	x	x

3
Theory

3.1 Troubleshooting/Fault Tracing Algorithms

The first thing that shall be considered here is the desired functionality of the algorithms. Note that the task for these algorithms is to find the most likely fault, i.e the first part of the problem, see Chapter 1. One desired attribute of the algorithms is the ability to update connections between faults, fault codes, and symptoms, if new data is available. This could for example, result in stronger connections or new ones. It is also important that the complexity of the algorithms is within a reasonable scope. In cases where computational power is limited this could be of great importance. If the basic idea of the algorithms is simple to grasp, it is likely to simplify usage. Classification problems, in the area of machine learning, fit well with these attributes.

There are many algorithms and ideas that are applicable to this problem. A Bayesian network can model dependencies and point out the fault if a fault case is given. The modelling of a Bayesian network [Jensen and Nielsen, 2007] can be done entirely from the repair manual. There are other methods that fit the problem as well, namely neural networks [Russell and Norvig, 2003], and one approach developed in this thesis called matrix correlation approach, which is similar to latent semantic indexing [Manning et al., 2008]. In this chapter, the theory behind all three candidates will be explained briefly.

3.2 Matrix Correlation Approach

The matrix correlation approach is inspired from latent semantic indexing [Manning et al., 2008] and similarity measures [Li and Han, 2013]. It is based on the fact that the relationships between malfunctions, symptoms and FMI's can be modelled by a matrix. Assume that the Tables 2.2, 2.3, and 2.4 in Chapter 2 are a big matrix with only zeros and ones. The zeros represent that there is no connection between a column and a row entry, while ones represent the opposite. If all the ones were replaced by a number that stated the strength of correlation between fault codes or symptoms and malfunctions, see Table 3.1 for an example. A matrix of correlation values would be obtained. A fault case vector consisting of zeros and ones to denote inactive and active fault codes and symptoms, can then be used to decide which malfunction is present, by taking for example the scalar product between the vector and each row in the matrix, see Example 3.1. The result is a vector where each value represent one malfunction, and the values represent a similarity measure that state how strongly each fault is connected to the fault case. The matrix containing all correlation values is denoted correlation matrix. Scalar product similarity measures will give crude correlation values since large values in the correlation matrix have great impact on the results, but the they are easy to interpret. The correlation matrix can be constructed by counting all occurrences of all fault codes and symptoms in regard to all malfunctions, i.e. correlation values. Note that the correlation matrix could be constructed by choosing the correlation values instead of learning them from a data set.

Table 3.1: Example of a table with relations between malfunctions (M_i), FMI's, SID's, and PID's.

	FMI_1	FMI_2	FMI_3	SID_1	SID_2	SID_3
M_1	0	2	1	1	0	0
M_2	3	0	1	0	1	0
M_3	2	1	0	0	0	1

Example 3.1

The fault codes in this example are FMI_1, FMI_2, FMI_3, SID_1, SID_2, and SID_3, see Table 3.1. The correlation matrix is specified in Table 3.1. A fault case vector contains zeros and ones, that represent active and inactive fault codes respectively. If a fault case vector, like $\begin{pmatrix}0 & 1 & 1 & 0 & 1 & 0\end{pmatrix}$ is obtained, the correlation matrix and the fault case vector can be used to deduce the fault by a similarity measure. The similarity measure can be obtained by calculating the scalar product between the fault case vector and each row in the matrix. The matrix (Table 3.1) will be denoted R, a matrix row i will be written as R_i, and the fault code vector is denoted F. Each value in the matrix R is a measure stating how closely related the column entry is to the row entry. The calculations with the scalar product as similarity measure is shown below:

Fault case:

$F = \begin{pmatrix} 0 & 1 & 1 & 0 & 1 & 0 \end{pmatrix}$

Matrix rows:

$R_1 = \begin{pmatrix} 0 & 2 & 1 & 1 & 0 & 0 \end{pmatrix}$

$R_1 = \begin{pmatrix} 3 & 0 & 1 & 0 & 1 & 0 \end{pmatrix}$

$R_1 = \begin{pmatrix} 2 & 1 & 0 & 0 & 0 & 1 \end{pmatrix}$

$\text{Correlation} = R_1 \cdot F = 3$

$\text{Correlation} = R_2 \cdot F = 2$

$\text{Correlation} = R_3 \cdot F = 1$

The best match for the fault case vector, can be found by comparing the values by size, i.e a larger value indicate a stronger connection.

In this example, the scalar product is used as a similarity measure to simplify the process. There are many other similarity measures like the cosine similarity, see equation (3.1). This similarity measure will be used in the case study. The reason behind this is illustrated in Example 3.2.

$$cos(\theta) = \frac{a \cdot b}{|a||b|} \tag{3.1}$$

Example 3.2

The vectors a, b, and c are used to demonstrate the difference between cosine similarity and the scalar product. The vectors a and b are similar, with the exception of one large value, 1000. If similarity measures, between these vectors and c, are calculated. The difference between scalar product and cosine similarity can be seen in the results, i.e. the range of the results.

Vectors:

$$a = \begin{pmatrix}1 & 0 & 6 & 1000 & 0\end{pmatrix}$$

$$b = \begin{pmatrix}1 & 0 & 6 & 10 & 0\end{pmatrix}$$

$$c = \begin{pmatrix}1 & 0 & 0 & 1 & 0\end{pmatrix}$$

Scalar products:

$$a \cdot c^T = 1001$$

$$b \cdot c^T = 11$$

Cosine similarities:

$$\frac{a \cdot c^T}{|a||c|} = 0.7078$$

$$\frac{b \cdot c^T}{|b||c|} = 0.6621$$

The scalar product can lead to similarity values in many different sizes, and one value from the scalar product can dominate the result. This in turn, makes it harder for the method to recognize patterns. Cosine similarity amends this, and will give results in the same range.

3.3 Bayesian Networks

Bayesian networks represent a way to do probabilistic reasoning. Imagine a network of nodes with links. All nodes could for example represent events with states, for example true or false. These nodes are connected through links that make them dependent on each other. Each node is assigned with probability values that indicate how likely one state is to happen given the states of the parent nodes. A state for one or more nodes may be known in one situation, i.e *evidence*. The network can then be used to calculate new probabilities (*posterior probabilities*) for all the other nodes, i.e *interference*. The general idea behind Bayesian networks is to construct a graph that represents conditional probabilities between different nodes with states. This graph represents the full joint probability distribution.

A Bayesian network can be defined as follows [Jensen and Nielsen, 2007]:

- A set of random variables (stochastic variables) and directed links between the variables.
- The variables can be either continuous or discrete.
- If there is a link from X_1 to X_2, X_1 is the parent of X_2. The network contains no cyclic connections and hence is a directed acyclic graph (DAG). Each

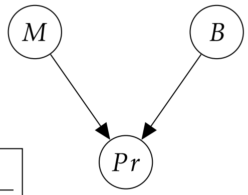
node X_i has a conditional probability distribution $P(X_i|\text{Parents}(X_i))$, that describes the parents influence.

The most common type of Bayesian networks are called causal models. This means that links only are drawn between the nodes X and Y, if X can cause Y to enter a certain state. A causal network is built by considering cause and effect.

Example 3.3

Consider a small example with the nodes memory, bus and program. If memory or the bus is broken, it will affect the program. This can be seen from the links between memory (M), bus (B) and program (Pr). The probabilities $P(M)$ and $P(B)$ represent the probability that the component is faulty. The conditional probability $P(Pr|M,B)$ states how likely the program is to be non functional depending on the states of M and B.

$P(M)$
0.15



$P(B)$
0.1

M	B	$P(Pr|M,B)$
t	t	0.99
f	t	0.7
t	f	0.4
f	f	0.05

Figure 3.1: Example of Bayesian network with three nodes, M (memory), B (Bus) and Pr(program). All values in the CPT's denote the probability of being true. The abbreviation for true and false are t and f.

The probabilities $P(Pr,M,B)$, in Table 3.2, are obtained by the following product rule, i.e:

$$P(Pr,M,B) = P(Pr|M,B)P(M|B)P(B) = P(Pr|M,B)P(M)P(B)$$

In which the equality $P(M|B) = P(M)$, has been used since the variables M and B are independent because of d-separation, see [Jensen and Nielsen, 2007]. The tables in Figure 3.1 represent conditional probability distributions (CPD). The CPD becomes a conditional probability table (CPT) because all variables are discrete. Nodes with this kind of CPT's are often denoted as *general type*. The full joint probability distribution can also be represented as a table, see Table 3.2.

One attribute of Bayesian networks is the way they express the full joint probability distribution as many small distributions, for example CPT's. Together all these distributions represent the full joint probability function, because of the chain rule for Bayesian networks, see equation (3.2) [Jensen and Nielsen, 2007].

$$P(X_1, X_2, ..., X_n) = \prod_{i=1}^{n} P(X_i|parents(X_i)) \tag{3.2}$$

Table 3.2: Full joint probability distribution

M	B	Pr	$P(Pr, M, B)$
t	t	t	0.0148
t	t	f	0.00015
t	f	t	0.054
t	f	f	0.081
f	t	t	0.0595
f	t	f	0.0255
f	f	t	0.0383
f	f	f	0.7268

3.3.1 Evidence

A Bayesian network consists of nodes with different states. If the state of one node is known, for example a faulty component, the state of that node can be set. In other words evidence is given to the Bayesian network. By using the new knowledge, new probability values for the other nodes can be calculated, see Section 3.3.3. Evidence, therefore, makes the Bayesian network re-evaluate the situation, and this is how known states are set in the fault tracing problem.

3.3.2 Noisy Or

Noisy-or is a node type that can be used to simplify a network [Jensen and Nielsen, 2007]. If a cause is present but the effect does not trigger, the effect has been inhibited, e.g. $P(X = false|Y = true)$. Consider the network below, Figure 3.2:

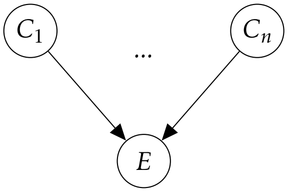


Figure 3.2: Example of Bayesian network with effect E and Causes C_i.

By specifying all the probabilities describing if one effect is being inhibited in respect to all causes, all necessary probabilities can be acquired. Some assumptions have to be made in order for this to be true:

- All variables are binary.

- All the causes C_i are independent of each other.
- $P(E = true|C_1 = false, \dots, C_n = false) = 0$

A probability called leak or background probability often exists in noisy-or models, which represents activation due to external circumstances. This is often needed due to the last assumption.

The variable q_i represents the chance of an effect being inhibited in regard to a certain cause. If the causes C_1 and C_2 are active the probability for $E = true$ is:

$$
\begin{aligned}
&P(E = true|C_1 = true, C_2 = true, C_3 = false, \dots, C_n = false) = \\
&1 - P(E = false|C_1 = true, C_2 = true, C_3 = false, \dots, C_n = false) = \\
&1 - q_1 * q_2
\end{aligned}
$$

Noisy-max [Jensen and Nielsen, 2007] is a generalized version of noisy-or that can handle variables that have more than two states.

3.3.3 Interference Methods

Interference in a Bayesian network is the calculation of the posterior probabilities given some evidence, E [Russell and Norvig, 2003]. One way to do this is to use the *chain rule*. Interference will find the posterior probability $P(Q|E)$, where Q is the query variables, and the Bayesian network consists of variables X. The hidden variables that are not written in the query will be denoted H. Note that $X = Q \cup E \cup H$.

One way to do interference would be to just marginalize the hidden variables H out of the full joint probability distribution, see equation (3.3). Another way to express this is to say that the hidden variables are removed from the full join probability distribution. The α in the equation represents a normalization constant.

$$P(Q|E) = \alpha \sum_{H} P(Q, E, H) \tag{3.3}$$

This interference method is not a good solution for large networks, due to the fact that a full joint distribution will have many probability values. In the case of boolean variables, the number of entries will be 2^n, which becomes very large if the network is big.

Another approach is to use the *variable elimination* algorithm [Russell and Norvig, 2003]. The network in Figure 3.3 will be used to demonstrate the algorithm.

If we have evidence $E = \{A = true, B = true\}$ and want to calculate $P(D|A = true, B = true)$, which we will denote as $P(D|a, b)$. The nested summation using the *chain rule* is:

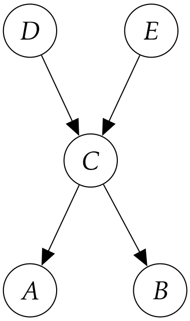


Figure 3.3: Example of Bayesian network with effect E and causes C_i.

$$P(D|a,b) = \alpha P(D) \sum_{E} P(E) \sum_{C} P(C|D,E)P(a|C)P(b|C) \tag{3.4}$$

Each term in the equation (3.4) will correspond to one *factor*. A factor can be viewed as a CPT, see for example Table 3.3. The equation gives us the factors, $f_1(D)$, $f_2(E)$, $f_3(C,D,E)$, $f_4(C)$, and $F_5(C)$. Notice that a and b are left out in the last two factors, because they are fixed in the query. Two operations will be used during the calculations, namely *pointwise product* and *summation* of factors. Pointwise product of factors is a union and if there exists one value in each factor corresponding to the same entry, the values are multiplied. Summing is the same as marginalizing, which means that a set of variables are summed out of the probability distribution. One example of this procedure can be seen in Table 3.4, where one variable has been summed out from Table 3.3. See [Russell and Norvig, 2003] for more information. In the end, the following expression with factors is evaluated from right to left:

$$P(D|a,b) = \alpha f_1(D) \times \sum_{E} f_2(E) \times \sum_{C} f_3(C,D,E) \times f_4(C) \times f_5(C)$$

Table 3.3: Example of one factor

C	D	E	$f(C,D,E)$
T	T	T	p_1
T	T	F	p_2
T	F	T	p_3
T	F	F	p_4
F	T	T	p_5
F	T	F	p_6
F	F	T	p_7
F	F	F	p_8

Table 3.4: D has been marginalized out from the probability distribution in Table 3.3

C	E	$f(C,E)$
T	T	$p_1 + p_3$
T	F	$p_2 + p_4$
F	T	$p_5 + p_7$
F	F	$p_6 + p_8$

The basic outline of the variable elimination algorithm:

- for each variable in network (note that variable order only matters in regard to complexity).
    1. Create and store factor.
    2. If the variable is hidden use summation to sum out the variable.
- Do *pointwise product* on all the factors.
- Normalize result.

If a Bayesian network is large and more complicated than the example, variable elimination can have exponential time and space complexity [Russell and Norvig, 2003]. Another type of algorithms that can be used to reduce time complexity are *clustering algorithms*. The network in the Figure 3.3 is a multiple connected network, see node C. The underlying idea in clustering is to create single connected network by combining nodes, for example one step is to combine D and E.

Approximate interference can be utilized if the networks are large and complicated. These methods often involve sampling the states from the prior probabilities in the network [Russell and Norvig, 2003]. The samples are used in calculations of the posterior probabilities.

3.3.4 Learning Parameters

One way to handle the learning problem is to use *maximum likelihood estimation*. The method is based on finding the parameters that maximize the maximum likelihood estimate. Maximum likelihood estimates for CPT's in Bayesian networks can be calculated by counting the number of occurrences for cases bound to the CPT's. For example a maximum likelihood estimate is obtained by dividing the number of occurrences for a specific case by total number of cases. The maximum likelihood of $P(x|\dots)$ is for example calculated by:

$$\hat{\theta} = \frac{N(x,\dots)}{N(\dots)}$$

$N(x,\dots)$ = number of cases where x is present

$N(\dots)$ = total number of cases with variables ...

If a data set contained a variable with zero occurrences, the corresponding parameter would receive the value zero. In other words the probability is zero. Consider a data set with a number of faults and fault codes. If one fault never occurs in the set, the maximum likelihood estimate for the faults CPT's are zero. This could be problem if a small data set is used.

Bayesian estimate is another way to learn parameters that handle this problem. The method is based on the fact that prior probabilities must be chosen before the estimation based on new data. The idea is to update the posterior probabilities based on the prior probabilities. The variable x symbolizes an event in the network and θ denotes the parameters, in other words, probabilities. The new probability f_n is calculated by using θ, $P(x)$ and the prior probability f,

$$f_n(\theta) = \frac{P(x|\theta)f(\theta)}{P(x)}$$

If a data set is incomplete, certain values are missing. This is a problem in the methods above, since these incomplete cases need to be removed. For example if one fault has a value that is always missing, the final data set would not represent the fault in a good way. There is another more advanced method compared to the ones above, called *EM-algorithm* that can handle incomplete data. To compensate for incomplete data, the algorithm uses known probabilities in estimation purpose, and then moves on to finding a new Bayesian estimate. See [Jensen and Nielsen, 2007], for more information regarding the *EM-algorithm*.

3.4 Neural Networks

A neural network consists of nodes and links, see Figure 3.4 and the idea behind this is to imitate how the brain works [Russell and Norvig, 2003]. The figure represents a feed-forward network, where all links point in the same direction, i.e. the network becomes a directed acyclic graph. Feed forward networks often consist of building blocks in the form of layers. The layers that are not input and output are called hidden layers. To each link, a weight value is attached, which determines how much the output value from one node will affect the next. All nodes with the exception of the input nodes are bound to one activation function, which is a function of the sum of all products between inputs and weights. A neural network need to learn the values for all weight values from data. When weights values have been calculated input can be propagated forward in the network. This is done by using the weight values and the activation functions connected to links and nodes respectively. Basically nodes in one layer send their results to the nodes in the next layer. These nodes evaluate the results for the previous layer by using weights and activation functions. The output from the current layer is sent to the next. This procedure continues until the last layer is reached. This is how output is obtained. To summarize, input values need to be propagated forward to calculate output, i.e. forward propagation [Ng, 2014]. See

Example 3.4. Input and output can be either continuous or discrete.

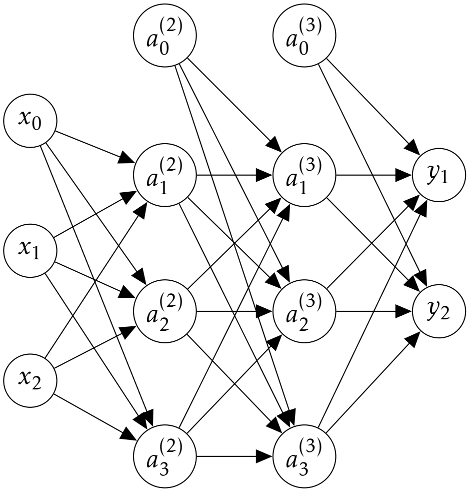


Figure 3.4: Example of neural network.

The input nodes in Figure 3.4 are x_1 and x_2, while the output nodes are y_1 and y_2. All a nodes reside in the hidden layers. The task for all hidden layer and output nodes are to evaluate the results from the previous nodes. The nodes x_0, $a_0^{(2)}$, and $a_0^{(3)}$ are bias values that usually are set to one.

Semantics of the neural network:

- The indexes i and j symbolize nodes and layers. In some cases like $\Theta_{i,j}$, i and j will symbolize matrix indexes.
- To each link in the network one weight is attached, all the weights are stored in the matrix $\Theta^{(j)}$. $\Theta_{1,0}^{(1)}$ is the weight connected the link between node x_1 and $a_1^{(2)}$, so $\Theta^{(j)}$ represents the weights between layer j and $j+1$.
- Each node ($a_i^{(j)}$) is connected to one activation function ($g(...)$) which determine the output from the node. $A^{(j)}$ is a vector that contains all $a_i^{(j)}$ nodes in the layer j. See equation below:

$$\begin{aligned} in^{(j)} &= \Theta^{(j)} A^{(j)} \\ A^{(j+1)} &= g(in^{(j)}) \end{aligned}$$

There are many different activation functions that determine the characteristics of the network. A commonly used soft threshold is the sigmoid function [Ng, 2014], see Figure 3.5.

$$g(z) = \frac{1}{1 + e^{-z}} \tag{3.5}$$

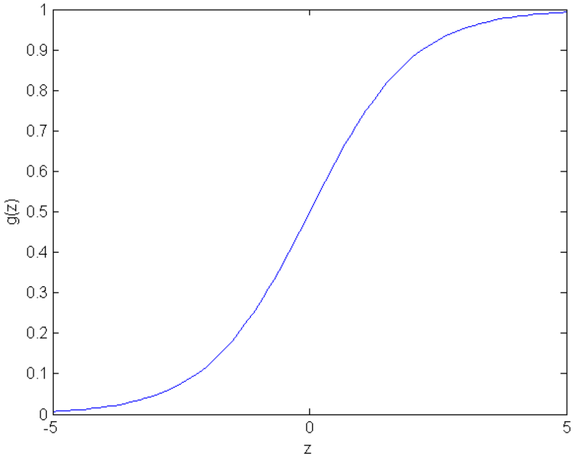


Figure 3.5: Sigmoid Function, $g(z)$

Example 3.4

If the inputs x_1, x_2 are known, the neural network (Figure 3.4) output is calculated by forward propagation as shown below. The output from hidden layer one (layer two) is $a_1^{(2)}$, $a_2^{(2)}$, and $a_3^{(2)}$. The output from hidden layer one is then handled by hidden layer two (layer three). There the output is $a_1^{(3)}$, $a_2^{(3)}$, and $a_3^{(3)}$. Finally these results are handled by the output nodes which give y_1 and y_2.

$$\begin{aligned}
a_1^{(2)} &= g(\Theta_{1,0}^{(1)}x_0 + \Theta_{1,1}^{(1)}x_1 + \Theta_{1,2}^{(1)}x_2)\\
a_2^{(2)} &= g(\Theta_{2,0}^{(1)}x_0 + \Theta_{2,1}^{(1)} * x_1 + \Theta_{2,2}^{(1)}x_2)\\
a_3^{(2)} &= g(\Theta_{3,0}^{(1)}x_0 + \Theta_{3,1}^{(1)}x_1 + \Theta_{3,2}^{(1)}x_2)\\
a_1^{(3)} &= g(\Theta_{1,0}^{(2)}a_0^{(2)} + \Theta_{1,1}^{(2)}a_1^{(2)} + \Theta_{1,2}^{(2)}a_2^{(2)} + \Theta_{1,3}^{(2)}a_3^{(2)})\\
a_2^{(3)} &= g(\Theta_{2,0}^{(2)}a_0^{(2)} + \Theta_{2,1}^{(2)}a_1^{(2)} + \Theta_{2,2}^{(2)}a_2^{(2)} + \Theta_{2,3}^{(2)}a_3^{(2)})\\
a_3^{(3)} &= g(\Theta_{3,0}^{(2)}a_0^{(2)} + \Theta_{3,1}^{(2)}a_1^{(2)} + \Theta_{3,2}^{(2)}a_2^{(2)} + \Theta_{3,3}^{(2)}a_3^{(2)})\\
y_1 &= g(\Theta_{1,0}^{(3)}a_0^{(3)} + \Theta_{1,1}^{(3)}a_1^{(3)} + \Theta_{1,2}^{(3)}a_2^{(3)} + \Theta_{1,3}^{(3)}a_3^{(3)})\\
y_2 &= g(\Theta_{2,0}^{(3)}a_0^{(3)} + \Theta_{2,1}^{(3)}a_1^{(3)} + \Theta_{2,2}^{(3)}a_2^{(3)} + \Theta_{2,3}^{(3)}a_3^{(3)})
\end{aligned}$$

3.4.1 Learning

In this section, layer will be denoted l and i, j will denote row and column in a matrix respectively.

If a neural network has been designed, the next step is to calculate all the weights. This can be done through an algorithm called back propagation and an error optimization function (minimizing error). A prerequisite is to choose a cost function [Ng, 2014], as in equation (3.6). The cost function is used to calculate how good the current weight values are. Some derivative of this function point to a better set of weight values. The idea is to use this derivative to obtain optimal weight values. Note that this is a very brief explanation. There are many things that need to be taken into account during calculations as the one described above.

$$J(\Theta) = -\frac{1}{m}\sum_{n=1}^{m}\sum_{k=1}^{K} y_k^{(n)} log(g(x^{(n)})_k) + (1 - y_k^{(n)}) log(1 - g(x^{(n)})_k) + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\Theta_{(i,j)}^{(l)} \tag{3.6}$$

m = number of entries in the training set

K = number of output nodes

s_l = number of nodes in layer l, without the bias node

In order to use an error minimizing method like gradient decent or the inbuilt *fminunc* in Matlab, the partial derivatives need to be calculated:

$$\frac{\delta}{\delta\Theta_{i,j}^{(l)}} J(\Theta)$$

This can be done through back propagation. The error in the output nodes needs to be propagated backwards. The idea is that all nodes in the previous layer contribute to the error in the current node. The equation (3.7) can be used to propagate the error backwards, where the dot represent pointwise product, [Ng, 2014].

$$\delta_l = \Theta^{(l)}\delta^{(l+1)} \cdot g'(\Theta^{(l)}A^{(l)}) \tag{3.7}$$

The errors are not the goal of back propagation. It is one necessary step in order to calculate the partial derivatives ($\frac{\delta}{\delta\Theta_{i,j}^{(l)}}$) of all weights.

Back propagation algorithm outline [Ng, 2014]:

- set all $\Delta_{i,j}^{(l)} := 0$

- For training all examples $(X_1, Y_1), ..., (X_n, Y_n)$ do
  - Set $A^{(1)} = X$
  - Do forward propagation
  - Calculate $\delta^{(L)} = Y - A^{(k)}$ (k = output layer).
  - Compute $\delta^{(l)} = \Theta^{(l)}\delta^{(l+1)} \cdot g'(\Theta^{(j)}A^{l})$ for $\delta^{(L-1)}, \delta^{(L-2)}, ..., \delta^{(2)}$
  - $\Delta^{(l)}_{i,j} := \Delta^{(l)}_{i,j} + a^{(l)}_j \delta^{l+1}_i$
- $D^{(l)}_{i,j} := \frac{1}{m}(\Delta^{(l)}_{i,j} + \lambda\Theta^{(l)}_{i,j})$ if $j \neq 0$
- $D^{(l)}_{i,j} := \frac{1}{m}\Delta^{(l)}_{i,j}$ if $j = 0$

The regularisation term λ is not applied to the bias values, i.e. the case $j = 0$. It is possible to show $D_{i,j}$ is equal to the partial derivatives [Ng, 2014].

$$\frac{\delta}{\delta\Theta^{(l)}_{i,j}} = D_{i,j}$$

The last step is to use an error optimization method that uses the partial derivatives like *fminunc*. This method is rather slow and a function called *fmincg* [Ng, 2014] has been used instead.

4

Analysis

4.1 Case Study

The methods Bayesian network, matrix correlation approach, and neural network will be evaluated through five data sets that have been generated for this purpose. Chapter 2 presented available data, and the difficulties of acquiring real data sets. In short the real data was inaccessible, due to the amount of time it would take to construct real data sets. This difficulty leads to the solution of generating data sets in order evaluate the chosen methods. Advantages and disadvantages of each method will be discussed. In the end, one method is chosen for further analysis.

The case study will be performed on the SUS unit, see Chapter 2 for faults, fault codes, and symptoms. The SUS is considered independent in the case study, which means that it is not affected by faults in other components. It is important to point out that this assumption does not reflect reality in a good way. Even with this disadvantage, the modelling will suffice to show strengths and weaknesses of the methods.

The **single fault assumption** is made, which means that it is assumed that only one fault can be present at a time. This assumption is made to give the methods a common evaluation base and a reasonable scope. Therefore, this assumption is reasonable, because the primary goal is to compare the methods. The assumption is necessary for the neural network, because the training set only consists of single faults. In order to be able to recognize these cases a neural network would need an extended data set with multiple faults. The Bayesian network and matrix correlation approach do not suffer in this way. In short, a scenario with multiple faults is likely to lead to posterior probabilities and similarity measures in the same range for the active faults.

4.1.1 Input and Output

The same input and output is used for all the methods. The input fault codes for the methods are FMI, PID, and SID. In total 15 FMI codes, 15 SID and PID codes, and 16 symptoms (denoted S) exist. The input fault codes and symptoms are represented by a vector, where each instance is either zero or one, which means active or not active. In this evaluation the assumption that the fault codes, symptoms, and malfunctions are either active or not is made, and thus they can be represented by zeros and ones. The number of malfunctions (M) are 15, which are represented by a vector in the same way as the input. See Example 4.1.

- Input: vector, dimension $\mathbb{R}^{46\times1}$ (46 fault codes and symptoms)
- Output: vector, dimension $\mathbb{R}^{15\times1}$ (15 malfunctions)

Example 4.1

One example of input and output vectors is presented below. The zeros denote inactive, while ones denote active.

$$\text{input} = \begin{bmatrix} 0 & 0 & 1 & 0 & \dots & 0 & 1 \end{bmatrix}$$
$$\text{output} = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 1 & 0 \end{bmatrix}^T$$

4.1.2 Generation of Data

Five data sets are generated through the ideal connections that can be seen in the Tables 2.1, 2.2 and 2.3 in Chapter 2. Each generation is done by evaluating probability values which will lead to different data sets each generation. These probability values are changed in certain data set generations in order to reflect different situations. By generating data sets this way irregularity in the real world is taken into account. Data set 1 is intended for learning parameters in the algorithms, while the other are validation sets. The validation sets reflect different situations. Data set 2 is generated the same way as data set 1, because the generation process will give a slightly different set. This is done to test the algorithms in a similar situation. In practise false alarms and changed dependencies between components might occur. Data sets 3 and 4 are generated to reflect this fact. Data set 5 contains all possible cases, which makes it possible to find weaknesses for certain test cases. See Table 4.1 for all data sets and their sizes.

The method behind the generation of data sets 1, 2, 3, and 4 are based on probabilities, that specify how likely fault codes and symptoms are to occur in regard to a certain malfunction. The probabilities for all fault codes and symptoms can be represented as the tables in Chapter 2, describing the ideal relationship between malfunctions, fault codes, and symptoms. All slots in the tables are replaced by probability values, see Tables 4.2, 4.3, and 4.4. The probability values represent how strongly the different fault codes, symptoms, and malfunctions are connected to each other. These values were arbitrarily chosen with the consideration that they should be reasonable. Reasonable means that dependencies given by the repair manual should have probability values large enough to represent them. The false alarm rates were chosen to be rather small in the training set, so they would not be overrepresented. In order to test how the methods handle changed dependencies validation set 3 and 4 contain different probabilities for false alarms and dependencies respectively. In Table 4.1 the sizes of the data sets vary. This is due to the fact that the generation process in some cases result in entries with no active fault codes, symptoms, or faults at all. These cases are removed from the data sets, and therefore the size vary. The generation process for all data sets, except data set 5, follows below:

Data generation with probabilities:

1. Set n = size of data set
2. Create empty input and output matrices, *input* and *output*
3. For each entry in n
   - (a) Choose one malfunction from the uniform distribution
   - (b) Get the corresponding fault codes and symptoms, c
   - (c) For each entry in c
     - i. Use the corresponding probability value to randomize the activity, either 0 or 1
     - ii. Assign the outcome to the result, x
   - (d) Set malfunction in result, y
   - (e) If *input* has at least one active fault code or symptom, save the results x and y in *input* and *output*

The generation process of data set 5 is different because this set contains all possible cases. In other words all combinations of malfunctions, fault codes, and symptoms that are possible are represented in data set 5. The generation procedure follow below:

Data generation of all cases:

- The matrix *ideal*, contain all ideal connections between malfunctions, fault codes, and symptoms
- For each malfunction, m
  1. Find all indexes i in *ideal* that are connected to m
  2. Find all combinations c of the entries in i
  3. For each entry in c

(a) Set all malfunctions in output y that are consistent with the combination

(b) Set input x to the combination

4. Save result x and y

Table 4.1: Data set description and size

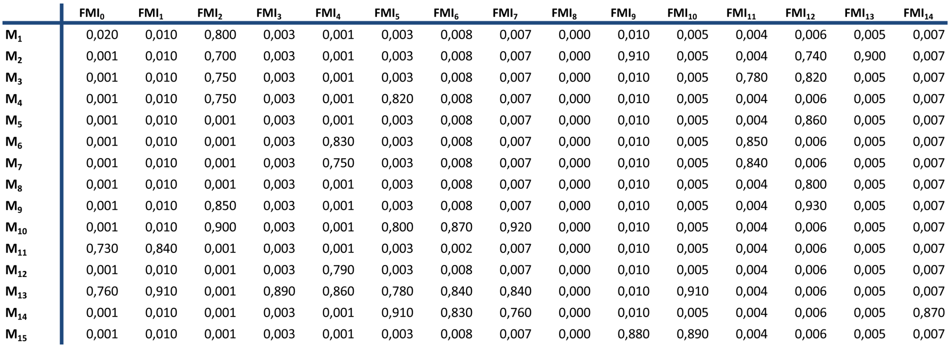
Data set	Description	No. of fault cases
1	training set	9943
2	validation set (generated the same way as the training set)	9936
3	validation set (higher false alarm rate)	9984
4	validation set (changed dependencies)	9984
5	validation set (all possible combinations)	1669

See Chapter A for implementation of data generation in Matlab.

4.1.3 Training Data Set

The probability values used for generation (Section 4.1.2) of the training set are shown in the Tables 4.2, 4.3 and 4.4. Note that false alarms are included as well, but the rate for false alarms is low.

Table 4.2: Probabilities for FMI fault codes during data generation



	FMI_0	FMI_1	FMI_2	FMI_3	FMI_4	FMI_5	FMI_6	FMI_7	FMI_8	FMI_9	FMI_{10}	FMI_{11}	FMI_{12}	FMI_{13}	FMI_{14}
M_1	0,020	0,010	0,800	0,003	0,001	0,003	0,008	0,007	0,000	0,010	0,005	0,004	0,006	0,005	0,007
M_2	0,001	0,010	0,700	0,003	0,001	0,003	0,008	0,007	0,000	0,910	0,005	0,004	0,740	0,900	0,007
M_3	0,001	0,010	0,750	0,003	0,001	0,003	0,008	0,007	0,000	0,010	0,005	0,780	0,820	0,005	0,007
M_4	0,001	0,010	0,750	0,003	0,001	0,820	0,008	0,007	0,000	0,010	0,005	0,004	0,006	0,005	0,007
M_5	0,001	0,010	0,001	0,003	0,001	0,003	0,008	0,007	0,000	0,010	0,005	0,004	0,860	0,005	0,007
M_6	0,001	0,010	0,001	0,003	0,830	0,003	0,008	0,007	0,000	0,010	0,005	0,850	0,006	0,005	0,007
M_7	0,001	0,010	0,001	0,003	0,750	0,003	0,008	0,007	0,000	0,010	0,005	0,840	0,006	0,005	0,007
M_8	0,001	0,010	0,001	0,003	0,001	0,003	0,008	0,007	0,000	0,010	0,005	0,004	0,800	0,005	0,007
M_9	0,001	0,010	0,850	0,003	0,001	0,003	0,008	0,007	0,000	0,010	0,005	0,004	0,930	0,005	0,007
M_{10}	0,001	0,010	0,900	0,003	0,001	0,800	0,870	0,920	0,000	0,010	0,005	0,004	0,006	0,005	0,007
M_{11}	0,730	0,840	0,001	0,003	0,001	0,003	0,002	0,007	0,000	0,010	0,005	0,004	0,006	0,005	0,007
M_{12}	0,001	0,010	0,001	0,003	0,790	0,003	0,008	0,007	0,000	0,010	0,005	0,004	0,006	0,005	0,007
M_{13}	0,760	0,910	0,001	0,890	0,860	0,780	0,840	0,840	0,000	0,010	0,910	0,004	0,006	0,005	0,007
M_{14}	0,001	0,010	0,001	0,003	0,001	0,910	0,830	0,760	0,000	0,010	0,005	0,004	0,006	0,005	0,870
M_{15}	0,001	0,010	0,001	0,003	0,001	0,003	0,008	0,007	0,000	0,880	0,890	0,004	0,006	0,005	0,007

Table 4.3: Probabilities for PID and SID fault codes during data generation

	SID_{240}	SID_{253}	SID_{254}	$PSID_{234}$	$PSID_1$	$PPID_{393}$	PID_{55}	$PPID_{55}$	$PPID_{424}$	$PPID_{426}$	$PPID_{427}$	$PSID_2$	$PSID_3$	$PSID_4$	$PSID_6$
M_1	0,600	0,005	0,004	0,003	0,006	0,003	0,006	0,008	0,004	0,005	0,007	0,008	0,002	0,010	0,005
M_2	0,004	0,690	0,004	0,003	0,006	0,003	0,006	0,008	0,004	0,005	0,007	0,008	0,002	0,010	0,005
M_3	0,004	0,005	0,700	0,003	0,006	0,003	0,006	0,008	0,004	0,005	0,007	0,008	0,002	0,010	0,005
M_4	0,004	0,005	0,004	0,790	0,006	0,003	0,006	0,008	0,004	0,005	0,007	0,008	0,002	0,010	0,005
M_5	0,004	0,005	0,004	0,003	0,650	0,003	0,006	0,008	0,004	0,005	0,007	0,008	0,002	0,010	0,005
M_6	0,004	0,005	0,004	0,003	0,006	0,760	0,006	0,008	0,004	0,005	0,007	0,008	0,002	0,010	0,005
M_7	0,004	0,005	0,004	0,003	0,006	0,003	0,680	0,008	0,004	0,005	0,007	0,008	0,002	0,010	0,005
M_8	0,004	0,005	0,004	0,003	0,006	0,003	0,006	0,850	0,004	0,005	0,007	0,008	0,002	0,010	0,005
M_9	0,004	0,005	0,004	0,003	0,006	0,003	0,006	0,008	0,740	0,005	0,007	0,008	0,002	0,010	0,005
M_{10}	0,004	0,005	0,004	0,003	0,006	0,003	0,006	0,008	0,004	0,720	0,007	0,008	0,002	0,010	0,005
M_{11}	0,004	0,005	0,004	0,003	0,006	0,003	0,006	0,008	0,004	0,005	0,700	0,008	0,002	0,010	0,005
M_{12}	0,004	0,005	0,004	0,003	0,006	0,003	0,006	0,008	0,004	0,005	0,007	0,750	0,002	0,010	0,005
M_{13}	0,004	0,005	0,004	0,003	0,006	0,003	0,006	0,008	0,004	0,005	0,007	0,008	0,650	0,010	0,005
M_{14}	0,004	0,005	0,004	0,003	0,006	0,003	0,006	0,008	0,004	0,005	0,007	0,008	0,002	0,710	0,005
M_{15}	0,004	0,005	0,004	0,003	0,006	0,003	0,006	0,008	0,004	0,005	0,007	0,008	0,002	0,010	0,680

Table 4.4: Probabilities for symptoms during data generation

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}
M_1	0,800	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,010	0,004	0,003	0,005	0,005	0,004
M_2	0,020	0,800	0,850	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,010	0,004	0,003	0,005	0,005	0,004
M_3	0,700	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,010	0,004	0,003	0,005	0,005	0,004
M_4	0,020	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,010	0,004	0,003	0,005	0,005	0,004
M_5	0,020	0,010	0,005	0,700	0,650	0,600	0,010	0,005	0,003	0,002	0,010	0,004	0,003	0,005	0,005	0,004
M_6	0,020	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,010	0,004	0,003	0,005	0,005	0,004
M_7	0,020	0,010	0,005	0,008	0,010	0,020	0,670	0,720	0,650	0,002	0,010	0,004	0,003	0,005	0,005	0,004
M_8	0,020	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,800	0,010	0,004	0,003	0,005	0,005	0,004
M_9	0,020	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,740	0,002	0,010	0,004	0,003	0,005	0,005	0,004
M_{10}	0,020	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,800	0,800	0,003	0,005	0,005	0,004
M_{11}	0,020	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,700	0,004	0,003	0,005	0,005	0,004
M_{12}	0,020	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,010	0,004	0,003	0,005	0,005	0,004
M_{13}	0,020	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,660	0,004	0,003	0,005	0,005	0,004
M_{14}	0,020	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,010	0,004	0,690	0,005	0,005	0,004
M_{15}	0,020	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,700	0,004	0,003	0,600	0,590	0,700

4.1.4 Validation Data Sets

The validation sets are generated by using the generation procedures presented in 4.1.2. A short summary of all the data sets can be seen below:

- **Second data set:** generated the same way as the training set
- **Third data set:** higher false alarm rate
- **Fourth data set:** changed probabilities for fault codes and symptoms
- **Fifth data set:** all possible cases

Data sets 3 and 4 are generated in the same manner as the training set, but differs due to different probabilities for fault codes and symptoms. See Chapter A for probability values that are used during the generation of data set 3 and 4.

Data set 5 contains all possible combinations of malfunctions, fault codes, and symptoms. Cases that are not likely in reality are therefore included, for example when almost all fault codes and symptoms fail to activate.

There is one big difference between the fifth data set and the others. **The fifth data set contain multiple faults as long as the consistency of the active fault codes and symptoms are preserved. The other sets only contain one active fault in each case**. All methods should perform well on the fifth data set, because a high error rate imply that the methods have problems with some combinations.

4.1.5 Validation method

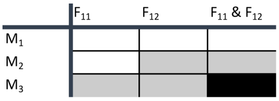
All methods are validated by the validation sets and a few selected cases from the repair manual. These specific fault cases are considered in order to analyse the behavior of the methods more closely. Table 4.5 shows the selected cases and brief explanations of each one. Ideal cases simply state dependencies exactly as the repair manual presents them. Therefore ideal cases correspond to a case there all fault codes, and symptoms that are sensitive to one malfunction are active. The specific cases have been constructed manually by setting zeros and ones in the input vector. Note that the input vector contains ones and zeros, which means active and not active.

Table 4.5: Selected validation cases

Case	Description
Ideal case	All the ideal connections shown in Tables 2.2, 2.3, 2.4 in Chapter 2
Single FMI's	The input is each FMI code individually
Specific fault case 1	Active: FMI_1 and FMI_2
Specific fault case 2	Active: FMI_4 and FMI_{11}
Specific fault case 3	Active: FMI_7 and S_{11}
Specific fault case 4	Active: FMI_7, S_{11}, and $PSID_3$

The tables in the validation sections will contain boxes highlighted grey and black. A grey box signifies that a malfunction is sensitive to either a single fault code or symptom, and combinations of these. A black box denotes a malfunction that have most in common with the given input. Table 4.6 show one example of one table with three different inputs, F_1, F_2, and F_1 & F_2. The grey boxes in the first two columns represent the malfunctions that are connected to the fault codes. Both fault codes in the last input have M_3 in common, which is the reason behind the black box in the table.

Table 4.6: Example of a table with results between malfunctions and fault codes with highlighted boxes.



	F_{11}	F_{12}	F_{11} & F_{12}
M_1			
M_2		▨	▨
M_3	▨	▨	■

The validation performance values, presented as percentage, are calculated by using a data set as input. The results are compared to the data set output, and from this a value of correctness in percentage is obtained. This correctness value is obtained by dividing the total number of correct outputs by the total number of outputs. See Chapter A for validation code.

Real data specified according to the repair manual could resemble the data sets 1, 2, 3, and 4. These data sets have all been generated with uncertainty in mind, which reflects reality in many ways. This makes the results from the case study credible.

4.1.6 Matrix Correlation Approach

The correlation matrix is created by counting each occurrence of fault codes and symptoms relative to all malfunctions. The correlation numbers are all divided by the total number of data entries. This is done to get an easier representation in the range zero to one. Cosine similarity [Li and Han, 2013] is used as similarity measure, and the reason is stated in Chapter 3.

No negative values exist in the correlation matrix, which means that the similarity measure will be in the range zero to one. Cosine similarity compensates for the fact that certain entries in the correlation matrix might be large, see Example 3.2.

Validation of Matrix Correlation

The highlighted boxes in Tables 4.7, 4.8, and 4.9 are explained in Section 4.1.5. Similarity measures state how much a fault case resemble the training set for each malfunction. Note that this is not a perfect measure of similarity, see Chapter 3. The resulting malfunction in all fault cases is the one with highest similarity measure in each column.

Table 4.7 shows the results for all the ideal cases, see Section 4.1.5 for explanation. All the values in the highlighted diagonal are considerably larger compared to the other values in each column, which shows that the matrix correlation approach can identify the correct fault. All case numbers correspond to the same malfunction number, e.g. case 1 and malfunction 1. All numbers are larger than zero in Table 4.7 due to the false alarms in the training set. This is realistic since real data is very likely to contain incorrect instances. Note that the individual fault codes in Table 4.8 are investigated in order to analyse behavior in some basic cases. First of all one observation is that all large values reside in the grey boxes. The correlation matrix learned from data set 1, therefore reflect the dependencies in Table 4.2. This table indicate how cosine similarity work. The fault code FMI_0 have correlation values 0.0508 and 0.0506 in regard to M_{11} and M_{13}. Note that the correlation values are in the same range but the results in column one in Table 4.2 does not follow suite. This is due to the cosine similarity. Basically the cosine similarity scales down the result for M_{13} more. This is due to the fact that malfunction M_{13} is sensitive to more fault codes and symptoms.

Table 4.9 reflects some interesting fault cases and their results. Columns one to three demonstrates similarity measures for fault cases with two fault codes. This is done to analyse how similarity measures reflect the dependencies shown by the grey and black boxes for fault case with multiple fault codes or symptoms.

Table 4.7: The results for the ideal cases. The black boxes represent rows and columns that are right according to the ideal connections.

Ideal Cases	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9	Case 10	Case 11	Case 12	Case 13	Case 14	Case 15
M1	0,990	0,245	0,564	0,361	0,017	0,003	0,009	0,005	0,565	0,246	0,028	0,003	0,023	0,007	0,017
M2	0,202	0,995	0,314	0,198	0,171	0,003	0,005	0,205	0,314	0,135	0,006	0,004	0,005	0,005	0,165
M3	0,494	0,367	0,996	0,265	0,237	0,273	0,198	0,291	0,606	0,180	0,007	0,005	0,009	0,012	0,010
M4	0,328	0,217	0,256	0,998	0,017	0,002	0,008	0,006	0,254	0,444	0,012	0,005	0,200	0,257	0,013
M5	0,008	0,213	0,249	0,002	0,993	0,002	0,009	0,316	0,251	0,009	0,010	0,004	0,011	0,008	0,010
M6	0,013	0,013	0,280	0,004	0,018	0,998	0,493	0,007	0,015	0,007	0,013	0,424	0,200	0,008	0,016
M7	0,010	0,009	0,218	0,002	0,012	0,521	0,996	0,004	0,175	0,007	0,006	0,308	0,141	0,004	0,011
M8	0,013	0,220	0,259	0,002	0,260	0,005	0,009	0,999	0,257	0,008	0,007	0,004	0,008	0,011	0,009
M9	0,305	0,417	0,490	0,297	0,264	0,002	0,188	0,334	0,896	0,204	0,010	0,005	0,010	0,014	0,009
M10	0,243	0,161	0,190	0,446	0,014	0,001	0,004	0,005	0,190	0,996	0,188	0,003	0,488	0,484	0,144
M11	0,013	0,011	0,012	0,003	0,015	0,002	0,008	0,005	0,014	0,183	0,995	0,005	0,492	0,011	0,185
M12	0,014	0,015	0,018	0,005	0,021	0,419	0,304	0,011	0,017	0,012	0,011	0,999	0,241	0,017	0,015
M13	0,006	0,006	0,008	0,173	0,010	0,196	0,141	0,004	0,008	0,460	0,455	0,240	0,993	0,392	0,235
M14	0,006	0,010	0,008	0,276	0,012	0,001	0,007	0,006	0,007	0,488	0,007	0,003	0,410	0,993	0,009
M15	0,010	0,177	0,007	0,004	0,010	0,001	0,006	0,002	0,006	0,147	0,189	0,002	0,267	0,012	0,989
Result	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15

Table 4.8: The results for each FMI. The grey boxes represent rows and columns that are correct according to the ideal connections. FMI 8 is never active in the SUS.

	FMI0	FMI1	FMI2	FMI3	FMI4	FMI5	FMI6	FMI7	FMI9	FMI10	FMI11	FMI12	FMI13	FMI14
M1	0,024	0,014	0,618	0,003	0,000	0,003	0,006	0,003	0,003	0,006	0,004	0,004	0,004	0,000
M2	0,001	0,004	0,340	0,000	0,001	0,001	0,002	0,001	0,421	0,004	0,001	0,350	0,421	0,003
M3	0,000	0,004	0,451	0,001	0,001	0,004	0,005	0,007	0,008	0,003	0,469	0,499	0,003	0,005
M4	0,001	0,009	0,544	0,000	0,000	0,600	0,009	0,003	0,007	0,002	0,002	0,003	0,003	0,000
M5	0,000	0,007	0,001	0,000	0,001	0,001	0,006	0,004	0,004	0,005	0,002	0,543	0,001	0,003
M6	0,000	0,009	0,000	0,001	0,599	0,003	0,003	0,000	0,009	0,004	0,598	0,007	0,005	0,003
M7	0,000	0,002	0,000	0,002	0,430	0,002	0,001	0,002	0,008	0,001	0,471	0,002	0,004	0,004
M8	0,000	0,007	0,000	0,001	0,001	0,002	0,001	0,005	0,007	0,001	0,006	0,554	0,005	0,007
M9	0,000	0,006	0,512	0,000	0,001	0,001	0,012	0,003	0,005	0,002	0,001	0,569	0,003	0,006
M10	0,000	0,006	0,410	0,002	0,001	0,360	0,393	0,419	0,004	0,002	0,001	0,003	0,003	0,005
M11	0,501	0,571	0,001	0,001	0,002	0,004	0,001	0,003	0,002	0,005	0,002	0,003	0,007	0,003
M12	0,000	0,009	0,001	0,001	0,721	0,003	0,009	0,007	0,005	0,005	0,005	0,008	0,007	0,012
M13	0,293	0,357	0,001	0,344	0,337	0,298	0,326	0,329	0,002	0,354	0,001	0,004	0,002	0,003
M14	0,000	0,006	0,000	0,001	0,000	0,476	0,417	0,390	0,007	0,001	0,001	0,004	0,002	0,443
M15	0,000	0,007	0,001	0,001	0,000	0,004	0,006	0,005	0,456	0,454	0,001	0,000	0,004	0,005
Result	M11	M11	M1	M13	M12	M4	M14	M10	M15	M15	M6	M9	M2	M14

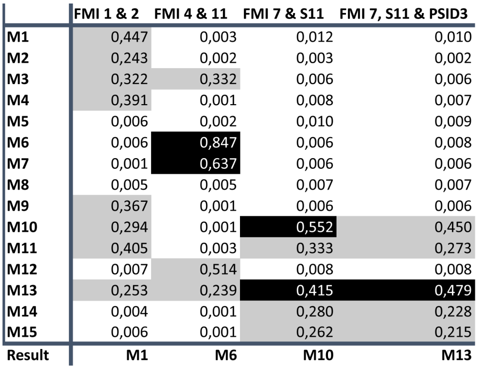
The last column demonstrate a case there one more fault code is know compared to column three. The results are consistent with the highlighted boxes, but the results from the case FMI_7, S_{11}, and $PSID_3$ are far from perfect. The similarity measures for M_{10} and M_{13} are close to each other. $PSID_3$ specifically points out M_{13}, so this result is not desired.

The calculation of the performance results in Table 4.10 are explained in Section 4.1.5. The MC approach manages to get good results in Table 4.10 for all data sets, but the result for the third and fifth data set is bit off compared to the others. The errors are due to the fact that the cosine similarity in some cases fail to give the correct result, see Example 4.2.

Example 4.2

The numbers in the vectors $input^i$ and $corr^i_{row}$ represent the column indexes of non-zero values of fault codes or symptoms in the input vector and correlation matrix rows. The input vector has size $\mathbb{R}^{46\times1}$ while the correlation matrix has the size $\mathbb{R}^{15\times46}$. Each row in the correlation matrix correspond to one malfunction. The interesting rows in the correlation matrix are row 11 and 13, i.e. $corr_{11}$ and $corr_{13}$. Note that the indexes in row $corr_{13}$ of the correlation matrix has more in common with the input indexes. The correlation values are not shown here, but

Table 4.9: Results for a few combinations of fault codes and symptoms. See Section 4.1.5 for an explanation of the highlighted boxes.



	FMI 1 & 2	FMI 4 & 11	FMI 7 & S11	FMI 7, S11 & PSID3
M1	0,447	0,003	0,012	0,010
M2	0,243	0,002	0,003	0,002
M3	0,322	0,332	0,006	0,006
M4	0,391	0,001	0,008	0,007
M5	0,006	0,002	0,010	0,009
M6	0,006	0,847	0,006	0,008
M7	0,001	0,637	0,006	0,006
M8	0,005	0,005	0,007	0,007
M9	0,367	0,001	0,006	0,006
M10	0,294	0,001	0,552	0,450
M11	0,405	0,003	0,333	0,273
M12	0,007	0,514	0,008	0,008
M13	0,253	0,239	0,415	0,479
M14	0,004	0,001	0,280	0,228
M15	0,006	0,001	0,262	0,215
Result	M1	M6	M10	M13

Table 4.10: Matrix correlation approach results from all data sets (MC).

	MC
Second data set	98.51%
Third data set	94.89%
Fourth data set	98.13%
Fifth data set	96.17%

the values in row 13 are considerably larger compared to those in row 11. The right answer is malfunction 13 but due to the cosine similarity the MC approach give malfunction 11 as the result.

Indexes in input represent active fault codes and symptoms:

$input^i = \begin{pmatrix} 1 & 2 & 5 & 8 & 41 \end{pmatrix}$

Indexes in the matrix rows represent if a fault are connected to a fault code or symptom:

$corr^i_{11} = \begin{pmatrix} 1 & 2 & 26 & 41 \end{pmatrix}$

$corr^i_{13} = \begin{pmatrix} 1 & 2 & 4 & 5 & 6 & 7 & 8 & 11 & 28 & 41 \end{pmatrix}$

The calculations of the cosine similarity for malfunction 11 and 13 is shown below:

Malfunction 11:

$$\begin{aligned} corr_{11} \cdot input &= 0.0265 \\ |corr_{11}||input| &= 0.0362 \\ \frac{corr_{11} \cdot input}{|corr_{11}||input|} &= 0.7317 \end{aligned}$$

Malfunction 13:

$$\begin{aligned} corr_{13} \cdot input &= 1.5626 \\ |corr_{13}||input| &= 2.2059 \\ \frac{corr_{11} \cdot input}{|corr_{13}||input|} &= 0.7084 \end{aligned}$$

The cosine similarity for malfunction 11 and 13 are 0.7317 respectively 0.7084. The norm of the two products in the second case will basically scale down the result too much and thus the cosine similarity will yield the wrong answer.

Example 4.2 applies to false alarms too, because the cosine similarity can scale up patterns that should not be that significant. The errors in the fifth set generally resemble Example 4.2.

4.1.7 Bayesian Networks

The Bayesian network has been constructed from the connections given by the repair manual, see Tables 2.2, 2.3, and 2.4. Note that all connections in a Bayesian network are drawn during the manual construction, compared to the data driven methods. Therefore it is not possible to get results that are not consistent with the repair manual. The Figure 4.1 show the Bayesian network used during the case study.

The Bayesian networks has been built and tested in the program GeNIe and with the C++ library SMILE [gen, 2014]. The interference algorithm that has been used is the *clustering algorithm*. The learning part of the parameters has been learned though the *EM-algorithm*. The EM-algorithm is basically the same as Bayesian estimation when the data set is complete, see Chapter 3. The node type is *Noisy-or* in the network with arbitrarily chosen CPT's, but of general type when parameters are learned. The learning algorithm could not be used with Noisy-or nodes. This does not pose as a problem, since GeNIe can convert Noisy-or nodes to general nodes.

Validation of Arbitrarily Chosen CPT's

The CPT's of the Bayesian network need conditional probability values. These values can be based on either expert knowledge or data sets. The probabilities representing the chance that a fault occur have been chosen to be rather small, since faults should not occur to frequently. The conditional probabilities have

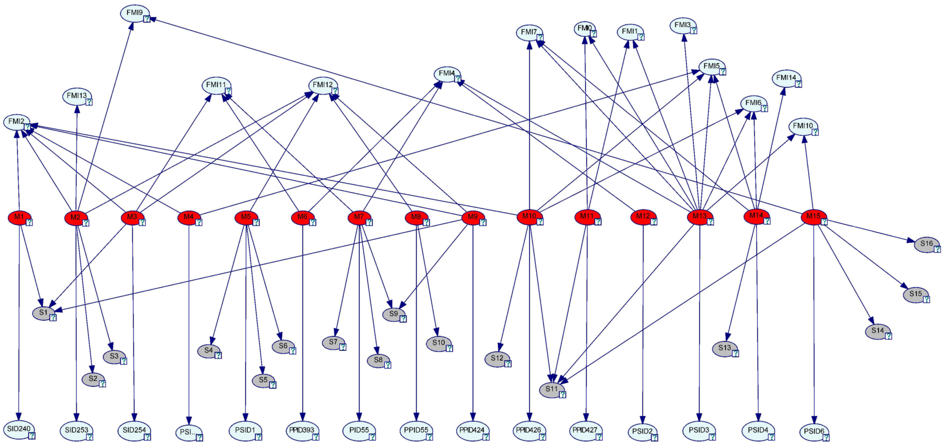


Figure 4.1: Bayesian network constructed from repair manual.

higher values, because malfunctions should activate fault codes and symptoms most of the time, but fail to do so at a lower frequency. This will incorporate uncertainty into the model.

The highlighted boxes in Tables 4.11, 4.12, and 4.13 are explained in Section 4.1.5. The resulting malfunction in all fault cases is the one with highest posterior probability in each column.

Table 4.11 shows the result for the ideal cases. The diagonal that corresponds to the correct malfunctions, holds the highest values, so the method can handle the ideal cases. The results in Table 4.12 behave as expected, since the large values are in the highlighted areas.

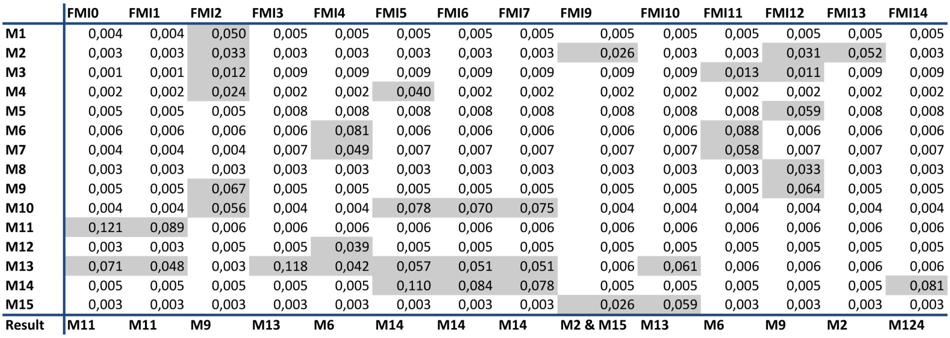
Table 4.11: The results for the ideal cases. The black boxes represent rows and columns that are right according to the ideal connections.

Ideal Cases	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	Result
case 1 (M1)	0,829	0,005	0,023	0,003	0,005	0,006	0,004	0,003	0,110	0,004	0,006	0,003	0,003	0,005	0,003	**M1**
case 2 (M2)	0,004	1,000	0,001	0,002	0,005	0,006	0,004	0,003	0,005	0,004	0,006	0,003	0,003	0,005	0,003	**M2**
case 3 (M3)	0,010	0,005	0,943	0,002	0,006	0,012	0,008	0,004	0,060	0,005	0,006	0,003	0,003	0,005	0,003	**M3**
case 4 (M4)	0,012	0,003	0,003	0,653	0,005	0,006	0,004	0,003	0,016	0,203	0,006	0,003	0,013	0,024	0,003	**M4**
case 5 (M5)	0,004	0,003	0,001	0,002	1,000	0,006	0,004	0,003	0,006	0,004	0,006	0,003	0,003	0,005	0,003	**M5**
case 6 (M6)	0,004	0,003	0,001	0,002	0,005	0,969	0,020	0,003	0,005	0,004	0,006	0,004	0,004	0,005	0,003	**M6**
case 7 (M7)	0,004	0,003	0,001	0,002	0,005	0,009	1,000	0,003	0,007	0,004	0,006	0,004	0,004	0,005	0,003	**M7**
case 8 (M8)	0,004	0,007	0,002	0,002	0,012	0,006	0,004	0,889	0,013	0,004	0,006	0,003	0,003	0,005	0,003	**M8**
case 9 (M9)	0,004	0,006	0,002	0,002	0,010	0,006	0,010	0,006	0,911	0,004	0,006	0,003	0,003	0,005	0,003	**M9**
case 10 (M10)	0,004	0,003	0,001	0,002	0,005	0,006	0,004	0,003	0,005	1,000	0,007	0,003	0,005	0,007	0,003	**M10**
case 11 (M11)	0,004	0,003	0,001	0,002	0,005	0,006	0,004	0,003	0,005	0,005	0,959	0,003	0,045	0,005	0,004	**M11**
case 12 (M12)	0,004	0,003	0,001	0,002	0,005	0,051	0,031	0,003	0,005	0,004	0,006	0,433	0,026	0,005	0,003	**M12**
case 13 (M13)	0,004	0,003	0,001	0,002	0,005	0,007	0,004	0,003	0,005	0,009	0,010	0,003	1,000	0,008	0,004	**M13**
case 14 (M14)	0,004	0,003	0,001	0,002	0,005	0,006	0,004	0,003	0,005	0,006	0,006	0,003	0,005	1,000	0,003	**M14**
case 15 (M15)	0,004	0,003	0,001	0,002	0,005	0,006	0,004	0,003	0,005	0,004	0,006	0,003	0,003	0,005	1,000	**M15**

The Bayesian network manages to recognize patterns in Table 4.13, since all the black boxes have significantly higher values. Note that FMI_1 and FMI_2 have no malfunction in common, and therefore all highlighted boxes are grey.

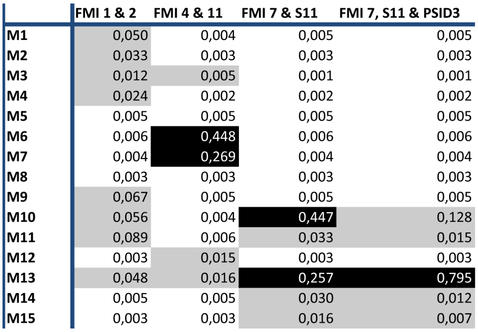
Table 4.14 shows results for all validation sets. The calculation of the performance results is explained in Section 4.1.5. Errors in the table are generally

Table 4.12: The results for each FMI. The grey boxes represent rows and columns that are correct according to the ideal connections. FMI 8 is never active in the SUS.



	FMI0	FMI1	FMI2	FMI3	FMI4	FMI5	FMI6	FMI7	FMI9	FMI10	FMI11	FMI12	FMI13	FMI14
M1	0,004	0,004	0,050	0,005	0,005	0,005	0,005	0,005	0,005	0,005	0,005	0,005	0,005	0,005
M2	0,003	0,003	0,033	0,003	0,003	0,003	0,003	0,003	0,026	0,003	0,003	0,031	0,052	0,003
M3	0,001	0,001	0,012	0,009	0,009	0,009	0,009	0,009	0,009	0,009	0,013	0,011	0,009	0,009
M4	0,002	0,002	0,024	0,002	0,002	0,040	0,002	0,002	0,002	0,002	0,002	0,002	0,002	0,002
M5	0,005	0,005	0,005	0,008	0,008	0,008	0,008	0,008	0,008	0,008	0,008	0,059	0,008	0,008
M6	0,006	0,006	0,006	0,006	0,081	0,006	0,006	0,006	0,006	0,006	0,088	0,006	0,006	0,006
M7	0,004	0,004	0,004	0,007	0,049	0,007	0,007	0,007	0,007	0,007	0,058	0,007	0,007	0,007
M8	0,003	0,003	0,003	0,003	0,003	0,003	0,003	0,003	0,003	0,003	0,003	0,033	0,003	0,003
M9	0,005	0,005	0,067	0,005	0,005	0,005	0,005	0,005	0,005	0,005	0,005	0,064	0,005	0,005
M10	0,004	0,004	0,056	0,004	0,004	0,078	0,070	0,075	0,004	0,004	0,004	0,004	0,004	0,004
M11	0,121	0,089	0,006	0,006	0,006	0,006	0,006	0,006	0,006	0,006	0,006	0,006	0,006	0,006
M12	0,003	0,003	0,005	0,005	0,039	0,005	0,005	0,005	0,005	0,005	0,005	0,005	0,005	0,005
M13	0,071	0,048	0,003	0,118	0,042	0,057	0,051	0,051	0,006	0,061	0,006	0,006	0,006	0,006
M14	0,005	0,005	0,005	0,005	0,005	0,110	0,084	0,078	0,005	0,005	0,005	0,005	0,005	0,081
M15	0,003	0,003	0,003	0,003	0,003	0,003	0,003	0,003	0,026	0,059	0,003	0,003	0,003	0,003
Result	**M11**	**M11**	**M9**	**M13**	**M6**	**M14**	**M14**	**M14**	**M2 & M15**	**M13**	**M6**	**M9**	**M2**	**M124**

Table 4.13: Results for a few combinations of fault codes and symptoms. See Section 4.1.5 for an explanation of the highlighted boxes.



	FMI 1 & 2	FMI 4 & 11	FMI 7 & S11	FMI 7, S11 & PSID3
M1	0,050	0,004	0,005	0,005
M2	0,033	0,003	0,003	0,003
M3	0,012	0,005	0,001	0,001
M4	0,024	0,002	0,002	0,002
M5	0,005	0,005	0,005	0,005
M6	0,006	0,448	0,006	0,006
M7	0,004	0,269	0,004	0,004
M8	0,003	0,003	0,003	0,003
M9	0,067	0,005	0,005	0,005
M10	0,056	0,004	0,447	0,128
M11	0,089	0,006	0,033	0,015
M12	0,003	0,015	0,003	0,003
M13	0,048	0,016	0,257	0,795
M14	0,005	0,005	0,030	0,012
M15	0,003	0,003	0,016	0,007

caused by the range interval of the calculated posterior probabilities after interference is done. When errors occur these values tend to be in the same range. All *PID* and *SID* codes have a strong connection to their respective malfunction in the Bayesian network, shown in Table 4.1. If a false alarm trigger one of these, there is a big risk that the connected malfunction gets the highest posterior probability value. This happens in many of the error cases. However some errors are caused by combinations of fault codes and symptoms that do not form a specific pattern, which tend to result in posterior probability values in the same range. Both error cases can be derived from the result by the third data set, which has a higher rate for false alarms. A higher rate for false alarms will result in more cases that correspond to the described error cases , i.e. the reason behind the result 84.08%. The Bayesian network scores 100% on the fifth data set. The reason behind this result is that the output in the fifth data set contain multiple faults. This is not the case in the other data sets where only one fault can be active. This further supports the conclusion that the probabilities tend to be in the same range. The Bayesian network is really sensitive to false alarms regarding *PID* and *SID*, these seem to be a bit more frequent in data set 2 compared to data set 4. Data set 4 gives a better result compared to data set 2, and the reason behind this is simply that data set 4 contains the same false alarm rates as data set 2. The conclusion is that the difference in the results depend on the random generation of the two data sets and the structure of the model. The similarity of the result from data set 2 and 4 show that the Bayesian network can handle changed dependencies.

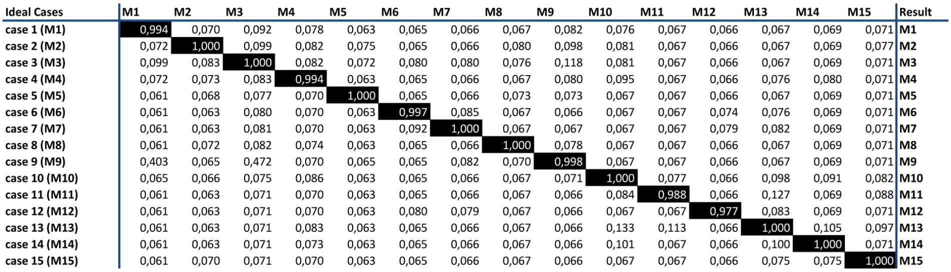
Table 4.14: The results from all validation sets on the Bayesian network with arbitrarily chosen CPT's. See Section 4.1.5 for more info about validation values.

	BN result
Second data set	92.88%
Third data set	84.08%
Fourth data set	93.33%
Fifth data set	100%

Validation of CPT's Learned from Training Set

The Bayesian network can learn parameters (CPT's) from a data set. This is done with the data set 1 in the program GeNIe. The initial parameters in the Bayesian network are the ones used in the previous section. The *EM-algorithm* is used for learning the parameters, and the confidence value indicates the reliability of the current parameters. If a high value is used, the changes in the parameters will be very small, therefore a rather small confidence value of 1 is used. This is done for demonstration purposes only. In reality it would depend on the credibility of the current parameters. Tables 4.15, 4.16, and 4.17 show the results for certain cases. All previously mentioned explanations of the highlighted boxes in the tables in Section 4.1.5 apply here as well. Table 4.17 shows that the learning procedure has changed the strength of the connections between patterns and malfunctions.

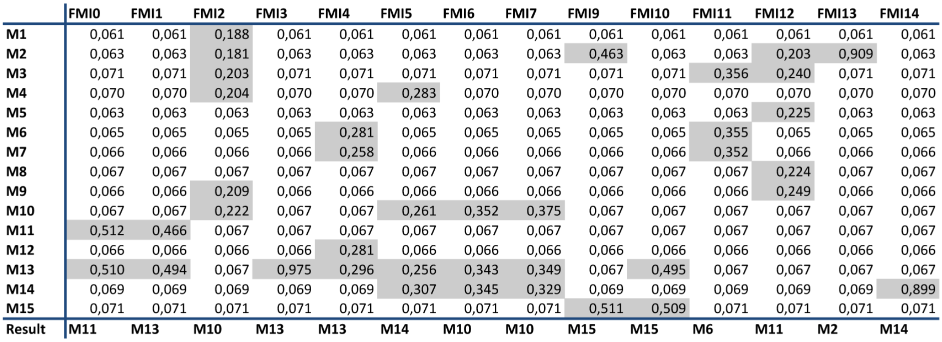
Table 4.15: The results for the ideal cases. The grey boxes represent rows and columns that are right according to the ideal connections.



Ideal Cases	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15	Result
case 1 (M1)	0,994	0,070	0,092	0,078	0,063	0,065	0,066	0,067	0,082	0,076	0,067	0,066	0,067	0,069	0,071	M1
case 2 (M2)	0,072	1,000	0,099	0,082	0,075	0,065	0,066	0,080	0,098	0,081	0,067	0,066	0,067	0,069	0,077	M2
case 3 (M3)	0,099	0,083	1,000	0,082	0,072	0,080	0,080	0,076	0,118	0,081	0,067	0,066	0,067	0,069	0,071	M3
case 4 (M4)	0,072	0,073	0,083	0,994	0,063	0,065	0,066	0,067	0,080	0,095	0,067	0,066	0,076	0,080	0,071	M4
case 5 (M5)	0,061	0,068	0,077	0,070	1,000	0,065	0,066	0,073	0,073	0,067	0,067	0,066	0,067	0,069	0,071	M5
case 6 (M6)	0,061	0,063	0,080	0,070	0,063	0,997	0,085	0,067	0,066	0,067	0,067	0,074	0,076	0,069	0,071	M6
case 7 (M7)	0,061	0,063	0,081	0,070	0,063	0,092	1,000	0,067	0,067	0,067	0,067	0,079	0,082	0,069	0,071	M7
case 8 (M8)	0,061	0,072	0,082	0,074	0,063	0,065	0,066	1,000	0,078	0,067	0,067	0,066	0,067	0,069	0,071	M8
case 9 (M9)	0,403	0,065	0,472	0,070	0,065	0,065	0,082	0,070	0,998	0,067	0,067	0,066	0,067	0,069	0,071	M9
case 10 (M10)	0,065	0,066	0,075	0,086	0,063	0,065	0,066	0,067	0,071	1,000	0,077	0,066	0,098	0,091	0,082	M10
case 11 (M11)	0,061	0,063	0,071	0,070	0,063	0,065	0,066	0,067	0,066	0,084	0,988	0,066	0,127	0,069	0,088	M11
case 12 (M12)	0,061	0,063	0,071	0,070	0,063	0,080	0,079	0,067	0,066	0,067	0,067	0,977	0,083	0,069	0,071	M12
case 13 (M13)	0,061	0,063	0,071	0,083	0,063	0,065	0,066	0,067	0,066	0,133	0,113	0,066	1,000	0,105	0,097	M13
case 14 (M14)	0,061	0,063	0,071	0,073	0,063	0,065	0,066	0,067	0,066	0,101	0,067	0,066	0,100	1,000	0,071	M14
case 15 (M15)	0,061	0,070	0,071	0,070	0,063	0,065	0,066	0,067	0,066	0,067	0,067	0,066	0,075	0,075	1,000	M15

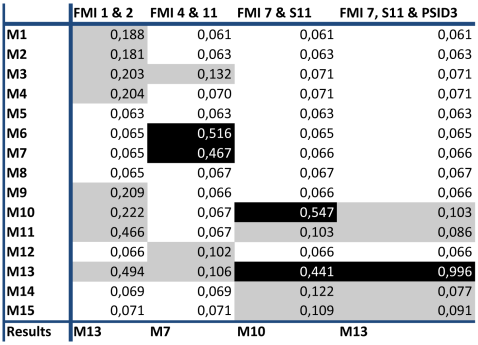
See Table 4.18 for results from all validation sets on the Bayesian network trained by the training set. The calculation of the performance results is explained in Section 4.1.5. The results resemble the results from the Bayesian network with arbitrarily chosen CPT's, and the reason behind the errors are the same.

Table 4.16: The results for each FMI. The grey boxes represent rows and columns that are right according to the ideal connections. FMI 8 is never active in the SUS.



	FMI0	FMI1	FMI2	FMI3	FMI4	FMI5	FMI6	FMI7	FMI9	FMI10	FMI11	FMI12	FMI13	FMI14
M1	0,061	0,061	0,188	0,061	0,061	0,061	0,061	0,061	0,061	0,061	0,061	0,061	0,061	0,061
M2	0,063	0,063	0,181	0,063	0,063	0,063	0,063	0,063	0,463	0,063	0,063	0,203	0,909	0,063
M3	0,071	0,071	0,203	0,071	0,071	0,071	0,071	0,071	0,071	0,071	0,356	0,240	0,071	0,071
M4	0,070	0,070	0,204	0,070	0,070	0,283	0,070	0,070	0,070	0,070	0,070	0,070	0,070	0,070
M5	0,063	0,063	0,063	0,063	0,063	0,063	0,063	0,063	0,063	0,063	0,063	0,225	0,063	0,063
M6	0,065	0,065	0,065	0,065	0,281	0,065	0,065	0,065	0,065	0,065	0,355	0,065	0,065	0,065
M7	0,066	0,066	0,066	0,066	0,258	0,066	0,066	0,066	0,066	0,066	0,352	0,066	0,066	0,066
M8	0,067	0,067	0,067	0,067	0,067	0,067	0,067	0,067	0,067	0,067	0,067	0,224	0,067	0,067
M9	0,066	0,066	0,209	0,066	0,066	0,066	0,066	0,066	0,066	0,066	0,066	0,249	0,066	0,066
M10	0,067	0,067	0,222	0,067	0,067	0,261	0,352	0,375	0,067	0,067	0,067	0,067	0,067	0,067
M11	0,512	0,466	0,067	0,067	0,067	0,067	0,067	0,067	0,067	0,067	0,067	0,067	0,067	0,067
M12	0,066	0,066	0,066	0,066	0,281	0,066	0,066	0,066	0,066	0,066	0,066	0,066	0,066	0,066
M13	0,510	0,494	0,067	0,975	0,296	0,256	0,343	0,349	0,067	0,495	0,067	0,067	0,067	0,067
M14	0,069	0,069	0,069	0,069	0,069	0,307	0,345	0,329	0,069	0,069	0,069	0,069	0,069	0,899
M15	0,071	0,071	0,071	0,071	0,071	0,071	0,071	0,071	0,511	0,509	0,071	0,071	0,071	0,071
Result	**M11**	**M13**	**M10**	**M13**	**M13**	**M14**	**M10**	**M10**	**M15**	**M15**	**M6**	**M11**	**M2**	**M14**

Table 4.17: Results for a few combinations of fault codes and symptoms. See Section 4.1.5 for an explanation of the highlighted boxes.



	FMI 1 & 2	FMI 4 & 11	FMI 7 & S11	FMI 7, S11 & PSID3
M1	0,188	0,061	0,061	0,061
M2	0,181	0,063	0,063	0,063
M3	0,203	0,132	0,071	0,071
M4	0,204	0,070	0,071	0,071
M5	0,063	0,063	0,063	0,063
M6	0,065	0,516	0,065	0,065
M7	0,065	0,467	0,066	0,066
M8	0,065	0,067	0,067	0,067
M9	0,209	0,066	0,066	0,066
M10	0,222	0,067	0,547	0,103
M11	0,466	0,067	0,103	0,086
M12	0,066	0,102	0,066	0,066
M13	0,494	0,106	0,441	0,996
M14	0,069	0,069	0,122	0,077
M15	0,071	0,071	0,109	0,091
Results	**M13**	**M7**	**M10**	**M13**

Table 4.18: The results from all validation sets on the Bayesian network (parameters learned by training set). See Section 4.1.5 for more info about validation values.

	BN result
Second data set	92.24%
Third data set	82.38%
Fourth data set	93.41%
Fifth data set	100%

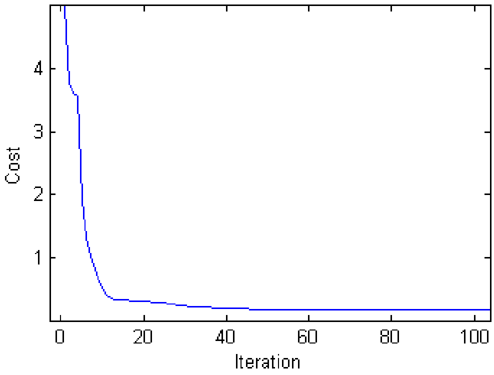
4.1.8 Neural Networks

The neural network will learn connections from the training set. The network that is used here consist of 46 input nodes, one hidden layer with 50 nodes, and one output layer with 15 nodes. All the parameters in the network has to be learned. This is done through back-propagation and minimizing the error in all nodes [Ng, 2014]. See Matlab code for neural network implementation in Chapter A.

Validation of Neural Network

The cost in each iteration is lowered until it converges, see Figure 4.2.

Figure 4.2: Cost value of cost function in every iteration



The highlighted boxes in Tables 4.19, 4.20, and 4.21 are explained in Section 4.1.5. The resulting malfunction in all fault cases is the one with highest value in each column.

The ideal cases in Table 4.19 are free from problems, since the diagonal that represents the right answers contains the largest values. The results in Table 4.20 are correct, but the values in the highlighted boxes differ greatly compared to each other. This is not a desired behavior, because the highlighted boxes points out malfunctions that the FMI's should be sensitive to. Table 4.21 show some bad results for the neural network. The boxes highlighted black should contain high values, since the patterns of fault codes and symptoms point to them specifically, see Tables 4.2, 4.3, and 4.4. For example FMI_7, S_{11}, and $PSID_3$ relate to M_{13}. FMI_4 and FMI_{11} should produce high values for both malfunction M_6 and M_7, but the neural network does this only for malfunction M_6. FMI_7 and symptom S_{11} should produce high values for malfunction M_{10} and M_{13}, but the neural network does this only for malfunction M_{10}. The last case consisting of FMI_7, S_{11}, and $PSID_3$ should produce the result M_{13}, the result is instead M_{10}.

Table 4.22 shows the results for all validation sets. The calculation of the results is explained in Section 4.1.5. The major drawback is the result from the fifth data set, which only is 83.82%. The neural network fails to make correct fault isolation based on data set 5, even if this set contains multiple faults as long as consistency of the repair manual is preserved. One explanation to this behavior is that the fifth data set contains cases that are not likely to occur in the training set, for example cases with single FMI codes or there all possible fault codes and symptoms are active. Further analysis shows that most errors happen when 2 to 6 fault codes and symptoms are active at the same time. The conclusion from

Table 4.19: The results for the ideal cases. The black boxes represent rows and columns that are correct according to the ideal connections.

Ideal Cases	Case 1	Case 2	Case 3	Case 4	Case 5	Case 6	Case 7	Case 8	Case 9	Case 10	Case 11	Case 12	Case 13	Case 14	Case 15
M1	0,999	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,001	0,000	0,001	0,000	0,000	0,000	0,000
M2	0,000	1,000	0,000	0,000	0,000	0,000	0,000	0,000	0,001	0,000	0,000	0,000	0,000	0,000	0,000
M3	0,003	0,001	1,000	0,000	0,001	0,000	0,000	0,001	0,003	0,000	0,000	0,000	0,000	0,000	0,000
M4	0,001	0,000	0,000	1,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
M5	0,000	0,000	0,000	0,000	1,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
M6	0,000	0,000	0,000	0,000	0,000	1,000	0,000	0,000	0,000	0,000	0,000	0,003	0,000	0,000	0,000
M7	0,000	0,000	0,000	0,000	0,000	0,003	1,000	0,000	0,000	0,000	0,000	0,001	0,000	0,000	0,000
M8	0,000	0,000	0,000	0,000	0,000	0,000	0,000	1,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
M9	0,000	0,001	0,000	0,000	0,000	0,000	0,000	0,001	1,000	0,000	0,000	0,000	0,000	0,000	0,000
M10	0,001	0,000	0,000	0,001	0,000	0,000	0,000	0,000	0,000	1,000	0,000	0,000	0,001	0,001	0,000
M11	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	1,000	0,000	0,000	0,000	0,000
M12	0,000	0,000	0,000	0,000	0,000	0,001	0,000	0,000	0,000	0,000	0,000	1,000	0,000	0,000	0,000
M13	0,000	0,000	0,000	0,000	0,000	0,001	0,000	0,000	0,000	0,001	0,002	0,001	1,000	0,001	0,001
M14	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	1,000	0,000
M15	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	1,000
Result	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10	M11	M12	M13	M14	M15

Table 4.20: The results for each FMI. The grey boxes represent rows and columns that are correct according to the ideal connections. FMI 8 is never active in the SUS.

	FMI0	FMI1	FMI2	FMI3	FMI4	FMI5	FMI6	FMI7	FMI9	FMI10	FMI11	FMI12	FMI13	FMI14
M1	0,039	0,006	0,564	0,075	0,002	0,001	0,023	0,007	0,016	0,024	0,001	0,001	0,078	0,021
M2	0,000	0,000	0,001	0,000	0,000	0,000	0,000	0,000	0,010	0,000	0,000	0,001	0,018	0,000
M3	0,000	0,000	0,008	0,000	0,000	0,000	0,000	0,001	0,000	0,000	0,193	0,066	0,001	0,001
M4	0,004	0,014	0,359	0,004	0,001	0,912	0,002	0,001	0,009	0,005	0,002	0,001	0,022	0,002
M5	0,002	0,004	0,000	0,005	0,001	0,001	0,006	0,002	0,004	0,005	0,002	0,193	0,004	0,007
M6	0,003	0,007	0,000	0,004	0,179	0,002	0,002	0,001	0,003	0,004	0,715	0,001	0,003	0,009
M7	0,001	0,001	0,000	0,001	0,009	0,000	0,000	0,000	0,002	0,001	0,025	0,000	0,001	0,001
M8	0,004	0,013	0,001	0,010	0,002	0,002	0,003	0,009	0,004	0,004	0,003	0,418	0,005	0,013
M9	0,001	0,001	0,045	0,000	0,000	0,000	0,000	0,001	0,000	0,000	0,000	0,054	0,000	0,001
M10	0,000	0,000	0,002	0,000	0,000	0,000	0,002	0,004	0,000	0,000	0,000	0,000	0,000	0,000
M11	0,798	0,761	0,004	0,016	0,002	0,002	0,008	0,008	0,005	0,005	0,004	0,004	0,021	0,025
M12	0,004	0,033	0,002	0,012	0,813	0,001	0,033	0,024	0,017	0,010	0,001	0,002	0,041	0,082
M13	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000	0,000
M14	0,001	0,001	0,000	0,001	0,000	0,029	0,026	0,017	0,001	0,000	0,001	0,001	0,001	0,113
M15	0,001	0,001	0,001	0,009	0,001	0,001	0,002	0,002	0,261	0,230	0,001	0,000	0,001	0,002
Result	M11	M11	M1	M1	M12	M4	M12	M12	M15	M15	M6	M8	M1	M14

Table 4.21: Results for a few combinations of fault codes and symptoms. See Section 4.1.5 for an explanation of the highlighted boxes.

	FMI 1 & 2	FMI 4 & 11	FMI 7 & S11	FMI 7, S11 & PSID3
M1	0,087	0,000	0,004	0,001
M2	0,001	0,000	0,000	0,000
M3	0,002	0,003	0,000	0,000
M4	0,234	0,000	0,000	0,000
M5	0,000	0,000	0,001	0,000
M6	0,000	0,950	0,000	0,001
M7	0,000	0,041	0,000	0,000
M8	0,000	0,000	0,002	0,001
M9	0,029	0,000	0,001	0,000
M10	0,001	0,000	0,115	0,065
M11	0,107	0,000	0,109	0,037
M12	0,001	0,017	0,006	0,001
M13	0,000	0,000	0,000	0,002
M14	0,000	0,000	0,003	0,001
M15	0,000	0,000	0,009	0,005
Result	M4	M6	M10	M10

this behavior is that the neural network fails to recognize certain patterns. The neural network results generated by the fifth data set has a consistency to the repair manual of 98, 80%, which is calculated by comparing the results with the relations specified in Chapter 2. This means that the neural network in some instances of the fifth data set, makes an incorrect choice between the valid faults, specified by the fault case. The problem in other words is that the neural network fails to recognize the patterns and instead chooses a suboptimal solution. Note that consistency to the model in this context, means that the results can be derived from at least one of the fault codes or symptoms in the input.

Table 4.22: The results from all validation sets on the Neural Network (NN). See Section 4.1.5 for more info about validation values.

	NN result
Second data set	98.68%
Third data set	94.81%
Fourth data set	98.28%
Fifth data set	83.82%

4.2 Training Set Size

The size of the training set can affect the performances of all methods. Tables 4.23, 4.24, and 4.25 show the validation results for the methods trained by three data sets with different sizes. The calculation of the performance results is explained in Section 4.1.5. The data sets are generated by the same probabilities as data set 1 (training set), see Tables 4.2, 4.3, and 4.4. The only difference between them is the size, and the data sets have sizes of approximately 1000, 500, and 100 entries. The data sets will not have sizes of exactly 1000, 500, and 100 entries, because the generation procedure sometimes generate empty fault cases. That are not included in the data sets. This was discussed in Section 4.1.2.

The results in Table 4.23 indicate that the matrix correlation method is stable for data sets of different sizes, because the difference between the results for each validation set is small.

Table 4.23: Table showing validation results for matrix correlation approach trained by data sets with different sizes (MC_{size}).

	MC_{1000}	MC_{500}	MC_{100}
Second data set	98.56%	97.77%	97.37%
Third data set	95.09%	94.58%	93.97%
Fourth data set	98.29%	97.48%	97.84%
Fifth data set	96.11%	95.03%	94.55%

Table 4.24 shows validation results for the Bayesian network trained by data sets of different sizes. All validation tests indicate that the characteristics of the results are roughly the same for all data set sizes.

Table 4.25 shows the validation scores for the neural network. Stable results are obtained if the neural network is trained by the sets with size 1000 and 500, which is not the case with the data set of size 100. The validation result for the neural network trained by the small data set gets lower scores compared to the

Table 4.24: Table showing validation results for Bayesian network trained by data sets with different sizes (BN_{size}).

	BN_{1000}	BN_{500}	BN_{100}
Second data set	92.33%	93.59%	92.00%
Third data set	81.52%	82.00%	78.76%
Fourth data set	93.52%	93.56%	92.78%
Fifth data set	100.00%	100.00%	99.88%

other tests. The neural network is more sensitive to small data sets compared to the other methods according to the results.

Table 4.25: Table showing validation results for neural network trained by data sets with different sizes (NN_{size}).

	NN_{1000}	NN_{500}	NN_{100}
Second data set	98.54%	97.59%	87.48%
Third data set	95.26%	94.66%	83.71%
Fourth data set	98.09%	96.76%	86.47%
Fifth data set	82.68%	80.65%	59.44%

4.3 Advantages and Disadvantages

In Table 4.26 advantages and disadvantages for all methods are shown. The cosine similarity did complicate the interpretation of the similarity measure in the matrix correlation method. These difficulties were discussed in Section 4.1.6. The matrix correlation approach is not considered to be intuitive due to the cosine similarity.

If a training set gives the algorithm undesired attributes, it would be an advantage to able to change the parameters in the model manually. In the Bayesian network and matrix correlation approach this can be done easily and the outcome of the changes is clear. The neural network offers no simple solution to this in the model. Neural networks are closely related to data in the learning process and a changed parameter shatter the result obtained by minimizing the error.

If the methods are considered, as they are in the case study, only the neural network and the matrix correlation approach can learn new relationships from data. This is not entirely true because it is possible to build a Bayesian networks from data [Jensen and Nielsen, 2007], but this will not be included in the case study.

Continuous variables can only be handled as they are in the Bayesian network and neural network [Russell and Norvig, 2003]. Many existing tools for Bayesian networks do not support the use of continuous variables, e.g. Genie. Since the neural network is purely data driven, a continuous variable could be used as it is. The matrix correlation approach cannot handle continuous input except in certain special cases. A continuous variable with a high value would always result in a high correlation. Some faults could, for example, be sensitive to lower values and therefore the best option in regard to the matrix correlation approach is to discretize the continuous variable. Discretizing is a valid option in all methods.

All methods have different prerequisites in order for the model construction to work. The Bayesian network, in the case study, needs expert knowledge in the form of model connections and probability estimation. If data is available, the probabilities can be calculated instead. The correlation matrix, in the matrix correlation approach, can be constructed either through data or by an expert. The neural network is dependent on data. The Bayesian network and the matrix correlation approach have an advantage, in this aspect, compared to the neural network, since they do not require data.

Table 4.26: Table showing advantages and disadvantages for the different methods

	Bayesian network	Neural network	Matrix correlation approach
Intuitive	Yes	No	No (due to cosine similarity)
Adjusting parameters manually	Yes	No	Yes
Learn new relationships between parameters	Yes	Yes	Yes
Continuous parameters without discretizing	Yes	Yes	No
Prerequisites needed	Expert knowledge of system or data	Data	Data or expert knowledge

Table 4.27 shows performance values of the different methods in regard to all validation sets. The calculation of the performance results is explained in Section 4.1.5. These numbers have been discussed in the sections above.

Table 4.27: Results for different data sets on the methods matrix correlation (MC), Bayesian network with arbitrarily chosen CPT's (BN_1), Bayesian network with trained parameters (BN_2) and the neural network (NN).

	MC	BN_1	BN_2	NN
Second data set	98.51%	92.88%	92.24%	98.68%
Third data set	94.89%	84.08%	82.38%	94.81%
Fourth data set	98.13%	93.33%	93.41%	98.28%
Fifth data set	96.17%	100%	100%	83.82%

The matrix correlation method had some problems with the cosine similarity measure, i.e. it can produce similar values for patterns that should not be similar, see Section 4.1.6. This leads to similarity measures that are in a very close range, which makes it harder separate them.

The neural network performance on all data sets except the fifth is good, see Section 4.1.8. The conclusion from the bad performance on the fifth data set is that the neural network have problems with certain patterns, i.e. it fails to recognize them.

The Bayesian networks has 100% accuracy when it comes to the fifth data set. The results with the second and fourth are good, but the third set gives bad results compared to the other methods. The reason is that the posterior probabilities for faults tend to get in the same range when no specific pattern is presented or certain false alarms are present, see Section 4.1.7. In the third set with more false alarms, this scenario is more likely to occur in regard to the other data sets. This weakness depend mostly on the structure of the model and the chosen false alarm rates.

4.4 Evaluation and Conclusion

The neural network, in this case study has shown some negative attributes regarding pattern recognition, and the good attributes cannot make up for this. The neural network can be tuned to get better results by for example changing the number of layers, and type activation functions. The thesis does not include any investigation of different neural network models. A few alternatives were tested during the construction of the network, but nothing extensive. This method will not be considered to be a candidate any more, but have the previous remarks in mind.

The matrix correlation method performed well in the case study, but cosine similarity values are not that easy to interpret compared to probabilities. Cosine similarity generates similarity values in the same range for cases that should not be close to each other, even in cases with a specific pattern. The conclusion of this is that the cosine similarity does not behave as desired in certain cases. Another similarity measure could probably remove this trait from the matrix correlation approach.

The Bayesian network has produced decent results in the case study, but not as good as the matrix correlation approach. The Bayesian network has the advantage of being easy to interpret and it recognises specific patterns without problems. The main drawback is the results from the data set with high false alarms, which depended on the attribute of producing posterior probabilities for faults in the same range. This can be turned into a strength, by presenting a list of possible faults instead of a single fault. It is not possible for the Bayesian network to produce results that are not consistent with the ideal connections, since it is built on these.

The method that will be used for implementation and further analysis is Bayesian networks, because the other methods had some issues that could not be ignored, like cosine similarity and pattern recognition. The intuitiveness of Bayesian networks and the results (posterior probabilities), compared to the other methods, play a major role in this decision.

4.5 Advantages of Methods Compared to Repair Manual

The repair manual offers all the relationships on paper. Certain cases could point out multiple faults and it might be hard to decide which of them is the best choice. All these methods will give a measure for each malfunction in a certain case, for example probabilities in the Bayesian network.

It also simplifies fault tracing because the methods will give results as long as input is provided, e.g fault codes and symptoms. In other words you could say that some of the work is transferred from the mechanic to the method. It is common that many new fault cases arise a while after a product release. If these new cases can be incorporated easily through learning or manually into the model new fault connections would be easier to identify.

5 Further Study and Implementation Description

5.1 Improvements to Suggested Method

The Bayesian network used in the case study could be improved in many ways. This part handle some possible elaborations on this topic.

5.1.1 Incorporate Dependencies to Other Components

The case study models take only the SUS unit into account. There are many ways to incorporate dependencies to other components. The existing model could be expanded in order to include dependencies to other components. This can be done by adding new nodes to the model. If data exists, the CPT's of the nodes can be learned. Otherwise they have to be instantiated by an expert. A more advanced and trickier approach would be to to learn the network from data [Jensen and Nielsen, 2007].

Another approach is to use the repair action set up that will be presented in Section 5.2. Repair actions lists are linked to specific faults, and the current fault is determined by the Bayesian network. These lists will be rearranged according to number of successful repair actions, cost, and repair time. The idea is to add repair actions to these lists that are linked to other components. This method will separate the Bayesian network model from external dependencies. If a fault case generally depends on a fault in another component, the idea is to add one repair action in the list that reflect this.

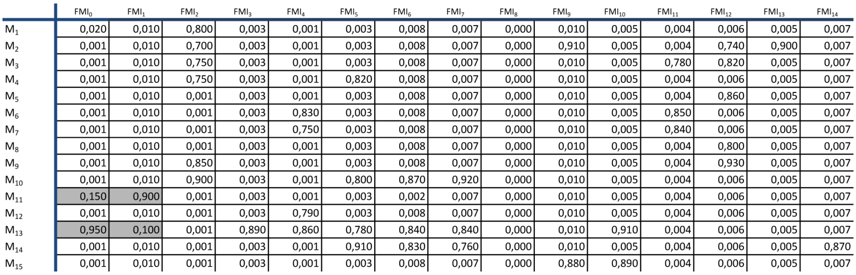
5.1.2 Adjusting Parameters to New Data

Parameter learning in Bayesian networks have been demonstrated in the case study in Chapter 4. The training set for the Bayesian network consisted of roughly

10000 entries, which implies that learning works for large data sets.

The case study left one Question unanswered. Namely how parameters change when new data is used to retrain the Bayesian network. In this section generated data with a few modifications is used to demonstrate this. The training sets are generated by changing the probabilities in the generation procedure, see Table 5.1. The data sets that are used to retrain the Bayesian network are generated in different sizes of 100, 500 and 1000 entries. This is done to decide how well the Bayesian network changes depending on training set size.

Table 5.1: Changed probabilities for FMI fault codes during data generation, highlighted in grey.



	FMI_0	FMI_1	FMI_2	FMI_3	FMI_4	FMI_5	FMI_6	FMI_7	FMI_8	FMI_9	FMI_{10}	FMI_{11}	FMI_{12}	FMI_{13}	FMI_{14}
M_1	0,020	0,010	0,800	0,003	0,001	0,003	0,008	0,007	0,000	0,010	0,005	0,004	0,006	0,005	0,007
M_2	0,001	0,010	0,700	0,003	0,001	0,003	0,008	0,007	0,000	0,910	0,005	0,004	0,740	0,900	0,007
M_3	0,001	0,010	0,750	0,003	0,001	0,003	0,008	0,007	0,000	0,010	0,005	0,780	0,820	0,005	0,007
M_4	0,001	0,010	0,750	0,003	0,001	0,820	0,008	0,007	0,000	0,010	0,005	0,004	0,006	0,005	0,007
M_5	0,001	0,010	0,001	0,003	0,001	0,003	0,008	0,007	0,000	0,010	0,005	0,004	0,860	0,005	0,007
M_6	0,001	0,010	0,001	0,003	0,830	0,003	0,008	0,007	0,000	0,010	0,005	0,850	0,006	0,005	0,007
M_7	0,001	0,010	0,001	0,003	0,750	0,003	0,008	0,007	0,000	0,010	0,005	0,840	0,006	0,005	0,007
M_8	0,001	0,010	0,001	0,003	0,001	0,003	0,008	0,007	0,000	0,010	0,005	0,004	0,800	0,005	0,007
M_9	0,001	0,010	0,850	0,003	0,001	0,003	0,008	0,007	0,000	0,010	0,005	0,004	0,930	0,005	0,007
M_{10}	0,001	0,010	0,900	0,003	0,001	0,800	0,870	0,920	0,000	0,010	0,005	0,004	0,006	0,005	0,007
M_{11}	0,150	0,900	0,001	0,003	0,001	0,003	0,002	0,007	0,000	0,010	0,005	0,004	0,006	0,005	0,007
M_{12}	0,001	0,010	0,001	0,003	0,790	0,003	0,008	0,007	0,000	0,010	0,005	0,004	0,006	0,005	0,007
M_{13}	0,950	0,100	0,001	0,890	0,860	0,780	0,840	0,840	0,000	0,010	0,910	0,004	0,006	0,005	0,007
M_{14}	0,001	0,010	0,001	0,003	0,001	0,910	0,830	0,760	0,000	0,010	0,005	0,004	0,006	0,005	0,870
M_{15}	0,001	0,010	0,001	0,003	0,001	0,003	0,008	0,007	0,000	0,880	0,890	0,004	0,006	0,005	0,007

The learning algorithm used in this section is the previously used *EM-algorithm*. The confidence value indicate how much the current parameters are to be trusted. A confidence value of 1 is used, which basically means that the current values should not be trusted. Tables 5.2, and 5.3 contain the old and new conditional probabilities before and after the Bayesian network has been retrained. Most of the values in the probability tables change as expected, except for the cases there both M_{11} and M_{13} are true simultaneously. The reason behind this, is that the generated data sets do not contain any cases where both faults are present at once. Therefore the probability is not updated. The generated data set contains all combinations of fault codes and faults except from the previously mentioned one. All cases with only one active fault change, and this is due to the changes in Table 5.1. The changes imply that FMI_0 will occur more frequently with M_{13} and FMI_1 with M_{11}, due to a higher probability value in the generation procedure. The Tables 5.2 and 5.3 reflect this too. The conditional probability value has decreased in row three, there FMI_0 and M_{11} are active. The opposite happened in row five there FMI_0 and M_{13} are active.

5.2 Repair Action Decision

The lists of repair actions that are available should be updated in regard to successful and failed repair actions. This will be done by defining two values, cost and repair time for each repair action. Equations that generate repair values, can be be constructed in many ways. Repair values denote how suitable repair actions are based on these parameters. The following equations below have different de-

Table 5.2: Conditional probabilities in different Bayesian networks. In the base network no additional data has been considered, while the other represent cases where the base network has been updated using the a new data set, with size of 1000 entries.

M_{11}	M_{13}	FMI_0	$P_{base}(FMI_0, M_{11}, M_{13})$	$P_{1000}(FMI_0, M_{11}, M_{13})$
T	T	T	0.9633	0.9633
T	T	F	0.0367	0.0367
T	F	T	0.7537	0.2336
T	F	F	0.2463	0.7664
F	T	T	0.7532	0.9094
F	T	F	0.2468	0.0906
F	F	T	0.0024	0.0024
F	F	F	0.9976	0.9976

Tables where the probabilities of the base network has been updated using the new data sets, with sizes of 500 and 100 entries.

M_{11}	M_{13}	FMI_0	$P_{500}(FMI_0, M_{11}, M_{13})$	$P_{100}(FMI_0, M_{11}, M_{13})$
T	T	T	0.9633	0.9633
T	T	F	0.0367	0.0367
T	F	T	0.1938	0.2923
T	F	F	0.8062	0.7077
F	T	T	0.9045	0.9589
F	T	F	0.0955	0.0411
F	F	T	0.0023	0.0000
F	F	F	0.9977	1.0000

sign aspects in mind. The first one guarantees that high values are assigned to expensive actions regarding cost and repair time. The second one is easier to tune due to weighting values that indicate how important the parameters are.

First repair value equation:

$$\text{repair value} = \left(\frac{\text{Number of succesful outcomes}}{\text{Total number of outcomes}}\right)^{-1} * C * T \tag{5.1}$$

there $C > 1$ and $T > 1$

Second repair value equation:

$$\text{repair value} = w_1 * \left(\frac{\text{Number of succesful outcomes}}{\text{Total number of outcomes}}\right)^{-1} + w_2 * C + w_3 * T \tag{5.2}$$

where $0 < C \leq 1$ and $0 < T \leq 1$

w_i = weighting value indicating importance of value

C = cost

T = repair time

The second approach is inspired from the paper [Yingping Huang and Zhang,

Table 5.3: Conditional probabilities in different Bayesian networks. In the base network no additional data has been considered, while the other represent cases where the base network has been updated using the a new data set, with size of 1000 entries.

M_{11}	M_{13}	FMI_1	$P_{base}(FMI_1, M_{11}, M_{13})$	$P_{1000}(FMI_1, M_{11}, M_{13})$
T	T	T	0.9863	0.9863
T	T	F	0.0137	0.0676
T	F	T	0.8597	0.9324
T	F	F	0.1403	0.0676
F	T	T	0.9192	0.5490
F	T	F	0.0808	0.4510
F	F	T	0.0109	0.0107
F	F	F	0.9891	0.9893

Tables where the probabilities of the base network has been updated using the new data sets, with sizes of 500 and 100 entries.

M_{11}	M_{13}	FMI_1	$P_{500}(FMI_1, M_{11}, M_{13})$	$P_{100}(FMI_1, M_{11}, M_{13})$
T	T	T	0.9863	0.9863
T	T	F	0.0137	0.0137
T	F	T	0.9464	0.8100
T	F	F	0.0535	0.1900
F	T	T	0.5564	0.6532
F	T	F	0.4436	0.3468
F	F	T	0.0024	0.0001
F	F	F	0.9976	0.9998

2014], see Chapter 1.

The repair manual contain lists of repair actions connected to one fault and fault code. The concept is that the fault tracing method first finds the most likely fault, and after that a repair list is chosen. The repair actions in the list are compared to each other by their repair value, then the repair action with the lowest value is chosen.

The repair action strategy is simple and was chosen due to the fact that the repair manual contains lists of repair actions. No data regarding how the system would respond to all actions existed in a simple context, and to collect this information is out of scope for the thesis.

5.2.1 Repair Action Decision Demonstration

A case study is used to demonstrate the calculation of repair action values. In the example FMI_0 is active, and the fault tracing method has pointed out that the Servo Motor is faulty. Both repair value equations are demonstrated in order to show how they behave differently. The repair list is shown below:

Servo Motor faulty and FMI$_0$ is active

1. Check if other error codes exist, that imply error in electrical system (R_1)
2. Check battery connection (R_2)
3. Measure battery voltage (R_3)
4. Check power cable connection between SUS and engine (R_4)
5. Measure the voltage on B+ and B– on the SUS (R_5)
6. Change Servo motor (R_6)

Note that the last action does not exist in the repair manual. It has been added to demonstrate an expensive action. Let all repair actions be denoted R_i according to the order in the list. The actions have their cost (C_i) and repair time (T_i) values listed below. Note that both repair action equations have a set of different test parameters because they are designed in different ways.

For first repair action value equation (5.1):

$$\begin{aligned} C_1 = 1,\ T_1 &= 25 \\ C_2 = 1,\ T_2 &= 20 \\ C_3 = 1,\ T_3 &= 15 \\ C_4 = 1,\ T_4 &= 20 \\ C_5 = 1,\ T_5 &= 25 \\ C_6 = 50,\ T_6 &= 50 \end{aligned}$$

For second repair action value equation (5.2):

$$\begin{aligned} C_1 = 0.1,\ T_1 &= 0.25 \\ C_2 = 0.1,\ T_2 &= 0.20 \\ C_3 = 0.1,\ T_3 &= 0.15 \\ C_4 = 0.1,\ T_4 &= 0.20 \\ C_5 = 0.1,\ T_5 &= 0.25 \\ C_6 = 0.5,\ T_6 &= 0.50 \end{aligned}$$

The Table 5.4 show the total number of actions and the number of successful actions.

Table 5.4: Table of an example with total number of actions and successful actions

Total number of actions	Successful actions
100	60
90	80
80	50
50	45
30	26
20	19

The Tables 5.5 and 5.6 show the results for the first equation (5.1) and second equation (5.2) respectively. The repair actions are ranked according to the lowest value in both tables, in ascending order. The last action which recommends change of component gets a high value with equation (5.1), since changing components should be the last resort. The ranking between the other actions in Table 5.5 are in a close range to each other, and will mostly be dependent on the number of successful actions. The results from equation (5.2) depend much on the choice of weight values. The repair values in this test end up in a similar range, see Table 5.6. Equation (5.1) represents a simple strategy that will make sure expensive actions do not climb up the list. This is due to the fact that the inverse of the probability of successful repair will always be greater than one. The values of cost and repair time will decide if the repair action value is large or small. It is, for example, impossible for R_6 to climb in ranking if the values of cost and repair time are not changed. The second repair value, equation (5.2) calculates the repair value by a summation of products between the attributes and weights. The weights symbolize the importance of each attribute. The second repair value equation offers more freedom compared to the first, since it is easier to tune the behavior with the weighting parameters. Both methods are shown to demonstrate that simple updating equations can be shaped in many different ways to suit different needs. The results in Tables 5.5 and 5.6 can not be judged as good or bad without a goal for the repair action strategy. If the goal is to guarantee that expensive actions are always in the bottom of the list, equation (5.1) is a suitable choice. One problem with equation (5.1) is that it can produce numbers in a big range, see for example R_6 in Table 5.5. Note that this goal could be achieved in a similar fashion by equation (5.2) by setting the weight value for the cost to a high value. Equation (5.2) has a big advantage compared to equation (5.1), because it is easier to tune.

Table 5.5: Results for repair action values for the first equation (5.1)

Repair actions	Result
R_1	33.3
R_2	22.5
R_3	24.0
R_4	27.8
R_5	28.8
R_6	5631.6

5.3 Implementation Description

The program GeNIe [gen, 2014] was used during the case study in order to evaluate the performed of the Bayesian network. GeNIe is a graphical interface for the C++ library, SMILE. In a real application the SMILE library could be used as a platform for all the calculations. The C++ library could be used with for example QT, if a C++ implementation is desired. A SMILE wrapper exists for .Net, which

Table 5.6: Results for repair action values for the second equation (5.1) with $w_1 = 1.5$, $w_2 = 2$, and $w_3 = 0.5$.

Repair actions	Result
R_1	2.8250
R_2	1.9875
R_3	2.6750
R_4	1.9667
R_5	2.0558
R_6	2.8289

could be an alternative to the previous proposal.

Figure 5.1 shows one possible way to implement the fault tracing scenarios discussed in this thesis. The Bayesian network box involves setting evidence in the network according to the fault case, for example $FMI_0 = true$. Then evidence has been set the posterior probabilities are calculated, which are the new probabilities related to the evidence. One strategy for choosing fault is to simply take the one with the highest probability, and then the corresponding repair list is forwarded to the mechanic. The mechanic need to give the system feedback on successful or unsuccessful repair actions. All the fault cases are then stored in a database in order to update the system. A validation step can prevent that bad data updates the system. This is the reason behind the box representing expert approval, in Figure 5.1.

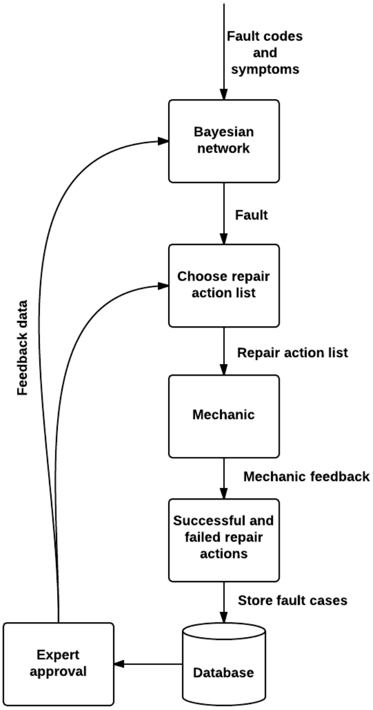


Figure 5.1: Implementation flowchart

6 Conclusions and Future Work

6.1 Conclusion

The problem that this study strives to address is how different algorithms perform in a troubleshooting scenario for the SUS unit, i.e., identifying faults and propose suitable repair actions. A major focus is placed on learning dependency structures from data, in other words, the strength of the connections between faults, fault codes, and symptoms. This is done in order to reflect the fact that real data might be scarce at release of a product, but the quantity of data is likely to increase over time.

There are many algorithms that are applicable to the problem of identifying faults based on fault codes and symptoms. The three candidates that were compared to each other, in this thesis, are Bayesian networks, neural networks, and the matrix correlation approach. Each method has it is own way of tackling the problem. All methods can be purely data driven, but only the neural network and the matrix correlation approach are used in this way. The Bayesian network is built from known dependencies between faults, fault codes, and symptoms. The Bayesian network model consists of probability values that need to be initiated from either data or expert knowledge. The study takes both these strategies into account.

No real data could be used in this study, because the conversion of the real data to usable data sets was out of scope for this thesis. Five sets were generated to evaluate all algorithms. There is one data set for training, while the others were created for validation purposes. The data sets where generated from the dependencies between faults, fault codes, and symptoms given in the repair manual [Penta, 2006]. The generation process was built on probabilities, that stated how likely it is that fault codes and symptoms are active in regard to all faults. This generation procedure gives data sets that are more realistic compared to the ideal cases in the repair manual.

Further analysis of the chosen algorithms indicated that all of them have strengths and drawbacks. The results from the validation sets showed that the matrix correlation approach had the best performance. The chosen test cases revealed problems with the neural network and matrix correlation approach, namely pattern recognition and cosine similarity measures. In the aspect of training set size, all methods have basically the same characteristics after being trained by the different data set sizes, except the neural network. The matrix correlation approach performed well in most tests, but has one issue that could not be discarded, i.e. cosine similarity. The cosine similarity has an undesired behavior in certain situations, i.e. the range of the similarity measures. This is the foremost argument against the matrix correlation method. The main reason behind the exclusion of the neural network is due to the problems with pattern recognition. Bayesian networks are rather easy to construct and can be adapted to suit many different situations. In general it suits the problem well. Bayesian networks can be quite advanced when it comes to different ways to implement interference and to learn parameters, but the modelling and the results are intuitive. These facts are some of the reasons behind the decision to deem the Bayesian network as the most appropriate method for the presented problem. The Bayesian network has flaws as well, for example the results on the validation sets that did not match the matrix correlation approach. The Bayesian network produced less ideal results for the data set with high false alarm rate, and the reason behind this was the model and data structure. It is important to note that this is not a general trait for Bayesian networks. The other algorithms cannot be entirely ruled out, because a valid statement concluding which algorithm is best, requires a more extensive study of all the algorithms.

Another topic addressed is how well the Bayesian network can adapt to new data sets that contain different dependencies, for example a weaker connection between a symptom and a fault in comparison to the original training set. The trust measure that the *EM-algorithm* uses to train the Bayesian network is a big advantage. In reality data sets can be of different quality, and the Bayesian network can adapt do this due to the trust measure. The results from this investigation showed that the Bayesian network managed to reflect the changes in the new data sets, with different data set sizes.

The task for the Bayesian network is to point out the most probable fault. After that the next step is to recommend a repair action. In the repair manual repair actions are given as lists connected to fault codes and faults. These lists could be made more accurate by allowing a updating equation to reorganize them based on for example successful repair actions, cost, and repair time. Two proposals for updating equations are presented, but the conclusion is that such equations can be tailored to suit the desires in different situations. The proposed fault tracing algorithm can adapt to new data, which will make new knowledge available to all users.

The two primary contributions of the thesis is a conceptual idea of how fault tracing can be implemented, and a demonstration through Matlab and the program GeNIe to show all results. Volvo Penta will get insight into how data should be structured, in order to simplify usage of data in fault tracing scenarios. The methods investigated are applicable in other similar areas, which makes it interesting from other point of views as well. For example one could construct a Bayesian network for many possible scenarios.

6.2 Future Work

The study has only scratched the surface of a complicated and advanced topic. All the methods that were compared have been utilized in their basic form, which means there is much more that can be uncovered by delving deeper into each method. Note that a further investigation of all methods could result in a different ranking between the presented methods. A list of possible elaboration topics are listed in below:

Investigating other similarity measures for the matrix correlation approach

The matrix correlation method was promising, but was dropped due to the fact that the cosine similarity measure presented some issues. There are many similarity measures and the possibility that another choice suits the problem better is high. Therefore one elaboration in the future is to investigate the matrix correlation based on LSI further, and especially similarity measures.

Building Bayesian networks from data

The Bayesian network has one disadvantage, the way it is presented in the thesis, it must be built on dependencies that are known. There are ways to construct Bayesian networks from data [Jensen and Nielsen, 2007]. By using this knowledge, it would be possible to learn new connections solely based on data, which is a big advantage if the models are complicated.

How well all methods handle continuous input

The case study merely pointed out if continuous variables could be included in the model or not. This is an interesting topic, because many data sources such as mileage is represented by continuous data. Future work could consist of trials that included tests with both discrete and continuous data.

Study different neural network models

The neural network model in the case study has not undergone any extensive testing to find the optimal parameters, for example number of hidden layers, number of nodes and so on. Neural network models can exist in many different shapes, and this can have a great impact on performance. This could therefore be an interesting topic to investigate further.

Investigate how well each method can handle multiple faults

Multiple faults is something that the case study did not take into account. In reality this could happen and to reflect that, the model should be able to handle both single and multiple faults. Models always strive to reflect reality as good as possible, and therefore this topic is interesting for future work on all the methods.

Develop a more sophisticated repair action decision strategy

The suggested ranking methods of repair actions in the implementation section is simple. There are much theory on decision making [Russell and Norvig, 2003] that could be included in the troubleshooting process to generate a more sophisticated repair action decision method.

Use more data

In Chapter 2 available data was listed, but all of it was not used. A more accurate model of reality could be obtained if the available data was incorporated into the algorithms. The algorithms could for example find out if a product is likely to be worn out with the help of product usage and mileage. However the data need to be analysed and rewritten into a form that the algorithms can use.

Appendix

A

Appendix

A.1 Data Generation

A.1.1 Probability Values

All the Tables in this section contain probability values used for data generation.

Table A.1: Probabilities for FMI fault codes during data generation of data set 3

	FMI_0	FMI_1	FMI_2	FMI_3	FMI_4	FMI_5	FMI_6	FMI_7	FMI_8	FMI_9	FMI_{10}	FMI_{11}	FMI_{12}	FMI_{13}	FMI_{14}
M_1	0,020	0,030	0,800	0,020	0,050	0,030	0,010	0,060	0,000	0,040	0,050	0,030	0,040	0,060	0,070
M_2	0,010	0,030	0,700	0,020	0,050	0,030	0,010	0,060	0,000	0,910	0,050	0,030	0,740	0,900	0,070
M_3	0,010	0,030	0,750	0,020	0,050	0,030	0,010	0,060	0,000	0,040	0,050	0,780	0,820	0,060	0,070
M_4	0,010	0,030	0,750	0,020	0,050	0,820	0,010	0,060	0,000	0,040	0,050	0,030	0,040	0,060	0,070
M_5	0,010	0,030	0,040	0,020	0,050	0,030	0,010	0,060	0,000	0,040	0,050	0,030	0,860	0,060	0,070
M_6	0,010	0,030	0,040	0,020	0,830	0,030	0,010	0,060	0,000	0,040	0,050	0,850	0,040	0,060	0,070
M_7	0,010	0,030	0,040	0,020	0,750	0,030	0,010	0,060	0,000	0,040	0,050	0,840	0,040	0,060	0,070
M_8	0,010	0,030	0,040	0,020	0,050	0,030	0,010	0,060	0,000	0,040	0,050	0,030	0,800	0,060	0,070
M_9	0,010	0,030	0,850	0,020	0,050	0,030	0,010	0,060	0,000	0,040	0,050	0,030	0,930	0,060	0,070
M_{10}	0,010	0,030	0,900	0,020	0,050	0,800	0,870	0,920	0,000	0,040	0,050	0,030	0,040	0,060	0,070
M_{11}	0,730	0,840	0,040	0,020	0,050	0,030	0,010	0,060	0,000	0,040	0,050	0,030	0,040	0,060	0,070
M_{12}	0,010	0,030	0,040	0,020	0,790	0,030	0,010	0,060	0,000	0,040	0,050	0,030	0,040	0,060	0,070
M_{13}	0,760	0,910	0,040	0,890	0,860	0,780	0,840	0,840	0,000	0,040	0,910	0,030	0,040	0,060	0,070
M_{14}	0,010	0,030	0,040	0,020	0,050	0,910	0,830	0,760	0,000	0,040	0,050	0,030	0,040	0,060	0,870
M_{15}	0,010	0,030	0,040	0,020	0,050	0,030	0,010	0,060	0,000	0,880	0,890	0,030	0,040	0,060	0,070

Table A.2: Probabilities for PID and SID fault codes during data generation of data set 3

	SID_{240}	SID_{253}	SID_{254}	$PSID_{234}$	$PSID_1$	$PPID_{393}$	PID_{55}	$PPID_{55}$	$PPID_{424}$	$PPID_{426}$	$PPID_{427}$	$PSID_2$	$PSID_3$	$PSID_4$	$PSID_6$
M_1	0,600	0,030	0,020	0,040	0,050	0,020	0,020	0,030	0,040	0,040	0,030	0,030	0,060	0,050	0,030
M_2	0,020	0,690	0,020	0,040	0,050	0,020	0,020	0,030	0,040	0,040	0,030	0,030	0,060	0,050	0,030
M_3	0,020	0,030	0,700	0,040	0,050	0,020	0,020	0,030	0,040	0,040	0,030	0,030	0,060	0,050	0,030
M_4	0,020	0,030	0,020	0,790	0,050	0,020	0,020	0,030	0,040	0,040	0,030	0,030	0,060	0,050	0,030
M_5	0,020	0,030	0,020	0,040	0,650	0,020	0,020	0,030	0,040	0,040	0,030	0,030	0,060	0,050	0,030
M_6	0,020	0,030	0,020	0,040	0,050	0,760	0,020	0,030	0,040	0,040	0,030	0,030	0,060	0,050	0,030
M_7	0,020	0,030	0,020	0,040	0,050	0,020	0,680	0,030	0,040	0,040	0,030	0,030	0,060	0,050	0,030
M_8	0,020	0,030	0,020	0,040	0,050	0,020	0,020	0,850	0,040	0,040	0,030	0,030	0,060	0,050	0,030
M_9	0,020	0,030	0,020	0,040	0,050	0,020	0,020	0,030	0,740	0,040	0,030	0,030	0,060	0,050	0,030
M_{10}	0,020	0,030	0,020	0,040	0,050	0,020	0,020	0,030	0,040	0,720	0,030	0,030	0,060	0,050	0,030
M_{11}	0,020	0,030	0,020	0,040	0,050	0,020	0,020	0,030	0,040	0,040	0,700	0,030	0,060	0,050	0,030
M_{12}	0,020	0,030	0,020	0,040	0,050	0,020	0,020	0,030	0,040	0,040	0,030	0,750	0,060	0,050	0,030
M_{13}	0,020	0,030	0,020	0,040	0,050	0,020	0,020	0,030	0,040	0,040	0,030	0,030	0,650	0,050	0,030
M_{14}	0,020	0,030	0,020	0,040	0,050	0,020	0,020	0,030	0,040	0,040	0,030	0,030	0,060	0,710	0,030
M_{15}	0,020	0,030	0,020	0,040	0,050	0,020	0,020	0,030	0,040	0,040	0,030	0,030	0,060	0,050	0,680

Table A.3: Probabilities for symptoms during data generation of data set 4

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}
M_1	0,800	0,030	0,050	0,040	0,060	0,040	0,060	0,040	0,030	0,040	0,040	0,030	0,050	0,040	0,030	0,030
M_2	0,040	0,800	0,850	0,040	0,060	0,040	0,060	0,040	0,030	0,040	0,040	0,030	0,050	0,040	0,030	0,030
M_3	0,700	0,030	0,050	0,040	0,060	0,040	0,060	0,040	0,030	0,040	0,040	0,030	0,050	0,040	0,030	0,030
M_4	0,040	0,030	0,050	0,040	0,060	0,040	0,060	0,040	0,030	0,040	0,040	0,030	0,050	0,040	0,030	0,030
M_5	0,040	0,030	0,050	0,700	0,650	0,600	0,060	0,040	0,030	0,040	0,040	0,030	0,050	0,040	0,030	0,030
M_6	0,040	0,030	0,050	0,040	0,060	0,040	0,060	0,040	0,030	0,040	0,040	0,030	0,050	0,040	0,030	0,030
M_7	0,040	0,030	0,050	0,040	0,060	0,040	0,670	0,720	0,650	0,040	0,040	0,030	0,050	0,040	0,030	0,030
M_8	0,040	0,030	0,050	0,040	0,060	0,040	0,060	0,040	0,030	0,800	0,040	0,030	0,050	0,040	0,030	0,030
M_9	0,040	0,030	0,050	0,040	0,060	0,040	0,060	0,040	0,740	0,040	0,040	0,030	0,050	0,040	0,030	0,030
M_{10}	0,040	0,030	0,050	0,040	0,060	0,040	0,060	0,040	0,030	0,040	0,800	0,800	0,050	0,040	0,030	0,030
M_{11}	0,040	0,030	0,050	0,040	0,060	0,040	0,060	0,040	0,030	0,040	0,700	0,030	0,050	0,040	0,030	0,030
M_{12}	0,040	0,030	0,050	0,040	0,060	0,040	0,060	0,040	0,030	0,040	0,040	0,030	0,050	0,040	0,030	0,030
M_{13}	0,040	0,030	0,050	0,040	0,060	0,040	0,060	0,040	0,030	0,040	0,660	0,030	0,050	0,040	0,030	0,030
M_{14}	0,040	0,030	0,050	0,040	0,060	0,040	0,060	0,040	0,030	0,040	0,040	0,030	0,690	0,040	0,030	0,030
M_{15}	0,040	0,030	0,050	0,040	0,060	0,040	0,060	0,040	0,030	0,040	0,700	0,030	0,050	0,600	0,590	0,700

Table A.4: Probabilities for FMI fault codes during data generation of data set 4

	FMI_0	FMI_1	FMI_2	FMI_3	FMI_4	FMI_5	FMI_6	FMI_7	FMI_8	FMI_9	FMI_{10}	FMI_{11}	FMI_{12}	FMI_{13}	FMI_{14}
M_1	0,020	0,010	0,700	0,003	0,001	0,003	0,008	0,007	0,000	0,010	0,005	0,004	0,006	0,005	0,007
M_2	0,001	0,010	0,850	0,003	0,001	0,003	0,008	0,007	0,000	0,750	0,005	0,004	0,690	0,800	0,007
M_3	0,001	0,010	0,900	0,003	0,001	0,003	0,008	0,007	0,000	0,010	0,005	0,950	0,750	0,005	0,007
M_4	0,001	0,010	0,950	0,003	0,001	0,760	0,008	0,007	0,000	0,010	0,005	0,004	0,006	0,005	0,007
M_5	0,001	0,010	0,001	0,003	0,001	0,003	0,008	0,007	0,000	0,010	0,005	0,004	0,720	0,005	0,007
M_6	0,001	0,010	0,001	0,003	0,700	0,003	0,008	0,007	0,000	0,010	0,005	0,800	0,006	0,005	0,007
M_7	0,001	0,010	0,001	0,003	0,900	0,003	0,008	0,007	0,000	0,010	0,005	0,700	0,006	0,005	0,007
M_8	0,001	0,010	0,001	0,003	0,001	0,003	0,008	0,007	0,000	0,010	0,005	0,004	0,900	0,005	0,007
M_9	0,001	0,010	0,700	0,003	0,001	0,003	0,008	0,007	0,000	0,010	0,005	0,004	0,750	0,005	0,007
M_{10}	0,001	0,010	0,950	0,003	0,001	0,900	0,720	0,750	0,000	0,010	0,005	0,004	0,006	0,005	0,007
M_{11}	0,900	0,750	0,001	0,003	0,001	0,003	0,002	0,007	0,000	0,010	0,005	0,004	0,006	0,005	0,007
M_{12}	0,001	0,010	0,001	0,003	0,700	0,003	0,008	0,007	0,000	0,010	0,005	0,004	0,006	0,005	0,007
M_{13}	0,850	0,700	0,001	0,800	0,900	0,900	0,950	0,700	0,000	0,010	0,970	0,004	0,006	0,005	0,007
M_{14}	0,001	0,010	0,001	0,003	0,001	0,710	0,700	0,950	0,000	0,010	0,005	0,004	0,006	0,005	0,870
M_{15}	0,001	0,010	0,001	0,003	0,001	0,003	0,008	0,007	0,000	0,950	0,800	0,004	0,006	0,005	0,007

Table A.5: Probabilities for PID and SID fault codes during data generation of data set 4

	SID_{240}	SID_{253}	SID_{254}	$PSID_{234}$	$PSID_1$	$PPID_{393}$	PID_{55}	$PPID_{55}$	$PPID_{424}$	$PPID_{426}$	$PPID_{427}$	$PSID_2$	$PSID_3$	$PSID_4$	$PSID_6$
M_1	0,700	0,005	0,004	0,003	0,006	0,003	0,006	0,008	0,004	0,005	0,007	0,008	0,002	0,010	0,005
M_2	0,004	0,800	0,004	0,003	0,006	0,003	0,006	0,008	0,004	0,005	0,007	0,008	0,002	0,010	0,005
M_3	0,004	0,005	0,600	0,003	0,006	0,003	0,006	0,008	0,004	0,005	0,007	0,008	0,002	0,010	0,005
M_4	0,004	0,005	0,004	0,650	0,006	0,003	0,006	0,008	0,004	0,005	0,007	0,008	0,002	0,010	0,005
M_5	0,004	0,005	0,004	0,003	0,800	0,003	0,006	0,008	0,004	0,005	0,007	0,008	0,002	0,010	0,005
M_6	0,004	0,005	0,004	0,003	0,006	0,950	0,006	0,008	0,004	0,005	0,007	0,008	0,002	0,010	0,005
M_7	0,004	0,005	0,004	0,003	0,006	0,003	0,600	0,008	0,004	0,005	0,007	0,008	0,002	0,010	0,005
M_8	0,004	0,005	0,004	0,003	0,006	0,003	0,006	0,900	0,004	0,005	0,007	0,008	0,002	0,010	0,005
M_9	0,004	0,005	0,004	0,003	0,006	0,003	0,006	0,008	0,600	0,005	0,007	0,008	0,002	0,010	0,005
M_{10}	0,004	0,005	0,004	0,003	0,006	0,003	0,006	0,008	0,004	0,600	0,007	0,008	0,002	0,010	0,005
M_{11}	0,004	0,005	0,004	0,003	0,006	0,003	0,006	0,008	0,004	0,005	0,800	0,008	0,002	0,010	0,005
M_{12}	0,004	0,005	0,004	0,003	0,006	0,003	0,006	0,008	0,004	0,005	0,007	0,900	0,002	0,010	0,005
M_{13}	0,004	0,005	0,004	0,003	0,006	0,003	0,006	0,008	0,004	0,005	0,007	0,008	0,900	0,010	0,005
M_{14}	0,004	0,005	0,004	0,003	0,006	0,003	0,006	0,008	0,004	0,005	0,007	0,008	0,002	0,600	0,005
M_{15}	0,004	0,005	0,004	0,003	0,006	0,003	0,006	0,008	0,004	0,005	0,007	0,008	0,002	0,010	0,800

Table A.6: Probabilities for symptoms during data generation of data set 4

	S_1	S_2	S_3	S_4	S_5	S_6	S_7	S_8	S_9	S_{10}	S_{11}	S_{12}	S_{13}	S_{14}	S_{15}	S_{16}
M_1	0,750	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,010	0,004	0,003	0,005	0,005	0,004
M_2	0,020	0,700	0,750	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,010	0,004	0,003	0,005	0,005	0,004
M_3	0,900	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,010	0,004	0,003	0,005	0,005	0,004
M_4	0,020	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,010	0,004	0,003	0,005	0,005	0,004
M_5	0,020	0,010	0,005	0,900	0,800	0,750	0,010	0,005	0,003	0,002	0,010	0,004	0,003	0,005	0,005	0,004
M_6	0,020	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,010	0,004	0,003	0,005	0,005	0,004
M_7	0,020	0,010	0,005	0,008	0,010	0,020	0,600	0,800	0,600	0,002	0,010	0,004	0,003	0,005	0,005	0,004
M_8	0,020	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,650	0,010	0,004	0,003	0,005	0,005	0,004
M_9	0,020	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,800	0,002	0,010	0,004	0,003	0,005	0,005	0,004
M_{10}	0,020	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,650	0,700	0,003	0,005	0,005	0,004
M_{11}	0,020	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,900	0,004	0,003	0,005	0,005	0,004
M_{12}	0,020	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,010	0,004	0,003	0,005	0,005	0,004
M_{13}	0,020	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,800	0,004	0,003	0,005	0,005	0,004
M_{14}	0,020	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,010	0,004	0,850	0,005	0,005	0,004
M_{15}	0,020	0,010	0,005	0,008	0,010	0,020	0,010	0,005	0,003	0,002	0,600	0,004	0,003	0,750	0,650	0,600

A.1.2 Data Generation Code

```
%% Cases
clc
clear
%%

load('Input_ideal.mat');
load('Output_ideal.mat');

IN_ideal = Input_ideal;
IN_ext = Input_ideal;
targets_ext = Output;

%% All combinations
numCases = 1;

for i = 1:15 % malfunction 1-15
    IN_row = IN_ideal(i,:);
    index = find(IN_row);
    l = length(index);
    numCases = 1;

    for j = 1:l % all fault codes belonging to malfunction i
         indexes = combnk(index, j);
         for k = 1:size(indexes,1)
```

```
             index_row = indexes(k,:);
             %numCases = randi(10,1,1);
            [IN_ext, targets_ext] = addCases(IN_ideal, IN_ext,...
                                             targets_ext, numCases,...
                                             index_row);
         end
    end

end

Input = IN_ext(16:end, :);
Output = targets_ext(16:end, :);

%% False alarms data

clear; clc;

load('prop_dataSet5.mat');
mat = prop_dataSet5;

num_cases = 10000;

Input = [];
Output = [];

for i = 1:num_cases

    % create empty case
    case_malfunc = zeros(1,46);
    out = zeros(1,15);

    % choose malfunction randomly
    malfunc = randi([1 15],1,1);

    % pick out the fault codes belonging to the malfunction
    row_fc = mat(malfunc,:);

    % choose which fault codes to put in case, by the probabilities in
    % row_fc
    for j = 1:size(mat,2)
        % fault codes/symptom equal to zero or one?
        x = sum(rand >= cumsum([1 - row_fc(j), row_fc(j)]));
        case_malfunc(j) = x;
    end

    % Set values
    sum_row = sum(case_malfunc);
    out(malfunc) = 1;

    if(sum_row > 0)
        Input = [Input; case_malfunc];
        Output = [Output; out];
    end
end


function [ IN_ext targets_ext ] = addCases( IN_ideal, IN, targets, numCases, in dexes)
%ADDCASE
```

```
IN_ext = IN;
targets_ext = targets;

input_t = zeros(1,size(IN,2));
targets_t = ones(1,15);

for i = 1:length(indexes)
    [row, col] = find(IN_ideal(:,indexes(i)));
    targets_index = zeros(1,15);
    targets_index(row) = 1;
    targets_t = targets_index & targets_t;
end

input_t(indexes) = 1;

for k = 1:numCases
   IN_ext = [IN_ext; input_t];
   targets_ext = [targets_ext; targets_t];
end

end
```

A.2 Matrix Correlation Approach

A.2.1 Create Correlation Matrix

```
function [ corrMatrix ] = createDependMatrix( Input, Output)
%CREATEDEPENDMATRIX


m = size(Output, 2);
n = size(Input, 2);

corrMatrix = zeros(m,n);

for k = 1:size(Input, 1);
    % find which indexes in Output are != 0
    index_output = find(Output(k, :));
    % find which indexes in Input are != 0
    index_input = find(Input(k,:));

    % add +1 to row i and col j in corrMatrix
    for i = 1:length(index_output)
        for j = 1:length(index_input)
            corrMatrix(index_output(i), index_input(j)) = corrMatrix(index_output(i),
        end
    end

end

corrMatrix = corrMatrix/size(Input,1);

end
```

A.2.2 Correlation

```
function [ corr1, corr2] = Correlation( m, faultcase )
%CORRELATION Summary of this function goes here
%   Detailed explanation goes here

corr1 = [];
corr2 = [];

n_m = size(m,2);
n_p = length(faultcase);

if(n_m == n_p)
    for i = 1:size(m,1)
       corr1 = [corr1; (m(i,:)*faultcase')/(norm(faultcase)*norm(m(i,:)'))];
       corr2 = [corr2; m(i,:)*faultcase'];
    end
else
   disp('Wrong size of input vector');
end

end
```

A.2.3 Validation

```
function [ rel, faulty_indexes ] = correctness( Res, Input, Input_ideal )
%CORRECTNESS


corr = 0;
faulty_indexes = [];

for i = 1:size(Input,1)

   faultcodes = find(Input(i,:));
   malfunctions = Input_ideal(:, faultcodes);
   [I,J]= find(malfunctions);
   if(sum(Res(i) == I))
       corr = corr + 1;
   else
       faulty_indexes = [faulty_indexes; i];
   end


end

rel = corr/size(Input,1);

end
```

A.3 Bayesian Network

A.3.1 Validation

```
#include <iostream>
#include <fstream>
#include <sstream>
#include <string>
#include "..\..\..\..\Diadrom\SMILE\smile.h"

using namespace std;

// Load data
void loadInputOutput( string file, vector<vector<int>> &input, vector<
    vector<int>> &output, vector<string> &fc_sym, int num_fc_sym)
{
    // read from cases from file

    // input file stream
    ifstream txt_file;
    // open file, only reading is permitted
    txt_file.open(file, ios::in);

    stringstream stream;
    string line;
    vector<int> row_input;
    vector<int> row_output;

    if(txt_file.is_open())
    {
        cout << "loading file..." << endl;

        // handle first line
        getline(txt_file, line);
        stream << line;
        while(!stream.eof())
        {
            string x;
            stream >> x;
            fc_sym.push_back(x);
        }

        // get line
        while(getline(txt_file, line))
        {
            // clear string stream and put the string line there
            stream.clear();
            stream << line;
            // clear vectors
            row_input.clear();
            row_output.clear();

            // counter variable
            int i = 1;

            // go though string stream and handle each int
            while(!stream.eof())
            {
                int n;
                stream >> n;

                if(i <= num_fc_sym)
```

```
                    row_input.push_back(n);
                else if(i > num_fc_sym)
                    row_output.push_back(n);

                i += 1;
            }

            // save result vector
            input.push_back(row_input);
            output.push_back(row_output);
        }
    }
    else
    {
        cout << "could not open file" << endl;
    }

    txt_file.close();
    cout << "finished" << endl;
}

// Set evidence, input is 1x46 with active and non-active fault codes (i.e
     0 or 1) vector and fc_sym are a 1x46 vector with all the names
void setEvidence( vector<int> input, vector<string> fc_sym, DSL_network &
    net)
{
    for(size_t i = 0; i < input.size(); ++i)
    {
        if(input[i] == 1)
        {
            const char* symbol = fc_sym[i].c_str();
            int handle = net.FindNode(symbol);

            net.GetNode(handle)->Value()->SetEvidence(0);
            //cout << "true: " << fc_sym[i] << endl;
        }
    }
}


// Do interference on the network
void interference( DSL_network &net)
{
    // use clustering algorithm
   net.SetDefaultBNAlgorithm(DSL_ALG_BN_LAURITZEN);
   // Update
   net.UpdateBeliefs();
}

void printResult( DSL_network &net, vector<string> &faults)
{
    DSL_sysCoordinates theCoordinates;

    // 46 is the number of fault codes + symptoms
    for( size_t i = 46; i < faults.size(); ++i)
    {
        const char* symbol = faults[i].c_str();
        int handle = net.FindNode(symbol);
```

```
            theCoordinates.LinkTo(*net.GetNode(handle)->Value());
            theCoordinates[0] = 0; // index of true
            theCoordinates.GoToCurrentPosition();
            double P_res = theCoordinates.UncheckedValue();
            cout << faults[i] << ": " << P_res << endl;
        }
}

int getResult( DSL_network &net, vector<string> &faults)
{
    DSL_sysCoordinates theCoordinates;
    int res = 0;
    double P_res_old = 0;

    // 46 is the number of fault codes + symptoms
    for(size_t i = 46; i < faults.size(); ++i)
    {
        // Set evidence
        const char* symbol = faults[i].c_str();
        int handle = net.FindNode(symbol);

        theCoordinates.LinkTo(*net.GetNode(handle)->Value());
        theCoordinates[0] = 0; // index of true
        theCoordinates.GoToCurrentPosition();
        double P_res = theCoordinates.UncheckedValue();

        // check if result (res) should be replaced
        if(P_res > P_res_old)
        {
            P_res_old = P_res;
            res = i - 45;
        }
    }

    return res;
}

// Validation, result from network, output from the data set
float matchResult( vector<int> result, vector<vector<int>> output)
{
    int corr = 0;

    if(result.size() == output.size())
    {
        for(size_t i = 0;  i < result.size(); ++i)
        {
            if(output[0].size() == output[i].size())
            {
                for(size_t j = 0; j < output[0].size(); ++j)
                {
                    if(output[i][j] == 1)
                    {
                        if((j + 1) == result[i])
                            corr += 1;
                    }
                }
            }
            else
                cout << "wrong size, output" << endl;
        }
    }
    else
        cout << "wrong size, result vector" << endl;

    float rel = (float)corr/(float)result.size();

    return rel;
}

template <class T>
void printVector(vector<T> x)
{
    for(size_t i = 0; i < x.size(); ++i)
    {
        cout << x[i] << " ";
    }

    cout << endl;
}

int main()
{
    DSL_network theNet;

    //theNet.ReadFile( "Bayesian_network_SUS_learn_parameters.xdsl");
    theNet.ReadFile( "Bayesian_network_SUS_prob.xdsl");

    vector<vector<int>> input;
    vector<vector<int>> output;
    vector<string> fc_sym;
    // inOutProb: contains the cases generated from probabilties
    // inOutExt: contains all possilbe cases
    loadInputOutput( "inOutProb.txt", input, output, fc_sym, 46);

    vector<int> result;

    // For validation in Matlab
    //ofstream outTest;
    //outTest.open("outTest.txt");

    cout << "Validation..." << endl;
    for(size_t i = 0; i < input.size(); ++i)
    {
        setEvidence( input[i], fc_sym, theNet);
        interference( theNet);
        int res = getResult( theNet, fc_sym);
        result.push_back( res );
        //outTest << res << "\n";
        theNet.ClearAllEvidence();
    }

    //outTest.close();

    float rel = matchResult( result, output);
    cout << "Validation finished" << endl;
    cout << "Correctness: " << rel << endl;


    string i;
    cin >> i;

    return 0;
}
```

A.4 Neural Network

A.4.1 Cost Function and Back propagation

```
function [ J, grad ] = costFunction( params, X, Y, input_layer_size,...
                                     hidden_layer_size,...
                                     output_layer_size, lambda)
%COSTFUNCTION

m = size(X, 1);
num_malfuncs = size(Y,2);
J = 0;
grad = 0;

Theta1 = reshape(params(1:hidden_layer_size * (input_layer_size + 1)),...
                 hidden_layer_size, (input_layer_size+1));

Theta2 = reshape(params((1 + ...
                 (hidden_layer_size * (input_layer_size + 1))):end),...
                 output_layer_size, (hidden_layer_size+1));

% feed forward propagation

X1 = [ones(size(X,1),1) X];
z2 = X1*Theta1';
a2 = sigmoid(z2);
a2 = [ones(size(a2,1),1) a2];
z3 = a2*Theta2';
a3 = sigmoid(z3);
h = a3;

% cost function + reg

for k = 1:num_malfuncs
     J = J + 1/m * (- log(h(:,k))' * Y(:,k) - log( 1 - h(:,k))' * ...
     (1 - Y(:,k)));
end

reg = lambda/(2*m) * ( sum(sum(Theta1(:,2:end).^2)) + ...
      sum(sum(Theta2(:,2:end).^2)));

J = J + reg;

% Back propagation

DELTA2 = 0;
```

```
DELTA1 = 0;

for t = 1:m

    % forward propagation
    a1 = [1 X(t,:)];
    z2 = a1 * Theta1';
    a2 = sigmoid(z2);
    a2 = [1 a2];
    z3 = a2 * Theta2';
    a3 = sigmoid(z3);

    % calc error in output nodes
    delta3 = (a3 - Y(t,:))';

    % calc delta 2 error in each node in layer 2 (hidden layer 1)
    g_grad = [1 sigmoidGradient(z2)];
    delta2 = (Theta2'*delta3).*g_grad';

    % add to earlier results
    DELTA2 = DELTA2 + delta3*a2;
    DELTA1 = DELTA1 + delta2(2:end)*a1;

end

Theta2_grad = 1/m*DELTA2;
Theta2_grad(:,2:end) = Theta2_grad(:,2:end) + lambda/m * Theta2(:,2:end);

Theta1_grad = 1/m*DELTA1;
Theta1_grad(:,2:end) = Theta1_grad(:,2:end) + lambda/m * Theta1(:,2:end);

grad = [Theta1_grad(:) ; Theta2_grad(:)];

end
```

A.4.2 Randomize Weight Initialization

```
function [ init_parameters ] = randInitWeights( num_parameters, interval)
%RANDINITIALIZEWEIGHTS

init_parameters = interval(1) + (interval(2) - interval(1)) .*...
                  rand(num_parameters,1);

end
```

A.4.3 Validation

```
function [ rel , faulty_indexes ] = predError( Output, p_res )
%PREDERROR

res_out = 0;
faulty_indexes = [];
for i=1:size(Output, 1)
    if(sum(p_res(i) == find(Output(i,:))))
```

```
       res_out = res_out + 1;
    else
        faulty_indexes = [faulty_indexes; i];
    end
end

rel = res_out/size(Output,1);

end
```

Bibliography

Genie and smile, 2014. GeNIe and SMILE. http://genie.sis.pitt.edu/index.php/downloads. Cited on pages 36 and 54.

Ahmad Taher Azar and Shaimaa Ahmed El-Said. Probabilistic neural network for breast cancer classification. *Neural Computing and Applications*, 21: 1 – 15, 2012. ISSN 09410643. URL https://lt.ltag.bibl.liu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=aph&AN=90472899&site=eds-live. Cited on page 5.

David Heckerman, John S. Breese, and Koos Rommelse. Decision-theoretic troubleshooting. March 1995. URL http://doi.acm.org/10.1145/203330.203341. Cited on page 5.

Finn V. Jensen and Thomas D. Nielsen. *Bayesian Networks and Decision Graphs*. Springer Science + Business Media, 2007. Cited on pages 13, 16, 17, 18, 19, 22, 44, 49, and 59.

Helge Langseth and Finn V. Jensen. Decision theoretic troubleshooting of coherent systems. *Reliability Engineering and System Safety*, 2003. doi: http://dx.doi.org/10.1016/S0951-8320(02)00202-8. URL http://www.sciencedirect.com/science/article/pii/S0951832002002028. Cited on page 5.

Baoli Li and Liping Han. *Intelligent Data Engineering and Automated Learning – IDEAL 2013*. Springer Berlin Heidelberg, 2013. Pages: 611-618. Cited on pages 5, 13, and 33.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval / Christopher D. Manning, Prabhakar Raghavan, Hinrich Schütze.* Cambridge : Cambridge University Press, 2008, 2008. ISBN 9780521865715. URL https://lt.ltag.bibl.liu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=cat00115a&AN=lkp.687061&site=eds-live. Cited on page 13.

Andrew Ng. Machine learning course, 2014. Coursera videos. https://class.coursera.org/ml-005. Cited on pages 22, 23, 25, 26, and 40.

Volvo Penta. Workshop manual, group 30 electrical system, b 2(0), 2006. Volvo Penta. Cited on pages 3, 7, 8, 10, and 57.

Anna Pernestål. *Probalistic fault Diagnosis with Automotive Applications*. Department of Electrical Engineering, Linköping University, 2009. Cited on page 5.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition, 2003. ISBN 0137903952. Cited on pages 13, 19, 20, 21, 22, 44, and 60.

Yousef Shatnawi and Mahmood Al-khassaweneh. Fault diagnosis in internal combustion engines using extension neural network. *IEEE Transactions on Industrial Electronics*, 61(3):1434 – 1443, 2014. ISSN 02780046. URL `https://lt.ltag.bibl.liu.se/login?url=http://search.ebscohost.com/login.aspx?direct=true&db=buh&AN=89975758&site=eds-live`. Cited on page 5.

Håkan Warnquist. Computer-assisted troubleshooting for efficient off-board diagnosis. *Reliability Engineering and System Safety*, 2011. URL `http://liu.diva-portal.org/smash/record.jsf?pid=diva2:411037`. Cited on page 5.

Yusha Wang Yingping Huang and Renjie Zhang. Fault troubleshooting using bayesian network and multicriteria decision analysis. *Advances in Mechanical Engineering, vol. 2014, Article ID 282013*, 2014. doi: 10.1155/2014/282013. URL `http://www.hindawi.com/journals/ame/2014/282013/abs/`. Cited on pages 5 and 51.

Upphovsrätt

Copyright