Automotive Applications of Diagnostic Condition Monitoring

CHRISTOPHE TREMBLIN

MSc by Research in Pattern Analysis and Neural Networks

Supervisor: Doctor Ian Nabney

ASTON UNIVERSITY

October 1999

Aston University

Automotive Applications of Diagnostic Condition Monitoring

Christophe Tremblin

MSc by Research in Pattern Analysis and Neural Networks, 1999

Thesis Summary

This project is sponsored by GenRad, a leading supplier of vehicle diagnostic and service information solutions for complex electrical and electronic systems in the automotive and aerospace industries. The long term goal is to create more sophisticated diagnostic systems that incorporate signal processing of noise/vibration signals and that can provide global diagnostic information (i.e. fuse information derived from multiple sources). The aim of this project is to test the feasibility of using neural networks and belief nets for condition monitoring and fault diagnosis in the automotive industry. Neural networks are used for signal processing at the individual sensor level, and Bayesian belief nets are used for reasoning about the predictions made by individual neural networks.

Keywords: condition monitoring, rotating machine, vibration analysis, belief network, neural network

Acknowledgements

Sincere thanks

- to Doctor Ian Nabney for supplying me good advises and making connections to the mechanical department.
- to Doctor John Penny and Mr. Andrew Glendenning of Modiarot project for the time they spend to explain me some vibration and mechanical notions.
- to Genrad and Mr. Marc White for sponsoring my project.
- to Institut d'Informatique d'Entreprise which permits me to go to the United Kingdom for one year.
- to my flatmates Pierre Alain Marzio and Bruno Vincent for a lot of interesting discussions not only about our projects.
- to all the MSc Students and NCRG staff for this year.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The objective of this work was to develop and evaluate a system for condition monitoring and fault diagnosis using neural networks and belief networks. This model is based on signal processing of vibration signals which is generally accepted as a good tool to diagnose the state of the rotating machine.

Vibration monitoring of the rotating machine has been used during the past 30 years. Since 1920's, experts have known that the spectrum of the vibration signal could be used to detect unbalance in rotating machines. In the sixties, it was identified too as a good means to detect cracks, misaligned couplings and other undesirable conditions. Such conditions introduce a perturbation in the vibration signal which experts are able to interpret. In the past 10 years, computer-based systems have been created to use the vibration signal for condition monitoring purposes. [7]

Our intention is to prove that belief networks and neural networks can be used to infer the state of a rotating machine from its vibration signals. Neural networks are used for signal processing at the individual sensor level, and Bayesian belief nets are used for reasoning about the predictions made by individual neural networks.

Building a condition monitoring tool requires background information from different fields: Vibration analysis, knowledge of rotating machines, etc. So before beginning to build a condition monitoring tool it is important to known where to search for expertise and how that expertise is structured already.

The objective of knowledge elicitation is to milk all available sources of information such as talking to experts, reading books, etc. There does not seem to be a recognised methodology for the process of knowledge elicitation. Some methods work well for certain domains and others don't. Typical sources for diagnostic information are:

- Literature on vibration analysis

- Manufacturer's operations and maintenance handbooks

- Expertise of a diagnostician

- Operator experience

- Cases derived from operation.

- ...

Nevertheless, the knowledge acquisition is a difficult process because the different sources can have different and sometimes conflicting opinions.

The chapter 2 reviews briefly how the vibration diagnosis is carried out today.
The chapter 3 describes the proposed model and the rig data used for test the model.
In the chapter 4, a method for finding the characteristic symptoms of a fault is described and the methods used to calculate the symptom value are explained.
Chapter 5 is a description of Belief Networks. In addition, some techniques used to build this type of networks are explained.
Chapter 6 shows the results we obtain with this model for a particular rotating machine.
The last chapter gives the conclusions of this work and some suggestions for the further work.

Chapter 2

Vibration analysis for diagnosis of machinery

Vibration monitoring and analysis is used in industry for troubleshooting rotating equipment problems and as a predictive maintenance tool. It is generally accepted as a good tool to diagnose a machine's health.

Typically, all these monitoring systems, whether they are designed for on-line or off-line use, can carry out relevant data reduction, trending and presentation but the actual diagnosis of machinery health is normally carried out manually by experts with experience both of instrumentation and of the dynamics of the machine or process. However, as the correct interpretation of such data is a fairly complex task, experts are rare and need a long time to build up experience.

2.1 Definition of the vibration diagnosis problem

Diagnosing a machinery failure from vibration data is a difficult and complex task due to :

- The large amount of data involved.

- The many possible ways of displaying and pre-processing data.

- Each fault may have a large number of characteristic symptoms which may or may not occur at the same time.

- Each machine has its own signature, which depends on where it is installed, its operating condition, machines installed in the neighbourhood etc.

In this section an attempt is made to highlight some of the problems which occur when trying to diagnosis the state of health of a rotating machine. The knowledge necessary for diagnosing the state of the rotating machine is difficult to acquire and needs a long learning time. So the experts are rare, this simple fact is perhaps the biggest problem for a person who attempts to built a condition monitoring tool.

Before considering mechanical faults, we have to examine the problems linked to failure of sensors and human errors. Many of the problems which occur are simply instrumentation failure, sensor and wiring failure being the most frequent.

A complete failure of a sensor may be interpreted as the machine running extremely smoothly. Nevertheless, this particular type of failure is easy to detect. More bothersome is partial failure which may result in an erratic signal, recognising this type of failure is often a problem for an expert but can be extremely difficult for an computer-based system.

Operating conditions have a great influence on the vibration level of the machine. A machine running with low load and low speed has a much lower vibration level than a machine running at maximum power.

The vibration signature can change during the lifetime of the machine. So, the diagnostic system has to be upgraded at regular interval in order to prevent the system becoming obsolete.

Some faults have different causes: unbalance may be caused by a bow in the shaft, by impact or by an sudden temperature variations. A fault can have different consequences. A coupling misalignment may produce a vibration of $1\times$ rotational speed in some cases, and produces a vibration of 2× rotational speed in other cases. Although a 1× component is usually blamed on unbalance or bearing clearance, a one-time rotational speed vibration can be often be cleared up by close alignment. Axial vibration is also a good indicator of misalignment.

As can be seem from the above the symptoms associated with the individual faults often overlap, making fault diagnosis difficult. There are also severe limitations on sensor types, locations and numbers as well as in the methods of data acquisition and there is a real need to ensure that the data being used for diagnosis are accurate. An other important aspect is the necessity of tailoring the system to the specific machine and its components [18].

2.2 How is vibration diagnosis carried out?

To measure the vibration signal, proximity sensors and acceleration sensors are used. A proximity sensor is a non-contact device, which measures the displacement or position of an observed surface relative to its mounting surface. An acceleration sensor measures the acceleration of the observed surface. The two types of sensor basically carry the same information. But, the proximity sensors are more efficient for lower frequencies and acceleration for higher frequency components.

To measure the relative movement of the shaft in the radial direction, a pair of proximity sensors and a pair of acceleration sensors equip each bearing of the rotating machine. To record the axial movement of the shaft as well, one or more probes may be placed at the end of the shaft.

The signal obtained by these sensors has to be pre-processed in order to be useful for the diagnostician. The following presentations are often used:

- Plots of the signal for examining the magnitude, stability, etc. of the vibration signal. Interpretation of these plots may indicate the nature of the vibration

(Figures 2.1 and 2.2).

- Comparative spectrum plots for tracking change in the spectral content over a period of time. Comparing new vibration data with old data, where the state of the machine is known can provide useful information (Figures 2.3 and 2.4).

- Position of the rotor in the bearing. This knowledge can provide warning of major changes in the alignment state of the machine and an indication of bearing wear.

The development of a conditional monitoring tool will, to a high degree, depend on the co-operation of an experienced diagnostician and has to be tailored to some particular machine in order to be successful. [13][3][12]

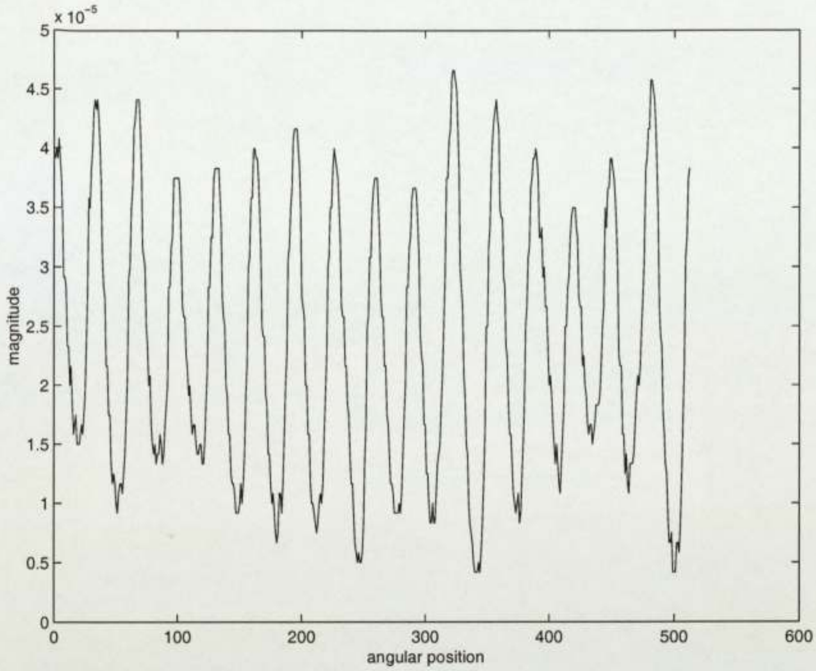


Figure 2.1: Vibration of a rotating machine in normal operating condition. This signal is obtained from a proximity sensor

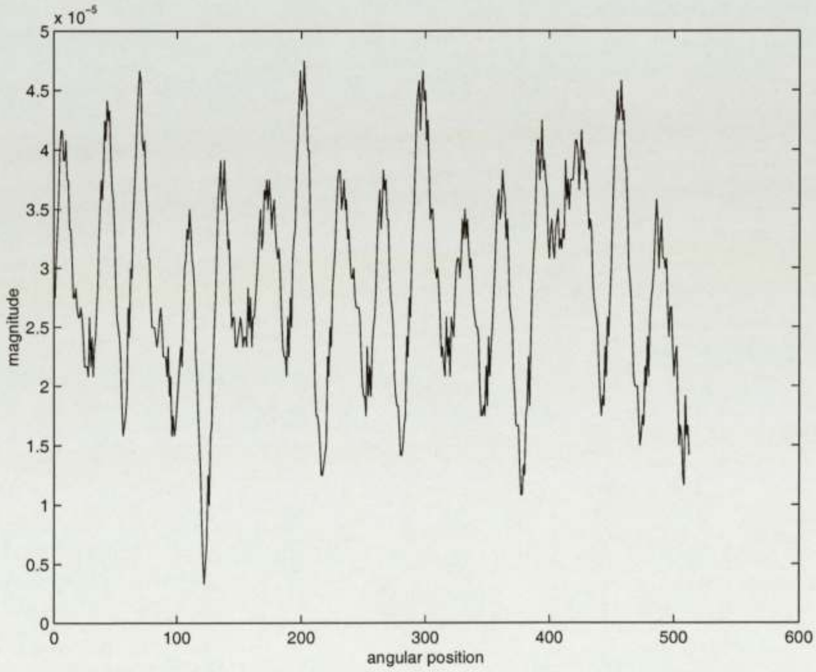


Figure 2.2: Vibration of a rotating machine in unbalance fault. This signal is obtained from a proximity sensor

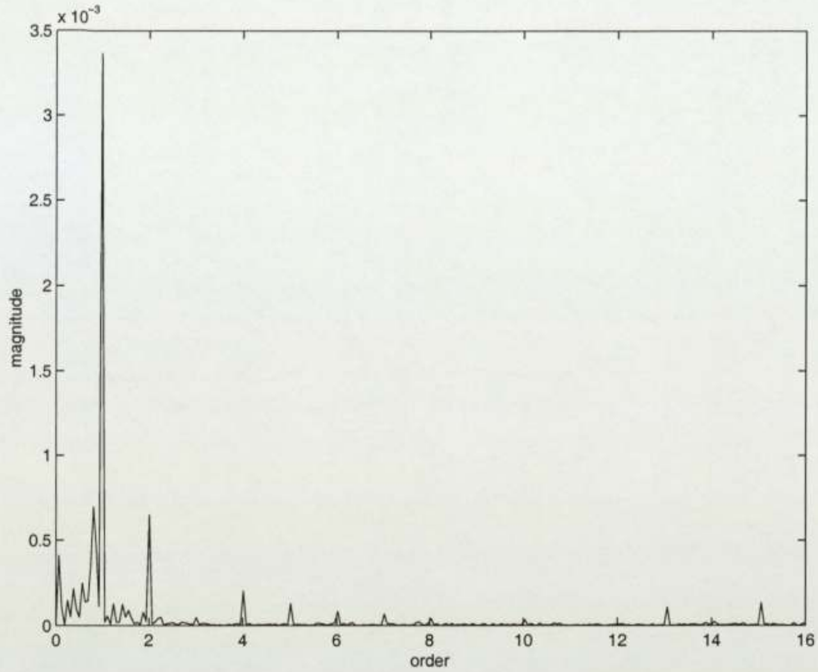


Figure 2.3: Spectrum plot of the signal in figure 2.1

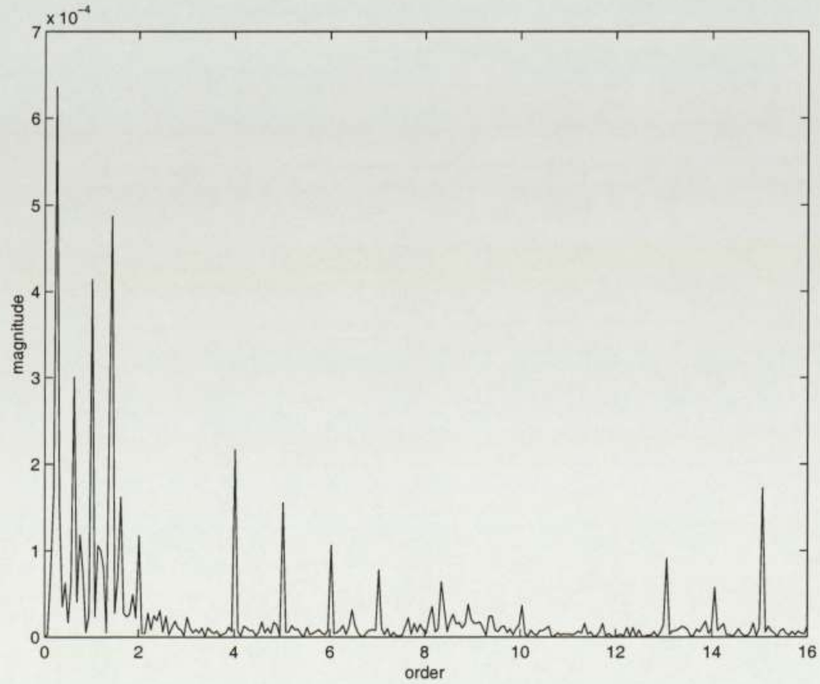


Figure 2.4: Spectrum plot of the signal in figure 2.2

Chapter 3

A Framework for a Condition Monitoring System

3.1 Description

The vibration of a mechanical system can be modelled as a time varying function of the machine's condition:

$$v(t) = f(c(k), t)$$

In some cases it may be possible to estimate this function $f(.)$ from the dynamics of the machinery. In general it is unknown. For condition monitoring purposes, the objective is to determine the machine's condition $c(k)$ from the vibration signal $v(t)$.

The inverse of the function $f(.)$ is required to extract a single sample of the condition from a large number of vibration data points. One method of obtaining the inverse of this function is by training a neural network, however this has the disadvantage of requiring a network with a large number of inputs.

A more common approach to this inversion process is to split the process into a symptom extraction phase and a condition classification phase. The symptom extraction phase transforms the vibration signal into a vector of symptoms which are characteristic of the different possible states of the machine. The second phase uses this vector to diagnostic the state of the machine.[1]

The model based diagnosis system being developed is shown schematically in the Figure 3.1. It uses neural networks to extract the symptoms from the vibration signal and a belief network to infer the state of the machine.

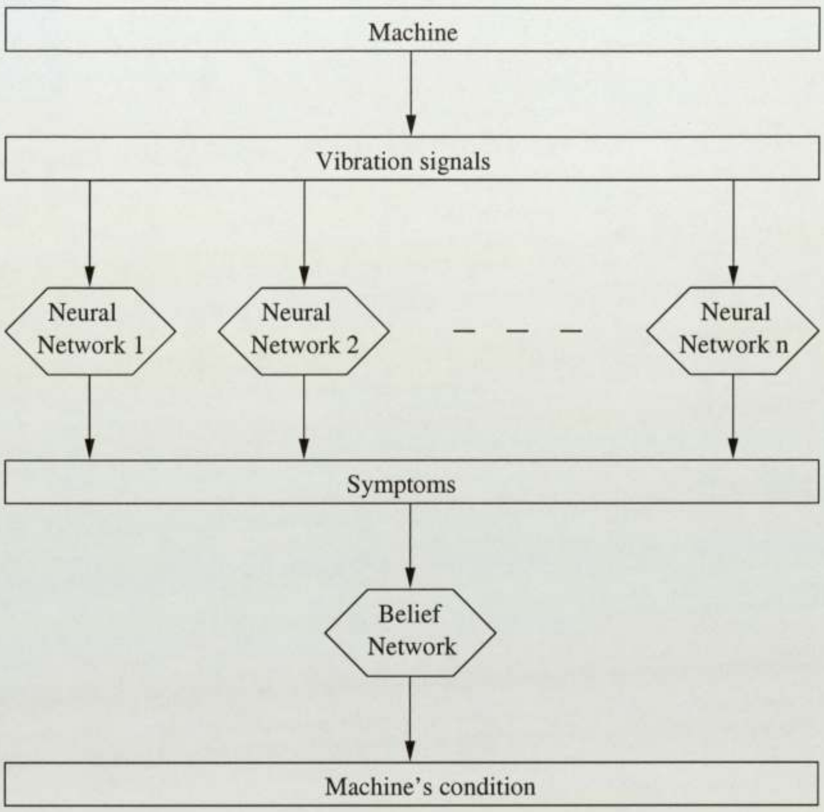


Figure 3.1: Simplified schematic of the model based diagnostic system

3.2 Interest of using Belief Network

Using Belief Networks could be profitable for several reasons. A belief network works in both directions (i.e. symptoms towards faults and faults towards symptoms). It is interesting because generally a condition monitoring system works in the direction symptoms towards faults, so this system needs $\sum_{i=1}^{Nfaults} 2^{Nsymptom_i}$ different probabilities to work ($Nsymptom_i$ is the number of characteristic symptoms of the fault i). The belief networks could works in both directions, so we can work in the direction faults towards symptoms. In this case, we need the probability $p(symptom_i/fault_{i_1}, ..., fault_{i_k})$ to train the network so we have to evaluate *only* $\sum_{i=1}^{Nsymptom} 2^{Nfault_i}$, where $Nfault_i$ is the number of faults which exihibit the symptom i.

The other interesting fact is that a belief networks is more flexible than a rule-based system. So, it is easier to experiment with a belief networks (for example see the effect of a sensor failure, change the location of a sensor ...).

In addition, a belief network could be updated in real time allowing the condition monitoring tool to evolve with its machine and with the environment of the machine. For example, the vibration signal could change with the machine's age (wear of pieces) so the condition monitoring tool has to be updated in order to be efficient.

3.3 The machine used

For this study, I used data from the Modiarot project, an EU Brite/EuRam collaborative research project to develop a method of fault diagnosis in rotating machinery.

3.3.1 Description

The rig consists of two rotors, which may be rigidly or flexibly coupled. Each rotor is supported by two bearings. To simulate the faults, the rig has five balancing disks, each of which has 24 equally spaced holes in which balancing weights may be mounted. To measure the signal, we have several sensors :

- two acceleration sensors (one horizontal the other vertical) for each bearing.

- two proximity sensors (+/− 45 degrees from Top vertical) for each bearing.

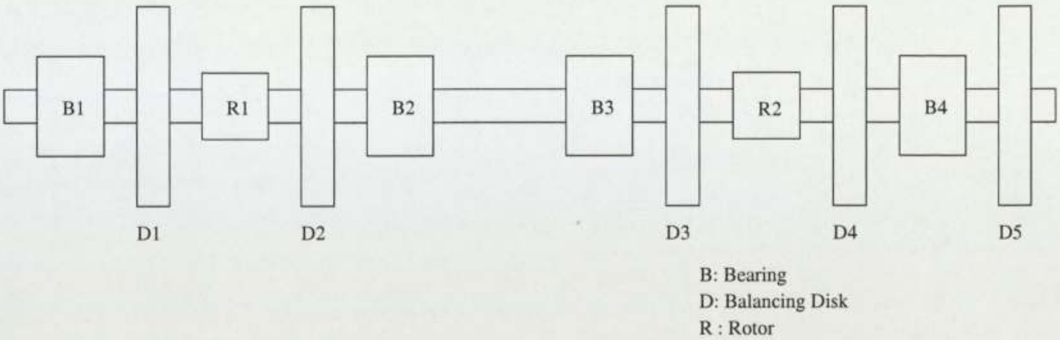


Figure 3.2: Rig of the Modiarot project

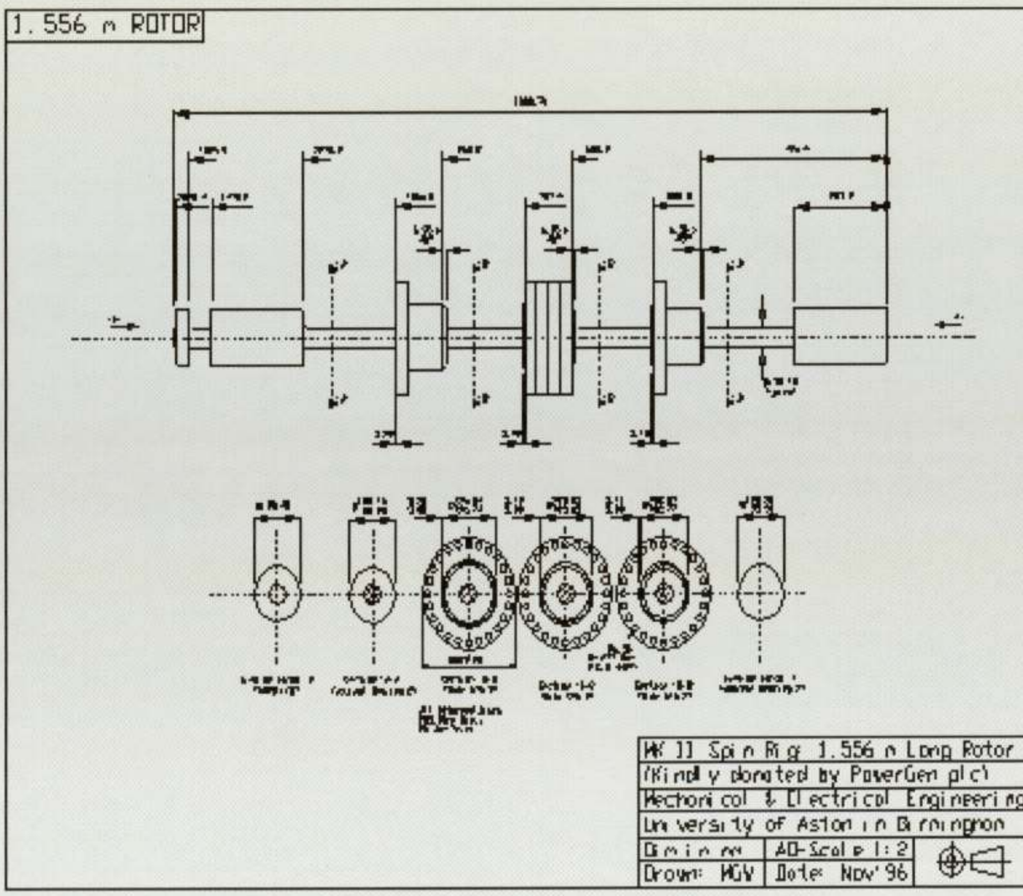


Figure 3.3: Technical Diagram of a Rotor

3.3.2 Rig data

The Modiarot team carried out several tests. The sensors take 32 measurements by turn (at regular angle space) of the shaft. So, we note that the vibration signal is not in the time domain but in the *order* domain.

For a rotating machine, the interesting information is extracted at frequencies related to the speed of the shaft. So in the time domain, experts have to run the shaft at several speeds in order to obtain useful data. They plot the spectrum of the data as a function of the frequency and the speed of the shaft. The interesting information appears in the plane of equation $x = y$. This plot is called the **order spectrum**.

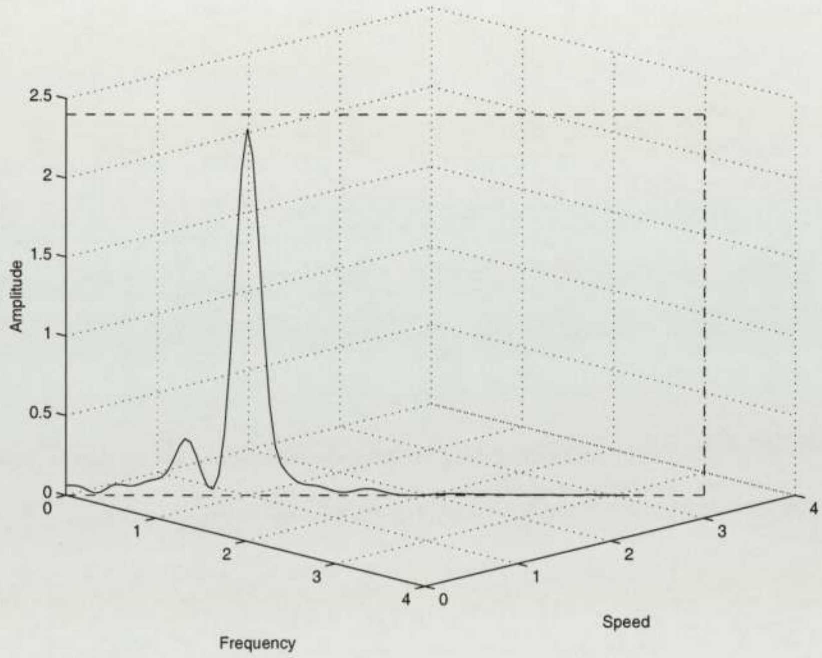


Figure 3.4: Order Spectrum

Obtaining this plot in the time domain presents several problems:

- need several measurement (different speed).
- the number of records per second increases with the speed in order to observe all the frequency spectrum.
- runout subtraction (i.e. even if the shaft is stopped the sensors record a value so we have to subtract this noise from the data).

However, from data recorded as a function of the angular position of the shaft, we can directly obtain the order spectrum. In this case, we speak about order instead of frequency [8].

The tests consist of a series of data where 16 turns of the shaft is recorded at 141 different speeds. The balancing disks allow the operator to simulate the **unbalance** fault under various conditions. The balancing weights have several locations (disk and hole in the case when several weights are mounted) and different weights.

Since the two rotors may be flexibly coupled, it is possible to simulate a **flexible coupling misalignment** under various conditions. To simulate this fault, the Modiarot team put a shim between the two rotor axis.

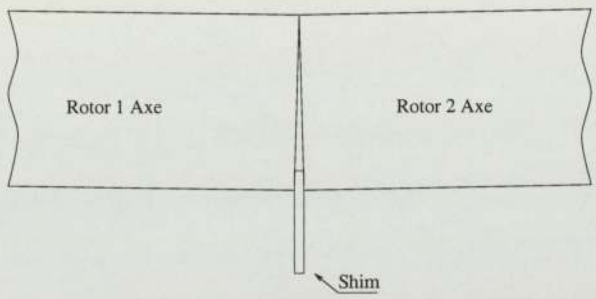


Figure 3.5: use of a shim to simulate a angular misalignment

Unfortunately, we do not have the data from all the sensors. Only the data from the 8 acceleration sensors and 4 proximity sensors are always recorded (the proximity sensor used is the "+45 degrees from Top vertical "for 3 first bearings, the "−45 degrees from Top vertical "for the last bearing).

For a complete description of the dataset see Annexe A.

Chapter 4

Symptom Selection

Symptom selection is a very important task in building diagnostic systems. The main aim of this selection is to find a set of symptoms which will, hopefully, contain all relevant and discriminatory information needed to infer the state of the rotating machine.

As an example consider the situation where three symptoms are used to distinguish between three faults. Each of the faults are characterised by the presence of two symptoms. The presence of a third symptom should have no influence. Now, as long as two symptoms are known to be true and one false, the diagnosis is clear and simple. If all three are known to be true, we have three different conclusions:

- we have a new fault which manifests itself with all three symptoms
- the three faults are present
- only two faults are present

4.1 Definition of the symptom selection problem

Like a human disease, each fault of a machine is characterised by a certain number of symptoms. To obtain these symptoms we need the expert's opinion and the available literature. But, this selection is not a easy task for several reasons.

All machines are different, even machines of the same type. We cannot be sure how a fault will manifest itself. Generally, we have little or no data available, simply because the fault has never appeared on the machine. In addition, a full mathematical description of a fault doesn't exist or demands too much effort to be adjusted to a particular machine. So, in deciding what to accept as a characteristic symptom of a fault, we have to proceed carefully and ask an expert's help. Nevertheless, the experts cannot solve all the problems. Different expert may have different opinions on how important certain symptoms are to diagnose a particular fault.

To built a computer-based diagnostic system, it is important to find symptoms which distinguish between the different states of a machine. But, experts and the available literature is more focused on what characterises a fault and not on what will discriminate it from the other faults.

The question is, which symptoms to use and how many of these should be extracted from the data raw. At least in theory, if the number of symptoms is sufficient it is possible to distinguish between all possible faults. That, however, is theory and only true in Boolean logic, where symptoms are either *false* or *true* and the absence of one symptom is sufficient to abandon a particular fault.

In practice, the more symptoms that are known to be present the more difficult it is to differentiate between faults. This paradox can be explain by the fact that the symptoms are more or less important for diagnostic a fault and are often characteristic of several faults. For example, if two faults have several symptoms, which have a little influence to diagnostic the fault, in common. If all these symptoms are present, the two faults may be diagnosted present, even when their most characteristic symptom is absent.

The point is to find as many symptoms as necessary to distinguish between faults and preferably no more than that. On the other hand, the more faults one wants to diagnose the more symptoms one needs, where several characteristics are common to several faults. If the distinguishing symptoms are overestimated in their importance, then a fault will possibly be diagnosed even in the case when only a few symptoms are present. If a relatively low weight is assigned to them, it will be impossible to distinguish between similar patterns. In addition, in the case of sensor failure, it could be useful to have some redundancy.

Unfortunately, there does not seem to be a general solution to the problem of symptom selection.

4.2 How find the characteristic symptom of a fault

Obviously, the best way to find the characteristic symptom of a fault is to have its description. Nevertheless, if such information is not available or not sufficient, it would be interesting to see the influence of the fault on some *general* symptoms.

Frequency component	Change in subsynchronous frequency components Change in principal frequency components Odd frequency components 3, 5, 7x Half harmonics 1.5, 2.5, 3.5x Rotor stator resonant frequency Amplitude when passing first critical
Orbit	Orbit stability Shape (i.e. round), elliptic, flat etc. Direction of rotation

Table 4.1: Symptoms for condition monitoring of rotating machinery

Speed related information	Variations of vibration with speed
	Speed of current first critical
	Operating speed
	Speed stability
	RMS when passing first critical
General information	Overall axial vibration
	Overall radial vibration
	Stability of vibration
	Possible increase of first order over period of time
	Possible slight increase of vibration at low speed
	Increase of horizontal and vertical vibration
	Vibration stability
	Possible change of vibration pattern after start/stop
	Axial and Radial shaft position
Operating condition	Load changes
	Hot/cold gas etc.
	Temperature variations

Table 4.2: Symptoms for condition monitoring of rotating machinery

4.3 Faults Description

From the dataset provided by the team of the Modiarot Project, it is possible to test a diagnostic system with two different faults. The intention of this section is to describes these faults in order to find their characteristic symptoms.

4.3.1 Normal operating condition

Of course, conditions where a machine has very a low level of vibration, usually indicate that no significant mechanical or operational problems are present. However sometimes considerable faults in the machine do not cause high vibration. For this reason it is recommended to make additional measurements, for example bearing oil temperature or noise control. However, if the level of vibration exceeds a defined value this is a reliable sign that significant mechanical or operational problems are present in the machine.

4.3.2 Unbalance

Unbalance is the most common cause of vibration at rotating machine. According to the International Standards Organisation (ISO), unbalance can be defined as:

> ...that condition which exists in a rotor when vibratory force or motion is imparted to its bearings as a result of centrifugal forces.

A more intuitive definition of unbalance is an unequal distribution of weight of a part about its rotating centreline. Another common and useful definition for unbalance is:

> ...a condition which exits whenever the rotating centreline (shaft axis) and central principal axis of a rotor do not coincide.

The central principal axis is the axis about which the weight of the rotor is equally distributed.

Unbalance is a linear problem. If a rotor is out of balance, it should be out of balance by the same quantity, through 360 degree of rotation. For this reason unbalance is characterised by a frequency component of $1 \times w$ (i.e. first order component). But, some other faults have dominant first order components too, such as shaft bow, bearing misalignment, etc. For this reason it is difficult to distinguish unbalance from these faults.

The following conditions can be helpful for determining unbalance in a system:

- Frequency component at half shaft or fractional shaft speed is not present.

- High frequency component ($2 \times w$, $3 \times w$, ...) are small.

- The unbalance has a fixed phase angle with respect to a shaft reference mark.

- Vibration is related to speed but is not related to load.

Under normal conditions, sinusoidal signals are obtained. When excessive mass unbalance is present, however, the behaviour of a bearing can cause truncated signals that introduce higher-order vibrations ($2 \times w$, $3 \times w$, ...) with amplitudes lower than the $1 \times w$ vibration.

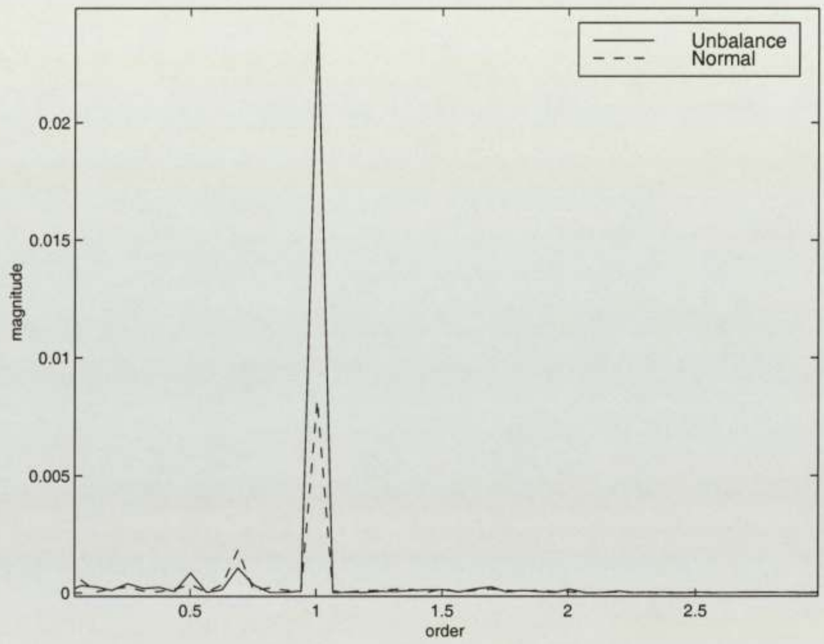


Figure 4.1: Order spectrum of a typical unbalance fault

Types of unbalance can be classified according to the geometric relationship between the center of mass, the shaft axis and principal inertia axis of a rotor. Essentially, there are three types of rotor unbalance :

- *static* is a condition that exists when the center of mass is not on the shaft axis and when the principal axis of inertia is parallel to the axis of rotation. Force unbalance is another name for static unbalance.

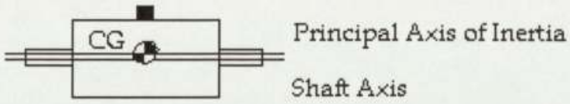


Figure 4.2: Static unbalance

- *couple* is a specific condition that exists when the center of mass is on the shaft axis and when the principal inertia axis is not parallel with the axis of rotation.

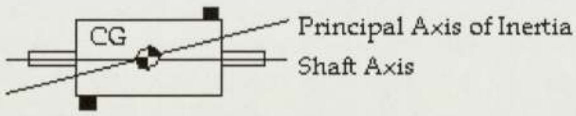


Figure 4.3: Couple unbalance

- *dynamic* is a specific condition that exist when the center of mass is not on the shaft axis and when the principal inertia axis is not parallel with the axis of rotation.

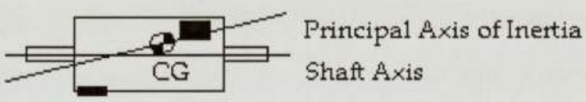


Figure 4.4: Dynamic unbalance

4.3.3 Flexible coupling misalignment

It is very difficult to align two shafts so that no forces exist which will cause vibration. For this reason vibration due to misalignment is almost as common as vibration due to unbalance. Flexible coupling misalignment results in two forces, axial and radial, which cause axial and radial vibration. These forces and therefore the level of generated vibration will increase with increased misalignment.

Misalignment occurs when two shafts are not aligned properly. Misalignment will cause a predominant vibration at a frequency of $1 \times w$ in many cases, but unlike unbalance, misalignment will often be accompanied by harmonically related frequencies, including $2 \times w$, $3 \times w$ and occasionally higher orders of rotating speed. In severe cases, the second-order component can exceed the first-order component.

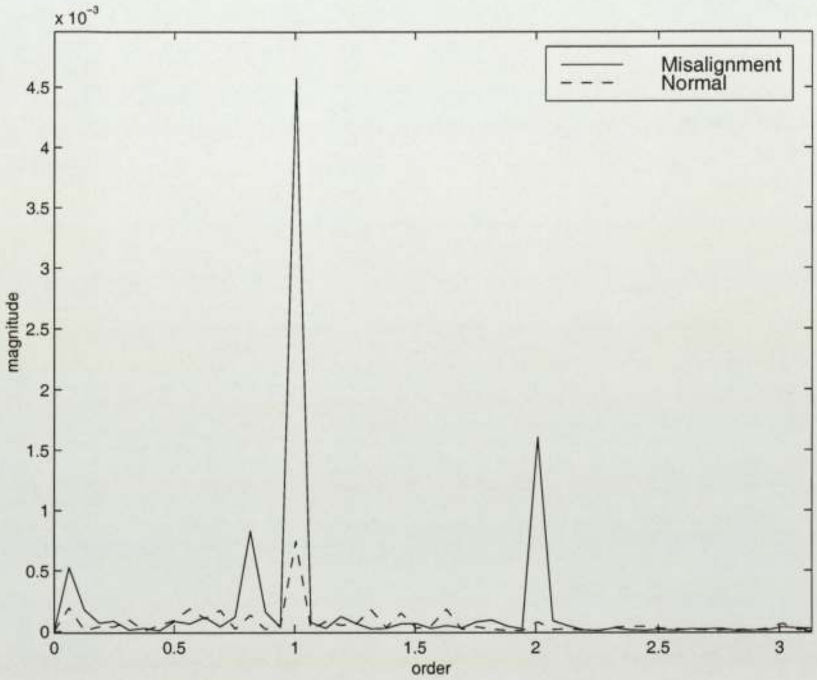


Figure 4.5: Power spectrum of a typical misalignment fault

Types of misalignment can be classified according to geometric relationship between the shaft axis of the rotors. Essentially, there are three types of misalignment between two rotors:

- *radial* is a specific condition that exist when the rotor's axis have a distance of a mm between them.

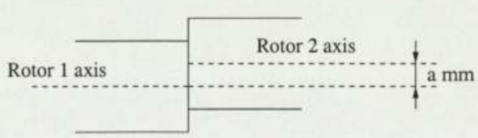


Figure 4.6: Radial misalignment

- *angular* is a specific condition that exist when the rotor's axis have a angle of *a* degree between them.

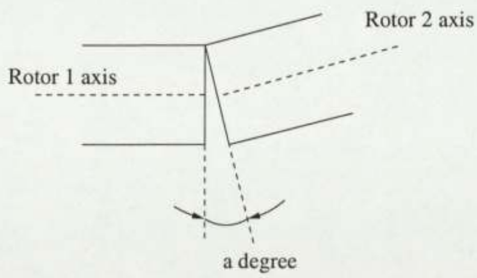


Figure 4.7: Angular misalignment

- *combination of these two.*

4.4 Characteristic Symptoms of the unbalance and misalignment faults

The team of the Modiarot project made a *rule base* or *fault matrix* which can be used as a guide to determine the relation between faults and symptoms. The fault matrix consists of a two-dimensional array of symptoms (columns) and faults (rows). At each (fault/symptom) pair, we associate two weights (α_{ij}/β_{ij}) where i is the fault index and j is the symptom index. The weight α_{ij} is to be used when symptom j is present. The weight β_{ij} is applied when the evidence is absent.

The rule base of the Modiarot team is used in a condition monitoring tool for rotating machines. This tool is built to distinguish between 11 different faults. So, we need only a part of the rule base to select the characterised symptoms of the unbalance and misalignment faults.

This rule base provides useful information: in particular, it helpful to decide which symptoms are the most useful for distinguishing between the two faults. For example, the symptom **2xR1x** is clearly important because its presences indicates a misalignment and the symptoms **05nld** and **05lod** can be ignored because they have the same influence for the two faults.

Symptom name	Unbalance	Misalignment
Major amplitude of axial vibration high (*a*)	0/0	30/0
Frequency Components at [0.10x-0.45x] (*sd*)	-10/0	0/0
Frequency Components at [0.45x-0.55x] (*sc*)	-10/0	0/0
Frequency Components at [0.55x-0.95x] (*su*)	-10/0	0/0
Frequency Component at 1x high (*1x*)	80/-50	30/-10
Frequency Component at 2x high (*2x*)	20/-10	40/-20
Frequency Component at 3x high (*3x*)	0/0	20/-10
The 1x is dominant (*1xd*)	35/-30	0/0
The 2x is dominant (*2xd*)	-30/0	20/0
Dominant frequency 0.5x not locked (*05nld*)	-25/0	-20/0
Dominant frequency 0.5x locked on (*05lod*)	-25/0	-20/0
Trend in 1x is accelerating (*1xT*)	-10/0	5/0
Trend in 2x is accelerating (*2xT*)	-5/0	10/0
Trend in 3x is accelerating (*3xT*)	-5/0	5/0
Background level increases (*b*)	-40/0	0/0
Relation (2xw/1xw) in percent (*2xR1x*)	0/0	60/0
The 1x vector is changed (*1xCaf*)	30/-40	10/0
The 2x vector is changed (*2xCaf*)	10/0	30/-10
The 3x vector is changed (*3xCaf*)	0/0	10/-5

Table 4.3: Extract of Modiarot's rule base

A description of these symptoms follows:

- **a**: Major amplitude of axial vibration is the amplitude of total vibration level measured in a direction parallel with the shaft axis.

- **sd, sc, and su**: these symptoms define whether subsynchronous components in a frequency spectrum are high.

- **1x, 2x, 3x**: these symptoms define whether the three first harmonics in a frequency spectrum are high.

- **1xd, 2xd**: these symptoms define the largest peak for some vibration components.

  1xd = $Present$ if harmonic $1xw$ is $> 1.41*$ all other frequency components.

  2xd = $Present$ if harmonic $2xw$ is $> 1.00*$ all other frequency components.

- **1xT, 2xT, 3xT**: if the level of the three first harmonics, or their vectors, show an increasing rate of change during several measurements over a period of time, these symptoms contain information about kind of change for these harmonics. The total change should be at least 25 percent.

- **b**: This symptom contains information about frequency ranges which are not specified above, for example due to changes in non-synchronous resonant response.

- **2xR1x**: information about the ratio between second order components and first order components.

- **1xCaf, 2xCaf, 3xCaf**: These symptoms contain information about whether amplitudes and phase angles of the first three harmonics are steady or have changed. These symptoms look for a difference in the magnitude of the complex vector (modulus and angle) between present and previous measurements.

We also need symptoms, which permit to distinguish between the normal operating condition and the two faults. The symptom **Major amplitude of radial vibration high** (r), according to the rule base, is enough for this purpose. For shaft vibration, this symptom is defined as the higher value of the peak displacement measured in two selected perpendicular measurement directions. For measurements on the bearing pedestal or casing it is defined as the maximum *Root Mean Square* vibration velocity.

4.5 Practical calculation of the symptoms

The value of a symptom is not interesting in itself but how it is related to other value or to a former measurement. In this study, we use the signals recorded under normal condition operation as references. The interesting value to have is not the numerical value of the symptom but its probability of presence according to a former measurement.

4.5.1 Signal partition

The signal is split in $\frac{N-Nsamp}{Nshift}$ portions where N is the number of point in the signal, $Nsamp$ the number of points in each portion of the signal and $Nshift$ the number of points between the beginning of each portion ($Nshift$ can be inferior at $Nsamp$).

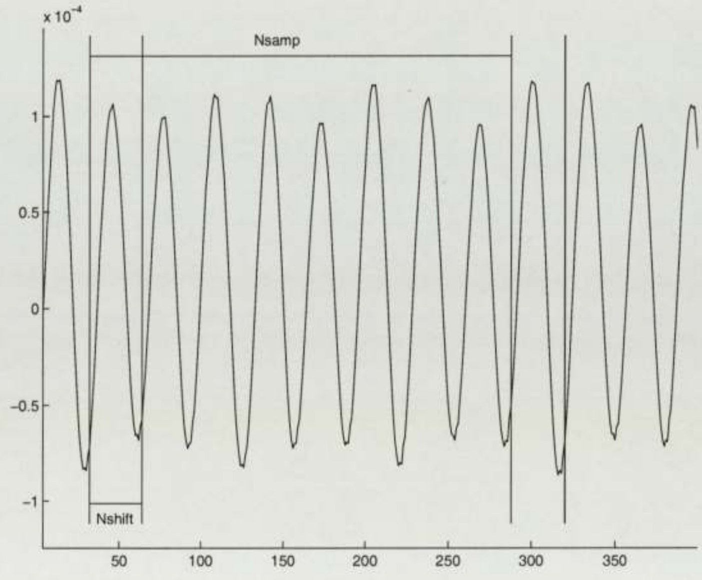


Figure 4.8: signal partition

For each portion of the signal, the symptoms are calculated. To choose the values of $Nsamp$ and $Nshift$ any particular techniques could be used. Nevertheless, for the symptom **r** the root mean square value for a digital signal is calculated thanks to the formula:

$$RMS^2 = \frac{1}{N}\sum_{i=1}^{N} x_i^2$$

where N is the number of data point and x_i the amplitude value at this data point. This quantity is a **constant** if N is enough great. So, we have to be sure that $Nsamp$ is large enough in order to respect this property.

4.5.2 Dependencies of the vibration signal

The symptoms are speed depend and bearing depend: the value of a symptom changes as a function of the localisation of the sensor which has recorded the signal.

So, the method we use to calculate the symptom probability has to respect care of these dependencies.

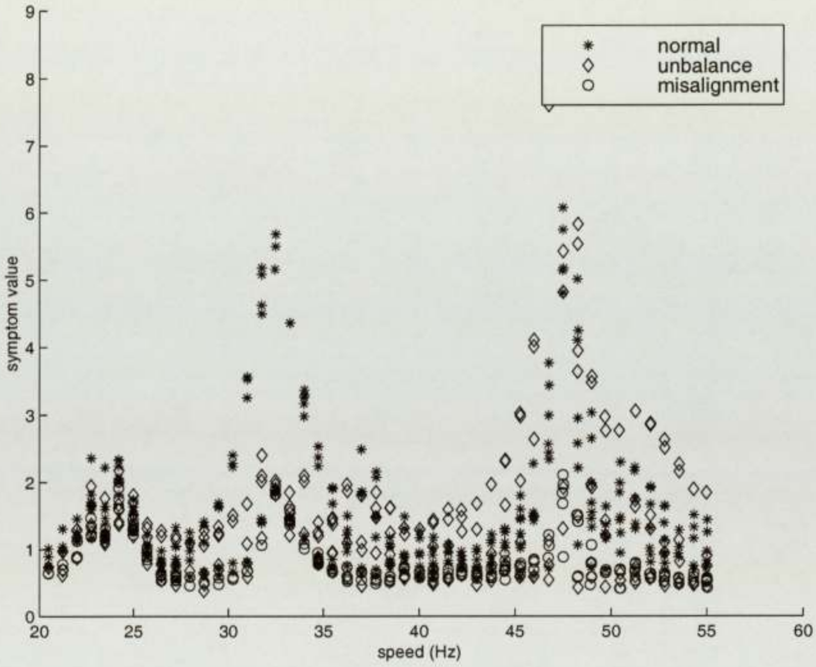


Figure 4.9: Numerical values of symptom **r** under different rotating speeds (data from proximity sensor of bearing 1)

4.5.3 Calculation with a threshold function

To express the probability of a symptom from its value, a soft threshold function can be used:

$$p = (s_1 = present) = y(x) = \frac{1}{1 + e^{(-wx - w_o)}} \tag{4.1}$$

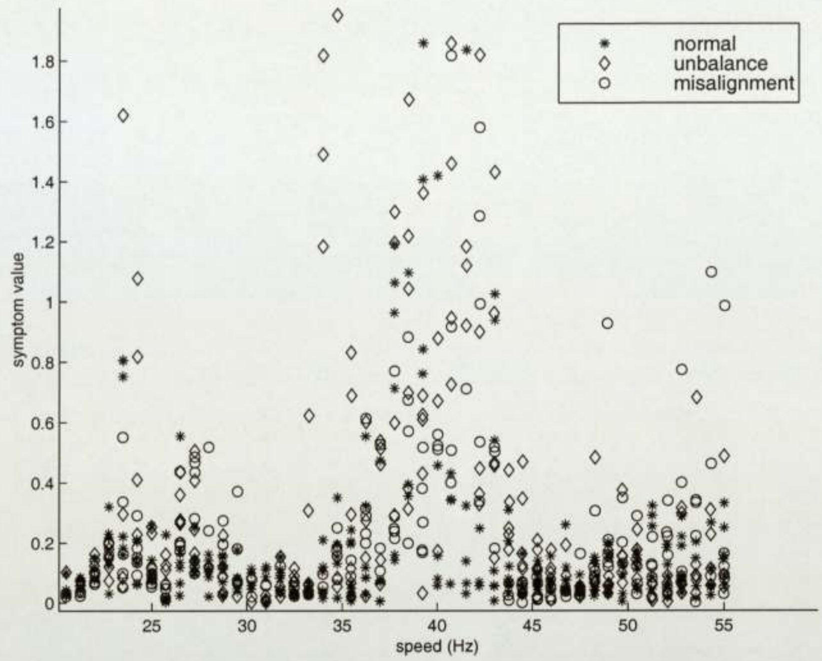


Figure 4.10: Numerical values of Symptom **2xR1x** under different rotating speed (data from proximity sensor of bearing 1)

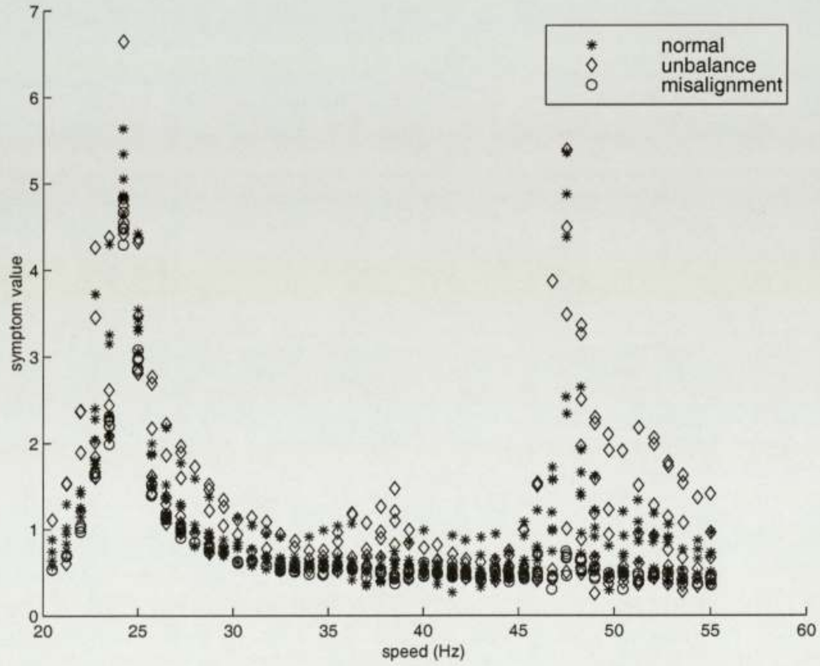


Figure 4.11: Numerical values of symptom **r** under different rotating speed (data from proximity sensor of bearing 3)

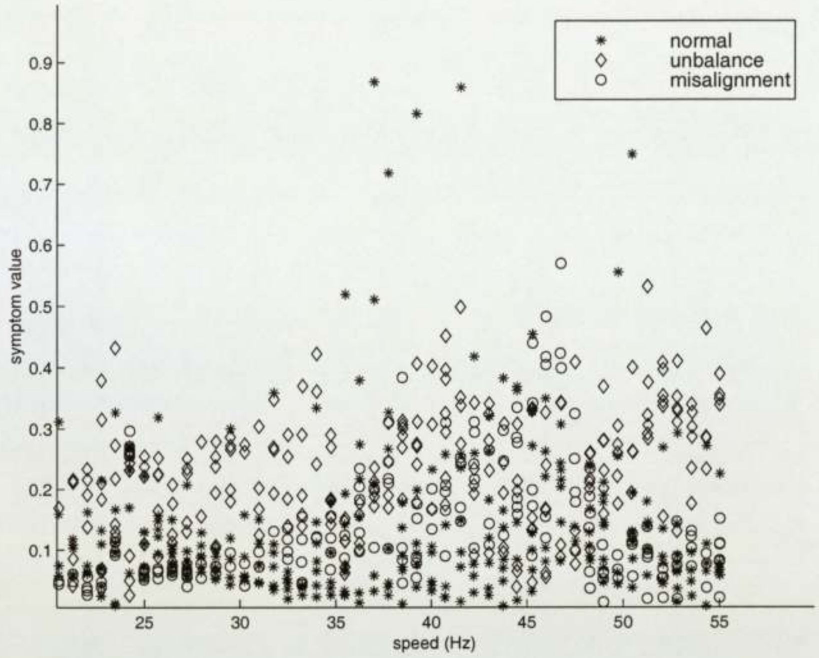


Figure 4.12: Numerical values of symptom **2xR1x** under different rotating speed (data from proximity sensor of bearing 3)

To choose the w and w_o coefficient we minimising the cross-entropy error function:

$$E = -\sum_{n=1}^{N}\{t_n \ln(y_n) + (1-t_n)\ln(1-y_n)\} \tag{4.2}$$

To perform this minimisation, we evaluate the derivatives $\frac{\partial E}{\partial w}$, $\frac{\partial E}{\partial w_o}$

$$\begin{aligned} \frac{\partial E}{\partial w} &= -\sum_{n=1}^{N} \frac{t_n}{y_n}\frac{\partial y_n}{\partial w} - \frac{1-t_n}{1-y_n}\frac{\partial y_n}{\partial w} && (4.3) \\ &= -\sum_{n=1}^{N} \frac{t_n - y_n}{y_n(1-y_n)}\frac{\partial y_n}{\partial w} && (4.4) \end{aligned}$$

with $(1-y_n) = y_n e^{(-wx-w_o)}$ so

$$\begin{aligned} \frac{\partial y_n}{\partial w} &= \frac{x_n e^{(-wx_n-w_o)}}{(1+e^{(-wx_n-w_o)})^2} && (4.5) \\ &= x_n e^{(-wx_n-w_o)}(y_n)^2 && (4.6) \\ &= x_n y_n(1-y_n) && (4.7) \end{aligned}$$

and finally

$$\frac{\partial E}{\partial w} = \sum_{n=1}^{N} x_n(y_n - t_n) \tag{4.8}$$

$$\begin{aligned}
\frac{\partial E}{\partial w_o} &= -\sum_{n=1}^{N} \frac{t_n - y_n}{y_n(1-y_n)} \frac{\partial y_n}{\partial w_o} && (4.9) \\
&= -\sum_{n=1}^{N} \frac{t_n - y_n}{y_n(1-y_n)} e^{(-wx_n - w_o)} (y_n)^2 && (4.10) \\
&= \sum_{n=1}^{N} (y_n - t_n) && (4.11)
\end{aligned}$$

Let's we introduce the gradient descent rule:

$$\Delta w = -\eta \frac{\partial E}{\partial w}, \qquad \Delta w_o = -\eta \frac{\partial E}{\partial w_o}$$

So we finally obtain the formulas:

$$\begin{aligned}
\Delta w &= -\eta \sum_{n=1}^{N} x_n (y_n - t_n) && (4.12) \\
\Delta w_o &= -\eta \sum_{n=1}^{N} (y_n - t_n) && (4.13)
\end{aligned}$$

which allows us to built a learning algorithm for the parameters w and w_o where η is learning rate parameter.

To initialise the algorithm, we can take any values for w and w_o. Nevertheless, we have to perform this algorithm for each bearing and for each rotating speed. So in order to reduce the computation time, we have to choose values for sensible initial w and w_o.

To initialise the algorithm, we assume that the probability of the mean of the *normal* value is 0.01

$$\begin{aligned}
w &= \frac{2\ln(99)}{x_2 - x_1} && (4.14) \\
w_o &= -\frac{(x_2 + x_1)\ln(99)}{x_2 - x_1} && (4.15)
\end{aligned}$$

where

$$\begin{aligned}
x_1 &= \frac{\sum_{n=1}^{N} x_n t_n}{\sum_{n=1}^{N} t_n} && (4.16) \\
x_2 &= \frac{\sum_{n=1}^{N} x_n (1 - t_n)}{\sum_{n=1}^{N} 1 - t_n} && (4.17)
\end{aligned}$$

4.5.4 Calculation with a Neural Network

since we have to train a networks for each bearing, for each symptoms and for each rotating speed, we have to choose a neural network which its training method could be performed quickly. There are two major class of neural network model:

1. **the multi-layer perceptron network**: model based on units which compute a non-linear function of the scalar product of the input vector and a weight vector.

2. **the radial basis functions network**: model in which the activation of a hidden unit is determined by the distance between the input vector and a prototype vector.

A Radial Basis Functions network (RBF) is chosen to perform this classification task. This choice is based on the fact that the procedures for training a RBF network can be substantially faster than the methods used to train a multi-layer perceptron model [2].

The radial basis function neural network structure consists of a network of processing elements, or nodes, arranged in layers. In this study, we use three layers of processing nodes: an input layer which accepts the input variable (numerical value of a symptom), one hidden layer, and an output layer (probability of presence of the symptom).

To train the RBF, we use the *Cross-validation* technique: the dataset is divided into S distinct segments. The network is trained with $S-1$ segments. We evaluate the error function using the last segment in order to test the network performance. This operation is repeated for each of S segments.

The Cross-Validation technique is used to find the best number of hidden neurones. In practice, we test the networks with 1, 2, 3, 4, 5 and 6 hidden neurones with $S = 3$.

Chapter 5

Belief Networks

A classic example of the use of belief networks is in the medical domain. Here each new patient typically corresponds to a new *case*, and the problem is to diagnose the patient (i.e. find beliefs for the unmeasurable disease variables), predict what is going to happen to the patient, or find an optimal prescription, given the values of observable variables (symptoms). A doctor may be the expert used to define the structure of the network, and provide the initial relations between variables (often in the form of conditional probabilities), based on his medical training and experience with previous cases. Then the network probabilities may be fine-tuned by using statistics from previous cases, and from new cases as they arrive.

In our case, a belief network is used to infer the state of a rotating machines from a symptom vector. The analogy with the medical network is obvious. Nevertheless, building a belief network for a rotating machines could be more difficult simply because, by luck, we have more doctors than rotating machine's experts. The other difficulty is that often, two humans are less differences than two different machines, so determines the symptoms probability could be easier.

5.1 Belief Networks and Probabilistic Inference

A belief network (BN) (also known as a Bayesian network or probabilistic causal network) is a directed acyclic graph where each node represents a scalar variable, which may be discrete, continuous, or propositiitional (true/false). The BN captures the relations (which may be uncertain, stochastic, or imprecise) between the set of variables represented by the nodes. Because of this, the words *node* and *variable* are used interchangeably throughout this document, but *variable* usually refers to the real world or the original problem, while *node* usually refers to its representation within the belief network .

A link between two nodes A and B exists if:

- A causes B.

- A partially causes or predisposes B.

- B is an imperfect observation of A.

- A and B are functionally related.

- A and B are statistically correlated.

the node A is called the *parent node* and the node B the *child node.* The precise definition of a link is based on conditional independence, and is explained in detail in [16] and [14]. However, most people use the notion of link without concentrating on the precise definition.

Furthermore, probabilistic relations are provided for each node, which express the probabilities of that node taking on each of its values, conditioned on the values of its parent nodes. These relations are express by a conditional probability table (CPT) (or in more generals terms a conditional probability function (CPF)).

After the belief network is constructed, it may be applied to a particular case. For each variable you know the value, you enter that value into its node. Then the belief network does probabilistic inference to find beliefs for all the other variables.

Parent A	**Parent B**	**Child C state Υ**	**Child C state Ψ**
state α	**state a**	p(C= Υ/A=α, B=a)	p(C= Ψ/A=α, B=a)
	state b	p(C= Υ/A=α, B=b)	p(C= Ψ/A=α, B=b)
state β	**state a**	p(C= Υ/A=β, B=a)	p(C= Ψ/A=β, B=a)
	state b	p(C= Υ/A=β, B=b)	p(C= Ψ/A=β, B=b)
state γ	**state a**	p(C= Υ/A=γ, B=a)	p(C= Ψ/A=γ, B=a)
	state b	p(C= Υ/A=γ, B=b)	p(C= Ψ/A=γ, B=b)

Table 5.1: Example of CPT

Depending on the structure of the network, and which nodes receive findings or display beliefs, a belief network may do diagnosis, prediction, classification, logic, arithmetic calculation, or any combination of these, to complete the probabilistic inference. The final beliefs are sometimes called posterior probabilities (with prior probabilities being the probabilities before any findings were entered). Probabilistic inference done using a belief network is called belief updating.

5.2 Example

This network is one of the simplest networks in condition monitoring because the assumption is made that all the faults and all the symptoms are independent. This network is a good starting point for beginning the construction of the real network.

Some faults can have a particular localisation on the shaft (unbalance for example), so it is interesting to create a symptom node for each bearing in order to permit the localisation of each fault in the shaft. In our application, links between faults and symptoms are created using the Modiarot base rule.

The following network is not complete: several nodes are omitted, simply because we don't need all the nodes for illustrate your purpose and the complete network is big.

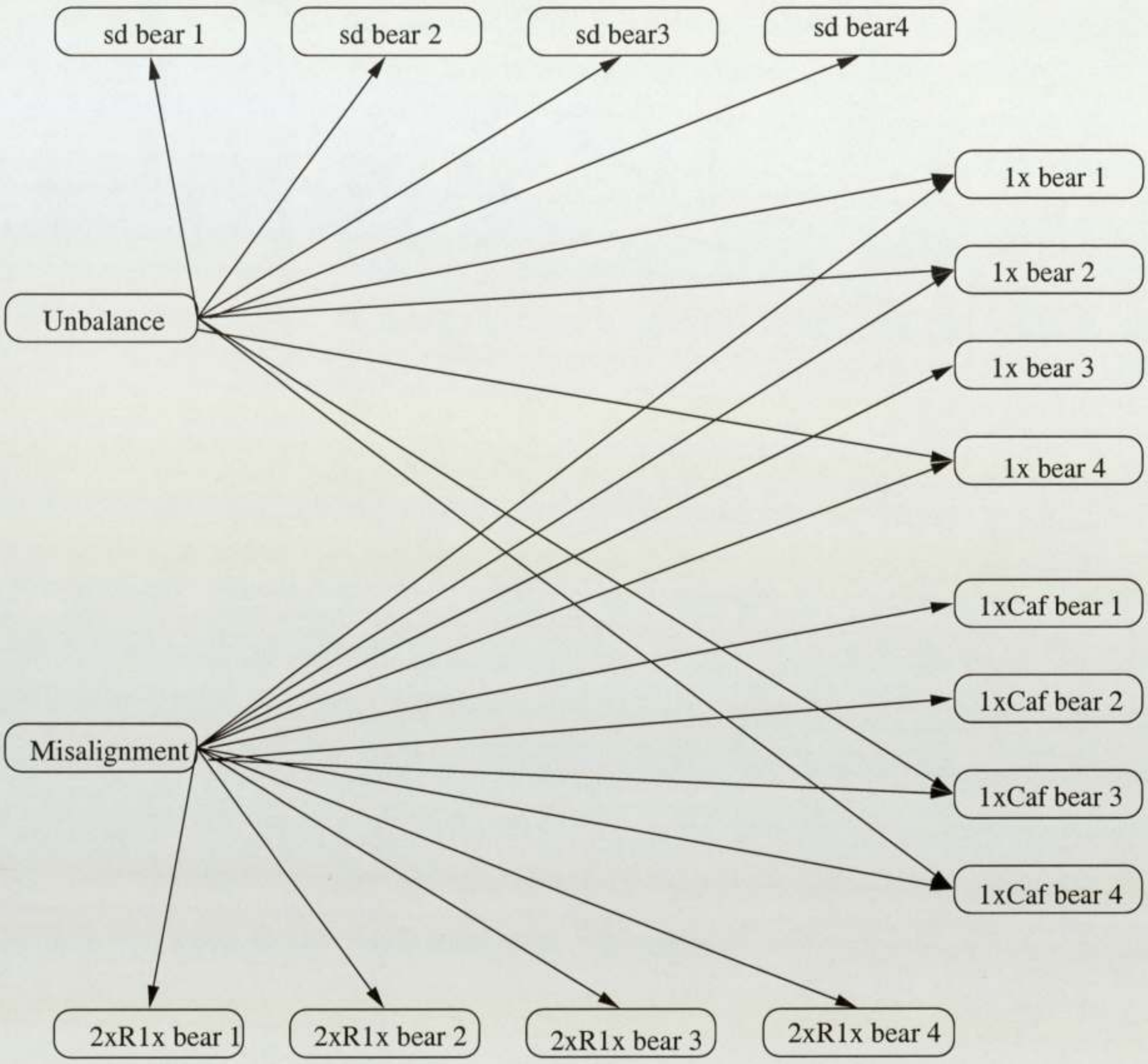


Figure 5.1: Belief network example

5.3 Constructing Belief Networks

The goal of this section is simply to provide a flavour of the process of structuring large networks and quantifying probabilistic influences. [9]

5.3.1 Structuring the network

When one tries to build a model, these is an inevitabe simplification of expert's knowledge, which is itself a simplification of the real world. The goal is to identify which elements or relationships are important and which can be omitted or abstracted. A model is generally built for a specific task so decisions about which elements to include in the model and which to ignore are made according to whether they seem likely to affect this decision. To refine the model a sensitivity analysis could be used.

A directed link shows a probabilistic dependency between two nodes; similarly a lack of link represents conditional independence. An expert is generally asked to find the links between the nodes; in this study the links were created according to the Modiarot's rule base. The guiding rule for selecting the direction of the link is to choose whichever direction feels most natural or the most easy to express.

We have to distinguish the two notions of influence and causality. Influence represents a probabilistic relation and can generally be expressed in either direction irrespective of the causal relation or lack of such between two nodes. So two nodes can be in *influence* but it is impossible to create a directed link between them (the belief network is acyclic). For example in a diagnostic belief we have the node *normal* and *fault X*. A relationships obviously exists between these two nodes, but is it the normal condition which cause the lack of fault or the presence of a fault which cause an abnormal condition?

The next work is to discritise the value of each node. For example, the node could take two states **present** or **absent**. A node is generally a binary variable, nevertheless some nodes could have several states. We have to be careful at this stage because increasing the number of a node's states, increases the size of its conditional probability table and more importantly the size of its childrens CPT.

When the discritisation step is over, we have to find the CPT of each node. Generally, a expert was asked to make qualitative judgements about the influence of each parent on theirs children. Nevertheless, if such knowledge is not available, some techniques permit the construction of the CPT from real data. This technique will be explained in the next section.

5.3.2 Refinement of model structure

When the CPT were built some refinements were be made. This happened whenever modifications to the network structure made it easier to assess, or the expert changed his mind after further consideration about earlier assumptions.

Violation of conditional independence

One important example concerned violations of conditional independence. If, on further reflection, the expert judged that two variables with a common parent but no direct link between them are not conditionally independent, then an additional node may be required to fix the problem.

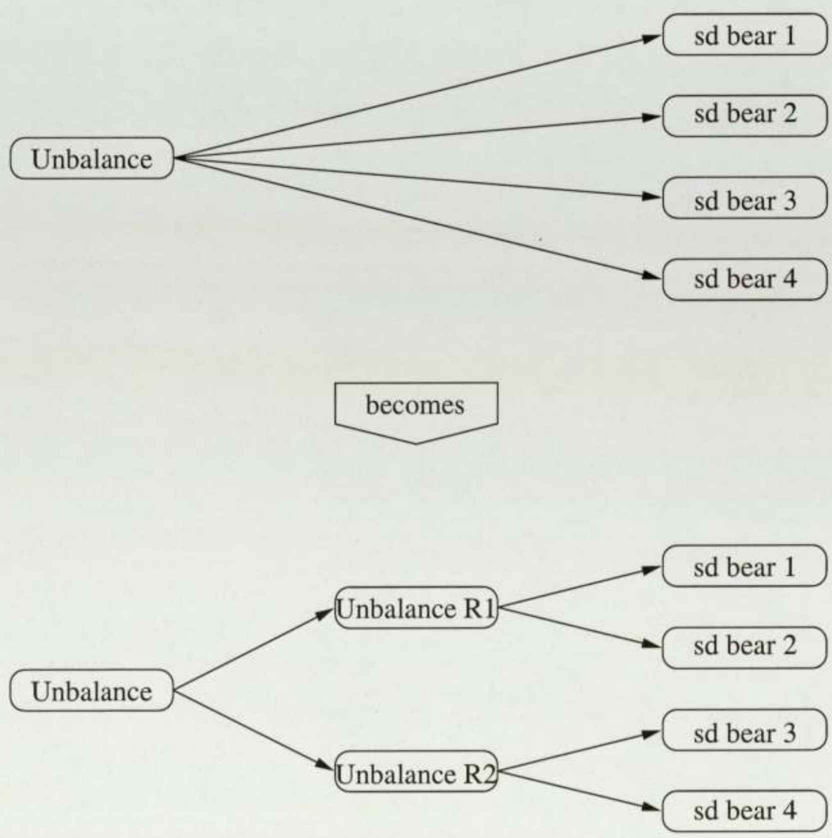


Figure 5.2: Addition of node renders the two lower ones conditionally independent

For example, the unbalance fault is located in one rotor of the rotating machines so its characteristic symptoms are occuring in the two bearings which support the rotor are not independent.

Discovery of a failure of conditional independence is often an indication that there may be a hidden variable whose introduction can facilitate the assessment process [15].

Identical effects

Another way in which an additional node can simplify the assessment process is where two factors are judged to have identical effects if either or both were present. For example, the nodes *Unbalance* and *Misalignment* at a given level were both judged to have the same impact on *1x bear 1*, *1x bear 2*, *1x bear 3* and *1x bear 4*. Thus the additional node *1x common* was defined.

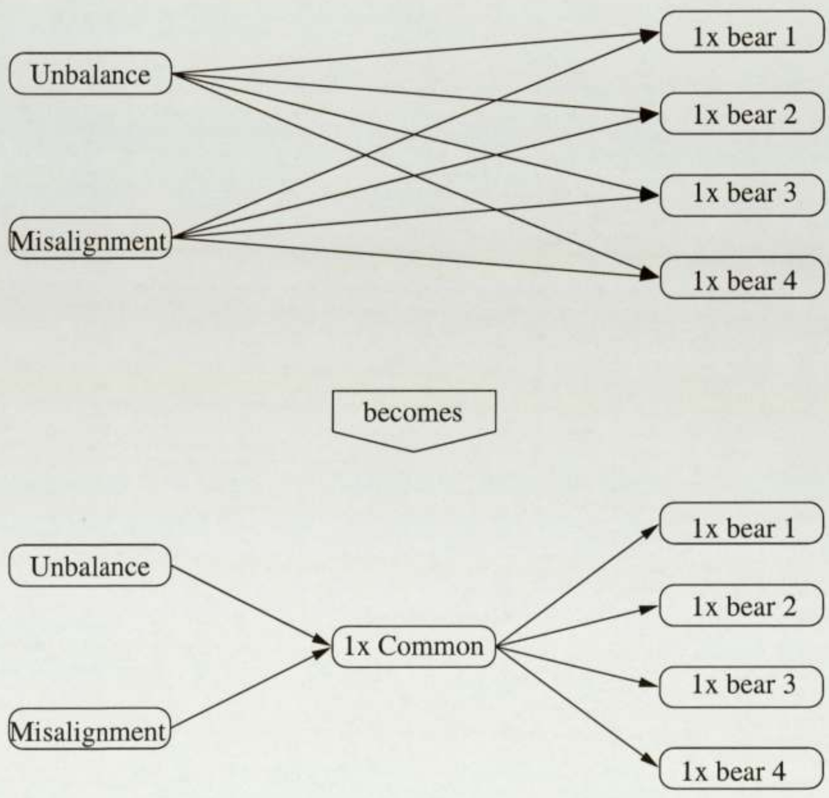


Figure 5.3: Unbalance and Misalignment have the same effect

So, after these refinements our example becomes:

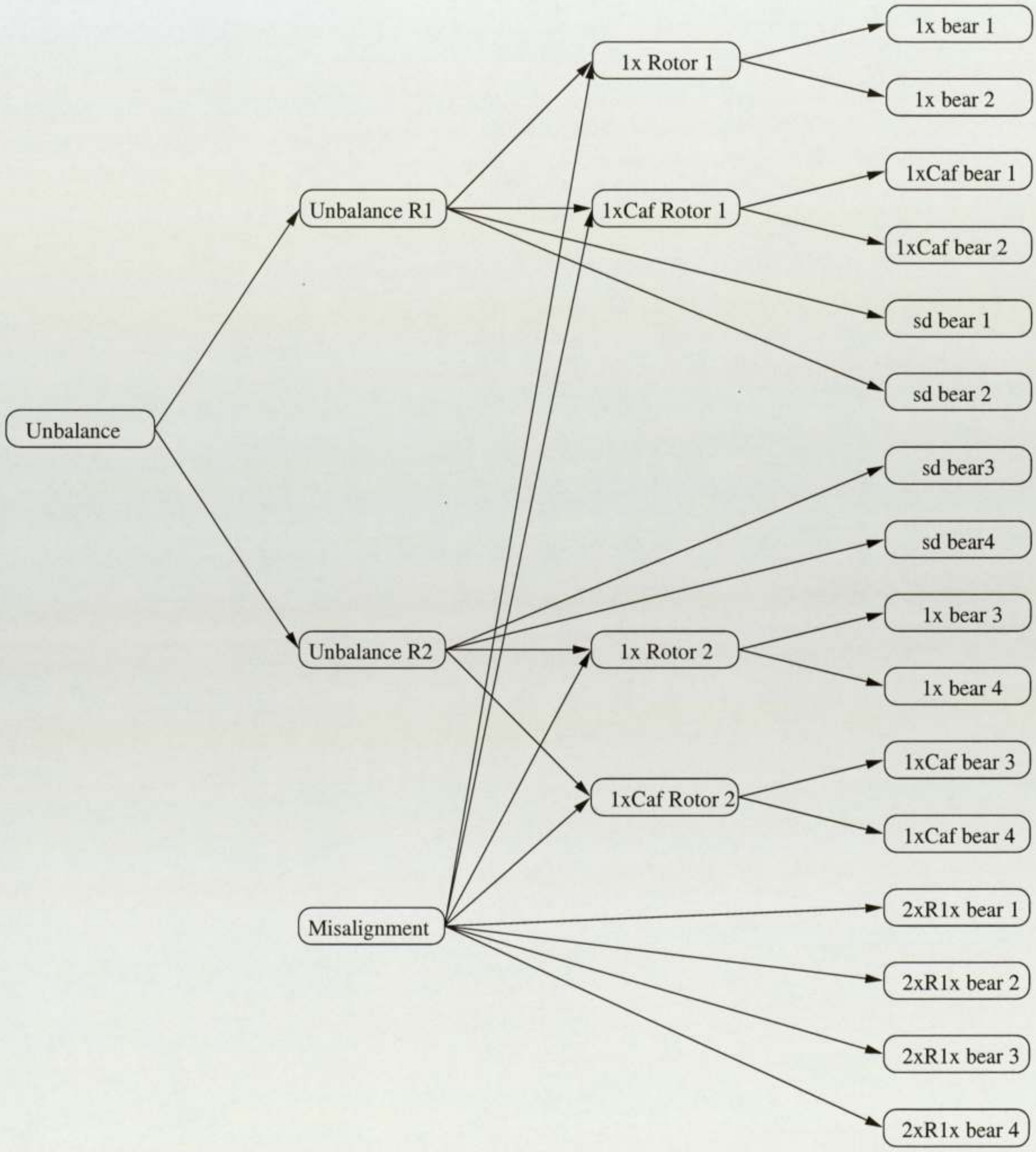


Figure 5.4: Belief network after refinements

5.3.3 Sensitivity analysis

Sensitivity analysis is useful to discover the relative importance of a variable in arriving at a diagnosis. During the construction of a network, sensitivity analysis can be also useful to identify which parts of the network may be critical and important for the diagnostic.

One useful measure of sensitivity to predictive or causal evidence is the *sensitivity range* of the probability of an event y with respect to an event x (y and x are binary variables). Suppose e is an assessment error (viewed as an event) which might affect the assessment of the probability of x, giving $p(x|e)$. Suppose that y is conditionally independent of e given x.

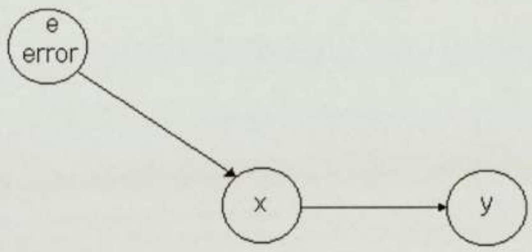


Figure 5.5: Network showing y conditionally independent of error e, given x

Then the sensitivity range is defined as the derivative of $p(y|e)$ with respect to $p(x|e)$:

$$SR(y,x) = \frac{dp(y|e)}{dp(x|e)} \tag{5.1}$$

Given conditional independence, so that $p(y|x) = p(y|x,e)$, we have,

$$p(y|e) = p(y|x).p(x|e) + p(y|\bar{x}).(1 - p(x|e)) \tag{5.2}$$

Taking the derivative with respect to $p(x|e)$, we get,

$$SR(y,x) = p(y|x) - p(y|\bar{x}) \tag{5.3}$$

So, it turns out that the sensitivity range is equal to the maximum possible change in the probability of y which can be caused if error e varies from 0 to 1.

Since a sensitivity range is a difference between two probabilities, its absolute magnitude cannot be greater than one. If the link is non-deterministic, then it must be strictly less than one:

$$| SR(y, x) | < 1 \tag{5.4}$$

Consequently the effect of any error in judging a probability of a causal or predictive variable cannot be greater than the magnitude of the error. Thus errors in probability assessment cannot be amplified by the model. In general they will be attenuated as the number of cascaded uncertain inferences is increased. Suppose we have a causal or predictive chain: It is easy to show that the sensitivity range of the end of a chain with respect to an error in the probability $p(x_1|e)$ is simply the product of the sensitivity ranges for each of the intermediate links:

$$SR(x_n, x_1) = \prod_{i}^{n-1} SR(x_{i+1}|x_i) \tag{5.5}$$

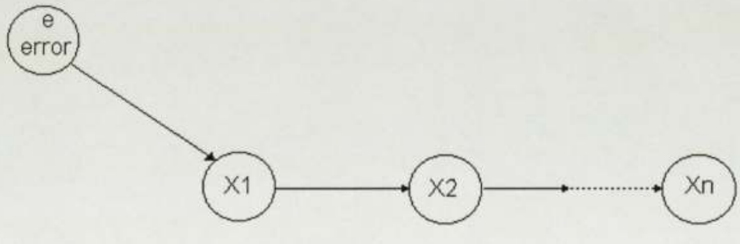


Figure 5.6: Propagating the effect of error e along a causal chain

We may also consider the potential impact in errors in assessments of the link probabilities, $p(x_{i+1}|x_i)$. Since the absolute sensitivity range of each link must be less than unity if it is non-deterministic, each link serves to dilute the impact of all the others, whether they come before or after it in the chain. This supports the intuition that the longer a chain of uncertain reasoning, the more tenuous the results.

However, things can be a little different for diagnostic links. Suppose variable a influences b, but there is some error e in the assessment of the influence of a on b. We assume that a and e are independent.

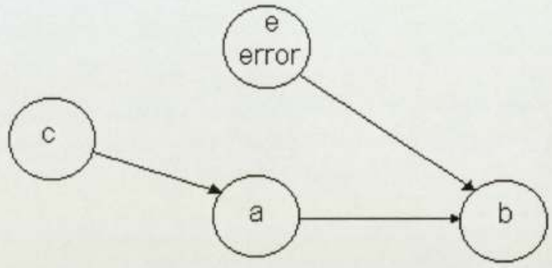


Figure 5.7: Examining the sensitivity of variable a to diagnostic information b

We define the prior of a as $O(a)$.The assessment of the likelihood ratio may be affected by the error e:

$$L(b, a|e) = \frac{p(b|a, e)}{p(b|\bar{a}, e)} \tag{5.6}$$

Bayes' rule gives the formula:

$$p(b, a|e) = \frac{p(b|e)p(a|b, e)}{p(a|e)} \tag{5.7}$$

so

$$L(b, a|e) = \frac{p(a|b, e)p(\bar{a}|e)}{p(a|e)p(\bar{a}|b, e)} L(b, a|e)O(a) = \frac{p(a|b, e)}{1 - p(a|b, e)} \tag{5.8}$$

we can restate the posterior probability for a in terms of the prior odds and likelihood:

$$p(a|b, e) = \frac{L(b, a|e)O(a)}{L(b, a|e)O(a) + 1} \tag{5.9}$$

Taking the derivative of this posterior with respect to the likelihood ratio as a measure of sensitivity, we get:

$$\frac{dp(a|b, e)}{dL(b, a|e)} = \frac{O(a)}{(L(b, a|e)O(a) + 1)^2} \tag{5.10}$$

This sensitivity can get large when $O(a)$ is large and $L(a, b|e)$ is small.

5.4 Learning

During the construction of a belief network, the most difficult task is to quantifier the value in the CPT. This section presents a method for learning this CPT from the data.

5.4.1 Learning from Cases

Belief network learning is the automatic process of determining a suitable belief network, given data in the form of cases. A case is a set of variables which describe a particular event (for example the probabilities of the symptoms when the machine run in normal condition operation at 30 Hz). Some cases may not have values for some variables that other cases do, which is known as missing data. We can ignore some of the variables in order to see its influence (or its lack of influence) to diagnostic a particular fault.

The belief network learning task has traditionally been divided into two parts:

1. **structure learning** determines the dependence and independence of variables and suggests a direction of causation, in other words, the placement of the links in the network.

2. **Parameter learning** determines the conditional probability relationship at each node, given the link structures and the data.

Here, we are interested in the second part, as the structure has been designed by hand.

5.4.2 Experience

There is considerable controversy over the best way to represent uncertainty, with some of the suggestions being probability, fuzzy logic, nonmonotonic logic, belief functions, Dempster-Shafer, etc. Currently probability and fuzzy logic are the most practical methods for most applications. Of these two, probability has a much sounder theoretical basis (at least with respect to the way it is actually used). However, a deficiency of using only probability is the inability to represent ignorance in an easy way.

As an example, suppose we work for a firm which has developed a diagnostics tool for rotating machine. Each time, we test a rotating machine we try to predict if the machine is in good state or not. If you go for the first time in the firm X, you known nothing about the rotating machine in this particular firm, so the probability that the machine has no fault is 0.5.

Now, you have visited the firm X several time, and you have a good knowledge of this firm. So two situations appear:

1. the employer takes care of their machines, and the firm has a good maintenance service. In this situation, the probability that the machine is in a good state, is more than 0.5.

2. the machines are being over-used. In this case, the probability that the machine is in a good state, decreases.

This example illustrates the concept of experience: We are able to do a better prediction of the state of the rotating machine simply because we known additional information.

One way to handle this using just probabilities is to keep track of the beliefs about the ratio of normal to abnormal condition. Then we will have many probabilities, one for each possible ratio. Each of these probabilities will change as , and when you are asked to supply a probability that the next state will be normal, they will all be involved in the calculation. These are sometimes called second order probabilities, but in this example they are really just a probability distribution over possible ratios.

Instead we can introduce the concept of experience, which is a measure of the confidence that the belief network has in its probabilities.

At each node we store one experience number for each possible configuration of states of the parent nodes, along with the CPT vector of conditional probabilities (one probability for each state of the node). The experience value corresponds closely to the number of cases that have been seen or its equivalent (normally it is 1 more than the number of cases). This form of experience has sometimes been called the "equivalent sample size (ess) ".

5.4.3 Learning Algorithm

This describes the algorithm used for parameter learning of conditional probability tables from a file of cases.

Before learning begins, the network starts off in a state of ignorance (providing there has been no previous learning or entry of probabilities by an expert). At each node, all CPT probabilities start as uniform, and each experience starts at its lowest value (normally 1.0).

So for each node i, for each state s_i of a node i and for each instantiation s_{π_i} of the set of parents π_i of the node i

$$p_i(s_i, s_{\pi_i}) = \frac{1}{n_i} \quad (5.11)$$

$$e_i(s_i, s_{\pi_i}) = 1 \quad (5.12)$$

where

- $p_i(s_i, s_{\pi_i})$ is the conditional probability that the node i has the state s_i when the set of parent π_i has the instantiation s_{π_i}.

- $e_i(s_i, s_{\pi_i})$ is the experience number corresponding at the conditional probability $p_i(s_i, s_{\pi_i})$.

- n_i the number of different states of the node i.

For each case to be learned the following is done. Only nodes for which the case supplies a value (finding), and supplies values for all of its parents, have their experience and conditional probabilities modified (i.e., no missing data for that node).

So for each node i, for each state s_i of a node i and for each instantiation s_{π_i} of π_i , the conditional probability $p_i(s_i, s_{\pi_i})$ is modified as follow:

$$p_i(s_i, s_{\pi_i}) = \frac{p_i(s_i, s_{\pi_i})e_i(s_i, s_{\pi_i}) + 1}{e_i(s_i, s_{\pi_i}) + 1} \tag{5.13}$$

$$e_i(s_i, s_{\pi_i}) = e_i(s_i, s_{\pi_i}) + 1 \tag{5.14}$$

The other probability $p_i(s'_i, s_{\pi_i})$ where s'_i is a different state of the node i are modified as follow:

$$p_i(s'_i, s_{\pi_i}) = \frac{p_i(s'_i, s_{\pi_i})e_i(s_i, s_{\pi_i})}{e_i(s_i, s_{\pi_i}) + 1} \tag{5.15}$$

in order to respect the constraint:

$$\sum_{k=1}^{n_i} p_i(s_i^k, s_{\pi_i}) = 1 \tag{5.16}$$

For more information see [6] section 4.1 (with the word *precision* equivalent to *experience*).

5.4.4 Fading

The previous algorithm allows us to built the CPT from a cases file. In this algorithm each case as the same influence in modifing each probability of each CPT. In a world which is constantly changing new cases have to be include in the CPT of each node in order to update the belief network. Nevertheless it is useful to treat more recent cases with a higher weight than older ones to match a changing world.

We can achieve this partial forgetting of the past by using fading. We will reduce the experience and smooth the probabilities of the selected nodes by an amount dictated by the degree, with 0 having no effect, and 1 creating uniform distributions with no experience (thereby undoing all previous learning). Then when you continue to learn new cases, they will effectively be weighted more than the cases you just faded.

Fading once with $\alpha = 1 - d$, and again with $\alpha = 1 - f$, is equivalent to a single fading with $\alpha = 1 - df$. So the effects of multiple fadings accumulate as they should.

To be most accurate you would fade a very small amount after each case, but for all practical purposes you can just fade a larger amount after a batch of cases.

If an occurrence time for each case is known, and the cases are learned sequentially through time, then the amount of fading to be done is: $\alpha = 1 - r^{\Delta t}$ where Δt is the amount of time since the last fading was done, and r is a positive number less than (but close to) 1, and depends on the units of time and how quickly the environment is changing. Different nodes may require different values of r.

During fading, each of the probabilities in the node's CPT is modified as follows:

$$\begin{aligned} p_i(s_i, s_{\pi_i}) &= C\left\{p_i(s_i, s_{\pi_i})e_i(s_i, s_{\pi_i}) - \alpha\left\{p_i(s_i, s_{\pi_i})e_i(s_i, s_{\pi_i}) - \frac{1}{n_i})\right\}\right\} && (5.17)\\ &= C\left\{(1-\alpha)p_i(s_i, s_{\pi_i})e_i(s_i, s_{\pi_i}) + \frac{\alpha}{n_i}\right\} && (5.18)\end{aligned}$$

We have to respect the constraint (5.14) so

$$\frac{1}{C} = \sum_{k=1}^{n_i}(1-\alpha)p_i(s_i^k, s_{\pi_i})e_i(s_i^k, s_{\pi_i}) + \frac{\alpha}{n_i} \tag{5.19}$$

Chapter 6

Results

6.1 Data set

To built the data set we use only a part of the data provided by the Modiarot's team (problem of size and computation time). We use the data from the series 1, 2, 3, and 4[1].

For each series, we take one third of the rotating speed (between 55Hz and 20Hz). The data from series 5 are not used because they duplicate with data provided by the first series (with extended rundown range 55Hz to 2Hz). The series 6 are not used because we do not have enough data in order to train the model.

To process the signal, we take $Nsamp = 256$ and $Nshift = 32$ so we have 16 different portion of signal for each rotating speed and for each test. Furthermore, half of the data is used to train and to construct the model; the second half is used to test it.

6.2 Classification results

We have decided that the nodes are binary variables (Present or Absent), so in order to calculate the classification error, we simply take the formulae:

[1]for details see annexe A

$$E_0 = \frac{\sum_{n=1}^{N}(1-t_n)g(y_n)}{\sum_{n=1}^{N}(1-tn)} \tag{6.1}$$

$$E_1 = \frac{\sum_{n=1}^{N} t_n(1-g(y_n))}{\sum_{n=1}^{N} tn} \tag{6.2}$$

$$E = \frac{1}{N}\sum_{n=1}^{N}(1-t_n)g(y_n) + t_n(1-g(y_n)) \tag{6.3}$$

where

$$g(x) = \begin{cases} 1 & \text{if } x > 0.5 \\ 0 & \text{otherwise} \end{cases} \tag{6.4}$$

E_0 represents the classification error for the output of *normal* class (i.e. $t_n = 0$), E_1 the classification error for the output of *fault* class (i.e. $t_n = 1$). E is simply the global classification error.

6.2.1 Using threshold functions

The classification error is very small for the symptom **r**. In particular the error of classification is below to 1 percent when the state of the machine is normal. So, the model using threshold functions could distinguish between the normal operating condition and the faults.

Unfortunately, the classification error for the symptoms **2xR1x, 3xCaf and 3x** is greater than 40 percent. So, it is seems that the model could have some difficulty in distinguishing between the misalignment fault and the other states of the rotating machine.

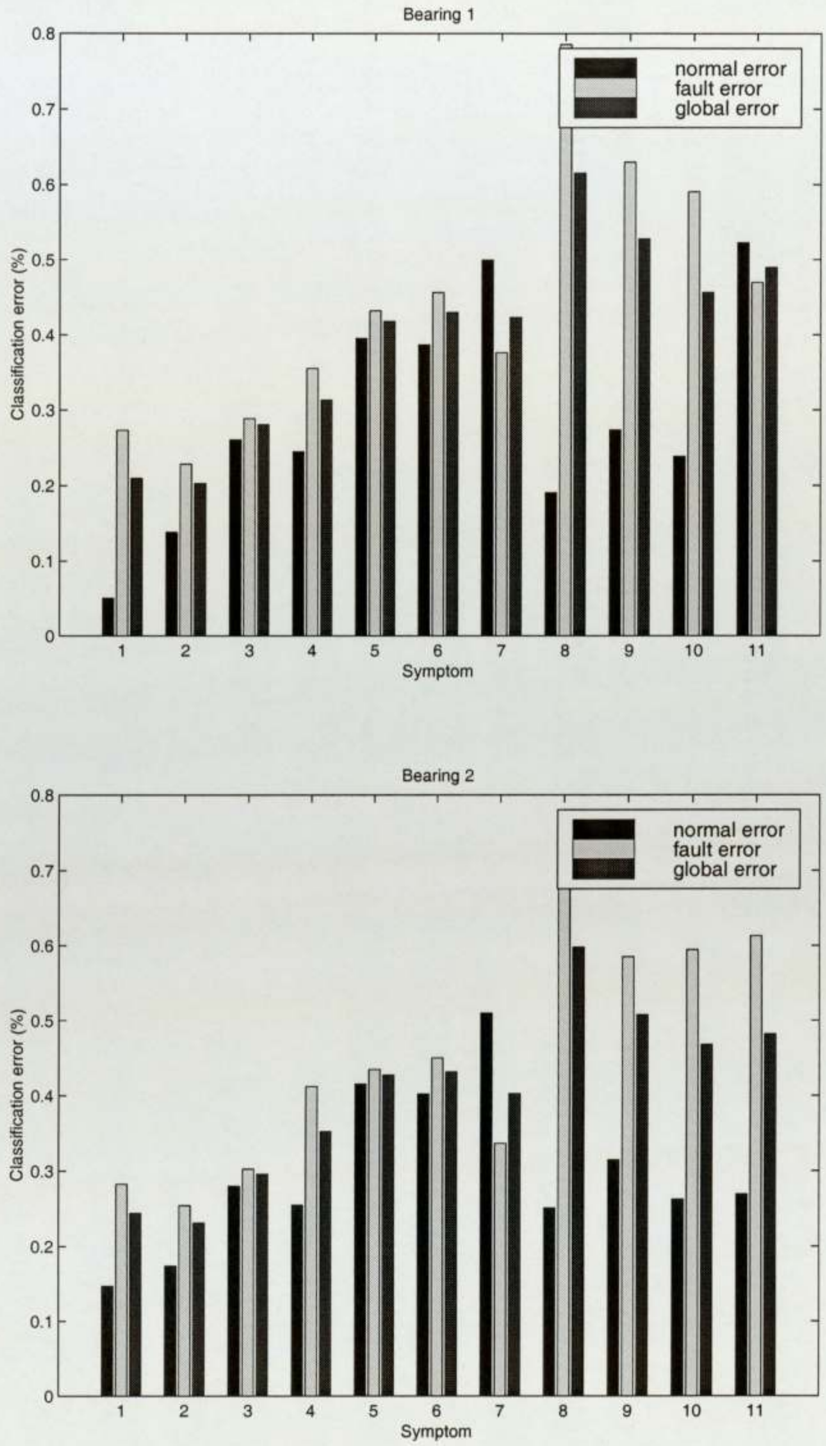
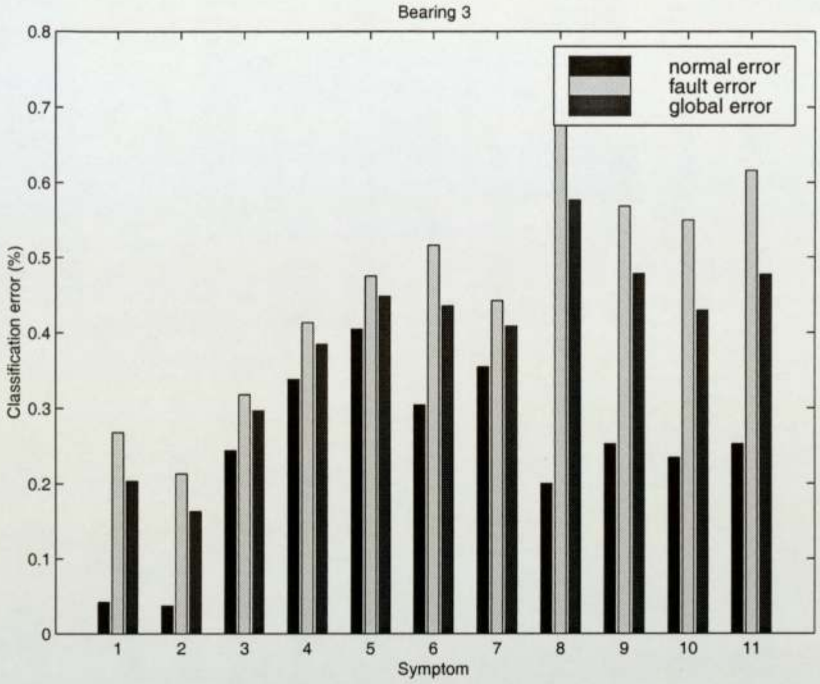


Figure 6.1: Classification error using threshold functions (1: r, 2: 1x, 3: 2x, 4: 3x, 5: sd, 6: sc, 7: su, 8: 1xCaf, 9: 2xCaf, 10: 3xCaf, 11: 2xR1x)



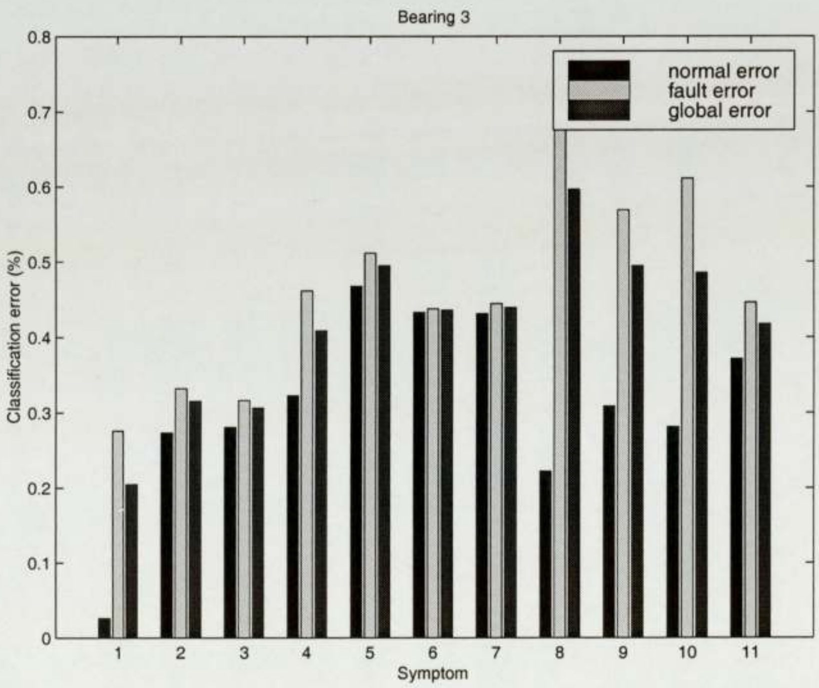


Figure 6.2: Classification error using threshold functions (1: r, 2: 1x, 3: 2x, 4: 3x, 5: sd, 6: sc, 7: su, 8: 1xCaf, 9: 2xCaf, 10: 3xCaf, 11: 2xR1x)

6.2.2 Using neural networks

The global error using neural networks is better than using threshold functions. So, we can hope have better results using neural networks than threshold functions. In particular, the diagnostic of the misalignment fault will be better because the classification error for the symptom **2xR1x** is less then 30 percent.

Nevertheless, for the symptom **1xCaf** and **2xCaf**, the neural networks are unable to classifis the target. So, these symptoms will have no influence to diagnostic the state of the machine when we use neural networks.

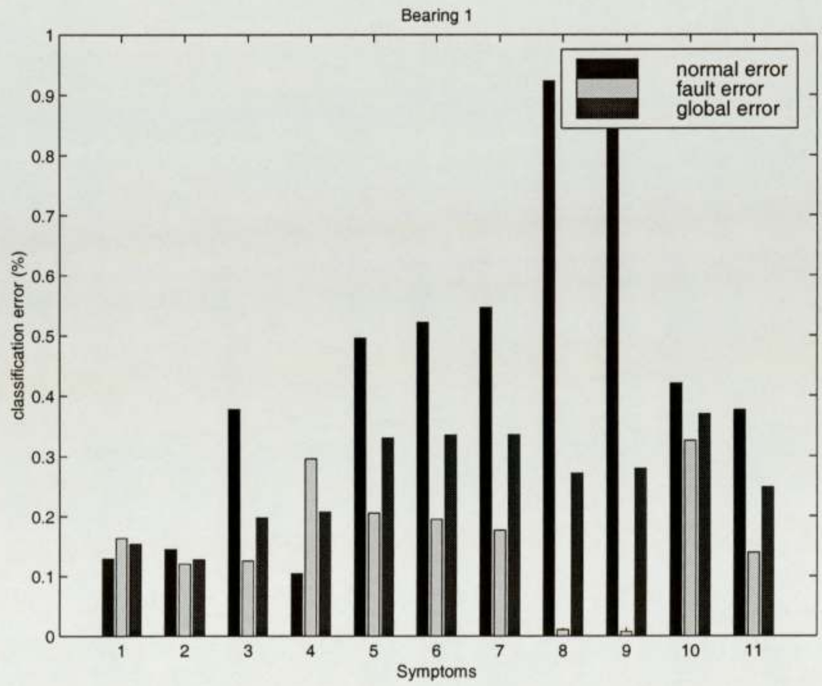


Figure 6.3: Classification error using neural networks (1: r, 2: 1x, 3: 2x, 4: 3x, 5: sd, 6: sc, 7: su, 8: 1xCaf, 9: 2xCaf, 10: 3xCaf, 11: 2xR1x)

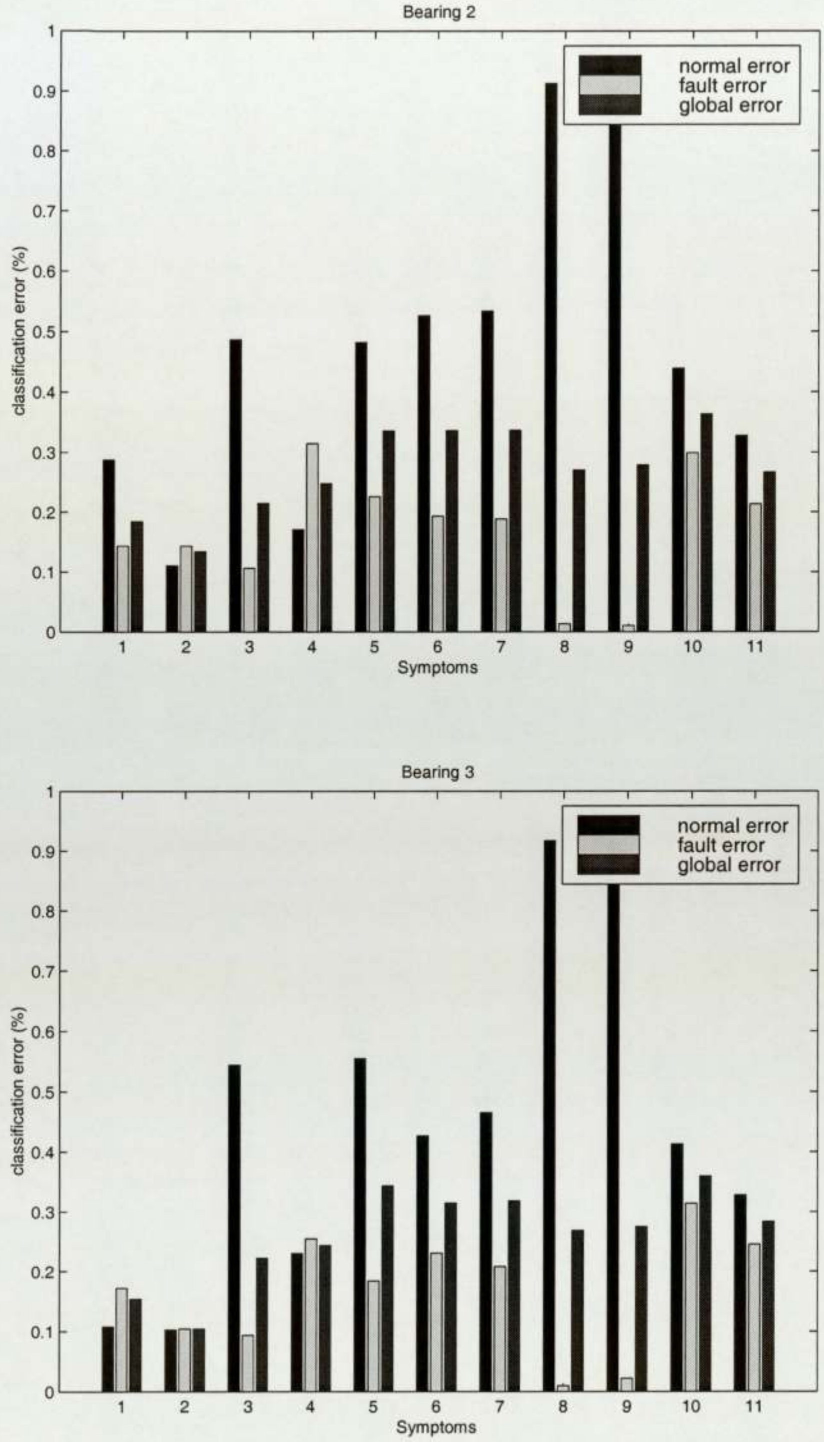


Figure 6.4: Classification error using neural networks (1: r, 2: 1x, 3: 2x, 4: 3x, 5: sd, 6: sc, 7: su, 8: 1xCaf, 9: 2xCaf, 10: 3xCaf, 11: 2xR1x)

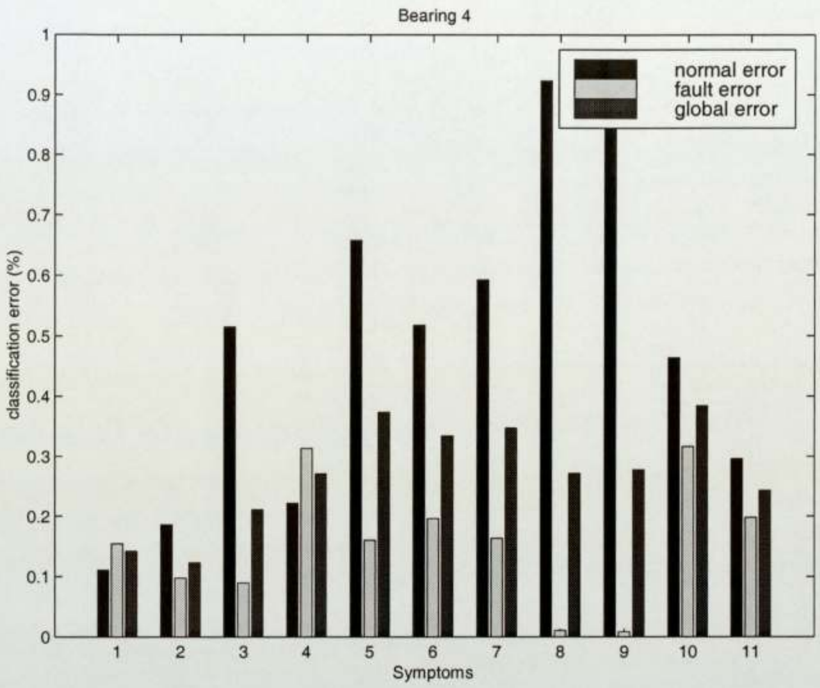


Figure 6.5: Classification error using neural networks (1: r, 2: 1x, 3: 2x, 4: 3x, 5: sd, 6: sc, 7: su, 8: 1xCaf, 9: 2xCaf, 10: 3xCaf, 11: 2xR1x)

6.3 General results

The following belief network is obtained using the techniques explained in Chapter 5. In addition of the symptom and fault nodes, we introduce the node **state**: this node permits us to link the different states of the rotating machine. The number of possible values for this node is equal to the number of different states that the belief network can distinguish (In our case, this node could take tree values: normal, unbalance and misalignment).

In the test step, the node **state** shows the performance of the networks using a matrix $(s_{i,j})$ where the $s_{i,i}$ elements are the number of good diagnostics for the state i and the $s_{i,j}$ elements the number of cases where the belief network diagnostics the state j instead of the state i.

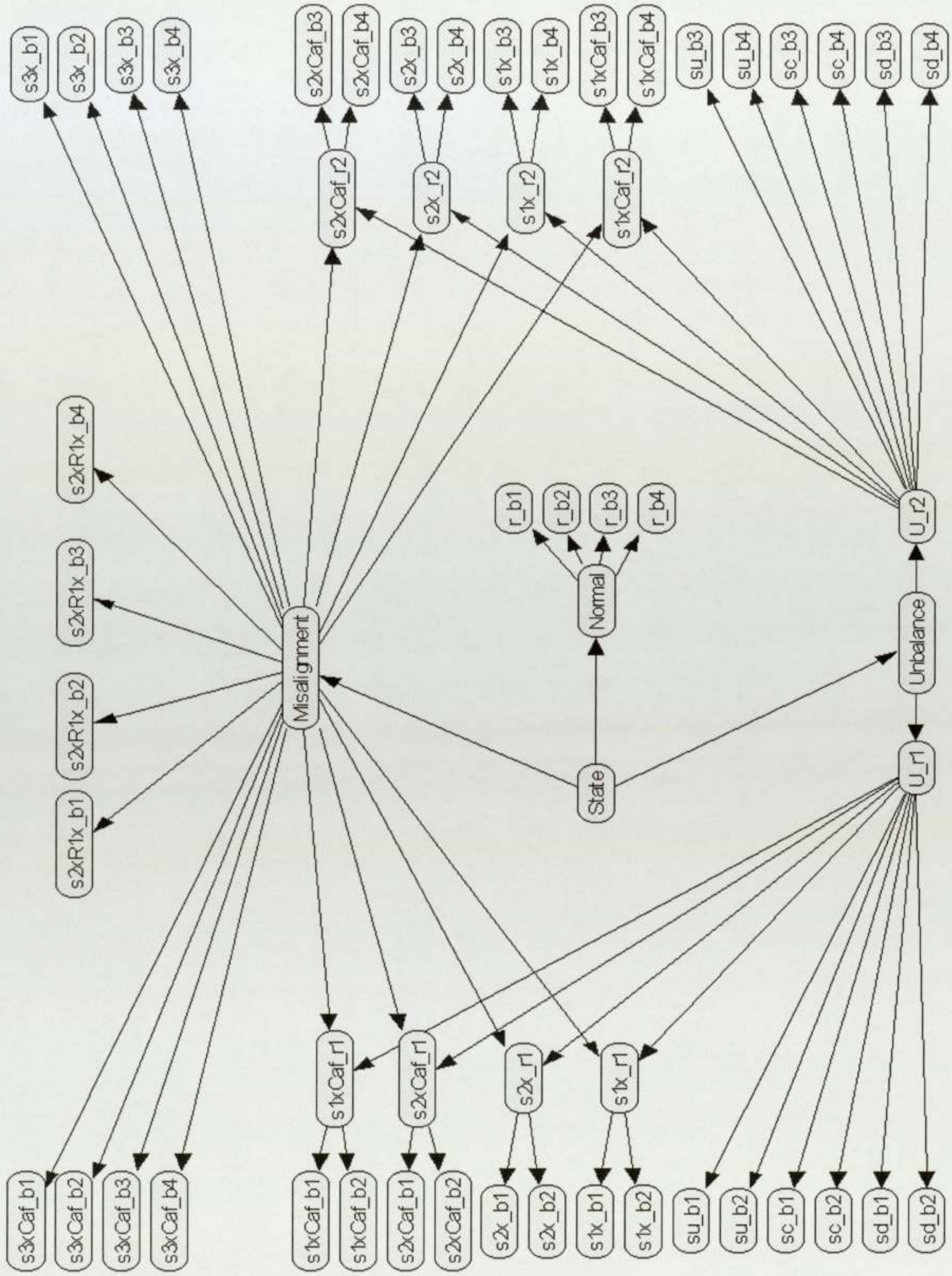


Figure 6.6: Final Belief Network

The belief network was built with the commercial software Netica. It is impossible in a 9-month project to develop a software able to build, train and test a belief network. The Netica software was chosen because it can train a belief network with a cases file [5] [4].

6.3.1 Using threshold functions

Actual	Predicted		
	Normal	Unbalance	Misalignment
Normal	1060	53	15
Unbalance	111	938	455
Misalignment	422	341	553

Table 6.1: Diagnostic error

We obtain a global error of 35 %

When a particular fault is present, the probability that the fault is detected is:

- 94 % for the normal operating condition
- 62 % for the unbalance fault
  - 20 % of error cases are diagnostic as normal operating condition
  - 80 % of error cases are diagnostic as misalignment fault
- 42 % for the misalignment fault
  - 55 % of error cases are diagnostic as normal operating condition
  - 45 % of error cases are diagnostic as unbalance fault

The bad result for the misalignment fault is due to the classification problem for the symptom **2xR1x**. With this poor result, the belief network is unable to detect the misalignment fault when it is present.

We do not obtain 100 % in diagnosing of normal operating condition because the Root Mean Square for some unbalance and misalignment fault is close to the RMS value for normal operating condition.

6.3.2 Using neural networks

Actual	**Predicted**		
	Normal	**Unbalance**	**Misalignment**
Normal	1082	42	4
Unbalance	41	967	496
Misalignment	308	140	868

Table 6.2: Diagnostic error

We obtain a global error of 26 %

When a particular fault is present, the probability that the fault is detected is:

- 96 % for the normal operating condition
- 64 % for the unbalance fault
  - 8 % of error cases are diagnostic as normal operating condition
  - 92 % of error cases are diagnostic as misalignment fault
- 66 % for the misalignment fault
  - 69 % of error cases are diagnostic as normal operating condition
  - 31 % of error cases are diagnostic as unbalance fault

We notice that the better classification result for the symptom **2xR1x** permits a better diagnostic for the misalignment fault.

These results are in the range define in [11], with the rule-based method they obtain between 60 % and 80 % in diagnosing the faults and 100 % for diagnostic normal operating condition.

6.3.3 Discussion

The belief network has better results when using neural networks than using threshold functions. This result is not a surprise according to the better classification error using neural networks than using threshold functions.

Nevertheless, in both methods more 90 % of the error is a mistake to distinguish the misalignment fault from the other machine's conditions. This error could be explained by the fact that we have no access to valuable information, in particular:

- **the orbital plot of the shaft**:
  this plot is obtained from the signal provided by two proximity sensors; the form of the shaft orbit is characteristic for several faults.

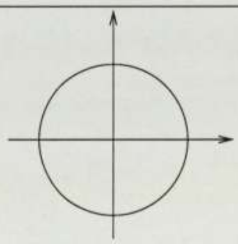
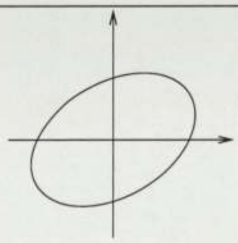
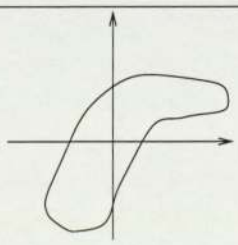
	characteristic orbital plot of normal condition operation
	characteristic orbital plot of unbalance fault
	characteristic orbital plot of misalignment fault

Table 6.3: Characteristic orbital plots

- **vibration signal during a long period of time**:
  according to the Modiarot team, change over a short (less than 1 minute), medium (between a few minutes and a few hours) and long (more than 12 hours) period of time in the two first harmonics gives useful information to detect misalignment. Unfortunately, we have only 16 turns of the machine recorded for each rotating speed.

We have introduce the nodes unbalance_r1 and unbalance_r2 in order to help the operator to find the localisation of the unbalance fault. The error for this two nodes is close to the unbalance error when this fault is not present. But when this fault is present their errors are 10 % greater than the error of the node unbalance. This difference between the unbalance node is explained by the fact that the node state is linked to the fault nodes. When the node normal and misalignment takes the *absent* value, the unbalance node takes the value *present* even if the both nodes unbalance_r1 and unbalance_r2 have the value *absent*.

Chapter 7

Conclusions and suggestions

7.1 Conclusions

The implementation of the complete model (i.e. using neural networks and belief network) is able to correctly diagnose the case studies. The results are in the range of the result obtain by the matrix method.

The mean used to obtain the probability of the symptoms are speed dependent. So, we need as many as neural networks than different rotating speeds. An attempt to use a neural network with the value of the symptom and the rotating speed as input variables was made. However, the classification results with this neural network were very bad (in particular the neural network was unable to class the symptom 1xCaf, 2xCaf, 3xCaf and 2xR1x). The neural network used was a simple radial basis functions network with one hidden layer. Tailoring a more complex neural network could solve this problem. Nevertheless, several rotating machines have one or a few number of rotating speed.

A big problem in developing a diagnostic system for vibration analysis is that the knowledge of the symptoms of possible faults is not complete. We have to tailor the model for each particular machine in order to maximise the number of good diagnostics. But, in general, we have no data to train the neural networks and the belief network (in particular, vibration signal when a fault is present). The only way to solve this problem is to use the expert's knowledge with all its inherent problems (see chapter 2).

Nevertheless, the introduction of new symptoms (such as the orbital plot, gas path analysis, temperature and pressure monitoring, oil analysis, etc.) could increase the performance of the model.

Furthermore, if vibration signals of the different states are available, the training procedure of the model can be entirely automotatic. This fact is a great advantage of the model and permits to find the best configuration for the sensor localisation on the machine.

Of course, the test of this model on only one machine is not a proof of the model's validity but the results are encouraging.

7.2 Suggestions for further work

Several different rotating machines exist (gearbox, centrifugal compressor, etc) and their characteristics could be very different. So, the first work to do is to test the model on different rotating machines in order to establish the model's validity for these type of machine.

More work has to be done in extracting symptoms from the measured data. In particular, building a neural network able to give the presence probability for each symptom with the numerical value of the symptom and the rotating speed of the machine as inputs would be interesting.

Rotating machines could have several faults: the Modiarot's team built a system to distinguish between 11 different faults. It would be interesting to known the model's response when the model copes with the 11 faults. Another point is to test the model with a unlearnt fault, however such test needs a more complex belief network which introduce the notion of abnormal state.

Another problem in condition monitoring is to deal with the case where two different faults are present at the same time. Such as there the case where we have and thus, we do not known the model's response to this particular situation.

In chapter 2, we have seen that many of the problems which occur are simply instrumentation failure. So, test the model's response to sensor failure could give us valuable information on the model performance.

Appendix A

Data set description

Description of the different dataset. For each serie of test the Modiarot team provides a test under normal operating condition.

1. A series of rundowns (55 Hz to 20 Hz) in various states of unbalance. 12 channels recorded (8x accelerometers, 4x mobile (RS) proximitors).

Weight (g)	Disk number	Hole number
17.4	1	1
17.4	2	1
17.4	3	1
17.4	4	1
17.4	5	1
8.6	3	7

Table A.1: Description of test for unbalance (one weight)

2. A series of rundowns(55 Hz to 20 Hz) in various states of angular misalignment (shim of 0.08, 0[1], 0.05 mm) . 12 channels recorded (8x accelerometers, 4x mobile (RS) proximitors).

[1]no misalignment repeat to check no permanent change

3. A series of rundowns (extended 55Hz to 2Hz) in further states of unbalance. 12 channels recorded (8x accelerometers, 4x mobile (RS) proximitors).

Weight 1			**Weight 2**		
Weight (g)	**Disk**	**Hole**	**Weight (g)**	**Disk**	**Hole**
20	1	7	20	3	22
20	2	18	20	5	13
20	1	10	20	5	5

Table A.2: Description of test for unbalance (two weights)

4. A series of rundowns (55Hz to 2Hz) in various states of angular misalignment(shim of 0.025, 0.050, 0.025 mm). 12 channels recorded (8x accelerometers, 4x mobile (RS) proximitors).

5. A series of rundowns (55Hz to 2Hz) in various states of unbalance. 16 channels recorded (8xaccelerometers, 8x mobile (RS) proximitors).

Weight (g)	**Disk number**	**Hole number**
17.4	1	1
17.4	2	1
17.4	3	1
17.4	4	1
17.4	5	1
8.6	3	7

Table A.3: Description of test for unbalance (one weight)

6. Rundown (55Hz to 2Hz) with 0.1 mm radially misaligned coupling. 16 channels (8x accelerometers, 8x proximitors).

Appendix B

Belief network softwares

To built, train and test a software has been used. To choose the software, several were tested. The following sites list different softwares available:

- http://www.sis.pitt.edu/ dsl/da-software.html

- http://www.cs.berkeley.edu/ murphyk/Bayes/bnsoft.html

- http://bayes.stat.washington.edu/almond/belief.html

B.1 JavaBayes

- **web** : http://www.cs.cmu.edu/ javabayes/Home/
- **advantages** : JavaBayes was written in Java so this software doesn't depend on the operating system you use.
- **disadvantages** : the graphic interface is not so easy (it is especially uneasy to see the belief values after a computation).

B.2 Genie

- **web** : http://www2.sis.pitt.edu/ genie/

- **advantages** : Compared with the other softwares Genie has two important advantages. First, it proposes several method of computation. Secondly , it can read several different types of files (hugins, netica ...) so we can create our belief network with these softwares and use Genie for the calculus.
- **disadvantages** : like JavaBayes , we can not easily read the belief values after a computation

B.3 Hugin

- **web** : http://www.hugin.dk
- **advantages** : like netica, hugin has a very efficient graphic interface and an application program interface. In addition, Hugin provides DDE (Dynamic Data Exchange). With DDE other programs (like Excel) are able to use the probabilities calculated in Hugin.
- **disadvantages** : Hugin has only one disadvantage : it is not a free software

B.4 Netica

- **web** : http://www.norsys.com
- **advantages** : Netica has a very efficient graphic interface. But its biggest advantage compared with the other softwares is that netica is able to learn from a case file. As a result, you can build the belief networks (in particular the conditional probability table) from a case file. In addition, netica has a application program interface (API). The Netica API is a complete library of functions for working with Bayesian belief networks and influence diagrams that you can call from your own program. It contains functions to build, learn, modify, transform, save and read networks, as well as a powerful inference engine. It may be embedded in programs written in any language (such as Java, C++, C, Visual Basic, Delphi, or Pascal), as long as the language can call C functions.

- **disadvantages** : Netica has only one disadvantage : it is not a free software

Bibliography

[1] A.K. Nandi A.C. McCormick and L. Jack. Digital signal processing algotithms in condition monitoring. *International Journal of COMADEM*, 3:5–14, 1998.

[2] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford University Press, Oxford, 1995.

[3] R.A. Collacott. *Vibration Monitoring and Diagnosis.* John Wiley & Sons, New York, 1979.

[4] Norsys Software Corp. *Netica API Programmer's library.* Norsys Software Corp, 1997.

[5] Norsys Software Corp. *Netica Application for Belief Networks and Influence Diagrams.* Norsys Software Corp, 1997.

[6] Steffen L. Lauritzen David J. Spiegelhalter, A. Philip Dawid and Robert G. Cowell. Bayesian analysis in expert systems. *Statistical Science*, 8:219–283, 1993.

[7] Andrew D. Dimarogonas. Vibration and acoustic monitoring of rotating shafts. October 1993.

[8] K.R. Fyfe and E.D.S Munck. Analysis of computed order tracking. *Mechanical Systems and Signal Processing*, pages 187–205, 1997.

[9] Max Henrion. Some praticale issues in constructing belief networks. *Uncertainty in Artificial Intelligence*, 3:161–173, 1989.

[10] Steffen L. Lauritzen and David J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *Royal Statistics Society B*, 50:157–194, 1988.

[11] M. F. White M. Jecmenica. Advanced computation methods applied for diagnosis matrix method. 1999.

[12] J.S. Mitchell. *Machinery analysis and Monitoring.* Penwell Press, 1981.

[13] M. Neale. *A Guide to the conditions monitoring of machinery.* Her Majesty's Stationery Office, 1981.

[14] Richard E. Neapolitan. *Probabilistic Reasoning in Expert Systems: Theory and Algorithms.* John Wiley & Sons, 1990.

[15] Judea Pearl. Fusion, propagation, and structuring in belief networks. *Artificial Intelligence*, 29:241–288, 1986.

[16] Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference.* Morgan Kaufmann, 1988.

[17] Jr. S. Lawrence Marple. *Digital spectral analysis with applications.* Prentice hall PTR, 1987.

[18] Christian Steinebach. Knowledge based systems for diagnosis of rotating machinery. Master's thesis, Institude of Technology Departement of Marine Enginneering Trondheim, January 1993.